版本信息：

版本

REV1.0

时间

11/23/2017

XILINX FPGA PCIE 修炼秘籍

基于 GTT/GTH 应用

电子版自学资料

常州一二三电子科技有限公司

溧阳米联电子科技有限公司

PCIE QQ 群：543731097

版本	时间	描述
Rev1.0	2017-12-09	PCIE_CH01 PCIE 协议理论基础部分

感谢您使用 CZ123/米联团队开发的开发板，在使用开发板前请认真阅读本手册，并且掌握如何正确使用开发板，不合理的操作会导致开发板损坏。

此手册不断更新中，请下载最新版本。

软件版本：VIVADO2016.4

使用本手册提供的 VIVADO 版本或者到赛灵思官网下载 VIVADO2016.4 版本

链接：http://pan.baidu.com/s/1i4WHtFB 密码：4drs

版权声明：

技术支持：

版主大神们都等着大家去提问--电子资源论坛 www.osrc.cn

微信公众平台：电子资源论坛

目录

【PCIE 专辑】本专辑讲解 XILINX FPGA PCIE 接口的使用

课程预计一共 10 节课左右，详细讲解 PCIE 的开发过程。

直播地址：https://osrc.ke.qq.com

录播地址：http://www.osrc.cn/forum.php?mod=forumdisplay&fid=339

PCIE_CH01 PCIE 协议理论基础部分

学习 PCIE 必须先掌握 PCIE 通信协议，本章转载了网络博客的关于 PCIE 的文章，并且摘录其中对我们学习 XILINX FPGA PCIE 通信技术具有理论基础的内容部分。

1.1 TLP 格式

PCIE 的数据报文首先通过事务层被封装为一个或者多个 TLP，之后通过 PCIe 总线的各个层次发送出去。TLP 的基本格式如下图所示。

一个完整的 TLP 由 1 个或者多个 TLP Prefix、TLP 头、Data Payload(数据有效负载)和 TLP Digest 组成。TLP 头是 TLP 最重要的标志，不同的 TLP 其头的定义并不相同。TLP 头包含了当前 TLP 的总线事务类型、路由信息等一系列信息。在一个 TLP 中，Data Payload 的长度可变，最小为 0，最大为 1024DW。

TLP Digest 是一个可选项,一个 TLP 是否需要 TLP Digest 由 TLP 头决定。Data Payload 也是一个可选项，有些 TLP 并不需要 Data Payload，如存储器读请求、配置和 I/O 写完成 TLP 并不需要 Data Payload。

TLP Prefix 由 PCIe V2.1 总线规范引入，分为 Local TLP Prefix 和 EP-EP TLP Prefix 两类。其中 Local TLP Prefix 的主要作用是在 PCIe 链路的两端传递消息，而 EP-EP TLP Prefix 的主要作用是在发送设备和接收设备之间传递消息。设置 TLP Prefix 的主要目的是为了扩展 TLP 头，并以此支持 PCIe V2.1 规范的一些新的功能。注意：我们的 PCIE 教程不涉及这部分。

TLP 头由 3 个或者 4 个双字(DW)组成。其中第一个双字中保存通用 TLP 头，其他字段与通用 TLP 头的 Type 字段相关。一个通用 TLP 头由 Fmt、Type、TC、Length 等字段组成，如下图所示

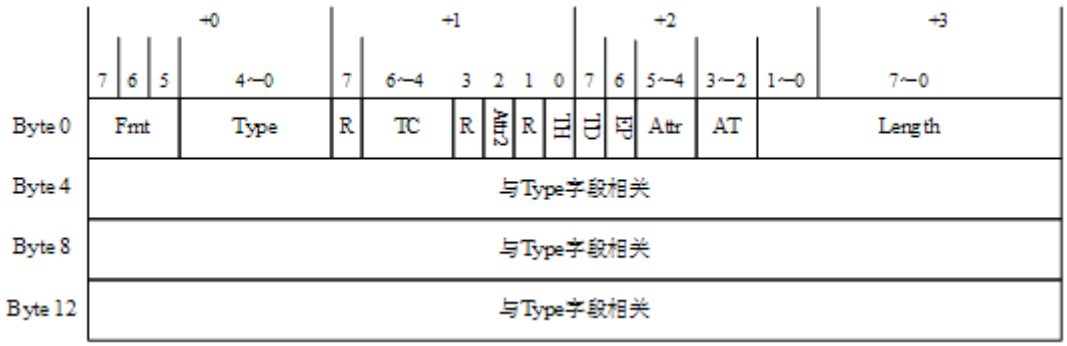

图 1-1

如果存储器读写 TLP 支持 64 位地址模式时，TLP 头的长度为 4DW，否则为 3DW。而完成报文的 TLP 头不含有地址信息，使用的 TLP 头长度为 3DW。其中 Byte 4~Byte 15 的格式与 TLP 相关，下文将结合具体的 TLP 介绍这些字段。

1.1.1 通用 TLP 头的 Fmt 字段和 Type 字段

Fmt 和 Type 字段确认当前 TLP 使用的总线事务，TLP 头的大小是由 3 个双字还是 4 个双字组成，当前 TLP 是否包含有效负载。其具体含义如下表所示。

表 1-1-1Fmt[1:0]字段

Fmt[2:0]	TLP 的格式
0b000	TLP 大小为 3 个双字，不带数据。
0b001	TLP 大小为 4 个双字，不带数据。
0b010	TLP 大小为 3 个双字，带数据。
0b011	TLP 大小为 4 个双字，带数据。
0b100	TLP Prefix
其他	PCIe 总线保留

其中所有读请求 TLP 都不带数据，而写请求 TLP 带数据，而其他 TLP 可能带数据也可能不带数据，如完成报文可能含有数据，也可能仅含有完成标志而并不携带数据。在 TLP 的 Type 字段中存放 TLP 的类型，即 PCIe 总线支持的总线事务。该字段共由 5 位组成，其含义如表所示。

表 1-1-1-1Type[4:0]字段

TLP 类型	Fmt[2:0]	Type[4:0]	描述
MRd	0b000 0b001	0b0 0000	存储器读请求；TLP 头大小为 3 个或者 4 个双字，不带数据。
MRdLk	0b000 0b001	0b0 0001	带锁的存储器读请求；TLP 头大小为 3 个或者 4 个双字，不带数据。

TLP 类型	Fmt[2:0]	Type[4:0]	描述
MWr	0b010 0b011	0b0 0000	存储器写请求；TLP 头大小为 3 个或者 4 个双字，带数据。
IORd	0b000	0b0 0010	IO 读请求；TLP 头大小为 3 个双字，不带数据。
IOWr	0b010	0b0 0010	IO 写请求；TLP 头大小为 3 个双字，带数据。
CfgRd0	0b000	0b0 0100	配置 0 读请求；TLP 头大小为 3 个双字，不带数据。
CfgWr0	0b010	0b0 0100	配置 0 写请求；TLP 头大小为 3 个双字，带数据。
CfgRd1	0b000	0b0 0101	配置 1 读请求；不带数据。
CfgWr1	0b010	0b0 0101	配置 1 写请求；带数据。
TCfgRd	0b010	0b1 1011	本书对这两种总线事务不做介绍。
TCfgWr	0b001	0b1 1011	
Msg	0b001	0b1 0r2r1r0	消息请求；TLP 头大小为 4 个双字，不带数据。 “rrr”字段是消息请求报文的 Route 字段，下文将详细介绍该字段。
MsgD	0b011	0b1 0r2r1r0	消息请求；TLP 头大小为 4 个双字，带数据。
Cpl	0b000	0b0 1010	完成报文；TLP 头大小为 3 个双字，不带数据。包括存储器、配置和 I/O 写完成。
CplD	0b010	0b0 1010	带数据的完成报文，TLP 头大小为 3 个双字，包括存储器读、I/O 读、配置读和原子操作读完成。
CplLk	0b000	0b0 1011	锁定的完成报文，TLP 头大小为 3 个双字，不带数据。
CplDLk	0b010	0b0 1011	带数据的锁定完成报文，TLP 头大小为 3 个双字，带数据。

TLP 类型	Fmt[2:0]	Type[4:0]	描述
FetchAdd	0b010 0b011	0b0 1100	Fetch and Add 原子操作。
Swap	0b010 0b011	0b0 1101	Swap 原子操作。
CAS	0b010 0b011	0b0 1110	CAS 原子操作。
LPrfx	0b100	0b0 L3L2L1L0	Local TLP Prefix
EPrfx	0b100	0b1 E3E2E1E0	End-End TLP Prefix

由上表所示，存储器读和写请求，IO 读和写请求，及配置读和写请求的 type 字段相同，如存储器读和写请求的 Type 字段都为 0b0 0000。此时 PCIe 总线规范使用 Fmt 字段区分读写请求，当 Fmt 字段是“带数据”的报文，一定是“写报文”；当 Fmt 字段是“不带数据”的报文，一定是“读报文”。

PCIe 总线的数据报文传送方式与 PCI 总线数据传送有类似之处。其中存储器写 TLP 使用 Posted 方式进行传送，而其他总线事务使用 Non-Posted 方式。

PCIe 总线规定所有 Non-Posted 存储器请求使用 Split 总线方式进行数据传递。当 PCIe 设备进行存储器读、I/O 读写或者配置读写请求时，首先向目标设备发送数据读写请求 TLP，当目标设备收到这些读写请求 TLP 后，将数据和完成信息通过完成报文(Cpl 或者 CplD)发送给源设备。

其中存储器读、I/O 读和配置读需要使用 CplD 报文，因为目标设备需要将数据传递给源设备；而 I/O 写和配置写需要使用 Cpl 报文，因为目标设备不需要将任何数据传递给源设备，但是需要通知源设备，写操作已经完成，数据已经成功地传递给目标设备。

在 PCIe 总线中，进行存储器或者 I/O 写操作时，数据与数据包头一起传递；而进行存储器或者 I/O 读操作时，源设备首先向目标设备发送读请求 TLP，而目标设备在准备好数据后，向源设备发出完成报文。

PCIe 总线规范还定义了 MRdLk 报文，该报文的主要作用是与 PCI 总线的锁操作相兼容，但是 PCIe 总线规范并不建议用户使用这种功能，因为使用这种功能将极大影响 PCIe 总线的数据传送效率。(我们的 PCIE 教程不涉及这部分内容)

与 PCI 总线并不相同，PCIe 总线规范定义了 Msg 报文，即消息报文。分别为 Msg 和 MsgD，这两种报文的区别在于一个报文可以传递数据，一个不能传递数据。

PCIe V2.1 总线规范还补充了一些总线事务，如 FetchAdd、Swap、CAS、LPrfx 和 EPrfx。其中 LPrfx 和 EPrfx 总线事务分别与 Local TLP Prefix 和 EP-EP TLP Prefix 对应。在 PCIe 总线规范 V2.0 中，TLP 头的大小为 1DW，而使用 LPrfx 和 EPrfx 总线事务可以对 TLP 头进行扩展，本节不对这些 TLP Prefix 做进一步介绍。PCIe 设备可以使用 FetchAdd、Swap 和 CAS 总线事务进行原子操作，本篇将在第 1.3.5 节详细介绍该类总线事务。

1.1.2TC 字段

TC 字段表示当前 TLP 的传送类型，PCIe 总线规定了 8 种传输类型，分别为 TC0 ~ TC7，缺省值为 TC0，该字段与 PCIe 的 QoS 相关。PCIe 设备使用 TC 区分不同类型的数据传递，而多数 EP 中只含有一个 VC，因此这些 EP 在发送 TLP 时，也仅仅使用 TC0，但是有些对实时性要求较高的 EP 中，含有可以设置 TC 字段的寄存器。

在 Intel 的高精度声卡控制器(High Definition Audio Controller)的扩展配置空间中含有一个 TCSEL 寄存器。系统软件可以设置该寄存器，使声卡控制器发出的 TLP 使用合适的 TC。声卡控制器可以使用 TC7 传送一些对实时性要求较强的控制信息，而使用 TC0 传送一般的数据信息。在具体实现中，一个 EP 也可以将控制 TC 字段的寄存器放入到设备的 BAR 空间中，而不必和 Intel 的高精度声卡控制器相同，存放在 PCI 配置空间中。

目前许多处理器系统的 RC 仅支持一个 VC 通路，此时 EP 使用不同的 TC 进行传递数据的意义不大。x86 处理器的 MCH 中一般支持两个 VC 通路，而多数 PowerPC 处理器仅支持一个 VC 通路。PLX 公司的多数 Switch 也仅支持两个 VC 通路。

有些 RC，如 MPC8572 处理器，也能决定其发出 TLP 使用的 TC。在该处理器的 PCIe Outbound 窗口寄存器(PEXOWARn)中，含有一个 TC 字段，通过设置该字段可以确定 RC 发出的 TLP 使用的 TC 字段。不同的 TC 可以使用 PCIe 链路中的不同 VC，而不同的 VC 的仲裁级别并不相同。EP 或者 RC 通过调整其发出 TLP 的 TC 字段，可以调整 TLP 使用的 VC，从而调整 TLP 的优先级。

1.1.3Attr 字段

Attr 字段由 3 位组成，其中第 2 位表示该 TLP 是否支持 PCIe 总线的 ID-based Ordering；第 1 位表示是否支持 Relaxed Ordering；而第 0 位表示该 TLP 在经过 RC 到达存储器时，是否需要进行 Cache 共享一致性处理。Attr 字段如下图所示。

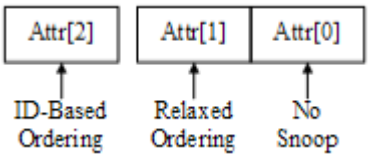

图 1-1-3-1

一个 TLP 可以同时支持 ID-based Ordering 和 Relaxed Ordering 两种位序。Relaxed Ordering 最早在 PCI-X 总线规范中提出，用来提高 PCI-X 总线的数据传送效率；而 ID-based Ordering 由 PCIe V2.1 总线规范提出。TLP 支持的序如表所示。

表 1-1-3TLP 支持的序

Attr[2]	Attr[1]	类型
0	0	缺省序，即强序模型
0	1	PCI-X Relaxed Ordering 模型
1	0	ID-Based Ordering(IDO)模型
1	1	同时支持 Relaxed Ordering 和 IDO 模型

当使用标准的强序模型时，在数据的整个传送路径中，PCIe 设备在处理相同类型的 TLP 时，如 PCIe 设备发送两个存储器写 TLP 时，后面的存储器写 TLP 必须等待前一个存储器写 TLP 完成后才能被处理，即便当前报文在传送过程中被阻塞，后一个报文也必须等待。

如果使用 Relaxed Ordering 模型，后一个存储器写 TLP 可以穿越前一个存储器写 TLP，提前执行，从而提高了 PCIe 总线的利用率。有时一个 PCIe 设备发出的 TLP，其目的地址并不相同，可能先进入发送队列的 TLP，在某种情况下无法发送，但这并不影响后续 TLP 的发送，因为这两个 TLP 的目的地址并不相同，发送条件也并不相同。

值得注意的是，在使用 PCI 总线强序模型时，不同种类的 TLP 间也可以乱序通过同一条 PCIe 链路，比如存储器写 TLP 可以超越存储器读请求 TLP 提前进行。而 PCIe 总线支持 Relaxed Ordering 模型之后，在 TLP 的传递过程中出现乱序种类更多，但是这些乱序仍然是有条件限制的。在 PCIe 总线规范中为了避免死锁，还规定了不同报文的传送数据规则，即 Ordering Rules。

PCIe V2.1 总线规范引入了一种新的“序”模型，即 IDO(ID-Based Ordering)模型，IDO 模型与数据传送的数据流相关，是 PCIe V2.1 规范引入的序模型。

Attr 字段的第 0 位是“No Snoop Attribute”位。当该位为 0 时表示当前 TLP 所传送的数据在通过 FSB 时，需要与 Cache 保持一致，这种一致性由 FSB 通过总线监听自动完成而不需要软件干预；如果为 1，表示 FSB 并不会将 TLP 中的数据与 Cache 进行一致，在这种情况下，进行数据传送时，必须使用软件保证 Cache 的一致性。

在 PCI 总线中没有与这个“No Snoop Attribute”位对应的概念，因此一个 PCI 设备对存储器进行 DMA 操作时会进行 Cache 一致性操作[1]。这种“自动的”Cache 一致性行为在某些特殊情况下并不能带来更高的效率。

当一个 PCIe 设备对存储器进行 DMA 读操作时，如果传送的数据非常大，比如 512MB，Cache 的一致性操作不但不会提高 DMA 写的效率，反而会降低。因为这个 DMA 读访问的数据在绝大多数情况下，并不会在 Cache 中命中，但是 FSB 依然需要使用 Snoop Phase 进行总线监听。而处理器在进行 Cache 一致性操作时仍然需要占用一定的时钟周期，即在 Snoop Phase 中占用的时钟周期，Snoop Phase 是 FSB 总线事务的一个阶段，如图 3-6 所示。

对于这类情况，一个较好的做法是，首先使用软件指令保证 Cache 与主存储器的一致性，并置“No Snoop Attribute”位为 1[2]，然后再进行 DMA 读操作。同理使用这种方法对一段较大的数据区域进行 DMA 写时，也可以提高效率。

除此之外，当 PCIe 设备访问的存储器，不是“可 Cache 空间”时，也可以通过设置“No Snoop Attribute”位，避免 FSB 的 Cache 共享一致性操作，从而提高 FSB 的效率。“No Snoop Attribute”位是 PCIe 总线针对 PCI 总线的不足，所作出的重要改动。

1.1.4 通用 TLP 头中的其他字段

除了 Fmt 和 Type 字段外，通用 TLP 头还含有以下字段。

1)、TH 位、TD 位和 EP 位

TH 位为 1 表示当前 TLP 中含有 TPH(TLP Processing Hint)信息，TPH 是 PCIe V2.1 总线规范引入的一个重要功能。TLP 的发送端可以使用 TPH 信息，通知接收端即将访问数据的特性，以便接收端合理地预读和管理数据。

TD 位表示 TLP 中的 TLP Digest 是否有效，为 1 表示有效，为 0 表示无效。而 EP 位表示当前 TLP 中的数据是否有效，为 1 表示无效，为 0 表示有效。

2)、AT 字段

AT 字段与 PCIe 总线的地址转换相关。在一些 PCIe 设备中设置了 ATC(Address Translation Cache)部件，这个部件的主要功能是进行地址转换。只有在支持 IOMMU 技术的处理器系统中，PCIe 设备才能使用该字段。

AT 字段可以用作存储器域与 PCI 总线域之间的地址转换，但是设置这个字段的主要目的是为了方便多个虚拟主机共享同一个 PCIe 设备。对这个字段有兴趣的读者可以参考 Address Translation Sevices 规范，这个规范是 PCI 的 IO Virtualization 规范的重要组成部分。对虚拟化技术有兴趣的读者可以参考清华大学出版社的《系统虚拟化——原理与实现》，以获得基本的关于虚拟化的入门知识。

3)、Length 字段

Length 字段用来描述 TLP 的有效负载(Data Payload)大小。PCIe 总线规范规定一个 TLP 的 Data Payload 的大小在 1B～4096B 之间。PCIe 总线设置 Length 字段的目的是提高总线的传送效率。

当 PCI 设备在进行数据传送时，其目标设备并不知道实际的数据传送大小，这在一定程度上影响了 PCI 总线的数据传送效率。而在 PCIe 总线中，目标设备可以通过 Length 字段提前获知源设备需要发送或者请求的数据长度，从而合理地管理接收缓冲，并根据实际情况进行 Cache 一致性操作。

当 PCI 设备进行 DMA 写操作，将 PCI 设备中 4KB 大小的数据传送到主存储器时，这个 PCI 设备的 DMA 控制器将存放传送的目的地址和传送大小，然后启动 DMA 写操作，将数据写入到主存储器。由于 PCI 总线是一条共享总线，因此传送 4KB 大小的数据，可能会使用若干个 PCI 总线写事务才能完成，而每一个 PCI 总线写事务都不知道 DMA 控制器何时才能将数据传送完毕。

如果这些总线写事务还通过一系列 PCI 桥才能到达存储器，在这个路径上的每一个 PCI 桥也无法预知，何时这个 DMA 操作才能结束。这种“不可预知”将导致 PCI 总线的带宽不能被充分利用，而且极易造成 PCI 桥数据缓冲的浪费。

而 PCIe 总线通过 TLP 的 Length 字段，可以有效避免 PCIe 链路带宽的浪费。值得注意的是，Length 字段以 DW 为单位，其最小单位为 1 个 DW。如果 PCIe 主设备传送的单位小于 1 个 DW 或者传送的数据并不以 DW 对界时，需要使用字节使能字段，即“DW BE”字段。有关“DW BE”字段。

1.2 TLP 的路由

TLP 的路由是指 TLP 通过 Switch 或者 PCIe 桥片时采用哪条路径，最终到达 EP 或者 RC 的方法。PCIe 总线一共定义了三种路由方法，分别是基于地址(Address)的路由，基于 ID 的路由和隐式路由(Implicit)方式。

存储器和 I/O 读写请求 TLP 使用基于地址的路由方式，这种方式使用 TLP 中的 Address 字段进行路由选径，最终到达目的地。

而配置读写报文、“Vendor_Defined Messages”报文、Cpl 和 CplD 报文使用基于 ID 的路由方式，这种方式使用 PCI 总线号[1](Bus Number)进行路由选径。在 Switch 或者多端口 RC 的虚拟 PCI-PCI 桥配置空间中，包含如何使用 PCI 总线号进行路由选径的信息。

而隐式路由方式主要用于 Message 报文的传递。在 PCIe 总线中定义了一系列消息报文，包括“INTx Interrupt Signaling”，“Power Management Messages”和“Error Signal Messages”等报文。在这些报文中，除了“Vendor_Defined Messages”报文，其他所有消息报文都使用隐式路由方式，隐式路由方式是指从下游端口到上游端口进行数据传递的使用路由方式，或者用于 RC 向 EP 发出广播报文。

1.2.1 基于地址的路由

在 PCIe 总线中，存储器读写和 I/O 读写 TLP 使用基于地址的路由方式。PCIe 设备使用的地址路由方式与 PCI 设备使用的地址路由方式类似。只是 PCIe 设备使用 TLP 进行数据传送，而 PCI 设备使用总线周期进行数据传送。使用地址路由方式进行数据传递的 TLP 格式如下图所示，在这类 TLP 中包含目的设备的地址。

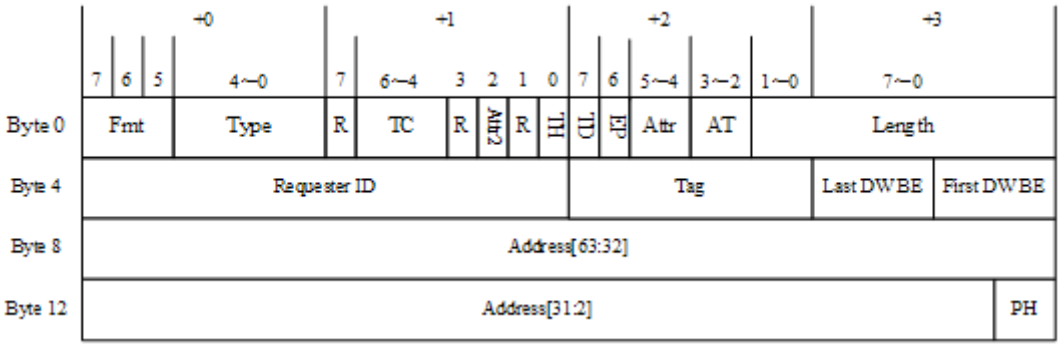

图 1-2-1-1 TLP 读请求

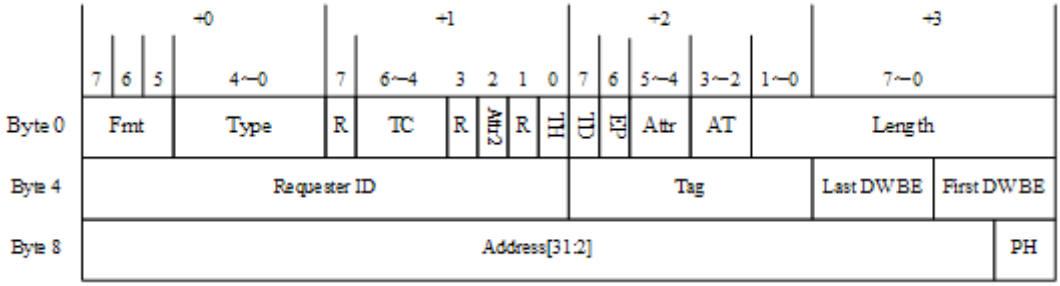

图 1-2-1-2 TLP 写请求

当一个 TLP 进行数据传递时，可能会经过多级 Switch，最终到达目的地。Switch 将根据存储器读写和 I/O 读写请求 TLP 的目的地址将报文传递到合适的 Egress 端口上。如图 8-10 所示，在一个 Switch 中包含了多个虚拟 PCI-to-PCI 桥。在 Switch 中有几个端口，就包含几个虚拟 PCI-to-PCI 桥。（此处的知识点对学习本套开发板 PCIE 通信处理关系不大，读者可以到网上补充这部分知识）

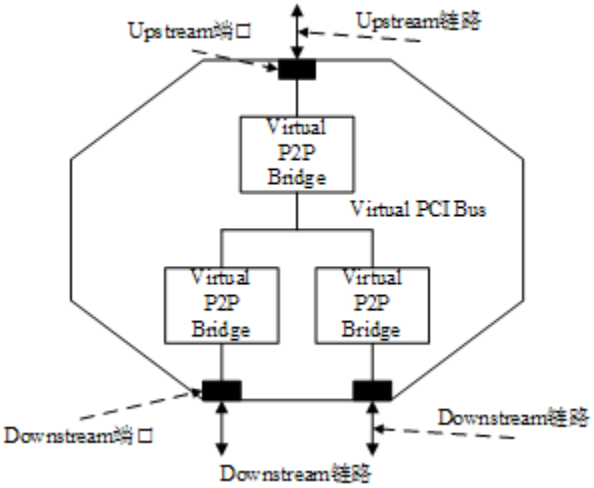

图 1-2-2-1 Switch 的等效逻辑图

在虚拟 PCI-to-PCI 桥的配置寄存器空间中，包含一个桥片能够接收的物理地址范围。PCIe 总线通过这个物理地址范围实现基于地址的路由。这段配置寄存器如图 1-2-1-2 所示。当系统软件初始化 PCI 总线时，将合理地设置这些寄存器，之后当 TLP 通过这些 Switch 时将根据这些寄存器选择合适的路径。

Secondary Status	I/O Limit	I/O Base	1Ch
Memory Limit	Memory Base		20h
Prefetchable Memory Limit	Prefetchable Memory Base		24h
Prefetchable Base Upper 32 Bits			28h
Prefetchable Limit Upper 32 Bits			2Ch
I/O Limit Upper 16 Bits	I/O Base Upper 16 Bits		30h

图 1-2-1-2 与地址路由相关的 PCIE 桥芯片配置寄存器

上图中的配置寄存器描述了该虚拟 PCI-to-PCI 桥下游 PCI 子树使用的三组空间范围，分别为 I/O、存储器和可预取的存储器空间，分别用 Base 和 Limit 两类寄存器描述，其中 Base 寄存器表示可访问空间的基地址，Limit 寄存器表示可访问空间的大小。TLP 使用基于地址的路由时，一定要通过查询这组寄存器之后，再决定传送路径。这组寄存器的使用方法与 PCI 总线中的 PCI 桥兼容。

其中 TLP 从“上游端口发送到下游端口”与“下游端口发送到上游端口”的路由过程略有不同，如下图所示。

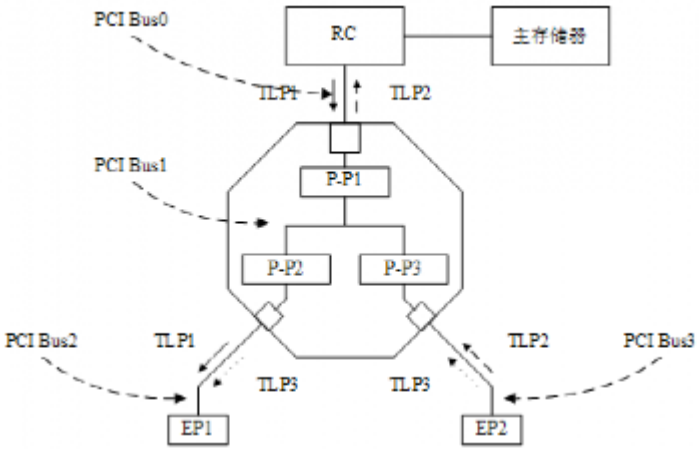

图 1-2-1-3 基于地址的路由寻径方式

下文以 TLP1~3 的发送过程对地址路由过程进行说明。TLP1~3 的描述如下。

TLP1 是一个存储器或者 I/O 请求 TLP，由 RC 发出，并通过一个 Switch 发向 EP1。存储器和 I/O 读写请求 TLP 使用这种地址路由方式。TLP1 将从 Switch 的上游端口传送到下游端口。

TLP2 是一个存储器或者 I/O 请求 TLP，由 EP2 发出，并通过一个 Switch 发向 RC。当 PCIe 设备进行 DMA 读写操作时，将使用这种地址路由方式。TLP2 将从 Switch 的下游端口传送到上游端口。

TLP3 是一个存储器或者 I/O 请求 TLP，由一个 EP2 发出，并通过一个 Switch 后发送到另外一个 EP。在 x86 处理器系统中，这种用法并不常见。但是在某些大规模处理器系统中，具有这种应用方式。此时 TLP3 将从 Switch 的下游端口传送到另外一个下游端口。

1)、TLP1 的传送过程

当 TLP1 从 RC 发向 EP1 时，这个 TLP1 为 I/O 或者存储器报文，其中 TLP1 目的地址在 EP1 的 BAR 空间中。当处理器访问 EP 的 BAR 空间时，需要使用该类 TLP。值得注意的是这个数据报文在通过 RC 时需要进行地址转换。

TLP1 首先通过 PCI Bus0 发向 Switch，并通过 Switch 的 Upstream 端口到达 P-P1 桥片，P-P1 桥片首先根据配置寄存器中的 Limit 和 Base 寄存器决定是否接收 TLP1。如果 Switch 不接收 TLP1，则将该 TLP 作为不支持的请求(Unsupported Request)处理，此时如果 TLP1 需要回应报文，Switch 将发出完成报文，该报文的状态为 UR(Unsupported Request)。

如果 Switch 接收 TLP1，则表示 TLP1 所访问的地址在该 Switch 下游端口所连接的 EP 或者 Switch 中，此时 Switch 将 TLP1 从 PCI Bus0 推至 PCI Bus1 中，即穿越 P-P1 桥片。TLP1 到达 PCI Bus1 后将同时查找 P-P2 和 P-P3 桥片配置寄存器中的 Limit 和 Base 寄存器，决定是 P-P2 还是 P-P3 桥片接收 TLP1。本小节中的例子将使用 P-P2 桥片接收 TLP1，并将 TLP1 推至 PCI Bus2，而 PCI Bus2 上的 EP1 将接收 TLP1，完成整个地址路由。

2)、TLP2 的传送过程

当 TLP2 从 EP2 发向 RC 时，一般来说该 TLP 将访问处理器系统的主存储器。此时 TLP2 首先将请求发至 P-P3 桥片，在 P-P3 桥片配置寄存器的 Limit 和 Base 寄存器中当然不会包含 TLP2 所访问的地址，此时 P-P3 桥片将 TLP2 推至 PCI Bus1。

TLP 从“下游端口向上游端口”与“TLP 从上游端口向下游端口”进行传递时，桥片的处理机制有所不同，从上游端口向下游端口传递时，如果桥片配置寄存器的 Limit 和 Base 寄存器包含该 TLP 的访问地址时，桥片将接收此 TLP，否则不接收该 TLP。而从下游端口向上游端口传递时，如果桥片配置寄存器的 Limit 和 Base 寄存器不包含该 TLP 的访问地址时，桥片将接收该 TLP，并将其推至桥片的上游 PCI 总线。值得注意的是，这两种地址译码方式都属于 PCI 总线的正向译码。

当 TLP2 到达 PCI Bus1 时，首先检查在 PCI Bus1 总线上的 P-P2 桥片是否可以接收此 TLP，如果不能接收则 TLP2 通过 P-P1 桥片传递到 PCI Bus0，即到达 RC。

在 MPC8548 处理器中，到达 RC 的 TLP 首先通过 Inbound 寄存器进行地址转换，将 TLP 的 PCI 总线地址转换为处理器的地址，然后访问处理器中相应的存储器空间；对于 x86 处理器而言，MCH 也会完成 PCI 域地址空间到存储器域地址空间的转换，然后访问处理器中相应的存储器空间。

3)、TLP3 的传送过程

TLP3 的传递方式与 TLP2 的传递方式有些类似，当 TLP3 传递到 PCI Bus1 时，P-P2 桥片将接收 TLP3，并将 TLP3 传递到 PCI Bus2 上的 EP1 中。由以上叙述可以发现，PCIe 总线中基于地址的路由方式与 PCI 总线上的基于地址的数据传递流程十分相近。TLP3 在 PCI 总线域上进行数据传递，因此不需要进行 PCI 总线域到存储器域的地址转换。

1.2.2 基于 ID 的路由

在 PCIe 总线中，基于 ID 的路由方式主要用于配置读写请求 TLP、Cpl 和 CplD 报文，此外 Vendor_Defined 消息报文也可以使用这种基于 ID 的路由方式。而在 PCI 总线中，只有配置读写周期才使用 ID 进行数据传递。

基于 ID 的路由方式与基于地址的路由方式有较大的不同，基于 ID 路由方式的 TLP 头格式也与基于地址路由方式的头格式不同，其报文格式如图 8-6 所示。

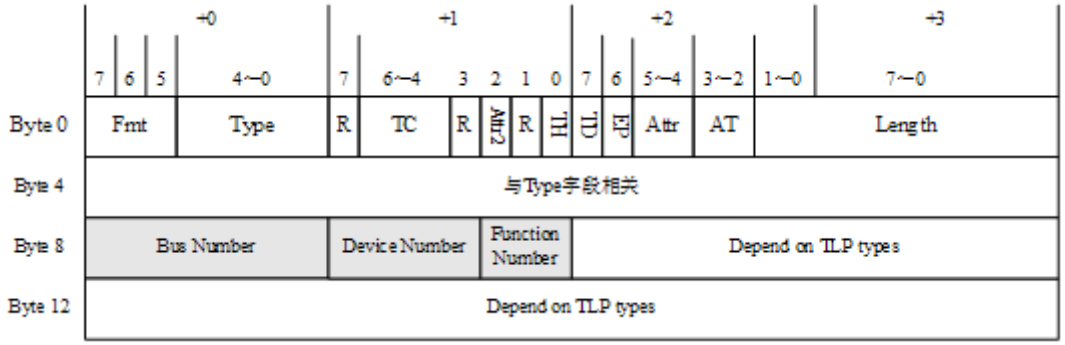

图 1-2-2-1 使用 ID 路由的 TLP 头格式

如上图所示，使用 ID 路由方式的 TLP 头，其 Byte8～11 字段与基于地址路由的 TLP 不同。基于 ID 路由的 TLP，使用 Bus Number、Device Number 和 Function Number 进行路由

寻址。从软件的角度上看，PCIe 总线与 PCI 总线兼容，只是在 PCIe 总线中，每一个 PCIe 设备使用唯一的 PCI 设备号，但是每一个设备仍然可以有多个子设备(Function)。

PCIe 总线规定，在一个 PCI 总线域空间中，最多只能有 256 条 PCI 总线，因此在一个 TLP 中，Bus Number 由五位组成；而在每一条总线中最多包含 32 个设备，因此 TLP 中的 Device Number 由 8 位组成（PCIe 链路采用端到端的通信方式，每一个链路只能挂接一个设备，因此在多数情况下，使用 3 位描述 Device Number 是多余的，因此 PCIe 总线提出了 ARI 格式）；而每一个设备中最多包含 8 个功能，因此一个 TLP 的 Function Number 由 3 位组成。

配置读写请求 TLP 是使用“基于 ID 路由”的一组重要报文，其主要作用是读写 PCIe 总线的 EP、Switch 及 PCIe 桥片的配置寄存器，以完成 PCIe 总线的配置。在处理器系统上电之后需要进行 PCI 总线系统的枚举，为 PCI 总线分配总线号，并设置 Switch、PCIe 桥片或者 EP 的配置寄存器，如 Limit 寄存器组、Base 寄存器组、BAR 寄存器、Subordinate Bus Number、Secondary Bus Number 和 Primary Bus Number 等一系列配置寄存器。

在上文中我们简单介绍了 Limit 寄存器组和 Base 寄存器组的用法，下文将重点描述 Subordinate Bus Number、Secondary Bus Number 和 Primary Bus Number 寄存器。Subordinate Bus Number、Secondary Bus Number 和 Primary Bus Number 寄存器在 Type 01h 配置寄存器中，用来描述 PCI-to-PCI 桥片的上游及下游总线号。这段寄存器在 PCI 配置寄存器中的位置如图下图所示。

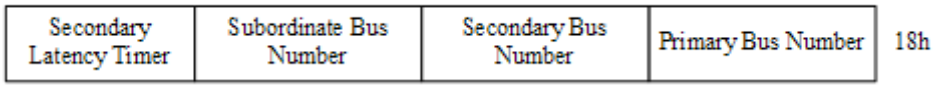

图 1-2-2-2 与 ID 路由相关的 PCIE 桥芯片配置寄存器

与 PCI 总线中的桥片类似，Primary Bus Number 记录 PCI-to-PCI 桥上游的 PCI 总线号，Secondary Bus Number 记录 PCI-to-PCI 桥下游的第一个 PCI 总线号，而 Subordinate Bus Number 记录 PCI-to-PCI 桥下游的最后一个 PCI 总线号。

如图 1-2-1-3 所示 P-P1 桥片的 Primary Bus Number 为 0，Secondary Bus Number 为 1，而 Subordinate Bus Number 为 3。这些总线号，在处理器系统对 PCI 总线进行枚举时由系统初始化程序设置，从系统初始化程序的角度上看，PCIe 总线与 PCI 总线基本兼容，只是 PCIe 总线对配置空间进行了一些扩展。

表 1-1-1-1Type[4:0]字段（此表在上节内容出现过，放在这里方便阅读）

TLP 类型	Fmt[2:0]	Type[4:0]	描述

TLP 类型	Fmt[2:0]	Type[4:0]	描述
MRd	0b000 0b001	0b0 0000	存储器读请求；TLP 头大小为 3 个或者 4 个双字，不带数据。
MRdLk	0b000 0b001	0b0 0001	带锁的存储器读请求；TLP 头大小为 3 个或者 4 个双字，不带数据。
MWr	0b010 0b011	0b0 0000	存储器写请求；TLP 头大小为 3 个或者 4 个双字，带数据。
IORd	0b000	0b0 0010	IO 读请求；TLP 头大小为 3 个双字，不带数据。
IOWr	0b010	0b0 0010	IO 写请求；TLP 头大小为 3 个双字，带数据。
CfgRd0	0b000	0b0 0100	配置 0 读请求；TLP 头大小为 3 个双字，不带数据。
CfgWr0	0b010	0b0 0100	配置 0 写请求；TLP 头大小为 3 个双字，带数据。
CfgRd1	0b000	0b0 0101	配置 1 读请求；不带数据。
CfgWr1	0b010	0b0 0101	配置 1 写请求；带数据。
TCfgRd	0b010	0b1 1011	本书对这两种总线事务不做介绍。
TCfgWr	0b001	0b1 1011	
Msg	0b001	0b1 0r2r1r0	消息请求；TLP 头大小为 4 个双字，不带数据。“rrr”字段是消息请求报文的 Route 字段，下文将详细介绍该字段。
MsgD	0b011	0b1 0r2r1r0	消息请求；TLP 头大小为 4 个双字，带数据。
Cpl	0b000	0b0 1010	完成报文；TLP 头大小为 3 个双字，不带数据。包括存储器、配置和 I/O 写完成。
CplD	0b010	0b0 1010	带数据的完成报文，TLP 头大小为 3 个双字，包括存储器读、I/O 读、配置读和原子操作读完成。

TLP 类型	Fmt[2:0]	Type[4:0]	描述
CplLk	0b000	0b0 1011	锁定的完成报文，TLP 头大小为 3 个双字，不带数据。
CplDLk	0b010	0b0 1011	带数据的锁定完成报文，TLP 头大小为 3 个双字，带数据。
FetchAdd	0b010 0b011	0b0 1100	Fetch and Add 原子操作。
Swap	0b010 0b011	0b0 1101	Swap 原子操作。
CAS	0b010 0b011	0b0 1110	CAS 原子操作。
LPrfx	0b100	0b0 L3L2L1L0	Local TLP Prefix
EPrfx	0b100	0b1 E3E2E1E0	End-End TLP Prefix

如表表 1-1-1-1 所示，RC 可以使用 Type 00h 和 Type 01h 读写请求 TLP，对 PCIe 设备的配置寄存器进行读写访问，配置读写请求 TLP 只能由 RC 发出，配置读写请求 TLP 使用基于 ID 的路由方式。

如图 1-2-1-3 所示的例子，RC 首先使用 Type 00h 配置请求 TLP 访问在 PCI Bus0 总线上的设备，PCI Bus0 上的所有设备，包括桥片都要监听 PCI Bus 0 上的配置请求，在本例中只有 Switch 挂接在 PCI Bus0 上，实际上是 Switch 的上游端口与 PCI Bus0 直接相连。因此 Switch 的上游端口将接收 RC 发出的 Type 00h 配置请求 TLP，之后 Switch 将向 RC 发出完成报文，结束配置请求。与 PCI 总线相同，PCIe 总线的 Type 00h 类型配置请求 TLP 不能够穿越桥片，在图 1-2-1-3 中这类请求只能访问 Switch 上游端口的配置空间。

PCI 总线是基于共享总线的数据传送方式，在一条 PCI 总线上可以连接多个 PCI Agent 设备，其中每一个 PCI Agent 都提供了一个 IDSEL#信号，这个信号与 PCI-to-PCI 桥片或者 HOST 主桥的地址线直接相连，PCI 总线根据与 IDSEL#信号与地址线的连接关系决定相应设备的 Device Number。

这与 PCIe 总线的使用方法不同，PCIe 总线使用“端对端”的连接方式，在 PCIe 链路只能连接一个下游设备，而这个下游设备的 Device Number 只能为 0。而只有在 Switch 的虚拟 PCI 总线上可以连接多个 Device Number 不同的端口。

当一个虚拟 PCI 总线上挂接 PCI-to-PCI 桥时，系统配置软件将使用 Type 01h 配置请求 TLP 访问 PCI-PCI 桥下游的 PCI 设备。如图 5-5 所示，RC 可以通过 Type 01h 配置请求 TLP 访问 P-P2 桥片、P-P3 桥片，EP1 和 EP2。

当 RC 使用 Type 01h 配置请求 TLP，直接访问 P-P1 桥的下游设备时，首先需要检查该 TLP 的 Bus Number 是否为 1，如果为 1 表示该 TLP 的访问目标在 PCI Bus 1 总线上，此时 PCI-to-PCI 桥将这个 Type 01h 类型的 TLP 转换为 Type 00h 类型的 TLP，然后推至 PCI Bus 1 总线，并访问其下的设备。

如果该 TLP 的 Bus Number 在 P-P1 桥片的 Secondary Bus Number 和 Subordinate Bus Number 寄存器之间，则 P-P1 桥片将该 Type 01h 类型的 TLP 直接转发到 PCI Bus 1 上，并不改变该 TLP 的类型，之后 Type 01 和类型的 TLP 将继续检查 P-P2 和 P-P3 桥片的配置空间，决定由 P-P2 还是 P-P3 接收该 TLP。如果 TLP 的 PCI Bus Number 为 2 时，P-P2 桥片将接收该 TLP，并将该 Type 01h 类型 TLP 转换为 Type 00h 类型的 TLP，然后发送给 EP1，并由 EP1 处理该 TLP。

上文简要讲述了配置请求 TLP 使用 ID 路由方式从上游端口向下游端口的传递规则，但是 Vendor_Defined 消息报文和 Cpl 和 CplD 报文还可能从下游端口向上游端口进行传递。此时 PCIe 总线处理方法略有不同。下文仍以图 5-5 为例说明这种情况。

当一个 TLP 从 EP2 传送到 EP1 或者 RC 时，首先检查 P-P3 桥片的配置空间，P-P3 桥片发现该 TLP 不是发向自己时，将该 TLP 推至上游总线，即 PCI Bus 1。如果 PCI Bus1 上 P-P1 桥片没有认领该 TLP，该 TLP 将继续向 P-P2 桥片传递，并由这个桥片将 TLP 转发给合适的 EP；如果 P-P1 桥片认领该 TLP，该 TLP 将继续向上游总线传递，直至 RC。

由以上描述可以发现，PCIe 总线使用的基于 ID 的路由方式与 PCI 总线中配置读写总线事务通过 PCI 桥的方法较为类似。

1.2.3 隐式路由

PCIe 总线规定消息请求报文使用隐式路由方式。在 PCIe 总线中，有许多消息是直接发向 RC 或者来自 RC 的广播报文，这些报文不使用地址或者 ID 进行路由，而是使用 Msg 和 MsgD 报文的 Route 字段进行路由，这种路由方式被称为隐式路由。

PCIe 总线定义了一些用于中断请求、错误状态处理、锁定总线事务、热插拔信号处理和“Vendor_Defined Messages”消息报文。这些消息报文需要使用隐式路由方式进行传递。消息报文的 Route 字段的含义如下表所示。

表 8-2-3-1Route[4:0]字段

Route[2:0]	描述
000	路由到 RC
001	使用地址路由
010	使用 ID 路由
011	来自 RC 的广播报文
100	本地消息，在接收端结束(Legacy 中断消息使用这种报文格式，传递来自 PCI 总线的中断报文)
101	用于 PCIe 电源管理(PME_TO_Ack 报文使用)
110~111	保留

由上表所示，使用隐式路由方式的 TLP，其 Route 字段为“000”，“011”，“100”或者“101”。当一个报文使用隐式路由向 EP 发送时，EP 将对 Route 字段进行检查，如果这个报文是“来自 RC 的广播报文”，或者是“本地报文”，EP 将接收此报文。

如果 Switch 收到一条使用隐式路由的 TLP 时，将根据报文 Route 字段的不同而分别处理。如果 Switch 的上游端口接收了一条来自 RC 的广播消息，则将该报文发向所有的下游端口；如果 Switch 接收了一条来自下游端口发向 RC 的消息报文时，Switch 将此报文直接转发到上游端口，直至 RC；如果 Switch 接收了一条使用隐式路由方式的本地消息报文，则 Switch 接收并终结此报文，不再上传或下推。

如果 RC 收到一个使用隐式路由的 TLP 时，将根据报文 Route 字段而分别处理这些 TLP。如果该 Route 字段为 0b000 和 0b101，RC 将接收该 TLP，并作相应的处理；如果为 0b100，RC 将接收该 TLP，并结束该 TLP 报文的传递。

1.3 存储器、I/O 和配置读写请求 TLP

本节讲述 PCIe 总线定义的各类 TLP，并详细介绍这些 TLP 的格式。在这些 TLP 中，有些格式对于初学者来说较难理解。读者需要建立 PCIe 总线中与 TLP 相关的一些基本概念，特别是存储器读写相关的报文格式。在 PCIe 总线中，存储器读写，I/O 读写和配置读写请求 TLP 由以下几类报文组成。

(1)、存储器读请求 TLP 和读完成 TLP

当 PCIe 主设备，RC 或者 EP，访问目标设备的存储器空间时，使用 Non-Posted 总线事务向目标设备发出存储器读请求 TLP，目标设备收到这个存储器读请求 TLP 后，使用存储器读完成 TLP，主动向主设备传递数据。当主设备收到目标设备的存储器读完成 TLP 后，将完成一次存储器读操作。

(2)、存储器写请求 TLP

在 PCIe 总线中，存储器写使用 Posted 总线事务。PCIe 主设备仅使用存储器写请求 TLP 即可完成存储器写操作，主设备不需要目标设备的回应报文。

(3)、原子操作请求和完成报文

原子操作由 PCIe V2.1 总线规范引入，一个完整的原子操作包括原子操作请求和原子操作完成报文组成。原子操作的使用方法与其他 Non-Posted 总线事务类似，首先 PCIe 主设备向目标设备发送原子操作请求，之后目标设备向主设备发送原子操作完成报文，结束一次原子操作。有关原子操作的详细说明见第 1.3.5 节。

(4)、I/O 读写请求 TLP 和读写完成 TLP

在 PCIe 总线中，I/O 读写操作使用 Non-Posted 总线事务，I/O 读写 TLP 都需要完成报文做为回应。只是在 I/O 写请求的完成报文中不需要“带数据”，而仅含有 I/O 写请求是否成功的状态信息。

(5)、配置读写请求 TLP 和配置读写完成 TLP

从总线事务的角度上看，配置读写请求操作的过程与 I/O 读写操作的过程类似。配置读写请求 TLP 都需要配置读写完成作为应答，从而完成一个完成的配置读写操作。

(6)、息报文

与 PCI 总线相比，PCIe 总线增加了消息请求事务。PCIe 总线使用基于报文的数据传送模式，所有总线事务都是通过报文实现的，PCIe 总线取消了一些在 PCI 总线中存在的边带信号。在 PCIe 总线中，一些由 PCI 总线的边带信号完成的工作，如中断请求和电源管理等，在 PCIe 总线中由消息请求报文实现。

1.3.1 存储器读写请求 TLP

存储器读写请求 TLP 的格式如下图所示。

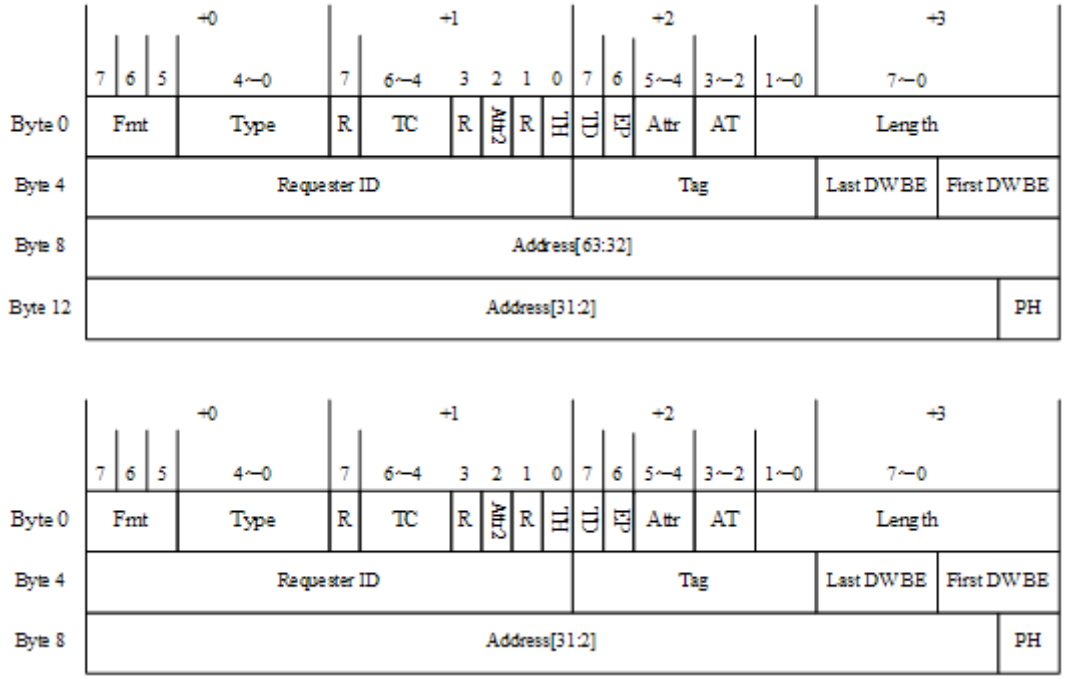

图 1-3-1-1 存储器和 IO 读写请求 TLP 头格式

在 PCIe 总线中，存储器写请求 TLP 使用 Posted 数据传送方式。而其他与存储器和 I/O 相关的报文都使用 Split 方式进行数据传送，这些请求报文需要完成报文，通知发送端之前的数据请求报文已经被处理完毕，有关完成报文的详细说明见第 1.3.2 节。

存储器读写请求 TLP 使用地址路由方式进行数据传递，在这类 TLP 头中包含 Address 字段，Address 字段具有两种地址格式，分别是 32 位和 64 位地址。在存储器读写和 I/O 读写请求的第 3 和第 4 个双字中，存放 TLP 的 32 或者 64 位地址。存储器、I/O 和原子操作读写请求使用的 TLP 头较为类似。本节仅介绍存储器、I/O 读写使用的 TLP 头，而在第 1.3.5 节详细介绍原子操作。

1)、Length 字段

在存储器读请求 TLP 中并不包含 Data Payload，在该报文中，Length 字段表示需要从目标设备数据区域读取的数据长度；而在存储器写 TLP 中，Length 字段表示当前报文的 Data Payload 长度。

Length 字段的最小单位为 DW。当该字段为 n 时，表示需要获得的数据长度或者当前报文的数据长度为 n 个 DW，其中 0≤n≤0x3FF。值得注意的是，当 n 等于 0 时，表示数据长度为 1024 个 DW。

2)、DW BE 字段

PCIe 总线以字节为基本单位进行数据传递，但是 Length 字段以 DW 为最小单位。为此 TLP 使用 Last DW BE 和 First DW BE 这两个字段进行字节使能，使得在一个 TLP 中，有效数据以字节为单位。

这两个 DW BE 字段各由 4 位组成，其中 Last DW BE 字段的每一位对应数据 Payload 最后一个双字的字节使能位；而 First DW BE 字段的每一位对应数据 Payload 第一个双字的字节使能位。其对应关系如表 1-3-1-所示。

表 1-3-1-1First 和 Last DW BE 字段

Last DW BE	第 3 位	为 1 表示数据 Payload 的最后一个双字的字节 3 有效
	第 2 位	为 1 表示数据 Payload 的最后一个双字的字节 2 有效
	第 1 位	为 1 表示数据 Payload 的最后一个双字的字节 1 有效
	第 0 位	为 1 表示数据 Payload 的最后一个双字的字节 0 有效
First DW BE	第 3 位	为 1 表示数据 Payload 的第一个双字的字节 3 有效
	第 2 位	为 1 表示数据 Payload 的第一个双字的字节 2 有效
	第 1 位	为 1 表示数据 Payload 的第一个双字的字节 1 有效

	第 0 位	为 1 表示数据 Payload 的第一个双字的字节 0 有效

Last DW BE 和 First DW BE 这两个字段的使用规则如下。

如果传送的数据长度在一个对界的双字(DW)之内，则 Last DW BE 字段为 0b0000，而 First DW BE 的对应位置 1；如果数据长度超过 1DW，Last DW BE 字段一定不能为 0b0000。PCIe 总线使用 Last DW BE 字段为 0b0000 表示所传送的数据在一个对界的 DW 之内。

如果传送的数据长度超过 1DW，则 First DW BE 字段至少有一个位使能。不能出现 First DW BE 为 0b0000 的情况。

如果传送的数据长度大于等于 3DW，则在 First DW BE 和 Last DW BE 字段中不能出现不连续的置 1 位。

如果传送的数据长度在 1DW 之内时，在 First DW BE 字段中允许有不连续的置 1 位。此时 PCIe 总线允许在 TLP 中传送 1 个 DW 的第 1，3 字节或者第 0，2 字节。

如果传送的数据长度为 2DW 之内时，则 First DW BE 字段和 Last DW BE 字段允许有不连续的置 1 位。

值得注意的是，PCIe 总线支持一种特殊的读操作，即“Zero-Length”读请求。此时 Length 字段的长度为 1DW，而 First DW BE 字段和 Last DW BE 字段都为 0b0000，即所有字节都不使能。此时与这个存储器读请求 TLP 对应的读完成 TLP 中不包含有效数据。再次提醒读者注意“Zero-Length”读请求使用的 Length 字段为 1，而不是为 0，为 0 表示需要获得的数据长度为 1024 个 DW。

“Zero-Length”读请求的引入是为了实现“读刷新”操作，该操作的主要目的是为了确保之前使用 Posted 方式所传送的数据，到达最终的目的地，与“Zero-Length”读对应的读完成报文中不含有负载，从而提高了 PCIe 链路的利用率。

在 PCIe 总线中，使用 Posted 方式进行存储器写时，目标设备不需要向主设备发送回应报文，因此主设备并不知道这个存储器写是否已经达到目的地。而主设备可以使用“读刷新”操作，向目标设备进行读操作来保证存储器写最终到达目的地。

在 PCIe 总线中，标准的存储器读请求也可以完成同样的刷新操作。但是“Zero-Length”读请求与这种读请求相比，其完成报文不需要“Data Payload”，因此在一定程度上提高了 PCIe 总线的效率。如果一个存储器读请求 TLP 报文的 TH 位为 1 时，DW BE 字段将被重新定义为 ST[7:0]字段，有关 ST 字段的详细说明见第 1.3.6 节。

3)、Requester ID 字段

Requester ID 字段包含“生成这个 TLP 报文”的 PCIe 设备的总线号(Bus Number)、设备号(Device Number)和功能号(Function Number)，其格式如图 8-3-1-1 所示。对于存储器写请求 TLP，Requester ID 字段并不是必须的，因为目标设备收到存储器写请求 TLP 后，不需要完成报文作为应答，因此 Requester ID 字段对于存储器写请求 TLP 并没有实际意义。

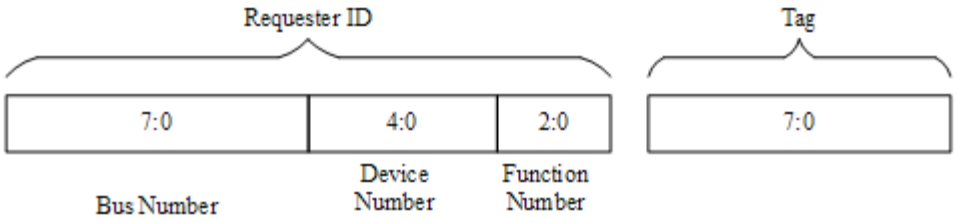

图 1-3-1-1Transaction ID 格式

但是 PCIe 总线规范并没有明确说明存储器写请求 TLP 究竟需不需要 Requester ID 字段，为此 IC 设计者依然需要将存储器写 TLP 的 Requester ID 字段置为有效。值得注意的是，如果一个存储器写请求 TLP 报文的 TH 位为 1 时，Tag 字段将被重新定义为 ST[7:0]字段，有关 ST 字段的详细说明见第 1.3.6 节。

对于 Non-Posted 数据请求，目标设备需要使用完成报文做为回应。在这个完成报文中，需要使用源设备的 Requester ID 字段。因此在 Non-Posted 数据请求 TLP 中，如存储器读请求、I/O 和配置读写请求 TLP，必须使用 Requester ID 字段。

存储器，I/O 读请求 TLP 中含有 Requester ID 和 Tag 字段。在 PCIe 总线中 Requester ID 和 Tag 字段合称为 Transaction ID，Transaction ID 字段的格式如图 8-9 所示。存储器读，I/O 和配置读写请求 TLP 使用 Transaction 字段的主要目的是使接收端通过分析报文的 Transaction ID，确认完成报文的目的地。

在 PCIe 总线中，所有 Non-Posted 数据请求都需要完成报文作为应答，才能结束一次完整的数据传递。一个源设备在发送 Non-Posted 数据请求之后，如果并没有收到目标设备回送的完成报文，TLP 报文的发送端需要保存这个 Non-Posted 数据请求，此时该设备使用的 Transaction ID(Tag 字段)不能被再次使用，直到一次数据传送结束，即数据发送端收齐与该 TLP 对应的所有完成报文。

PCIe 设备发出的每一个 Non-Posted 数据请求 TLP，在同一个时刻段内 Transaction ID 必须是唯一的。即在同一时间段内，在当前 PCI 总线域中不能存在两个或者两个以上的存储器读请求 TLP，其 Transaction ID 完全相同。

源设备发送 Non-Posted 数据请求后，在没有获得全部完成报文之前，不能释放这个 Transaction ID 占用的资源。在同一个 PCIe 设备发送的 TLP 中，其 Requester ID 字段是相同的，因此 PCIe 设备的设计者所能管理的资源是 Tag 字段。PCIe 设备的设计者需要合理地管理 Tag 资源，以保证数据传送的正确性。

PCIe 设备在发送 Non-Posted 数据请求时，需要暂存这些 Non-Posted 数据请求。其中 Tag 字段的长度决定了发送端能够暂存多少个同类型的 TLP，如果 Tag 字段长度为 5，发送端能够暂存 32 个报文；如果 PCIe 设备使能了 Extended Tag 位，Tag 字段可以由 8 位组成，此时发送端能够暂存 256 个报文。

通过 Tag 字段的长度，可以发现每个 PCIe 设备最多可以暂存 256 个同类型的 Non-Posted 报文。但是在多数情况下，一个 PCIe 设备可能只包含 1 个 Function。因此 PCIe 设备还可以使用 Function 号扩展 Tag 字段，从而扩展“暂存 TLP 报文”的数目。

PCIe 设备在 PCI Express Capability 结构的 Device Control 寄存器中，设置了一个 Phantom Functions Enable 位，。当一个 PCIe 设备仅支持一个 Function 时，Phantom Functions Enable 位可以被设置为 1，此时 PCIe 设备可以使用 Requester ID 的 Function Number 字段对 Tag 字段进一步扩展，此时一个 PCIe 设备最多可以支持 2048 个同类型的数据请求。

由以上分析可以发现，一个 PCIe 设备最多可以支持 2048 个存储器读数据请求，基本上可以满足绝大多数需要。但是在某些特殊应用场合，PCIe 设备即使可以暂存 2048 个存储器读请求，也并不足够。

与 PCI 总线相比，PCIe 总线的数据传送延时较长，而为了弥补这个传送延时，PCIe 设备通常使用流水线技术。此时 PCIe 设备必须能够连续发送多个存储器读请求报文，随后 RC 也将连续回送多个存储器读完成报文，在 PCIe 设备的实现中，需要保证能够源源不断地从 RC 接收这些报文，以充分利用报文接收流水线。

PCIe V2.1 总线规范还提出了另一种 Requester ID 格式，即 ARI (Alternative Routing-ID Interpretation)格式，除了 Requester ID 外，在完成报文中使用的 Completer ID 也可以使用这种格式。ARI 格式将 ID 号分为两个字段，分别为 Bus 号和 Function 号，而不使用 Device 号，ARI 格式如下图所示。

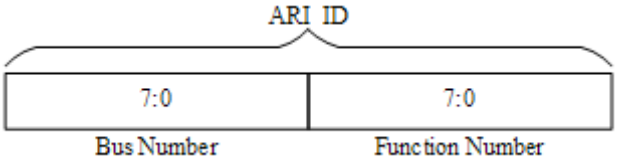

图 1-3-1-2 ARI 格式

PCIe 总线引入 ARI 格式的依据是在一个 PCIe 链路上仅可能存在一个 PCIe 设备，因而其 Device 号一定为 0。在多数 PCIe 设备中，Requester ID 和 Completion ID 包含的 Device 号是没有意义的。使用 ARI 格式时，一个 PCIe 设备最多可以支持 256 个 Function，而传统的 PCIe 设备最多只能支持 8 个 Function。

4）、I/O 读写请求 TLP 的规则

I/O 读写请求与存储器读写请求 TLP 的格式基本类似，只是 I/O 读写请求 TLP 只能使用 32 位地址模式和基于地址的路由方式，而且 I/O 读写请求 TLP 只能使用 Non-Posted 方式进行传递。PCIe 总线并不建议 PCIe 设备支持 I/O 地址空间，但是 Switch 和 RC 需要具备接收和发送 I/O 请求报文的能力，因为许多老的 PCI 设备依然使用 I/O 地址空间，这些 PCI 设备可以通过 PCIe 桥连接到 PCIe 总线中。因此虽然支持 I/O 读写请求的 PCIe 设备极少，但是在 PCIe 体系结构中，依然需要支持 PCI 总线域的 I/O 地址空间。

与存储器读写请求 TLP 不同，I/O 读写请求 TLP 的某些字段必须为以下值。

TC[2:0]必须为 0，I/O 请求报文使用的 TC 标签只能为 0。

TH 和 Attr2 位保留，而 Attr[1:0]必须为“0b00”，这表示 I/O 请求报文必须使用 PCI 总线的强序数据传送模式，而且在传送过程中，硬件保证其传送的数据与 Cache 保持一致，实际上 I/O 地址空间都是不可 Cache 的。

AT[1:0]必须为“0b00”，表示不支持地址转换，因此在虚拟化技术中，并不处理 PCI 总线域中的 I/O 空间。

Length[9:0]为“0b00 0000 0001”，表示 I/O 读写请求 TLP 最大的数据 Payload 为 1DW，该类 TLP 不支持突发传送。

Last DW[3:0]为“0b0000”。

1.3.2 完成报文

PCIe 总线支持 Split 传送方式，目标设备使用完成报文向源设备主动发送数据。完成报文使用 ID 路由方式，由 TLP Predix、报文头和 Data Payload 组成，但是在某些完成报文可以不含有 Data Payload，如 I/O 或者配置写完成和 Zero-Length 读完成报文。在 PCIe 总线中，有一下几类数据请求需要收到完成报文之后，才能完成整个数据传送过程，完成报文格式如图 5-11 所示。

所有的数据读请求，包括存储器、I/O 读请求、配置读请求和原子操作请求。当 一个 PCIe 设备发出这些数据请求报文后，必须收到目标设备的完成报文后，才能结束一次数据传送。这一类完成报文必须包含 Data Payload。

所有的 Non-Posted 数据写请求，包括 I/O 和配置写请求。当一个 PCIe 设备发出这些数据请求报文后，必须收到目标设备的完成报文后，才能结束数据传送。但是这一类完成报文不包含数据，仅包含应答信息。

与 ATS 机制相关的一些报文。

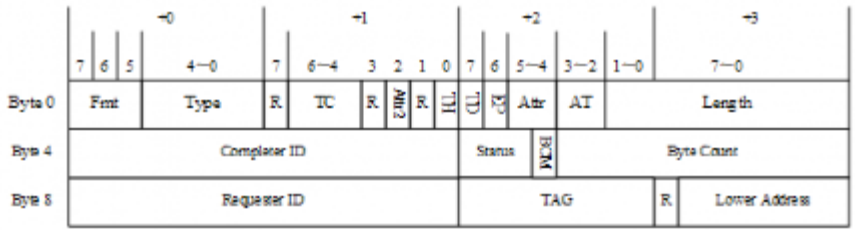

图 1-3-2-1 完成报文头格式

完成报文“Byte 0”中的大部分字段与“存储器，I/O、配置请求报文”的对应字段的含义相同。完成报文一次最多能够传送的报文大小不能超过 Max_Payload_Size 参数。在多数处理器中，完成报文中包含的数据在一个 Cache 行之内，完成报文使用 RCB 参数来处理数据对界，RCB 参数的大小与处理器系统的 Cache 行长度和 DDR-SDRAM 的一次突发传送长度相关在 x86 和 PowerPC 处理器中，一个存储器读完成报文一般不超过 RCB 参数。

1)、Requester ID 和 Tag 字段

完成报文使用 ID 路由方式，ID 路由方式详见第 1.2.2 节。完成报文头的长度为 3DW，完成报文头中包含 Transaction ID 信息，由 Requester ID 和 Tag 字段组成，这个 ID 必须与

源设备发送的数据请求报文的 Transaction ID 对应，完成报文使用 Transaction ID 进行 ID 路由，并将数据发送给源设备。

当 PCIe 设备收到存储器读、I/O 读写或者配置读写请求 TLP 时，需要首先保存这些报文的 Transaction ID，之后当该设备准备好完成报文后，将完成报文的 Requester ID 和 Tag 字段赋值为之前保存的 Transaction ID 字段。

2)、Completer ID 字段

Completer ID 字段的含义与 Requester ID 字段较为相似，只是该字段存放“发送完成报文”的 PCIe 设备的 ID 号。PCIe 设备进行数据请求时需要在 TLP 字段中包含 Requester ID 字段；而使用完成报文结束数据请求时，需要提供 Completer ID 字段。

3)、Status 字段

Status 字段保存当前完成报文的完成状态，表示当前 TLP 是正确地将数据传递给数据请求端；还是在数据传递过程中出现错误；或者要求数据请求方进行重试。PCIe 总线规定了几类完成状态，如表 5-6 所示。

表 1-3-2-1Status 字段

Status[2:0]	描述
0b000	SC(Sucessful Completion)，正常结束
0b001	UR(Unsupported Request)，不支持的数据请求
0b010	CRS(Configuration Request Retry Status)，要求数据请求方进行重试。当 RC 对一个 PCIe 目标设备发起配置请求时，如果该目标设备没有准备好，可以向 RC 发出 CRS 完成报文，当 RC 收到这类报文时，不能结束本次配置请求，必须择时重新发送配置请求
0b100	CA(Completion Abort)，数据夭折。表示目标设备无法完成本次数据请求
其他	保留

4)、BCM 位与 Byte Count 字段

BCM(Byte Count Modified)字段由 PCI-X 设备设置。PCI-X 设备也支持 Split Transaction 传送方式，当 PCI-X 设备进行存储器读请求时，目标设备不一定一次就能将所有数据传递给源设备。此时目标设备在进行第一次数据传送时，需要设置 Byte Count 字段和 BCM 位。

BCM 位表示 Byte Count 字段是否被更改，该位仅对 PCI-X 设备有效，而 PCIe 设备不能操纵 BCM 位，只有 PCI-X 设备或者 PCIe-to-PCI-X 桥可以改变该位。本节对此位不做进一步介绍，对此位感兴趣的读者可以参考 PCI-X Addendum to the PCI Local Bus Specification, Revision 2.0。

Byte Count 字段记录源设备还需要从目标设备中，获得多少字节的数据就能完成全部数据传递，当前 TLP 中的有效负载也被 Byte Count 字段统计在内。该字段由 12 位组成。该字段为 0b0000-0000-0001 表示还剩一个字节，为 0b1111-1111-1111 表示还剩 4095 个字节，而为 0b0000-0000-0000 表示还剩 4096 个字节。除了存储器读请求的完成报文外，大多数完成报文的 Byte Count 字段为 4。

如一个源设备向目标设备发送一个“读取 128B 的存储器读请求 TLP”，而目标设备收到这个读请求 TLP 后，可能使用两个存储器读完成 TLP 传递数据。其中第 1 个存储器读完成 TLP 的有效数据为 64B，而 Byte Count 字段为 128；第 2 个存储器读完成 TLP 中的有效数据为 64B，而 Byte Count 字段也为 64。当数据请求端接收完毕第 1 个存储器读完成 TLP 后，发现还有 64B 的数据没有接收完毕，此时必须等待下一个存储器读完成 TLP。等到数据请求端收齐所有数据后，才能结束整个存储器读请求。

目标设备发出的第 2 个读完成 TLP 中的有效数据为 64B，而 Byte Count 字段为 64，当数据请求端接收完毕这个读完成 TLP 后，将完成一个完整的存储器读过程，从而可以释放这个存储器读过程使用的 Tag 资源。存储器读请求的完成报文的拆分方式较为复杂，Byte Count 字段的设置也相对较为复杂。

5)、Lower Address 字段

如果当前完成报文为存储器读完成 TLP，该字段存放在存储器读完成 TLP 中第一个数据所对应地址的最低位。值得注意的是，在完成报文中，并不存在 First DW BE 和 Last DW BE 字段，因此接收端必须使用存储器读完成 TLP 的 Low Address 字段，识别一个 TLP 中包含数据的起始地址。

1.3.3 配置读写请求 TLP

配置读写请求 TLP 由 RC 发起，用来访问 PCIe 设备的配置空间。配置请求报文使用基于 ID 的路由方式。PCIe 总线也支持两种配置请求报文，分别为 Type 00h 和 Type 01h 配置请求。配置请求 TLP 的格式如图 5-12 所示。

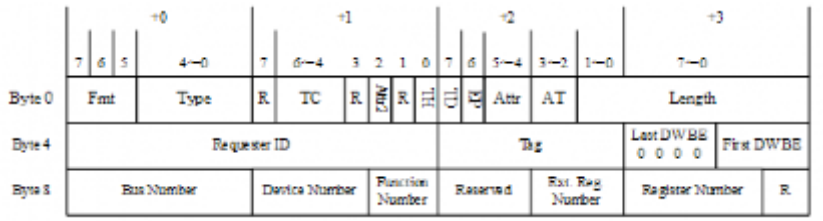

图 1-3-3-1 配置请求报文头格式

配置请求 TLP 的第 4~7 字节与存储器请求 TLP 类似。而第 8~11 字节的 Bus、Device 和 Function Number 中存放该 TLP 访问的目标设备的相应的号码，而 Ext Register 和 Reigister Number 存放寄存器号。配置请求报文的其他字段必须为以下值。

TC[2:0]必须为 0，I/O 请求报文的传送类型(TC)只能为 0。

TH 位为保留位；Attr2 位为保留，而 Attr[1:0]必须为“00b”，这表示 I/O 请求报文使用 PCI 总线的强序数据传送模式；AT[1:0]必须为“0b00”，表示不进行地址转换。

Length[9:0]为“0b00 0000 0001”，表示配置读写请求最大 Payload 为 1DW。

Last DW BE 字段为“0b0000”。而 First DW BE 字段根据配置读写请求的大小设置。

1.3.4 消息请求报文

在 PCIe 总线中，多数消息报文使用隐式路由方式，其格式如图 5-13 所示。其中 Byte 0 字段为通用 TLP 头，而 Byte 4 的第 3 字节中存放 Message Code 字段。

本节讲述 PCIe 总线定义的各类 TLP，并详细介绍这些 TLP 的格式。在这些 TLP 中，有些格式对于初学者来说较难理解。读者需要建立 PCIe 总线中与 TLP 相关的一些基本概念，特别是存储器读写相关的报文格式。在 PCIe 总线中，存储器读写，I/O 读写和配置读写请求 TLP 由以下几类报文组成。

(1)、存储器读请求 TLP 和读完成 TLP

当 PCIe 主设备，RC 或者 EP，访问目标设备的存储器空间时，使用 Non-Posted 总线事务向目标设备发出存储器读请求 TLP，目标设备收到这个存储器读请求 TLP 后，使用存储器读完成 TLP，主动向主设备传递数据。当主设备收到目标设备的存储器读完成 TLP 后，将完成一次存储器读操作。

(2)、存储器写请求 TLP

在 PCIe 总线中，存储器写使用 Posted 总线事务。PCIe 主设备仅使用存储器写请求 TLP 即可完成存储器写操作，主设备不需要目标设备的回应报文。

(3)、原子操作请求和完成报文

原子操作由 PCIe V2.1 总线规范引入，一个完整的原子操作包括原子操作请求和原子操作完成报文组成。原子操作的使用方法与其他 Non-Posted 总线事务类似，首先 PCIe 主设备向目标设备发送原子操作请求，之后目标设备向主设备发送原子操作完成报文，结束一次原子操作。有关原子操作的详细说明见第 1.3.5 节。

(4)、I/O 读写请求 TLP 和读写完成 TLP

在 PCIe 总线中，I/O 读写操作使用 Non-Posted 总线事务，I/O 读写 TLP 都需要完成报文做为回应。只是在 I/O 写请求的完成报文中不需要“带数据”，而仅含有 I/O 写请求是否成功的状态信息。

(5)、配置读写请求 TLP 和配置读写完成 TLP

从总线事务的角度上看，配置读写请求操作的过程与 I/O 读写操作的过程类似。配置读写请求 TLP 都需要配置读写完成作为应答，从而完成一个完成的配置读写操作。

(6)、消息报文

与 PCI 总线相比，PCIe 总线增加了消息请求事务。PCIe 总线使用基于报文的数据传送模式，所有总线事务都是通过报文实现的，PCIe 总线取消了一些在 PCI 总线中存在的边带信号。在 PCIe 总线中，一些由 PCI 总线的边带信号完成的工作，如中断请求和电源管理等，在 PCIe 总线中由消息请求报文实现。

1.4 TLP 中与数据负载相关的参数

在 PCIe 总线中，有些 TLP 含有 Data Payload，如存储器写请求、存储器读完成 TLP 等。在 PCIe 总线中，TLP 含有的 Data Payload 大小与 Max_Payload_Size、Max_Read_Request_Size 和 RCB 参数相关。下文将分别介绍这些参数的使用。

1.4.1Max_Payload_Size 参数

PCIe 总线规定在 TLP 报文中，数据有效负载的最大值为 4KB，但是 PCIe 设备并不一定能够发送这么大的数据报文。PCIe 设备含有“Max_Payload_Size”和“Max_Payload_Size Supported”参数，这两个参数分别在 Device Capability 寄存器和 Device Control 寄存器中定义。

“Max_Payload_Size Supported”参数存放在一个 PCIe 设备中，TLP 有效负载的最大值，该参数由 PCIe 设备的硬件逻辑确定，系统软件不能改写该参数。而 Max_Payload_Size 参数存放 PCIe 设备实际使用的，TLP 有效负载的最大值。该参数由 PCIe 链路两端的设备协商决定，是 PCIe 设备进行数据传送时，实际使用的参数。

PCIe 设备发送数据报文时，使用 Max_Payload_Size 参数决定 TLP 的最大有效负载。当 PCIe 设备的所传送的数据大小超过 Max_Payload_Size 参数时，这段数据将被分割为多个 TLP 进行发送。当 PCIe 设备接收 TLP 时，该 TLP 的最大有效负载也不能超过 Max_Payload_Size 参数，如果接收的 TLP，其 Length 字段超过 Max_Payload_Size 参数，该 PCIe 设备将认为该 TLP 非法。

RC 或者 EP 在发送存储器读完成 TLP 时，这个存储器读完成 TLP 的最大 Payload 也不能超过 Max_Payload_Size 参数，如果超过该参数，PCIe 设备需要发送多个读完成报文。值得注意的是，这些读完成报文需要满足 RCB 参数的要求，有关 RCB 参数的详细说明见下文。

在实际应用中，尽管有些 PCIe 设备的 Max_Payload_Size Supported 参数可以为 256B、512B、1024B 或者更高，但是如果 PCIe 链路的对端设备可以支持的 Max_Payload_Size 参数为 128B 时，系统软件将使用对端设备的 Max_Payload_Size Supported 参数，初始化该设备的 Max_Payload_Size 参数，即选用 PCIe 链路两端最小的 Max_Payload_Size Supported 参数初始化 Max_Payload_Size 参数。

在多数 x86 处理器系统的 MCH 或者 ICH 中，Max_Payload_Size Supported 参数为 128B。这也意味着在 x86 处理器中，与 MCH 或者 ICH 直接相连的 PCIe 设备进行 DMA 读写时，数据的有效负载不能超过 128B。而在 PowerPC 处理器系统中，该参数大多为 256B。

目前在大多数 EP 中，Max_Payload_Size Supported 参数不大于 512B，因为在大多数处理器系统的 RC 中，Max_Payload_Size Supported 参数也不大于 512B。因此即便 EP 支持较大的 Max_Payload_Size Supported 参数，并不会提高数据传送效率。

而 Max_Payload_Size 参数的大小与 PCIe 链路的传送效率成正比，该参数越大，PCIe 链路带宽的利用率越高，该参数越小，PCIe 链路带宽的利用率越低。

PCIe 总线规范规定，对于实时性要求较高的 PCIe 设备，Max_Payload_Size 参数不应设置过大，因此这个参数有时会低于 PCIe 链路允许使用的最大值。

1.4.2Max_Read_Request_Size 参数

Max_Read_Request_Size 参数由 PCIe 设备决定，该参数规定了 PCIe 设备一次能从目标设备读取多少数据。

Max_Read_Request_Size 参数在 Device Control 寄存器中定义。该参数与存储器读请求 TLP 的 Length 字段相关，其中 Length 字段不能大于 Max_Read_Request_Size 参数。在存储器读请求 TLP 中，Length 字段表示需要从目标设备读取多少数据。

值得注意的是，Max_Read_Request_Size 参数与 Max_Payload_Size 参数间没有直接联系，Max_Payload_Size 参数仅与存储器写请求和存储器读完成报文相关。

PCIe 总线规定存储器读请求，其读取的数据长度不能超过 Max_Read_Request_Size 参数，即存储器读 TLP 中的 Length 字段不能大于这个参数。如果一次存储器读操作需要读取的数据范围大于 Max_Read_Request_Size 参数时，该 PCIe 设备需要向目标设备发送多个存储器读请求 TLP。

PCIe 总线规定 Max_Read_Request_Size 参数的最大值为 4KB，但是系统软件需要根据硬件特性决定该参数的值。因为 PCIe 总线规定 EP 在进行存储器读请求时，需要具有足够大的缓冲接收来自目标设备的数据。

如果一个 EP 的 Max_Read_Request_Size 参数被设置为 4KB，而且这个 EP 每发出一个 4KB 大小存储器读请求时，EP 都需要准备一个 4KB 大小的缓冲(这是流量控制 Infinite FC Unit

的要求)。这对于绝大多数 EP，这都是一个相当苛刻的条件。为此在实际设计中，一个 EP 会对 Max_Read_Request_Size 参数的大小进行限制。

1.4.3RCB 参数

RCB 位在 Link Control 寄存器中定义。RCB 位决定了 RCB 参数的值，在 PCIe 总线中，RCB 参数的大小为 64B 或者 128B，如果一个 PCIe 设备没有设置 RCB 的大小[2]，则 RC 的 RCB 参数缺省值为 64B，而其他 PCIe 设备的 RCB 参数的缺省值为 128B。PCIe 总线规定 RC 的 RCB 参数的值为 64B 或者 128B，其他 PCIe 设备的 RCB 参数为 128B。

在 PCIe 总线中，一个存储器读请求 TLP 可能收到目标设备发出的多个完成报文后，才能完成一次存储器读操作。因为在 PCIe 总线中，一个存储器读请求最多可以请求 4KB 大小的数据报文，而目标设备可能会使用多个存储器读完成 TLP 才能将数据传递完毕。

当一个 EP 向 RC 或者其他 EP 读取数据时，这个 EP 首先向 RC 或者其他 EP 发送存储器读请求 TLP；之后由 RC 或者其他 EP 发送存储器读完成 TLP，将数据传递给这个 EP。

如果存储器读完成报文所传递数据的地址范围没有跨越 RCB 参数的边界，那么数据发送端只能使用一个存储器完成报文将数据传递给请求方，否则可以使用多个存储器读完成 TLP。

假定一个 EP 向地址范围为 0xFFFF-0000~0xFFFF-0010 这段区域进行 DMA 读操作，RC 收到这个存储器读请求 TLP 后，将组织存储器读完成 TLP，由于这段区域并没有跨越 RCB 边界，因此 RC 只能使用一个存储器读完成 TLP 完成数据传递。

如果存储器读完成报文所传递数据的地址范围跨越了 RCB 边界，那么数据发送端(目标设备)可以使用一个或者多个完成报文进行数据传递。数据发送端使用多个存储器读完成报文完成数据传递时，需要遵循以下原则。

第一个完成报文所传送的数据，其起始地址与要求的起始地址相同。其结束地址或者为要求的结束地址(使用一个完成报文传递所有数据)，或者为 RCB 参数的整数倍(使用多个完成报文传递数据)。

最后一个完成报文的起始地址或者为要求的起始地址(使用一个完成报文传递所有数据)，或者为 RCB 参数的整数倍(使用多个完成报文传递数据)。其结束地址必须为要求的结束地址。

中间的完成报文的起始地址和结束地址必须为 RCB 参数的整数倍。

当 RC 或者 EP 需要使用多个存储器读完成报文将 0xFFFE-FFF0~0xFFFF-00C7 之间的数据发送给数据请求方时，可以将这些完成报文按照表 5-9 方式组织。

表 x-4-3-1 存储器读完成报文的拆分方法

方式 1	方式 2	方式 3
0xFFFE-FFF0~0xFFFE-FFFF	0xFFFE-FFF0~0xFFFE-FFFF	0xFFFE-FFF0~0xFFFE-FFFF
0xFFFF-0000~0xFFFF-003F	0xFFFF-0000~0xFFFF-007F	0xFFFF-0000~0xFFFF-00C7
0xFFFF-0040~0xFFFF-007F	0xFFFF-0080~0xFFFF-00C7	
0xFFFF-0080~0xFFFF-00BF		
0xFFFF-00C0~0xFFFF-00C7		

上表提供的方式仅供参考，目标设备还可以使用其他拆分方法发送存储器读完成 TLP。PCIe 总线使用多个完成报文实现一次数据读请求的主要原因是考虑 Cache 行长度和流量控制。在多数 x86 处理器系统中，存储器读完成报文的数据长度为一个 Cache 行，即一次传送 64B。除此之外，较短的数据完成报文占用流量控制的资源较少，而且可以有效避免数据拥塞。

1.5 MSI 和 MSI-X 中断机制

在 PCI 总线中，所有需要提交中断请求的设备，必须能够通过 INTx 引脚提交中断请求，而 MSI 机制是一个可选机制。而在 PCIe 总线中，PCIe 设备必须支持 MSI 或者 MSI-X 中断请求机制，而可以不支持 INTx 中断消息。

在 PCIe 总线中，MSI 和 MSI-X 中断机制使用存储器写请求 TLP 向处理器提交中断请求，下文为简便起见将传递 MSI/MSI-X 中断消息的存储器写报文简称为 MSI/MSI-X 报文。不同的处理器使用了不同的机制处理这些 MSI/MSI-X 中断请求，如 PowerPC 处理器使用 MPIC 中断控制器处理 MSI/MSI-X 中断请求；而 x86 处理器使用 FSB Interrupt Message 方式处理 MSI/MSI-X 中断请求。

不同的处理器对 PCIe 设备发出的 MSI 报文的解释并不相同。但是 PCIe 设备在提交 MSI 中断请求时，都是向 MSI/MSI-X Capability 结构中的 Message Address 的地址写 Message Data 数据，从而组成一个存储器写 TLP，向处理器提交中断请求。

有些 PCIe 设备还可以支持 Legacy 中断方式(通过发送 Assert_INTx 和 Deassert_INTx 消息报文进行中断请求，即虚拟中断线方式,本教程使用的就是这种方式)。但是 PCIe 总线并不鼓励其设备使用 Legacy 中断方式，在绝大多数情况下，PCIe 设备使用 MSI 或者 MSI/X 方式进行中断请求。

PCIe 总线提供 Legacy 中断方式的主要原因是，在 PCIe 体系结构中，存在许多 PCI 设备，而这些设备通过 PCIe 桥连接到 PCIe 总线中。这些 PCI 设备可能并不支持 MSI/MSI-X 中断机制，因此必须使用 INTx 信号进行中断请求。

当 PCIe 桥收到 PCI 设备的 INTx 信号后，并不能将其直接转换为 MSI/MSI-X 中断报文，因为 PCI 设备使用 INTx 信号进行中断请求的机制与电平触发方式类似，而 MSI/MSI-X 中断机制与边沿触发方式类似。这两种中断触发方式不能直接进行转换。因此当 PCI 设备的 INTx 信号有效时，PCIe 桥将该信号转换为 Assert_INTx 报文，当这些 INTx 信号无效时，PCIe 桥将该信号转换为 Deassert_INTx 报文。

与 Legacy 中断方式相比，PCIe 设备使用 MSI 或者 MSI-X 中断机制，可以消除 INTx 这个边带信号，而且可以更加合理地处理 PCIe 总线的“序”。目前绝大多数 PCIe 设备使用 MSI 或者 MSI-X 中断机制提交中断请求。

MSI 和 MSI-X 机制的基本原理相同，其中 MSI 中断机制最多只能支持 32 个中断请求，而且要求中断向量连续，而 MSI-X 中断机制可以支持更多的中断请求，而并不要求中断向量连续。与 MSI 中断机制相比，MSI-X 中断机制更为合理。本章将首先介绍 MSI/MSI-X Capability 结构，之后分别以 PowerPC 处理器和 x86 处理器为例介绍 MSI 和 MSI-X 中断机制。

PCIe 设备可以使用 MSI 或者 MSI-X 报文向处理器提交中断请求，但是对于某个具体的 PCIe 设备，可能仅支持一种报文。在 PCIe 设备中含有两个 Capability 结构，一个是 MSI Capability 结构，另一个是 MSI-X Capability 结构。通常情况下一个 PCIe 设备仅包含一种结构，或者为 MSI Capability 结构，或者为 MSI-X Capability 结构。

1.5.1MSI Capability 结构

MSI Capability 结构共有四种组成方式，分别是 32 和 64 位的 Message 结构，32 位和 64 位带中断 Masking 的结构。MSI 报文可以使用 32 位地址或者 64 位地址，而且可以使用 Masking 机制使能或者禁止某个中断源。MSI Capability 寄存器的结构如下图所示。

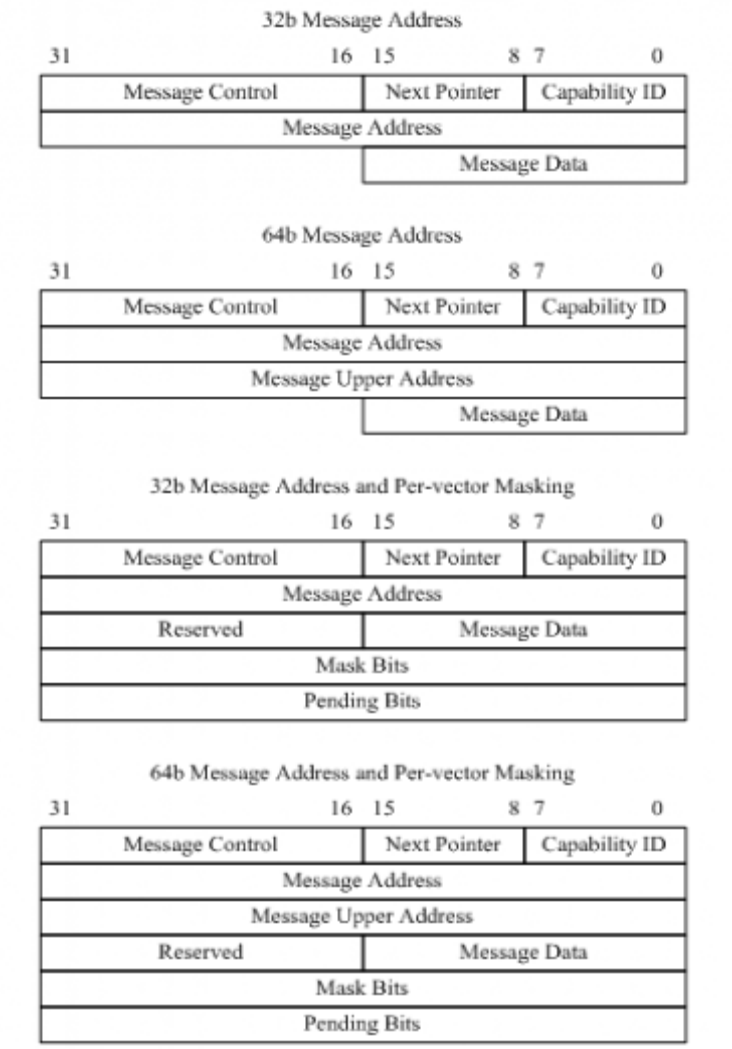

图 1-5-1-1 MSI Capability 结构

Capability ID 字段记载 MSI Capability 结构的 ID 号，其值为 0x05。在 PCIe 设备中，每一个 Capability 结构都有唯一的 ID 号。

Next Pointer 字段存放下一个 Capability 结构的地址。(这个实验用 PCITtree 来做比较方便，但是 PCITtree 只支持 32bit 系统，后续如果有支持 64bit 系统再讲解)

Message Control 字段。该字段存放当前 PCIe 设备使用 MSI 机制进行中断请求的状态与控制信息，如下表所示。

表 1-5-1-2 MSI Cabalibities 结构的 Message Control 字段

Bits	定义	描述
15:9	Reserved	保留位。系统软件读取该字段时将返回全零，对此字段写无意义。
8	Per-vector Masking Capable	该位为 1 时，表示支持带中断 Masking 的结构；如果为 0，表示不支持带中断 Masking 的结构。该位对系统软件只读，该位在 PCIe 设备初始化时设置。
7	64 bit Address Capable	该位为 1 时，表示支持 64 位地址结构；如果为 0，表示只能支持带 32 位地址结构。该位对系统软件只读，该位在 PCIe 设备初始化时设置。
6:4	Multiple Message Enable	该字段可读写，表示软件分配给当前 PCIe 设备的中断向量数目。系统软件根据 Multiple Message Capable 字段的大小确定该字段的值。在系统的中断向量资源并不紧张时，Multiple Message Capable 字段和该字段的值相等；而资源紧张时，该字段的值可能小于 Multiple Message Capable 字段的值。
3:1	Multiple Message Capable	该字段对系统软件只读，表示当前 PCIe 设备可以使用几个中断向量号，在不同的 PCIe 设备中该字段的值并不不同。当该字段为 0b000 时，表示 PCIe 设备可以使用 1 个中断向量；为 0b001、0b010、0b011、0b100 和 0b101 时，表示使用 4、8、16 和 32 个中断向量；而 0b110 和 0b111 为保留位。 该字段与 Multiple Message Enable 字段的含义不同，该字段表示，当前 PCIe 设备支持的中断向量个数，而 Multiple Message Enable 字段是系统软件分配给 PCIe 设备实际使用的中断向量个数。

Bits	定义	描述
0	MSI Enable	该位可读写，是 MSI 中断机制的使能位。该位为 1 而且 MSI-X Enable 位为 0 时，当前 PCIe 设备可以使用 MSI 中断机制，此时 Legacy 中断机制被禁止。一个 PCIe 设备的 MSI Enble 和 MSI-X Enable 位都被禁止时，将使用 INTx 中断消息报文发出/结束中断请求[1]。

Message Address 字段。当 MSI Enable 位有效时，该字段存放 MSI 存储器写事务的目的地址的低 32 位。该字段的 31:2 字段有效，系统软件可以对该字段进行读写操作；该字段的第 1~0 位为 0。

Message Upper Address 字段。如果 64 bit Address Capable 位有效，该字段存放 MSI 存储器写事务的目的地址的高 32 位。

Message Data 字段，该字段可读写。当 MSI Enable 位有效时，该字段存放 MSI 报文使用的数据。该字段保存的数值与处理器系统相关，在 PCIe 设备进行初始化时，处理器将初始化该字段，而且不同的处理器填写该字段的规则并不相同。如果 Multiple Message Enable 字段不为 0b000 时(即该设备支持多个中断请求时)，PCIe 设备可以通过改变 Message Data 字段的低位数据发送不同的中断请求。

Mask Bits 字段。PCIe 总线规定当一个设备使用 MSI 中断机制时，最多可以使用 32 个中断向量，从而一个设备最多可以发送 32 种中断请求。Mask Bits 字段由 32 位组成，其中每一位对应一种中断请求。当相应位为 1 时表示对应的中断请求被屏蔽，为 0 时表示允许该中断请求。系统软件可读写该字段，系统初始化时该字段为全 0，表示允许所有中断请求。该字段和 Pending Bits 字段对于 MSI 中断机制是可选字段，但是 PCIe 总线规范强烈建议所有 PCIe 设备支持这两个字段。

Pending Bits 字段。该字段对于系统软件是只读位，PCIe 设备内部逻辑可以改变该字段的值。该字段由 32 位组成，并与 PCIe 设备使用的 MSI 中断一一对应。该字段需要与 Mask Bits 字段联合使用。

当 Mask Bits 字段的相应位为 1 时，如果 PCIe 设备需要发送对应的中断请求时，Pending Bits 字段的对应位将被 PCIe 设备的内部逻辑置 1，此时 PCIe 设备并不会使用 MSI 报文向中断控制器提交中断请求；当系统软件将 Mask Bits 字段的相应位从 1 改写为 0 时，PCIe 设备将发送 MSI 报文向处理器提交中断请求，同时将 Pending Bit 字段的对应位清零。

在设备驱动程序的开发中，有时需要联合使用 Mask Bits 和 Pending Bits 字段防止处理器丢弃中断请求[2]。

1.5.2MSI-X Capability 结构

MSI-X Capability 中断机制与 MSI Capability 的中断机制类似。PCIe 总线引出 MSI-X 机制的主要目的是为了扩展 PCIe 设备使用中断向量的个数，同时解决 MSI 中断机制要求使用中断向量号连续所带来的问题。

MSI 中断机制最多只能使用 32 个中断向量，而 MSI-X 可以使用更多的中断向量。目前 Intel 的许多 PCIe 设备支持 MSI-X 中断机制。与 MSI 中断机制相比，MSI-X 机制更为合理。首先 MSI-X 可以支持更多的中断请求，但是这并不是引入 MSI-X 中断机制最重要的原因。因为对于多数 PCIe 设备，32 种中断请求已经足够了。而引入 MSI-X 中断机制的主要原因是，使用该机制不需要中断控制器分配给该设备的中断向量号连续。

如果一个 PCIe 设备需要使用 8 个中断请求时，如果使用 MSI 机制时，Message Data 的[2:0]字段可以为 0b000~0b111，因此可以发送 8 种中断请求，但是这 8 种中断请求的 Message Data 字段必须连续。在许多中断控制器中，Message Data 字段连续也意味着中断控制器需要为这个 PCIe 设备分配 8 个连续的中断向量号。

有时在一个中断控制器中，虽然具有 8 个以上的中断向量号，但是很难保证这些中断向量号是连续的。因此中断控制器将无法为这些 PCIe 设备分配足够的中断请求，此时该设备的“Multiple Message Enable”字段将小于“Multiple Message Capable”。

而使用 MSI-X 机制可以合理解决该问题。在 MSI-X Capability 结构中，每一个中断请求都使用独立的 Message Address 字段和 Message Data 字段，从而中断控制器可以更加合理地为该设备分配中断资源。

与 MSI Capability 寄存器相比，MSI-X Capability 寄存器使用一个数组存放 Message Address 字段和 Message Data 字段，而不是将这两个字段放入 Capability 寄存器中，本篇将这个数组称为 MSI-X Table。从而当 PCIe 设备使用 MSI-X 机制时，每一个中断请求可以使用独立的 Message Address 字段和 Message Data 字段。

除此之外 MSI-X 中断机制还使用了独立的 Pending Table 表，该表用来存放与每一个中断向量对应的 Pending 位。这个 Pending 位的定义与 MSI Capability 寄存器的 Pending 位类似。MSI-X Table 和 Pending Table 存放在 PCIe 设备的 BAR 空间中。MSI-X 机制必须支持这个 Pending Table，而 MSI 机制的 Pending Bits 字段是可选的。

1)、MSI-X Capability 结构

MSI-X Capability 结构比 MSI Capability 结构略微复杂一些。在该结构中，使用 MSI-X Table 存放该设备使用的所有 Message Address 和 Message Data 字段，这个表格存放在该设备的 BAR 空间中，从而 PCIe 设备可以使用 MSI-X 机制时，中断向量号可以并不连续，也可以申请更多的中断向量号。MSI-X Capability 结构的组成方式如下图所示。

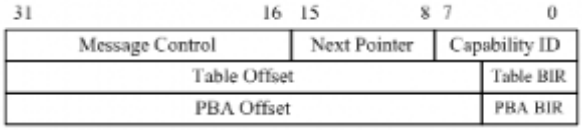

图 1-5-2-1 MSI-X Capability 结构的组成方式

上图中各字段的含义如下所示。

Capability ID 字段记载 MSI-X Capability 结构的 ID 号，其值为 0x11。在 PCIe 设备中，每一个 Capability 都有唯一的一个 ID 号。

Next Pointer 字段存放下一个 Capability 结构的地址。

Message Control 字段，该字段存放当前 PCIe 设备使用 MSI-X 机制进行中断请求的状态与控制信息，如下表所示。

表 1-5-2-1MSI-X Capability 结构的 Message Control 字段

Bits		定义	描述
15		MSI-X Enable	该位可读写，是 MSI-X 中断机制的使能位，复位值为 0，表示不使能 MSI-X 中断机制。该位为 1 且 MSI Enable 位为 0 时，当前 PCIe 设备使用 MSI-X 中断机制，此时 INTx 和 MSI 中断机制被禁止。当 PCIe 设备的 MSI Enble 和 MSI-X Enable 位为 0 时，将使用 INTx 中断消息报文发出/结束中断请求。
14		Function Mask	该位可读写，是中断请求的全局 Mask 位，复位值为 0。如果该位为 1，该设备所有的中断请求都将被屏蔽；如果该位为 0，则由 Per Vector Mask 位，决定是否屏

Bits		定义	描述
			蔽相应的中断请求。Per Vector Mask 位在 MSI-X Table 中定义，详见下文。
10：0		Table Size	MSI-X 中断机制使用 MSI-X Table 存放 Message Address 字段和 Message Data 字段。该字段用来存放 MSI-X Table 的大小，该字段对系统软件只读。

Table BIR(BAR Indicator Register)。该字段存放 MSI-X Table 所在的位置，PCIe 总线规范规定 MSI-X Table 存放在设备的 BAR 空间中。该字段表示设备使用 BAR0~5 寄存器中的哪个空间存放 MSI-X table。该字段由三位组成，其中 0b000~0b101 与 BAR0~5 空间一一对应。

Table Offset 字段。该字段存放 MSI-X Table 在相应 BAR 空间中的偏移。

PBA(Pending Bit Array) BIR 字段。该字段存放 Pending Table 在 PCIe 设备的哪个 BAR 空间中。在通常情况下，Pending Table 和 MSI-X Table 存放在 PCIe 设备的同一个 BAR 空间中。

PBA Offset 字段。该字段存放 Pending Table 在相应 BAR 空间中的偏移。

2)、MSI-X Table

MSI-X Table 的组成结构如下图所示。

DWORD 3	DWORD 2	DWORD 1	DWORD 0	
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	Entry 0
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	Entry 1
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	Entry 2
...	...	...	...	...
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	Entry N-1

图 1-5-2-2 MSI-X Table 的组成结构

由上图可见，MSI-X Table 由多个 Entry 组成，其中每个 Entry 与一个中断请求对应。其中每一个 Entry 中有四个参数，其含义如下所示。

Msg Addr。当 MSI-X Enable 位有效时，该字段存放 MSI-X 存储器写事务的目的地址的低 32 位。该双字的 31:2 字段有效，系统软件可读写；1:0 字段复位时为 0，PCIe 设备可以根据需要将这个字段设为只读，或者可读写。不同的处理器填入该寄存器的数据并不相同。

Msg Upper Addr，该字段可读写，存放 MSI-X 存储器写事务的目的地址的高 32 位。

Msg Data，该字段可读写，存放 MSI-X 报文使用的数据。其定义与处理器系统使用的中断控制器和 PCIe 设备相关。

Vector Control，该字段可读写。该字段只有第 0 位(即 Per Vector Mask 位)有效，其他位保留。当该位为 1 时，PCIe 设备不能使用该 Entry 提交中断请求；为 0 时可以提交中断请求。该位在复位时为 0。Per Vector Mask 位的使用方法与 MSI 机制的 Mask 位类似。

3)、Pending Table

Pending Table 的组成结构如图 6-4 所示。

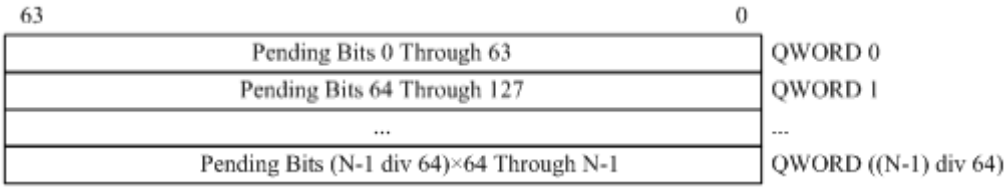

图 1-5-2-3 Pending Table 的组成结构

如上图所示，在 Pending Table 中，一个 Entry 由 64 位组成，其中每一位与 MSI-X Table 中的一个 Entry 对应，即 Pending Table 中的每一个 Entry 与 MSI-X Table 的 64 个 Entry 对应。与 MSI 机制类似，Pending 位需要与 Per Vector Mask 位配置使用。

当 Per Vector Mask 位为 1 时，PCIe 设备不能立即发送 MSI-X 中断请求，而是将对应的 Pending 位置 1；当系统软件将 Per Vector Mask 位清零时，PCIe 设备需要提交 MSI-X 中断请求，同时将 Pending 位清零。

1.6 x86 处理器如何处理 MSI-X 中断请求

PCIe 设备发出 MSI-X 中断请求的方法与发出 MSI 中断请求的方法类似，都是向 Message Address 所在的地址写 Message Data 字段包含的数据。只是 MSI-X 中断机制为了支持更多的中断请求，在 MSI-X Capablity 结构中存放了一个指向一组 Message Address 和 Message Data 字段的指针，从而一个 PCIe 设备可以支持的 MSI-X 中断请求数目大于 32 个，而且并不要求中断向量号连续。MSI-X 机制使用的这组 Message Address 和 Message Data 字段存放在 PCIe 设备的 BAR 空间中，而不是在 PCIe 设备的配置空间中，从而可以由用户决定使用 MSI-X 中断请求的数目。

当系统软件初始化 PCIe 设备时，如果该 PCIe 设备使用 MSI-X 机制传递中断请求，需要对 MSI-X Capability 结构指向的 Message Address 和 Message Data 字段进行设置，并使能 MSI-X Enable 位。x86 处理器在此处的实现与 PowerPC 处理器有较大的不同。

1.6.1 Message Address 字段和 Message Data 字段的格式

在 x86 处理器系统中，PCIe 设备也是通过向 Message Address 写入 Message Data 指定的数值实现 MSI/MSI-X 机制。在 x86 处理器系统中，PCIe 设备使用的 Message Adress 字段和 Message Data 字段与 PowerPC 处理器不同。

1)、 PCIe 设备使用 Message Adress 字段

在 x86 处理器系统中，PCIe 设备使用的 Message Address 字段仍然保存 PCI 总线域的地址，其格式如下图所示。

图 1-5-1-1 Message Address 字段的格式

其中第 31~20 位，存放 FSB Interrupts 存储器空间的基地址，其值为 0xFEE。当 PCIe 设备对 0xFEE5-XXXX 这段“PCI 总线域”的地址空间进行写操作时，MCH/ICH 将会首先进行“PCI 总线域”到“存储器域”的地址转换，之后将这个写操作翻译为 FSB 总线的 Interrupt Message 总线事务，从而向 CPU 内核提交中断请求。

x86 处理器使用 FSB Interrupt Message 总线事务转发 MSI/MSI-X 中断请求。使用这种方法的优点是向 CPU 内核提交中断请求的同时，提交 PCIe 设备使用的中断向量，从而 CPU 不需要使用中断响应周期从寄存器中获得中断向量。FSB Interrupt Message 总线事务的详细说明见下文。

Message Address 字段其他位的含义如下所示。

Destination ID 字段保存目标 CPU 的 ID 号，目标 CPU 的 ID 与该字段相等时，目标 CPU 将接收这个 Interrupt Message。FSB Interrupt Message 总线事务可以向不同的 CPU 提交中断请求。

RH(Redirection Hint Indication)位为 0 时，表示 Interrupt Message 将直接发向与 Destination ID 字段相同的目标 CPU；如果 RH 为 1 时，将使能中断转发功能。

DM(Destination Mode)位表示在传递优先权最低的中断请求时，Destination ID 字段是否被翻译为 Logical 或者 Physical APIC ID。在 x86 处理器中 APIC ID 有三种模式，分别为 Physical、Logical 和 Cluster ID 模式。

如果 RH 位为 1 且 DM 位为 0 时，Destination ID 字段使用 Physical 模式；如果 RH 位为 1 且 DM 位为 1，Destination ID 字段使用 Logical 模式；如果 RH 位为 0，DM 位将被忽略。

以上这些字段的描述与 x86 处理器使用的 APIC 中断控制器相关。对 APIC 的详细说明超出了本书的范围，对此部分感兴趣的读者请参阅 Intel 64 and IA-32 Architectures Software Developer’ s Manual Volume 3A: System Programming Guide, Part 1。

2)、Message Data 字段

Message Data 字段的格式如下图所示。

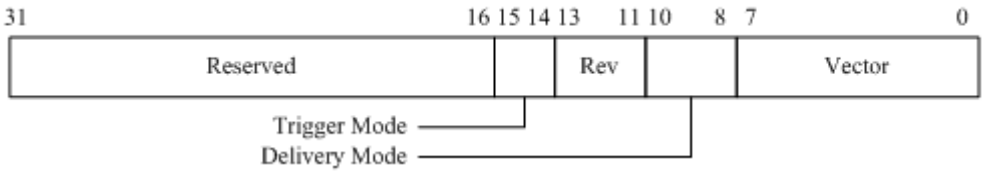

图 1-5-1-2 Message Data 字段的格式

Trigger Mode 字段为 0b0x 时，PCIe 设备使用边沿触发方式申请中断；为 0b10 时使用低电平触发方式；为 0b11 时使用高电平触发方式。MSI/MSI-X 中断请求使用边沿触发方式，但是 FSB Interrupt Message 总线事务还支持 Legacy INTx 中断请求方式，因此在 Message Data 字段中仍然支持电平触发方式。但是对于 PCIe 设备而言，该字段为 0b0x。

Vector 字段表示这个中断请求使用的中断向量。FSB Interrupt Message 总线事务在提交中断请求的同时，将中断向量也通知给处理器。因此使用 FSB Interrupt Message 总线事务时，处理器不需要使用中断响应周期通过读取中断控制器获得中断向量号。与 PowerPC 的传统方式相比，x86 处理器的这种中断请求的效率较高(P4080 处理器也提供了一种类似于 FSB Interrupt Message 总线事务的中断请求方法)。

值得注意的是，在 x86 处理器中，MSI 机制使用的 Message Data 字段与 MSI-X 机制相同。但是当一个 PCIe 设备支持多个 MSI 中断请求时，其 Message Data 字段必须是连续的，

因而其使用的 Vector 字段也必须是连续的，这也是在 x86 处理器系统中，PCIe 设备支持多个 MSI 中断请求的问题所在，而使用 MSI-X 机制有效避免了该问题。

Delivery Mode 字段表示如何处理来自 PCIe 设备的中断请求。

该字段为 0b000 时，表示使用“Fixed Mode”方式。此时这个中断请求将被 Destination ID 字段指定的 CPU 处理。

该字段为 0b001 时，表示使用“Lowest Priority”方式。此时这个中断请求将被优先权最低的 CPU 处理。当使用“Fixed Mode”和“Lowest Priority”方式时，如果 Vector 字段有效，CPU 接收到这个中断请求之后，将使用 Vector 字段指定的中断向量处理这些中断请求；而当 Delivery Mode 字段为其他值时，Message Data 字段中所包含的 Vector 字段无效。

该字段为 0b010 时，表示使用 SMI 方式传递中断请求，而且必须使用边沿触发，此时 Vector 字段必须为 0。这个中断请求将被 Destination ID 字段指定的 CPU 处理。

该字段为 0b100 时，表示使用 NMI 方式传递中断请求，而且必须使用边沿触发，此时 Vector 字段和 Trigger 字段的内容将被忽略。这个中断请求将被 Destination ID 字段指定的 CPU 处理。

该字段为 0b101 时，表示使用 INIT 方式传递中断请求，Vector 字段和 Trigger 字段的内容将被忽略。这个中断请求将被 Destination ID 字段指定的 CPU 处理。

该字段为 0b111 时，表示使用 INTR 信号传递中断请求且使用边沿触发。此时 MSI 中断信息首先传递给中断控制器，然后中断控制器在通过 INTR 信号向 CPU 传递中断请求，之后 CPU 在通过中断响应周期获得中断向量。上文中 PowerPC 处理器使用的方法与此方法类似。而在 x86 处理器中多使用 Interrupt Message 总线事务进行 MSI 中断信息的传递，因此这种模式很少被使用。

边沿触发和电平触发是中断请求常用的两种方式。其中电平触发指外部设备使用逻辑电平 1(高电平触发)或者 0(低电平触发)，提交中断请求。使用电平或者边沿方式提交中断请求时，外部设备一般通过中断线(IRQ_PIN#)与中断控制器相连，其中多个外部设备可能通过相同的中断线与中断控制器相连(线与或者与门)。

外部设备在使用低电平触发，提交中断请求的过程中，首先需要将 IRQ_PIN#信号驱动为低。当中断控制器将该中断请求提交给处理器，而且处理器将这个中断请求处理完毕后，处理器将通过写外部设备的某个寄存器来清除此中断源，此时外部设备将不再驱动 IRQ_PIN#信号线，从而结束整个中断请求。

IRQ_PIN#信号线可以被多个外部设备共享，在这种情况之下，只有所有外部设备都不驱动 IRQ_PIN#信号线时，IRQ_PIN#信号才为高电平。采用电平触发方式进行中断请求的优点是不会丢失中断请求，而缺点是一个优先权较高的中断请求有可能会长期占用中断资源，从而使其他优先权较低的中断不能被及时提交。因为优先级别较高的中断源有可能会持续不断地驱动 IRQ_PIN#信号。

而边沿触发使用上升沿(0 到 1)或者下降沿(1 到 0)作为触发条件，但是中断控制器并不是使用这个“边沿”作为触发条件。中断控制器使用内部时钟对 IRQ_PIN#信号进行采样，如果在前一个时钟周期，IRQ_PIN#信号为 0，而后一个时钟周期，IRQ_PIN#信号为 1，中断控制器认为外部设备提交了一个有效“上升沿”，中断控制器会锁定这个“上升沿”并向处理器发出中断请求。这也是外部设备至少需要将 IRQ_PIN#信号保持一个时钟采样周期的原因，否则中断控制器可能无法识别本次边沿触发的中断请求，从而产生 Spurious 中断请求。

外部设备使用“上升沿”进行中断申请时，不需要持续地将 IRQ_PIN#信号驱动为 1，而只需要保证中断控制器可以进行正确采样这些中断信号即可。在处理边沿触发中断请求时，处理器不需要清除中断源。

使用边沿触发可以有效避免“优先级别”较高的中断源长期占用 IRQ_PIN#信号的情况，使用“下降沿”触发进行中断请求与“上升沿”触发类似。

但是外部设备使用边沿触发方式时，有可能会丢失一些中断请求。例如在一个处理器系统中，存在一个定时器，这个定时器使用上升沿触发方式向中断控制器定时提交中断。如果当处理器正在处理这个定时器的上一个中断请求时，将不会处理这个定时器发出的其他“边沿”中断请求，从而导致中断丢失。而使用电平触发方式不会出现这类问题，因为电平触发方式是一个“持续”过程，处理器只有处理完毕当前中断，并清除相应中断源之后，才会处理下一个中断源。

MSI 中断请求实际上和边沿触发方式非常类似，MSI 中断请求通过存储器写 TLP 实现，这个写动作是一个瞬间的动作，并不是一个持续请求，因此在 x86 处理器中 MSI 中断请求使用边沿触发进行中断请求。

还有一些外部设备可以通过 I/O APIC 进行中断请求(与 I/O APIC 的 IRQX#引脚链接的外部设备)，这些 I/O APIC 接收的外部中断需要标明是使用边沿或者电平触发，I/O APIC 使用 FSB Interrupt Message 总线事务将中断请求发向 Local APIC，并由 Local APIC 向处理器提交中断请求。

1.6.2 FSB Interrupt Message 总线事务

与 MPC8572 处理器处理 MSI 中断请求不同，x86 处理器使用 FSB 的 Interrupt Message 总线事务，处理 PCIe 设备的 MSI/MSI-X 中断请求。由上文所示，MPC8572 处理器处理 MSI 中断请求时，首先由 MPIC 中断控制器截获这个 MSI 中断请求，之后由 MPIC 中断控制器向 CPU 提交中断请求，而 CPU 通过中断响应周期从 MPIC 中断控制器的 ACK 寄存器中获得中断向量。

采用这种方式的主要问题是，当一个处理器中存在多个 CPU 时，这些 CPU 都需要通过中断响应周期从 MPIC 中断控制器的 ACK 寄存器中获得中断向量。在一个中断较为密集的应用中，ACK 寄存器很可能会成为系统瓶颈。而采用 Interrupt Message 总线事务可以有效地避免这种系统瓶颈，因为使用这种方式中断信息和中断向量将同时到达指定的 CPU，而不需要使用中断响应周期获得中断向量。

x86 处理器也具有通过中断控制器提交 MSI/MSI-X 中断请求的方法，在 I/O APIC 具有一个 “The IRQ Pin Assertion Register” 寄存器，该寄存器地址为 0xFEC00020(该寄存器在存储器域和 PCI 总线域中的地址都为 0xFEC00020)，其第 4~0 位存放 IRQ Number。系统软件可以将 PCIe 设备的 Message Address 寄存器设置为 0xFEC00020，将 Meaasge Data 寄存器设置为相应的 IRQ Number。

当 PCIe 设备需要提交 MSI 中断请求时，将向 PCI 总线域的 0xFEC00020 地址写入 Message Data 寄存器中的数据。此时这个存储器写请求将数据写入 I/O APIC 的 The IRQ Pin Assertion Register 中，并由 I/O APIC 将这个 MSI 中断请求最终发向 Local APIC，之后再由 Local APIC 通过 INTR#信号向 CPU 提交中断请求。

上述步骤与 MPC8572 处理器传递 MSI 中断的方法类似。在 x86 处理器中，这种方式基本上已被弃用。下文以图 1-5-2-1 为例，说明 x86 处理器如何使用 FSB 总线的 Interrupt Message 总线事务，向 CPU 提交 MSI/MSI-X 中断请求。

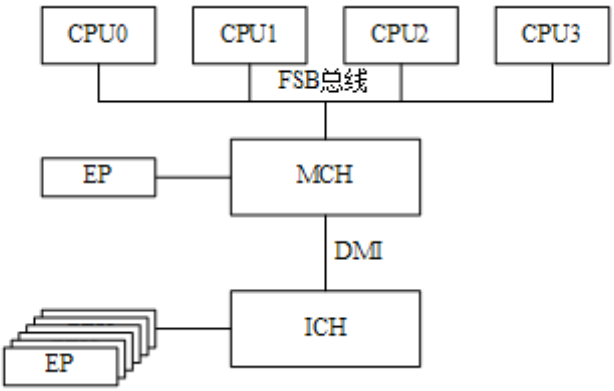

图 1-5-2-1 Interrupt Message 总线食物传递 MSI 中断请求

PCIe 设备在发送 MSI/MSI-X 中断请求之前，系统软件需要合理设置 PCIe 设备 MSI/MSI-X Capability 寄存器，使 Message Address 寄存器的值为 0xFEExx00y(其中 xx 表示 APIC ID，而 y 为 RH+DM)，同时合理地设置 Message Data 寄存器 Vector 字段。

PCIe 设备提交 MSI/MSI-X 中断请求时，需要向 0xFEExx00y 地址写 Message Data 寄存器中包含的数据，并以存储器写 TLP 的形式发送到 RC。如果 ICH 收到这个存储器写 TLP 时，将通过 DMI 接口将这个 TLP 提交到 MCH。MCH 收到这个 TLP 后，发现这个 TLP 的目的地址在 FSB Interrupts 存储器空间中，则将 PCIe 总线的存储器写请求转换为 Interrupt Message 总线事务，并在 FSB 总线上广播。

FSB 总线上的 CPU，根据 APIC ID 信息，选择是否接收这个 Interrupt Message 总线事务，并进入中断状态，之后该 CPU 将直接从这个总线事务中获得中断向量号，执行相应的中断服务例程，而不需要从 APIC 中断控制器获得中断向量。与 PowerPC 处理器的 MPIC 中断控制器相比，这种方法更具优势。

1.7 本章小结

本章主要讲解了 TLP 包的格式和类型、PCIE 的中断机制，掌握这些内容是我们后面分析代码的必要条件之一。

本章内容通篇转载于网络博客：

http://blog.sina.com.cn/s/blog_6472c4cc0102dskl.html

一个完美的结束，

意味着一个新的开始！