# 第12章 PCI

PCI是外围部件互连（Peripheral Component Interconnect）的简称，它是一种描述如何将一个系统的外围部件用一种结构化和易于控制的方式连接在一起的标准。 PCI标准具体描述了系统部件的电气特性和它们应该具有的接口。下面我们看看 Linux系统内核是如何初始化系统的PCI总线和PCI设备的。

## 12.1 PCI 系统

图12-1是一个PCI总线系统的逻辑示意图。 PCI总线和PCI-PCI桥将系统的各部件连接在一起，其中CPU和Video设备连接到PCI总线0，也就是主PCI总线。PCI-PCI桥一个特殊的PCI设备，它将主PCI总线和从PCI总线，也就是PCI总线1，连接在一起。PCI总线1也称为PCI-PCI桥的下游，而PCI总线0称为PCI-PCI桥的上游。连接到从PCI总线上的是系统的SCSI设备和以太网设备。在物理上， PCI-PCI桥、从PCI总线和连接到从PCI总线上的这两个设备将有可能做在一块PCI卡上。PCI-ISA桥用于连接比较老的ISA设备。

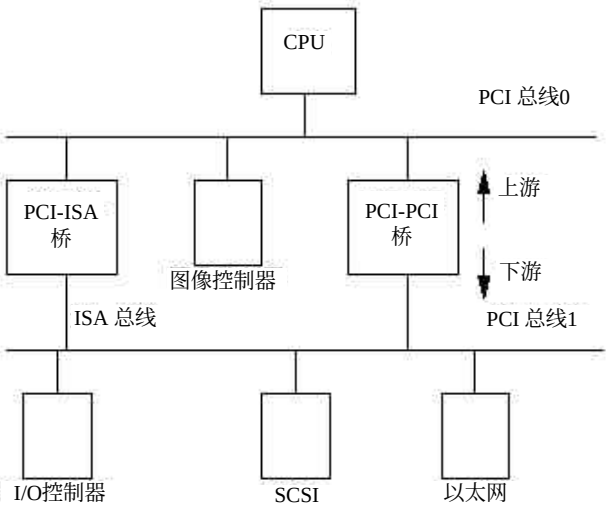


图12-1 PCI系统示意图

## 12.2 PCI地址空间

CPU和各种PCI设备都需要存取系统内存。设备驱动程序需要使用系统内存控制 PCI设备，并且可以通过系统内存在PCI设备之间传递信息。一般情况下，共享内存中包括设备的控制及状态寄存器。驱动程序可以利用这些寄存器来控制和读取设备的状态。例如， PCI SCSI设备驱动程序可以读取设备的状态寄存器，以便了解PCI SCSI设备是否已经准备好了向PCI SCSI磁盘

写入信息；或者，驱动程序可以在设备打开以后，向控制寄存器中写入命令启动设备。

CPU的系统内存可以用来作为这样的共享内存。但如果使用 CPU的系统内存，那么每当PCI 设备存取内存时，CPU都要停止以等待PCI设备完成存取操作。并且，一般一次只能有一个系统部件存取内存。这样，系统的性能将大大地降低。如果让系统的外围设备无限制地存取系统的主内存也十分的危险，因为一个失去控制的设备有可能使得系统瘫痪。

所以，系统的外围设备一般自己带有内存。CPU可以存取这些外围设备的内存，但外围设备只能通过DMA(Direct Memory Access)方式存取系统的内存。ISA设备有两种的地址空间：ISA I/O (Input/Output) 和 ISA 内存。PCI设备则有三种地址空间：PCI I/O, PCI 内存 和 PCI 设置。CPU可以通过设备驱动程序存取PCI I/O和 PCI 内存地址空间，并且可以通过Linux系统内核中的PCI初始化程序存取 PCI设置地址空间。

## 12.3 PCI设置头

系统中每一个PCI设备，包括PCI-PCI桥，都包含一个存储在PCI地址空间中有关设置的数据结构，即PCI设置头。系统通过PCI设置头识别和控制设备。设置头在PCI地址空间的具体位置依据PCI的拓扑结构的不同而不同。例如，一个 PCI视频卡插入到主板的某一个 PCI插槽中，那么它的设置头将在地址空间中有一个固定的位置，但如果将 PCI视频卡插入到主板的另一个PCI插槽中，它将又有另外的一个位置。但无论 PCI设备的设置头在哪，系统都能找到它，并且可以通过设置头中的状态及设置寄存器来设置它。

一般情况下，主板上的每一个 PCI插槽都对应一个相对固定的PCI设置头的地址。例如，主板上的第一个插槽的设置头从地址空间位移 0开始，第二个插槽的设置头从位移256开始（所有的设置头都是256字节），以此类推。根据系统硬件的不同，在一个给定的PCI总线中，PCI设置程序可以试着检查所有可能的PCI设置头，同时只需简单地读取设置头中的某一个字段（通常是厂家标识符字段），就可以知道系统中有无PCI设备。例如，对于一个没有 PCI设备的PCI插槽，如果试图读取其设置头的厂家标识符字段，将会返回一个0xFFFFFFFF错误。

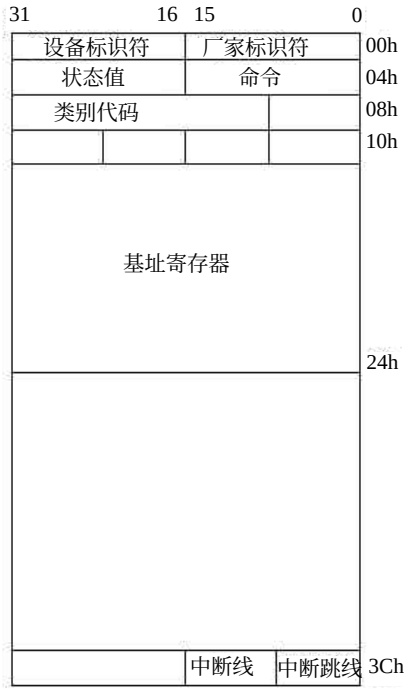


图12-2 PCI设置头示意图

图12-2是一个包括256个字节的设置头，其中的字段有：

- 厂家标识符：代表PCI设备厂家的一个唯一的数值。例如，Intel公司的厂家标识符是0x8086。
- 设备标识符：代表设备自身的一个唯一的数值。例如，Digital公司的21141型快速以太网卡的设备标识符是0x0009。
- 状态：此字段是设备的状态，其中的每一位的含义都有标准的规定。
- 命令：通过向此字段中写入命令，系统可以控制设备。
- 类别代码 :此字段代表设备的类型。例如，SCSI的类别代码是0x0100。
- 基址寄存器：这些寄存器用来决定和分配设备可以使用的 PCI I/O 和 PCI内存地址空间的

类型，数量和位置。

- 中断跳线：代表PCI卡上的4根中断跳线。通过这4根跳线，可以在PCI卡和PCI总线之间传递中断。4根跳线的标准标志为A、B、C、D。中断跳线字段描述PCI设备使用了哪一个中断跳线。一般情况下，对于某一个设备来说，它是由硬件决定的。中断处理程序可以通过此字段管理设备的中断。
- 中断线：此字段用来在PCI初始化程序、设备驱动程序以及Linux系统的中断处理程序之间传递中断句柄。此字段的数值对于设备来说没有任何意义。但它可以使得中断处理程序在Linux操作系统内正确地把PCI设备的一个中断送到正确的设备驱动程序的中断处理过程中。

## 12.4 PCI I/O 和 PCI 内存地址

PCI设备和它们运行在Linux系统内核上的设备驱动程序之间进行通信的地址空间有两种：PCI I/O 和 PCI 内存。例如，DEC公司的21141快速以太网设备把它的内部寄存器映射到 PCI I/O地址空间。Linux系统中的设备驱动程序通过读写这些寄存器来控制 PCI设备。而视频驱动程序一般使用较大的PCI内存地址空间来保存视频信息。

在PCI系统建立之前，或者在使用 PCI设置头的命令字段打开设备对地址空间的存取之前，任何程序都不能存取这些地址空间。应该注意的是，只有 PCI设置程序才可以读写包含PCI设置头的PCI设置地址，而Linux系统的设备驱动程序只能读写 PCI I/O 和 PCI 内存地址。

## 12.5 PCI-ISA桥

PCI-ISA桥通过将对PCI I/O 和 PCI 内存地址空间的访问转换成对ISA I/O 和 ISA内存地址空间的访问来兼容对以前的ISA设备的支持。系统保留低端的PCI I/O 和 PCI 内存地址空间提供给ISA外围设备使用，同时使用一个 PCI-ISA桥来把对此内存区域的访问转换成对 ISA的访问。

## 12.6 PCI-PCI 桥

PCI-PCI 桥是一种特殊的PCI设备，它把系统中的多个PCI总线集成为一个。单个PCI总线支持的设备数目有一定的限制。使用PCI-PCI 桥可以使系统支持更多的PCI设备。

PCI-PCI 桥只传送其下游需要的读写内存地址。例如，在PCI总线逻辑示意图中，PCI-PCI桥只传递从PCI总线0到PCI总线1中SCSI设备和快速以太网设备拥有的读写地址，所有的其他PCI I/O 和 PCI 内存地址都将被忽略。这样就防止了系统中不必要的地址传播。为了达到此目的，PCI-PCI桥必须设置一个它要传递的PCI I/O 和 PCI内存地址的起点和大小限制。一旦系统中的PCI-PCI桥设置完毕，那么Linux系统的设备驱动程序只通过设置的窗口来访问 PCI I/O 和PCI内存地址，PCI-PCI桥本身对驱动程序来说则是透明的。这对 PCI设备驱动程序的编写者来说十分重要。这也使得Linux系统对PCI-PCI桥的设置更为复杂。

因为PCI初始化程序需要寻找那些不在主 PCI总线上的设备，所以PCI-PCI桥必须可以判断是否需要从主PCI总线向从PCI总线传递设置环路。一个环路就是 PCI总线上的地址。 PCI中规定了两种PCI设置地址的格式：类型0和类型1，如图12-3所示：

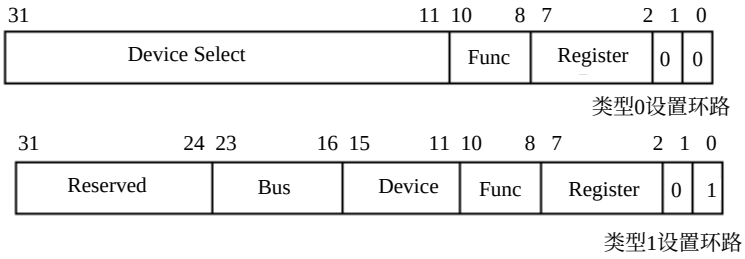


图12-3 PCI地址格式

在类型0设置环路中不包括总线值，所以它可以被PCI总线上的所有设备视为设置地址。类型0设置环路的31位到11位被视为设备选择字段。你可以使字段的每一位都代表一个不同的设备，那么第11位将会是插槽0中的PCI设备，第12位是插槽1中的PCI设备，以此类推。也可以把设备的插槽号直接地写入字段中。系统中到底使用哪一种机制取决于系统的 PCI内存控制器。

类型1设置环路包括一个PCI总线值，所以这种总线环路只有对PCI-PCI桥才有效。PCI-PCI桥根据自己的设置决定是否将类型1的设置环路传送到下游。每一个PCI-PCI桥都有一个主总线接口号和一个从总线接口号。主总线是靠近 CPU的总线，从总线是远离 CPU的总线。每一个PCI-PCI桥还有一个下属总线数目，也就是PCI-PCI桥下游的最多可以有的总线数。当一个PCI-PCI桥检测到一个类型1的设置环路时，它将按以下的方式之一来处理：

- 如果设置环路中的总线号不在 PCI-PCI桥的从总线号和从总线的下属总线号之间， PCI-PCI桥将忽略此设置环路。
- 如果设置环路中的总线号和PCI-PCI桥的从总线号相符， PCI-PCI桥将会把类型1的设置环路转化成为类型0的设置环路。
- 如果设置环路中的总线号大于 PCI-PCI桥的从总线号，但小于等于从总线的下属总线号，PCI-PCI桥将会把此设置环路不加任何改变地传送到从总线的接口中。

## 12.7 PCI初始化

Linux系统中PCI设备的初始化程序分为以下三个逻辑部分：

- PCI 设备驱动程序

这是一个伪设备驱动程序，它从总线 0开始搜索整个PCI系统，并且定位系统的所有 PCI设备和总线桥。它创建一个描述系统拓扑结构的数据结构链表。同时，它也将可以找到的所有的总线桥编号。

- PCI BIOS

此软件层提供在bib-pci-bios-specification中规定的各种函数。

- PCI Fixup

不同系统的Fixup程序用来处理不同系统中PCI初始化后的遗留问题。

### 12.7.1 Linux系统内核有关PCI的数据结构

当Linux系统内核初始化PCI系统时，它将创建一些代表系统中实际 PCI拓扑结构的数据结构。图12-4所示的数据结构关系即反映了本章开始时的PCI系统的拓扑结构。

每个数据结构pci_dev都描述一个PCI设备，包括PCI-PCI桥。每个pci_bus结构则描述一个PCI总线。这是一个树型的PCI总线结构，每一个PCI总线下面带有多个PCI设备。因为除了主

PCI总线以外的其他PCI总线只能由PCI-PCI桥连接，所以每个描述从PCI总线的pci_bus结构中都包括一个指向PCI-PCI桥的指针，而此PCI-PCI桥是此从PCI总线的父亲PCI总线的孩子。

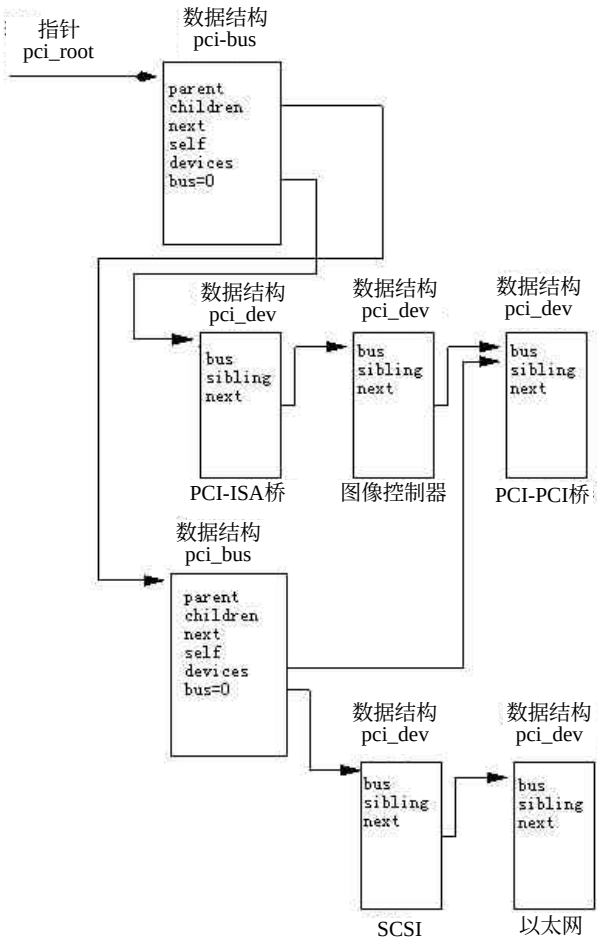


图12-4 PCI结构示意图

系统中所有的PCI设备的pci_dev结构组成了一个队列，pci_devices指针指向此队列。Linux系统内核使用此队列快速搜索系统的PCI设备。

### 12.7.2 PCI 设备驱动程序

这里的PCI设备驱动程序不是真正的设备驱动程序，而只是操作系统在系统初始化时调用的一个函数。PCI初始化程序必须搜索系统中所有的PCI总线，以便定位系统中全部的PCI设备，包括PCI - PCI桥。

PCI初始化程序通过PCI BIOS程序确定当前正在搜索的PCI总线中的每一个插槽是否被占用。PCI初始化程序从PCI总线0开始扫描，它首先试着读取每一个PCI插槽中可能存在的PCI设备的厂家标识符和设备标识符。如果对某一个插槽读取成功，那么说明此PCI插槽已经被占用，PCI设备驱动程序将创建一个描述此设备的数据结构pci_dev，并且将它插入到已经存在的所有PCI设备的链表。（pci_devices指针指向此链表）

PCI初始化程序如果发现设备是一个 PCI-PCI桥，那么将创建一个pci_bus结构，然后将它插入到pci_bus结构树中。PCI初始化程序通过类别代码 0x060400来确定此设备是否为PCI-PCI桥。如果是PCI-PCI桥，系统内核则设置PCI-PCI桥下游的PCI总线。如果在其中再发现PCI-PCI桥,则继续设置PCI-PCI桥下游的PCI总线。

只有知道了以下内容，PCI-PCI桥才能传递PCI I/O, PCI 内存或 PCI设置地址：

- 主总线号
- 从总线号
- 下属总线号：PCI-PCI桥的下游的所有总线的最大的总线号。
- PCI I/O 和 PCI内存窗口：PCI-PCI桥的下游地址的PCI I/O 和 PCI内存地址空间的窗口起点和大小。

问题是当你希望设置某一个PCI-PCI桥时，你并不知道该PCI-PCI桥的下属的总线号。也不知道PCI-PCI桥的下游是否还有 PCI-PCI桥，如果有的话，它们的总线号是多少。 Linux系统内核使用深度搜索算法搜索每一个总线号。每当找到一个 PCI-PCI桥和指定它的从总线号时，同时赋给该PCI-PCI桥一个临时的下属号 —— 0xFF，然后搜索和指定该 PCI-PCI桥下游所有 PCI-PCI桥的总线号。下面举例说明如何设置总线号：

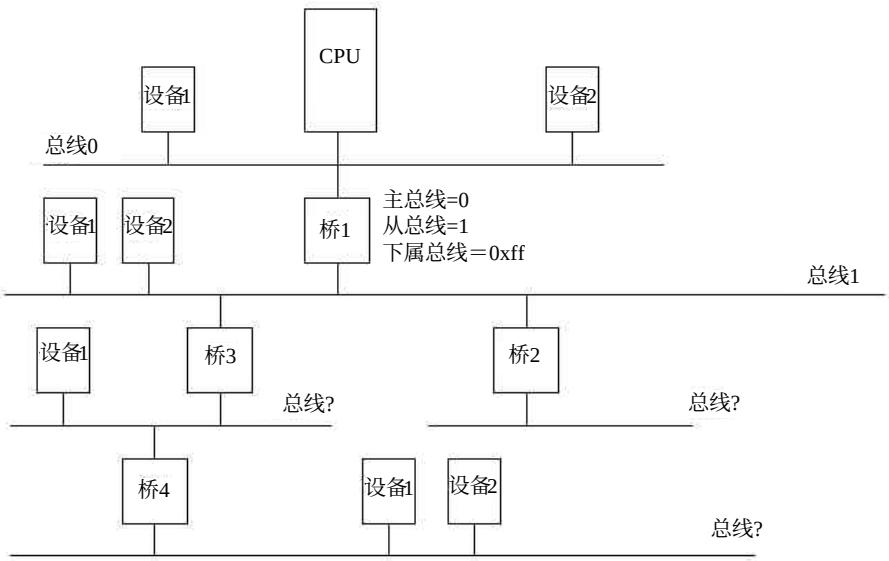


图12-5 PCI-PCI 设置示意图一

第一步：

如图12-5所示，初始化程序首先找到的是PCI - PCI桥1。它下游的总线将被设置为1，同时指定下属总线号为0xFF。这意味着所有类型1设置环路中凡是PCI总线号大于1的都将通过PCI-PCI桥1进入 PCI总线1。如果设置环路中的总线号是1，那么类型1设置环路将会被转换成类型0设置环路，但其他总线号的设置环路的类型则保持不变。这也正是 Linux系统的PCI初始化程序通过和扫描PCI总线1时所需要做的。

第二步：

Linux系统的PCI初始化程序使用的是深度搜索算法，所以它继续搜索 PCI总线1。如图12 - 6所示，初始化程序找到了 PCI-PCI桥2。因为PCI-PCI桥2以下再没有PCI - PCI桥了，所以PCI-

PCI桥2的下属总线号设置为2，同指定给PCI-PCI桥2的从总线号相同。

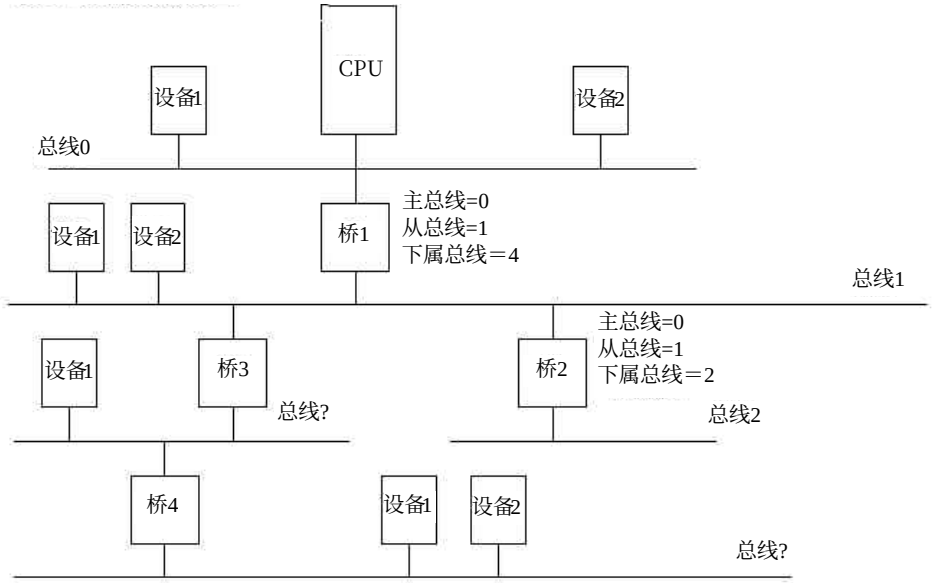


图12-6 PCI-PCI设置示意图二

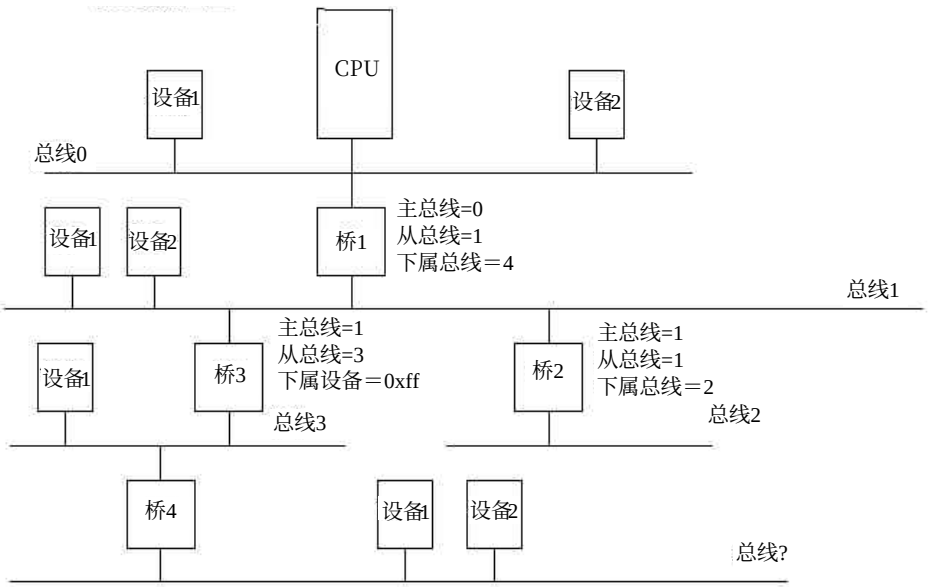


图12-7 PCI-PCI设置示意图三

第三步

PCI初始化程序接着返回重新去搜索PCI总线1，这样找到了另一个PCI-PCI桥3，如图12-7所示。初始化程序将它的主总线号指定为 1，从总线号指定为 3，同时指定下属的总线号为0xFF。现在，类型1设置环路中带有总线号1、2、3的都将可以正确地传送到相应的PCI总线。

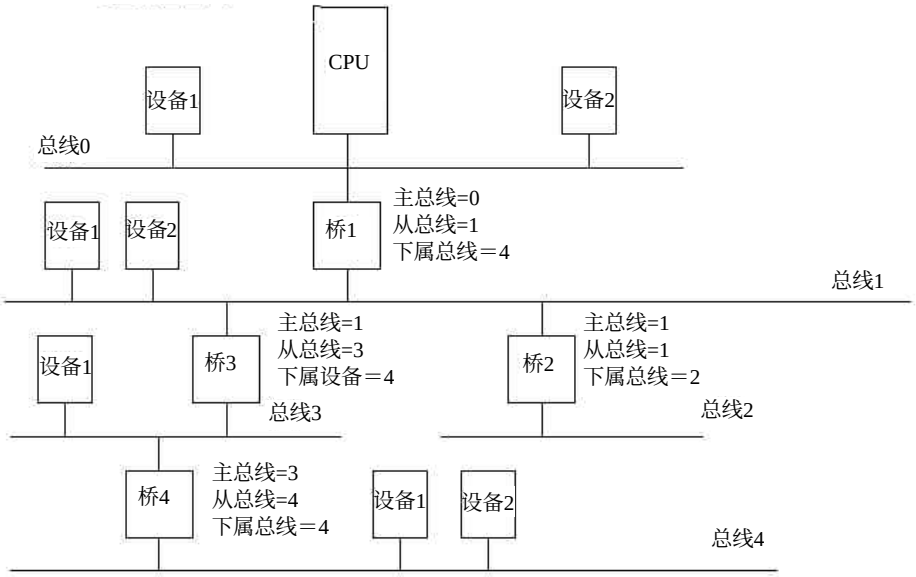


图12-8 PCI-PCI设置示意图四

第四步：

Linux系统的PCI初始化程序开始搜索PCI总线3，以及PCI-PCI桥3的下游。如图12-8所示，PCI总线3上有一个PCI-PCI桥4，所以初始化程序指定它的主总线号为3，从总线号为4。因为它是此分支上的最后一个总线桥，所以初始化程序指定它的从属总线号为 4。这样此系统中的所有PCI总线就都指定了总线号。

### 12.7.3 PCI BIOS 函数

PCI BIOS函数是一系列标准的跨平台子程序。它允许CPU控制对所有PCI地址空间的存取。只有Linux内核代码和设备驱动程序可以使用PCI BIOS函数。

### 12.7.4 PCI Fixup

对于基于Intel的系统来说，系统的BIOS已经在启动时比较全面地设置了 PCI系统。这样，Linux 系统就只需将这些设置映射到系统中了。