**BIOS**

# 第十一课 PCI & PCIe

## PCI Architecture

各种硬件控制模块大部分是以PCIe设备的形式挂载到了一颗或者几颗PCI/PCIe设备树上。固件和操作系统正是通过枚举设备树们才能发现绝大多数即插即用（PNP）设备的。

### PCI(Peripheral Component Interconnect )

PCI本地总线是具有多路复用地址和数据线的高性能32位或64位总线。 该总线旨在用作高度集成的外围控制器组件，外围外接卡和处理器/内存系统之间的互连机制。

* 连接到PCI总线的设备在处理器的地址空间中分配了地址
* 它是并行总线，与单个总线时钟同步。
* PC中使用的典型PCI卡包括：网卡，声卡，调制解调器，USB或串行端口等额外端口，电视调谐器卡和磁盘控制器。
* PCI是用于在计算机中连接硬件设备的本地计算机总线

### 

![desc](data:None;base64,)

### 

![desc](data:None;base64,)

### pci总线树

![desc](data:None;base64,)

PCI 设备。符合 PCI 总线标准的设备就被称为 PCI 设备，PCI 总线架构中可以包含多个 PCI 设备。e.g.LAN、AudioPCI 总线。PCI 总线在系统中可以有多条，类似于树状结构进行扩展，每条 PCI 总线都可以连接多个 PCI 设备/桥e.g,PCI local Bus#0、PCI Local Bus#1PCI 桥。当一条 PCI 总线的承载量不够时，可以用新的 PCI 总线进行扩展，而 PCI 桥则是连接 PCI 总线之间的纽带

### 

![desc](data:None;base64,)

### PCI Local Bus

![desc](data:None;base64,)

### 典型的桌面系统PCI架构

![desc](data:None;base64,)

桌面系统一般只有一个Host Bridge用于隔离处理器系统的存储器域与PCI总线域，并完成处理器与PCI设备间的数据交换。每个Host Bridge单独管理独立的总线空间，包括PCI Bus, PCI I/O, PCI Memory, and PCI Prefetchable Memory Space。桌面系统也一般只有一个Root Bridge，每个Root Bridge管理一个Local Bus空间，它下面挂载了一颗PCI总线树，在同一颗PCI总线树上的所有PCI设备属于同一个PCI总线域。桌面系统也一般只有一个Root Bridge，每个Root Bridge管理一个Local Bus空间，它下面挂载了一颗PCI总线树，在同一颗PCI总线树上的所有PCI设备属于同一个PCI总线域。一颗典型的PCI总线树如图：

### 一个HB和四个RB

![desc](data:None;base64,)

一个Host Bridge，但有四个Root Bridge，管理了四颗单独的PCI树，树之间共享Bus PCI空间。

### 

![desc](data:None;base64,)

## PCIE Architecture

### PCI Express - Peripheral Component Interconnect Express

PCI Express基于PCI架构、差分信号 、点对点拓扑 、内存访问 PCI Express支持4096字节的配置空间，而PCI允许的256字节

* PCIe是高速串行计算机扩展总线标准，旨在替代较旧的PCI，PCI-X和AGP总线标准
* PCIe对旧标准进行了许多改进，包括更高的最大系统总线吞吐量，更少的I / O引脚数和更小的物理占用空间，更好的总线设备性能扩展，更详细的错误检测和报告机制（Advanced Error Reporting -AER）， 以及本机热插拔功能，PCIe标准的最新版本为I / O虚拟化提供了硬件支持。

### pcie系统框图

![desc](data:None;base64,)

从图中可以看出，PCIE是点对点结构一个root port和一个endpoint直接组成一个点对点连接对Switch可以同时连接几个endpoint

### 

![desc](data:None;base64,)

## PCI Configuration Space（配置空间）

### PCI 配置空间

![desc](data:None;base64,)

### PCI桥片配置空间

![desc](data:None;base64,)

### 

![desc](data:None;base64,)

### 

![desc](data:None;base64,)

### PCI配置空间访问

* 使用IO空间的CF8h/CFCh地址来访问PCI设备的配置寄存器:
* CF8h: PCI配置空间地址端口。

![desc](data:None;base64,)

通过Bus Number, Device Number和Function Number 三段编码区分PCI设备在CONFIG\_ADDRESS端口填入BDF,即可以在CONFIG\_DATA上写入或者读出PCI配置空间的内容

* CFCh: PCI配置空间数据端口。

![desc](data:None;base64,)

### PCIE配置空间访问

PCIe规范在PCI规范的基础上，将配置空间扩展到4KB。原来的CF8/CFC方法仍然可以访问所有PCIe设备配置空间的头255B，但是该方法访问不了剩下的（4K-255）配置空间；通过将配置空间映射到Memory map IO（MMIO）空间，对PCIe配置空间可以像对内存一样进行读写访问了。



![desc](data:None;base64,)



![desc](data:None;base64,)

* RC (Root Complex)

• RC内部集成了一些PCI设备、RCRB(RC Register Block)和Event Collector等组成部件。其中RCRB由一系列的寄存器组成的大杂烩，而仅存在于x86处理器中；而Event Collector用来处理来自PCIe设备的错误消息报文和PME消息报文。RCRB的访问基地址一般在LPC设备寄存器上设置。

• 如果将RC中的RCRB、内置的PCI设备和Event Collector去除，该RC的主要功能与PCI总线中的Host Bridge类似，其主要作用是完成存储器域到PCI总线域的地址转换。但是随着虚拟化技术的引入，尤其是引入MR-IOV技术之后，RC的实现变得异常复杂

### Device Identification

* Vender ID

此字段标识设备的制造商。 有效的供应商标识符由 PCI SIG 分配以确保唯一性。 0FFFFh 是供应商 ID 的无效值

* Device ID

该字段标识特定设备。 此标识符由供应商分配

* Revision ID

该寄存器指定特定于设备的修订标识符。 该值由供应商选择

* Header Type

此字节标识预定义标头的第二部分的布局（从配置空间中的字节 10h 开始）以及设备是否包含多个功能

* Class Code

类代码寄存器是只读的，用于标识设备的通用功能，在某些情况下，还用于标识特定的寄存器级编程接口。 例如。 偏移量0x0B = 03h表示显示控制器

### Device Control & Status

* Device Control

命令寄存器提供对设备生成和响应 PCI 周期的能力的粗略控制，当向该寄存器写入 0 时，该设备在逻辑上与 PCI 总线断开连接，以便进行除配置访问之外的所有访问

•

![desc](data:None;base64,)

* Device Status

Status寄存器用于记录PCI总线相关事件的状态信息

•

![desc](data:None;base64,)

### Interrupt Line

该寄存器中的值表明设备的中断引脚连接到系统中断控制器的哪个输入端。 例如。 偏移量 0x3C=05h 表示设备的 PIRQ 路由到 8259 IRQ5

### Interrupt Pin

中断引脚寄存器告诉设备（或设备功能）使用哪个中断引脚



![desc](data:None;base64,)

### Capabilities Pointer

此可选寄存器用于指向此设备实现的新功能的链接列表。 该寄存器仅在状态寄存器中的“功能列表”位被设置时有效

### Capabilities List

该可选数据结构在 PCI 状态寄存器中通过设置Capabilities List 位（位 4）来指示能力指针位于偏移量 34h，该寄存器指向Capabilities List中的第一项。 指针值00h用于指示列表中的最后一个功能



![desc](data:None;base64,)

### Capability ID

每个定义的能力都必须有一个 SIG 分配的 ID 代码。 这些代码的分配和处理非常类似于类代码。 当前定义的功能列表如下所示



![desc](data:None;base64,)



![desc](data:None;base64,)



![desc](data:None;base64,)

### PCI Bus Enumeration

系统复位后，所有 PCI 设备都处于非活动状态。 必须枚举连接在 PCI 总线上的 PCI 设备，然后 BIOS 或操作系统才能对这些 PCI 设备进行寻址。 总线枚举是通过尝试读取设备功能 #0 处总线编号和设备编号的每个组合的供应商寄存器和设备 ID 寄存器来执行的。 如果没有收到来自设备函数#0 的响应，总线主设备将执行中止并返回一个全位值 (0xFFFFFFFF)，这是一个无效的 VID/DID 值，因此设备驱动程序可以判断指定的组合 总线/设备/功能不存在。 当对供应商 ID 寄存器的读取成功时，设备驱动程序知道它存在，它将所有 1 写入其 BAR 并读回设备请求的内存大小。 如果找到了PCI到PCI的桥接器，则系统必须为桥接器以外的辅助PCI总线分配一个总线号（非零）。 然后枚举该辅助总线上的设备。 如果找到更多的PCI桥接器，则继续进行直到扫描所有可能的总线/设备。

### Bus Number Registers

* Primary Bus Number :

表示主干的总线号

* Secondary Bus Number ：

表示端口的总线号。

* Subordinate Bus Number：

表示桥下最高的 PCI 总线编号



![desc](data:None;base64,)

### Resource Registers

* Memory Base & Memory Limit:

•

![desc](data:None;base64,)

* I/O Base & I/O Limit:

•

![desc](data:None;base64,)

### Resource Sizing

例子-----写-1到PCI配置offset10h，如果返回0FFF00000h，那么这个设备这里需要(0FFFFFh+1)=100000h字节的内存资源



![desc](data:None;base64,)



![desc](data:None;base64,)

### Expansion ROM Base

1. 一些 PCI 设备，特别是那些用于 PC 架构中的附加卡的设备，需要本地 EPROM 来扩展 ROM。 2. 用途：用于设备特定的初始化（例如 VGA）和可能的系统引导功能（例如 SCSI）。 3、扩展ROM基地址：偏移30h BIT0表示扩展ROM使能/禁用，高21位（bits[31:11]）为扩展ROM基地址。 4. 执行：扩展ROM总是从ROM复制到RAM并从RAM执行。

* ROM 头

![desc](data:None;base64,)