

## PCI 局部总线及其接口

## 目 录

第一章 前言.....	3
§1.1 简介.....	3
§1.2 PCI 总线的主要特点.....	4
§1.2.1 PCI 总线是一种局部总线.....	4
§1.2.2 PCI 总线是一种独立于处理器的同步总线.....	4
§1.2.3 高的传输速度.....	4
§1.2.4 具有热插拔能力.....	4
§1.2.5 PCI 总线是一种立足现在放眼未来的标准.....	5
第二章 信号定义.....	6
§2.1 信号类型定义.....	6
§2.2 引脚功能组.....	6
§2.2.1 系统引脚.....	7
§2.2.2 地址和数据引脚.....	8
§2.2.3 接口控制引脚.....	8
§2.2.4 仲裁引脚(只对总线主控).....	9
§2.2.5 错误反馈引脚(所有设备都要求有错误反馈引脚).....	9
§2.2.6 中断引脚.....	10
§2.2.7 高速缓存(cache)支持引脚(可选用).....	10
§2.2.8 64 位总线扩充引脚(集体可选用).....	10
§2.2.9 JTAG/边缘扫描引脚(任选).....	11
第三章 总线命令.....	12
§3.1 命令编码.....	12
§3.2 命令使用规则.....	14
第四章 PCI 协议的主要内容.....	15
§4.1 操作规则:.....	16
§4.1.1 何时信号稳定:.....	16
§4.1.2 控制信号:.....	17
§4.1.3 闭锁操作.....	19
§4.1.4 仲裁:.....	20
§4.1.5 奇偶校验:.....	20
§4.2 寻址.....	20
§4.3 总线传送.....	22
§4.3.1 读传送.....	23
§4.3.2 写传送.....	24
§4.3.3 传送终止.....	24
§4.4 仲裁.....	29
§4.5 仲裁放置(PARKING).....	31
§4.6 延迟.....	32
§4.6.1 PCI 上的延迟.....	32

§4.6.2 延迟指导原则.....	34
§4.7 快速背对背传送.....	35
§4.8 闭锁操作.....	37
§4.8.1 开始闭锁操作.....	39
§4.8.2 继续闭锁操作.....	40
§4.8.3 对锁定操作进行非闭锁操作.....	41
§4.8.4 完成闭锁操作.....	41
§4.8.5 对 LOCK# 和高速缓存器的连续回写的支持.....	42
§4.8.6 完整的总线锁定.....	43
§4.9 PCI 协议对 CACHE 的支持.....	43
§4.9.1 Cache 的作用.....	43
§4.9.2 Cache 的组织和访问.....	44
§4.9.3 PCI 协议下 Cache 的状态.....	47
§4.9.4 Cache 检查状态的转换关系: .....	48
§4.9.5 时序关系说明.....	49
§4.10 其它总线锁定.....	53
§4.10.1 设备选择.....	53
§4.10.2 特殊周期.....	54
§4.10.3 地址/数据分步.....	56
§4.10.4 中断应答.....	57
§4.10.5 错误功能.....	58
<b>第五章 配置空间.....</b>	<b>62</b>
§5.1 配置空间的组织.....	63
§5.2 补充说明: .....	68
§5.3 扩展 ROM 的组织.....	69
§5.4 INTEL X86, PC-AT 兼容的扩展 ROMS 的进一步说明.....	72
§5.5 配置周期操作: .....	74
§5.6 配置机制.....	76
§5.6.1 配置机制 1#.....	76
§5.6.2 配置机制 2#.....	77
§5.7 PCI BIOS 对配置空间的支持: .....	79

## 第一章 前言

### § 1.1 简介

微机总线是一些公共信号线的集合。机器内部各芯片之间，各插件板、各功能部件之间，微机与外部设备之间，大都通过某种总线传递和交换信息。

PCI (Peripheral Component Interconnect)总线是一种同步的、独立于处理器的、32 位或 64 位局部总线，其目的是在高集成度的外设控制器件、扩展板(add-in board)、和处理器/存储器系统之间提供一种内部连接机制。图 1.1 是一个典型 PCI 系统框图。

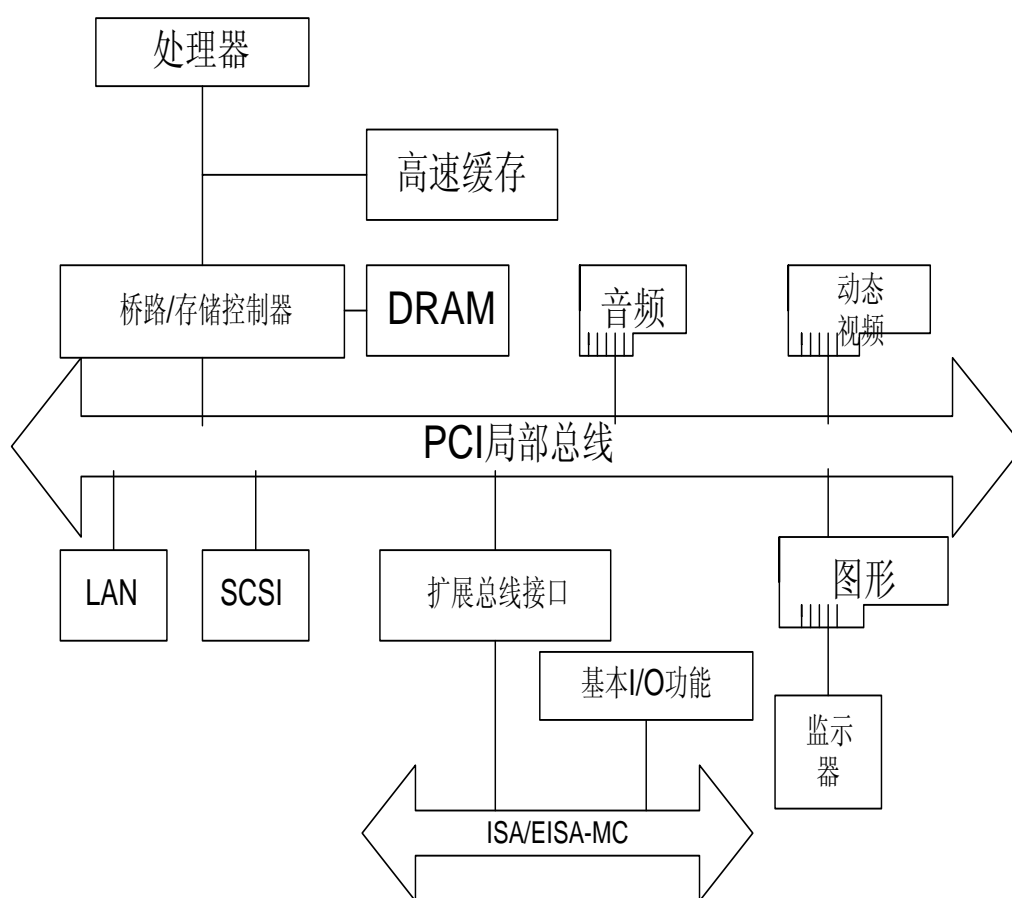


图 1.1 PCI 系统框图

在图 1.1 中，处理器/高速缓存/存储器子系统通过桥路与 PCI 总线相连接，该桥路提供一条低时间延迟的通道，通过它，处理器能直接操作任何映射到存储器或 I/O 空间的设备，它也提供一条高带宽通道，使 PCI 总线主控能直接操作主存储器。该桥路可以选择包括以下功能：数据缓存/停驻和 PCI 核心功能(即仲裁)。

## § 1.2 PCI 总线的主要特点

### § 1.2.1 PCI 总线是一种局部总线

局部总线就是 CPU 总线或称芯片总线，它将 CPU 芯片、存储器、外围接口器件等连接在一起，构成主系统板或某种 CPU 插件板，为主系统的各器件之间提供标准的信息接口及高速信息通道，并为 Cache、主存分系统和有关高速控制卡等服务。正是由于它将外部设备直接挂接到 CPU 而不是通过 ISA 或 EISA 总线，大大提高了系统的性能。

### § 1.2.2 PCI 总线是一种独立于处理器的同步总线

PCI 总线不受处理器类型的限制，任何设备只要其接口满足 PCI 总线规范，便可进行互连。通过桥路缓冲，独立于处理器的 PCI 总线可与处理器/存储器子系统同时工作。同时，它支持总线主控，将 DMA 功能和总线主控能力混在一起，具有主控能力的设备都可以获得总线控制权，进而对系统资源进行访问。PCI 总线是一种同步总线，时钟频率为 0~33MHz，和微处理器工作频率无关，这是它区别于 AT 总线的重要标志，也使其接口控制器的实现变得复杂。

### § 1.2.3 高的传输速度

PCI 总线数据宽度为 32 位，可扩展到 64 位，最高工作频率 33MHz，峰值吞吐率在 32 位时为 132Mb/s，64 位时为 264Mb/s。它支持快速背对背传输、猝发传输及 Cache 操作，具有主控能力的设备可以直接对主存进行操作。从而大大提高了总线的数据吞吐量，为外设的高速工作提供了保证。

### § 1.2.4 具有热插拔能力

PCI 总线规范规定了配置空间，配置空间定义了该设备的特征、功

能、传输能力、中断需求、及所需要的存储器空间和 I/O 空间。POST 程序会根据设备的要求自动为其分配存储器空间、I/O 空间。因而，基于 PCI 总线的设备都具有即插即用 (PNP-plug and play) 能力。

#### § 1.2.5 PCI 总线是一种立足现在放眼未来的标准

PCI 总线提供了 32/64 位数据线，满足了相当长一段时间内数据传输的要求，它提供了 5V，5V/3.3V，3.3V 电源标准，迈向绿色电脑标准。此外，独立于处理器的特点，也为 PCI 打开了广阔的应用领域，包括桌面电脑、笔记本电脑及服务器等各式机种。

## 第二章 信号定义

为处理数据、寻址、接口控制、仲裁及系统功能，PCI 接口要求作为目标的设备至少有 47 条引脚，作为总线主控的设备至少有 49 条引脚，图 2.1 为按功能分类的引脚，必要的引脚在左边，任选的引脚在右边。信号的方向说明是针对总线主控/目标组合设备而言的。

### § 2.1 信号类型定义

in input(输入)是一种只用于输入的标准信号。

out output(输出)是一种标准的有效驱动器。

t/s Tri-state(三态)是一种双向、三态输入/输出引脚，无效时是高阻态。

s/t/s Sustained Tri-state(持续三态)是一种每次由且只由一个单元拥有并驱动的低有效双向、三态信号。

o/d Open Drain(漏极开路)允许多器件共享，可作线或。

### § 2.2 引脚功能组

在 PCI 协议中，中央资源用来表示由主系统所支持的总线支持功能，特别是 PCI 所用的桥路和标准芯片集，这些功能包括中央仲裁；在复位期间驱动 REQ64#；在系统配置操作时产生有效的 IDSEL 信号给每个设备；反向解码；提拉电阻或称保持器。PCI 控制信号常常要求提拉电阻以保证在无单元有效地驱动总线时它们保持稳定值。这些信号包括：FRAME#，TRDY#，IRDY#，DEVSEL#，STOP#，PERR#，SERR#，在用到时也包括 LOCK#，REQ64#，ACK64#。点到点及共享的 32 位信号不要求提拉电阻，总线放置保证它们稳定。

64 位数据通道的扩展信号 AD[63..32]、C/BE[7..4]#和 PAR64 在接上时，也要求提拉电阻，如果它们未连接上，器件必须自己处理这些浮空输入。

在图 2.1 中, PCI 引脚定义按功能组组织。在信号名之后的一个“#”标志说明该信号是低电平有效的, 当无“#”标志时, 信号是高电平有效的。每条引脚上的信号类型跟在信号名之后。

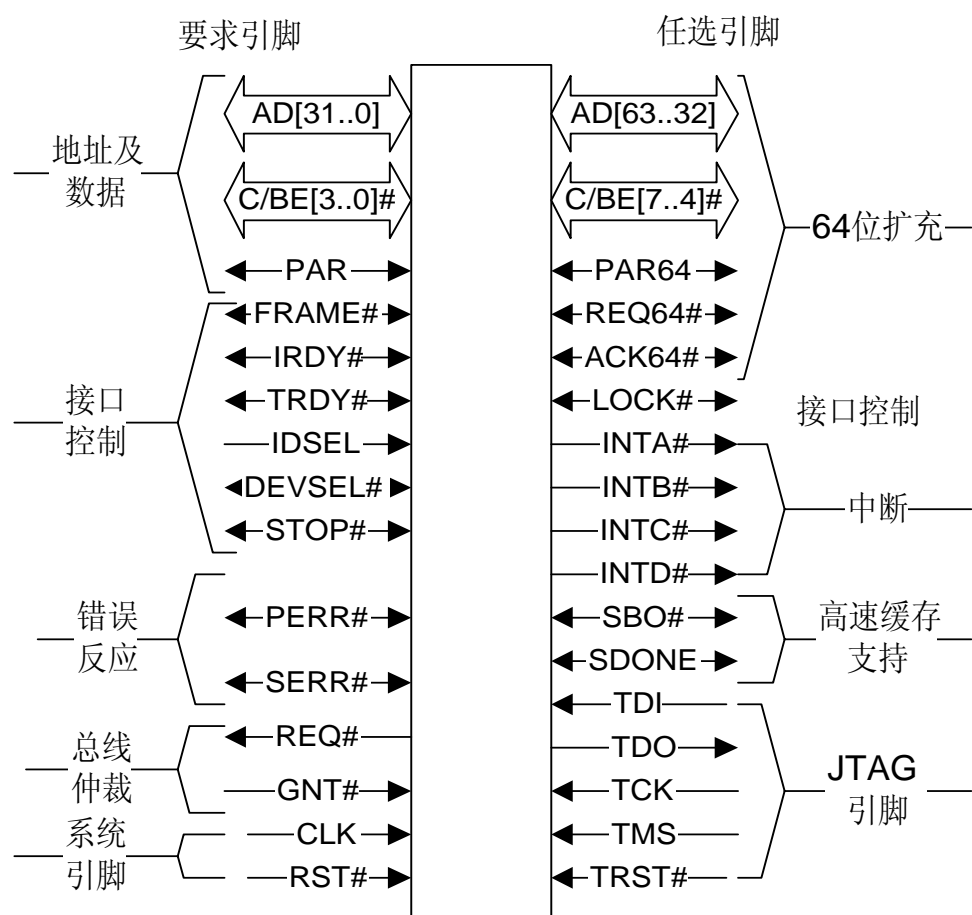


图 2.1 PCI 引脚列表

### § 2.2.1 系统引脚

CLK— in 系统时钟, 为所有 PCI 上的传输及总线仲裁提供时序。除 RST#及四个中断引脚外, 其它的 PCI 信号都在 CLK 信号的上升沿采样。CLK 最小频率是直流 (0Hz), 最高可达 33Mhz。

RST#— in 异步复位, 用于使 PCI 确定的寄存器、配置寄存器、顺序发生器和信号置于一个固定的状态。无论何时, 在 RST#有效期间, 所有 PCI 信号必须被驱动到它们的起始状态。通常情况下, 这意味这它们必须是三态(高阻态)。SERR#被浮空。REQ#和 GNT#都必须是三态。为防止



AD、C/BE#和 PAR 信号在复位期间被浮空，中央设备可以在复位期间驱动这些线，但是只能驱动到逻辑低——它们不可以驱动到高。REQ64#在复位结束时有意义。这些基本都是由中央设备驱动的。

### § 2.2.2 地址和数据引脚

一个 PCI 总线传输由一个地址段及相随的一个或多个数据段组成。

AD[31..00]— t/s 地址数据复用引脚。FRAME#开始变为有效的那个时钟周期为地址段。AD[31..00]包含有一个物理地址。对于配置空间和存储器空间，这是一个双字地址，对于 I/O 空间，这是一个字节地址。在数据段，AD[7..0]包含最低字节数据，而 AD[31..24]包含最高字节数据。

C/BE[3..0]#— t/s 总线命令和字节允许复用引脚。在地址段，C[3..0]上定义了总线命令。在数据段期间，BE[3..0]#用作字节允许，BE0#对应于最低字节而 BE3#对应于最高字节。

PAR— t/s AD[31..00] 和 C/BE[3..0]#上的数据偶校验。PAR 与 AD[31..00]有相同的时序，但延迟一个时钟，在地址段后一个时钟，PAR 稳定并有效；对于数据段，在写传输中，PAR 在 IRDY#有效后一个时钟稳定并有效，而在读传输中，PAR 在 TRDY#有效后一个时钟稳定并有效。一旦 PAR 有效，它必须保持有效直到当前数据段完成后一个时钟。在地址段和写数据段，总线主控驱动 PAR，在读数据段，目标驱动 PAR。

### § 2.2.3 接口控制引脚

FRAME#— s/t/s 周期构成(cycle Frame)。由当前总线主控驱动，以说明一个操作的开始和延续。FRAME#有效，说明总线传输开始。当 FRAME#维持有效时，说明总线传输继续进行，当 FRAME#是无效状态时(高电平)，说明传送的最后一个字节正在进行。

IRDY#— s/t/s 启动者准备好(Initiator Ready)。说明传输的启动者完成当前数据传输的能力。在读操作中，IRDY#有效说明总线主控已准备好接收数据。在写操作中，它说明 AD[31..00]上已有有效数据。在 IRDY#和 TRDY#都有效的时钟期间完成数据传输。在 IRDY#和 TRDY#都有效之前，需要插入等待状态。

TRDY#-- s/t/s 目标准备好(Target Ready)。说明传输的目标完成当前数据传输的能力,在写操作中,TRDY#有效说明目标已准备好接收数据。在读操作中,它说明 AD[31..00]上已有有效数据。在 IRDY#和 TRDY#都有效的时钟期间完成数据传输。在 IRDY#和 TRDY#都有效之前,需插入等待状态。

STOP#-- s/t/s 停止。说明当前的目标要求总线主控停止当前传输。

LOCK#-- s/t/s 锁定信号。说明一种需要多个传输完成的原子级操作。当 LOCK#有效时,非独占传输可以对当前非锁定的地址进行。当只有一个总线主控拥有 LOCK#时,不同的单元也可以使用 PCI 总线。对 LOCK#的控制必须由 LOCK#的拥有协议与 GNT#来完成。

IDSEL-- in 初始化设备选择(Initialization Device Select)。在配置空间读写操作中,用作片选。

DEVSEL#-- s/t/s 设备选择。当有效驱动时,说明驱动它的设备已将其地址解码为当前操作的目标。作为输入信号,DEVSEL#说明了总线上是否有目标被选中。

#### § 2.2.4 仲裁引脚(只对总线主控)

REQ#-- t/s 申请。向仲裁器说明该单元想使用总线。

GNT#-- t/s 允许。仲裁器向申请单元说明其对总线的操作已被允许。

#### § 2.2.5 错误反馈引脚(所有设备都要求有错误反馈引脚)

PERR#-- s/t/s 奇偶校验错误(Parity Error)。该引脚只用于反馈在除特殊周期外的其它传送过程中的数据奇偶校验错误。PERR#维持三态,并在检测到传送数据中的奇偶错误后,在数据结束后两个时钟,由接收数据的单元驱动 PERR#有效,并至少持续一个时钟周期。只有发出 DEVSEL#的单元才能发出 PERR#。

SERR#-- o/d 系统错误。用于反馈地址奇偶错误、特殊周期命令中的数据奇偶错误和将引起重大事故的其它灾难性的系统错误。如果一个单元不想产生不可屏蔽中断(NMI)则可用 SERR#反馈给系统。支持 SERR#的单元在采样到 SERR#有效时。就向操作系统报告系统错误。

### § 2.2.6 中断引脚

中断引脚是“电平触发”，低有效，用漏极开路输出驱动器驱动，与时钟异步。PCI 为每一个单一功能设备定义一根中断线。

INTA#--o/d 中断 A，用于单一功能设备请求一次中断。

INTB#--o/d 中断 B，用于多功能设备请求一次中断。

INTC#--o/d 中断 C，用于多功能设备请求一次中断。

INTD#--o/d 中断 D，用于多功能设备请求一次中断。

多功能设备的任何一种功能都能连到任何一条中断线上。中断引脚寄存器决定该功能用哪一条中断线去请求中断。如果一个设备只用了一条中断线，则这条中断线就被称为 INTA#，如果该设备用了两条中断线，那么它们就被称为 INTA#和 INTB#，依次类推。对于多功能设备，可以是所有功能用一条中断线，也可以是每种功能有自己的一条中断线(最多 4 种功能)，还可以是上述两种情况的综合。一个单功能设备不能用一条以上的中断线去请求中断。

系统商可以将 PCI 连接器上任何组合方式将中断线连接到中断控制器上，可以是线或方式，也可以是程序控制的电子开关切换方式，或是别的任何组合方法。这意味着设备驱动器对共用中断不能作任何假定。所有设备驱动器必须能与任何别的逻辑设备共用中断，包括同一多功能封装中不同设备之间的情况。

### § 2.2.7 高速缓存(cache)支持引脚(可选用)

一个能高速缓存的 PCI 存储器必须利用这两条高速缓存支持引脚作为输入，以支持写通(write-through)和回写(write-back)。如果可高速缓存的存储器是位于 PCI 上，则连接回写高速缓存到 PCI 的桥路必须利用两条引脚，且作为输出。连接写通高速缓存的桥路可以只使用一条引脚 SDONE。

SB0#-- in/out 监视补偿(Snoop Backoff)。当其有效时，说明对某条变化线的一次命中。当 SB0#无效而 SDONE 有效时，说明了一次“干净”的监视结果。

SDONE-- in/out 监视进行(Snoop Done)。表明对当前操作的监视状态。当其无效时，说明监视结果仍未定；当有效时，说明监视已有结果。

### § 2.2.8 64 位总线扩充引脚(集体可选用)

AD[63..32]--t/s地址数据复用引脚提供 32 个附加位。在一个地址段(用

DAC 指令且 REQ64#已有效), 传送 64 位地址的高 32 位; 如无高 32 位地址, 这些引脚就被保留, 其上数据是稳定的, 但值是不确定的。在数据段期间, 当 REQ64#和 ACK64#都有效时, 传送 64 位数据中的高 32 位。

C/BE[7..4]# --t/s 总线命令和字节允许复用引脚。在一个地址段(用 DAC 指令且 REQ64#已有效), 在 C/BE[7..4]上传送有效总线命令; 否则, 这些引脚被保留且其值不定。在数据段期间, 当 REQ64#和 ACK64#都有效时, C/BE[7..4]#是字节允许, 说明那些字节通道上含有有意义的数据。C/BE4#相应于第四字节而 C/BE7#相应于第七字节。

REQ64#-- /t/s 请求 64 位传输。当其被当前总线主控有效地驱动时, 说明该总线主控想作 64 位的传送。REQ64#与 FRAME#有相同的时序。在复位结束后, 若 REQ64#有效, 该设备就已连到 64 位通道上, 否则, 就没有。

ACK64#-- s/t/s 应答 64 位传送。在当前操作所寻址的目标有效驱动该信号时, 说明该目标能够进行 64 位传送, ACK64#和 DEVSEL#有相同的时序。

PAR64--t/s 高双字偶校验。是 AD[63..32]和 C/BE[7..4]#的偶校验位。当 REQ64#有效且 C/BE[7..4]#上有 DAC 命令时, 第一个地址段后一个时钟周期 PAR64 有效。DAC 命令的第二地址段后的哪一个时钟周期, PAR64 也有效。对于数据段, 当 REQ64#及 ACK64#均有效时, 读操作中, TRDY#有效后, PAR64 是稳定且有效的, 写操作中, IRDY#有效后, PAR64 是稳定且有效的, 一旦 PAR64 有效, 必须保持有效直到数据段完成后一个时钟(PAR64 时序与 AD[63..32]相同但延迟一个时钟)。在地址段和写数据段, 总线主控驱动 PAR64, 在读数据段, 则由目标驱动 PAR64。

在总线主控和目标之间, 64 位传送是动态协调的(每个地址段一次)。而且, 只有存储器命令支持 64 位传送。总线主控使 REQ64#有效, 目标则通过使 ACK64#加以应答。REQ64#和 ACK64#是外部上拉的, 以保证 64 位和 32 位单元混用。一旦 64 位传送建立, 就一直保持到这次传送结束。

## § 2.2.9 JTAG/边缘扫描引脚(任选)

IEEE 标准 1149.1, 测试存取口及边缘扫描结构(Test Access Port and Boundary scan Architecture)。在一个设备中包含测试存取口(TAP), 使得在测试该设备或装有该设备的板时可以使用边缘扫描。TAP 由 4 至 5 条引脚组成, 它们用于和 PCI 设备中的 TAP 控制器作串行接口。

TCK-- in 测试时钟。在 TAP 操作期间记录状态信息和测试设备的输入

输出数据。

TDI-- in 测试数据输入。用来在 TAP 操作期间将测试数据和测试指令串行移入设备中。

TDO-- out 测试输出。用来在 TAP 操作期间将测试数据和测试指令串行移出设备中。

TMS-- in 测试模式选择。用来控制设备中的 TAP 控制器的状态。

TRST#--in 测试复位。这个可选引脚给 TAP 控制器提供了一个异步初始化。

PCI 规范支持带有包含 1149.1 边缘扫描信号的联接器的扩展板。扩展板上的设备要连接到主板上的 1149.1 环。为了不中断设备间的串行链码，不支持 IEEE1149.1 标准接口的扩展板在硬件上要将其 TDI 引脚接到其 TDO 引脚上。

### 第三章 总线命令

#### § 3.1 命令编码

总线命令对目标说明当前总线主控正在请求的传输类型。总线命令在地址段期间 C/BE[3..0]#上，其编码如表 3.1 所示

表 3.1 PCI 总线命令

C/BE[3..0]#	总线命令
0000	中断应答(interrupt Acknowledge)命令是一个寻址系统中断控制器的隐性读。在地址段，地址位逻辑上无关紧要，字节允许说明返回矢量的大小。
0001	特殊周期(Special Cycle)命令提供一种简单的、广播式的信息传播机制。
0010	I/O 读命令用于从一个映射于 I/O 地址空间的单元中读取数据。AD[31..00]提供某个字节的地址，全部 32 位都必须解码。字节允许说明传送的大小且必须与字节地址段一致。
0011	I/O 写命令用于写数据到一个映射于 I/O 地址空间的单元中去。全部 32 位都必须解码。字节允许说明传送的大



	小且必须与字节地址段一致。
0100 0101 1000 1001	保留命令。这些命令编码留作将来使用。PCI 目标不能将保留编码与其它编码混同。目标不能对保留编码作出应答。如果接口上用了保留编码，操作时总线主控将用总线主控失败终止此操作。
0110	存储器读命令用于从一个映射于存储器地址空间的单元中读取数据。只要目标保证预取没有副作用，便可以用该命令预取。进而，目标必须保证在 PCI 传送完成之后，保留在暂时缓冲区中的数据的一致性(包括顺序)。在任何同步事件通过该通道之前，这种缓存必须初始化(清空)。
0111	存储器写命令用于写数据到一个映射于存储器地址空间的单元中去。当目标返回“准备好”时，它表明能正确接收对象数据。实现这一点，或者以一种完全同步的方式实现这条命令。或者保证在任何同步事件(更新 I/O 状态寄存器或存储器标志)通过这种操作通道之前，任何软件透明中继缓存器都被刷新。这表明总线主控在执行这条命令之后可立即处理同步事件。
1010	配置读命令用于配置空间的读操作。当 IDSEL 信号有效且 AD[1..0]是 00 时，就选中了一个单元。在配置周期的地址段期间，AD[7..2]寻址每个设备配置空间 64 个双字寄存器之一，字节允许寻址每个双字中的字节，且 AD[31..11]上的逻辑是不必关心的，AD[10..8]说明寻址多功能单元的哪个设备。
1011	配置写命令用于传送数据到配置空间。当 IDSEL 信号有效且 AD[1..0]是 00 时，就选中了一个单元。在配置周期的地址段期间，AD[7..2]寻址每个设备配置空间 64 个双字寄存器之一，字节允许寻址每个双字中的字节，且 AD[31..11]上的逻辑是不必关心的，AD[10..8]说明寻址多功能单元的哪个设备。
1100	存储器重复读(memory Read multiple)命令除说明总线主控在解除连接前打算读取一条以上高速缓存线的数据外，其它与存储器读命令相同。只要 FRAME#有效，存储控制器就将维持其流水线存储器请求。该命令为大块数据传送而设计的，此时，如果一个软件透明缓冲器能用作暂时存储，超前顺序读取额外的一条高速缓存线上的数据可以

	提高存储器系统的性能。
1101	双地址周期 (DAC) 命令用于传送 64 位地址给支持 64 位寻址的设备。只支持 32 位寻址的目标把该命令当作保留对待而对当前传送不作任何反应。

续表 3.1 PCI 总线命令

C/BE[3..0]#	总线命令
1110	高速缓存线存储器读(memory Read Line)命令说明总线主控打算完成两个以上的 32 位数据段外, 其余与存储器读类似。该命令为大块数据传送而设计。作为对传送申请的响应, 一次读取一条行缓存线范围内的所有数据, 而不是单个存储器读周期, 从而提高存储器系统的性能。和存储器读命令一样, 在任何同步事件提供这种操作路径之前, 预取缓存器必须初始化为空。
1111	高速缓存存储器写(Memory Write and Invalidate)命令除保证最小传输为一个完整的高速缓存线数据外, 其余与存储器写命令相同。如果总线主控想在一次 PCI 传送中, 写完被寻址的高速缓存线上的全部字节, 就可用该命令说明。如果总线主控想将下一条高速缓存线也全部传送完, 那么传送就可以跨越高速缓存线边界。该命令要求在总线主控中有一个配置寄存器说明高速缓存线的范围, 该命令可以通过不要求有效回写周期而将一条“脏”线初始化到回写高速缓存, 从而缩短操作时间而使存储器性能优化。

### § 3.2 命令使用规则

所有 PCI 设备对配置读/写命令而言, 都是目标, 都必须作出应答, 对别的命令则有选择余地。I/O 读/写命令是可选的, 命令执行规则保证 I/O 读写命令的执行。有重定位功能或寄存器的目标要求, 能通过配置寄存器而映射到存储器空间, 并响应基本的存储器读/写命令, 这就为没有 I/O 空间的设备提供了一种选择。当这种映射实现时, 无论设备映射到 I/O 空间还是存储器空间, 命令执行规则都由系统设计者来保证。对一个被映射设备的存储器读和写都构成“存储器映射 I/O 口”。

总线主控可以根据需要使用任选命令。目标也可根据需要而选用指令，但如果它选用了基本存储器命令，它就必须支持所有存储器命令，包括高速缓存存储器写命令，高速缓存线存储器读命令和存储器重复读命令。如果不能全部使用，这些性能已优化的命令必须转化为基本存储器命令。例如，一个目标可以不用高速缓存线存储器读命令，但它必须接收这种申请并把它当成存储器读命令。同样，一个目标可以不用高速缓存存储器写命令，但它必须接收这种申请并把它当成存储器写命令。

对于进/出系统存储器的块数据传输，对能支持高速缓存存储器写和高速缓存线存储器读的总线主控，建议采样这两条命令。如果由于某些原因，总线主控不能使用性能已优化的命令，那么就用存储器读和存储器写命令。

对于使用存储器读命令的总线主控，对所有命令都可作任意长度的操作，但最优方法如下所列。只有高速缓存存储器写命令要求实现高速缓存线范围寄存器，建议存储器读命令也用它。所有情况下，桥路保证任何隐含数据的正确性。

使用高速缓存线范围寄存器时的最优方法：

存储器读命令：当猝发传送少于半条高速缓存线数据时使用。

高速缓存线存储器读命令：当猝发传送半条到三条高速缓存线数据时使用。

存储器重复读命令：当猝发传送三条以上高速缓存线数据时使用。

未用高速缓存线范围寄存器时的最优方法：

存储器读命令：当猝发传送两个或更少的数据时使用。

高速缓存线存储器读命令：当猝发传送 3 到 12 个数据时使用。

存储器重复读命令：做长猝发时使用。

## 第四章 PCI 协议的主要内容

PCI 总线的基本传送机制是猝发传送。一个猝发传送由一个地址段和一个或多个数据段组成。它要求目标和总线主控都必须能理解隐含寻址，PCI 支持对存储器和 I/O 地址空间的猝发。主桥路在无副作用的情况下可以将多个存储器写操作合并为一个猝发传送。设备通过设置基本地址寄存器中的预取位来表明没有副作用。桥路可以通过初始化期间配置软件提供的地址范围来判断哪里允许合并、哪里不允许。当接下来的



是一个不可预取的读或写时，合并数据到缓存器必须停止(且缓存器被刷新)。如果在可预取范围，跟在上述两事件之后的写传送可以与后续的写合并，但不合并前面的数据。

主桥路通常可以将处理器产生的连续双字按原顺序组合到一个猝发传送中，例如，当处理器写的顺序是双字 0、双字 2、双字 3 时，桥路可以产生一次猝发传送。该 PCI 猝发顺序可以是双字 0、双字 1(无字节允许)、双字 2、双字 3。组合任何时候都使得后一个双字的地址比前一个双字的地址更有意义。当读猝发对被寻址的目标无副作用时，桥路可以将处理器的单个存储器读请求转换为读猝发。

因为从处理器中发出的 I/O 操作不能被组合，所以这种操作将按正常情况只有一个数据段。目前尚没有已知处理器和总线主控能对 I/O 空间产生猝发操作。但当 I/O 猝发成为现实时，目标和总线主控都必须能理解隐含寻址。不能进行多数据段处理的 PCI 设备必须在第一个数据段后脱离操作。所有 I/O 操作必须正确出现在 PCI 上犹如处理器产生它那样(如果 I/O 操作中有目标被选中，但字节允许却说明传送字节数大于该目标所支持的字节数，则目标用目标失败终止传送)。

#### § 4.1 操作规则:

##### § 4.1.1 何时信号稳定:

在 PCI 总线信号中，除了 RST#、INTA#、INTB#、INTC#和 INTD#外，其余信号的都在时钟信号的上升沿采样。每个信号对应于时钟上升沿都有建立和保持时间的限制。在这样的时间范围内，不允许传送。在该时间范围以外，信号值或传送就没有意义了。

1. 一旦复位完成，下列信号要保证在每个 CLK 的上升沿是稳定的：LOCK#，IRDY#，TRDY#，FRAME#，DEVSEL#，STOP#，REQ#，GNT#，REQ64#，ACK64#，SBO#，SDONE，SERR#和 PERR#。
2. 在下列特定的时钟沿，地址数据段必须是稳定的：
  - a. 地址——在采样到 FRAME #有效后的第一个时钟，AD[31..00]应是稳定的，无论某些线是否在逻辑上有意义。
  - b. 地址——在采样到 REQ64 #有效后的第一个时钟，AD[63..32]应是稳定的，无论某些线是否在逻辑上有意义。
  - c. 数据——在读操作中一旦 TRDY#有效或写操作中一旦 IRDY#有效时，AD[31..00]稳定并有效，无论哪些字节通道参与了 this 个传送。在别

的时间它们可以是不定的。在写传送中一旦 IRDY#有效或读传送中一旦 TRDY#有效, AD 线就不能改变直到当前数据段完成。

d. 数据——在 ACK64#有效及读操作中 TRDY#有效或写操作中 IRDY#有效时, AD[63..32]稳定并有效, 无论哪些字节通道参与了这个传送。在别的时间它们可以是不定的。

e. 数据——特殊周期命令, 当 IRDY#有效时, AD[31..00]稳定并有效, 无论哪些字节通道参与了这个传送。

f. 在读传送中 TRDY#有效后或写传送中 IRDY#有效后, 不要直接选通异步数据到 PCI 上。

3. 命令/字节允许在下列特定的时钟沿要保证达到稳定:

a. 命令——C/BE[3..0]#和 C/BE[7..4]#在第一次采样到 FRAME#和 REQ64#有效时稳定并有效, 并且包含有命令编码。

b. 字节允许——在地址段后那个时钟或(和)数据段中每个时钟沿, 无论是否插入等待状态, C/BE[3..0]#和 C/BE[7..4]#是稳定并有效的。在猝发传送期间, 从每个数据段完成(IRDY#和 TRDY#都有效)的那个时钟起, 总线主控修改字节允许。在下一个时钟, 修改后的值有效。字节允许在数据段之间可以自由改变, 但在每个数据段开始的那个时钟沿必须是有效的, 并在每个数据段期间保持有效。

c. C/BE#输出缓冲器在从数据段的第一个时钟到传送结束这段时间内都必须保持输出允许。这样就能确保 C/BE#不长时间浮空。——  
 $C/BE\#.oe = !IRDY\# \# !TRDY\#$

4. PAR 在 AD[31..00]有效后的那个时钟沿稳定并有效。PAR64 在 AD[63..32]有效后的那个时钟沿稳定并有效。

5. 当操作是一个配置命令时, 只有在采样到 FRAME#有效的第一个时钟 IDSEL 稳定并有效。在任何别的时间 IDSEL 都是不定的。

6. RST#、INTA#、INTB#、INTC#和 INTD#无限制或不同步。

#### § 4.1.2 控制信号:

7. 在 FRAME#和 IRDY#无效而 GNT#有效时, 该单元可能启动一次操作。

8. 当第一次采样 FRAME#有效时, 一个传送开始。

9. 在所有 PCI 传送中都有下列对 FRAME#的约束:

a. FRAME#及其相应的 IRDY#决定了总线的忙/闲状态, 当两者之一有效时, 总线忙, 当二者都无效时总线闲。——  
 $IDLE =$

$FRAME\# \& IRDY\#$

b. 一旦 FRAME# 无效, 在同一次传送中, 它就不能再有效。

c. 在 IRDY# 有效前, FRAME# 不能无效, 在 FRAME# 无效后, IRDY# 应维持有效至少一个时钟周期, 甚至在该传送由总线主控失败来终止时也一样。

d. 一旦总线主控发出 IRDY# 有效, 无论 TRDY# 状态如何, 它必须在当前数据段完成之后才能改变 IRDY# 和 FRAME#。

10. 最后数据段完成, 当:

a. FRAME# 无效而 IRDY# 有效 (正常终止)。在目标说明最后数据传送后 (TRDY# 有效), 接口恢复到 IDLE 状态。 — — last Data =

FRAME# & !IRDY

b. FRAME# 无效而 STOP# 有效 (目标终止)。

c. FRAME# 无效而设备选择计时器溢出 (总线主控失败)。

d. DEVSEL# 无效而 STOP# 有效 (目标失败)。

11. 当 FRAME# 和 IRDY# 均无效时, 传送结束。

12. 在所有 PCI 传送中, FRAME#, IRDY#, TRDY# 和 STOP# 都有下述规则:

a. 无论何时, 只要 STOP# 有效, 按 FRAME# 无效规则, 应尽快使 FRAME# 无效 (IRDY#

必须有效)。使 FRAME# 无效应在 STOP# 有效后尽快进行, 可能是 2~3 个时钟。目标不能假定任何 STOP# 有效和 FRAME# 无效之间的时间关系, 但必须保持 STOP# 有效直到 FRAME# 无效。当总线主控采样到 STOP# 有效, 它必须在 IRDY# 有效后的第一个时钟使 FRAME# 无效, 此后 IRDY# 仍有效。IRDY# 有效可以作为总线主控的 IRDY# 正常操作之结果 (当前传送未被目标终止), 并根据总线主控何时做好完成数据传送的准备而被延迟零个或多个时钟周期。相应地, 如果 TRDY# 无效时, 总线主控可立即使 IRDY# 有效, 说明再也没有数据传送了。

b. 一旦 STOP# 有效, 它必须保持有效直到 FRAME# 无效。在 FRAME# 无效的下一个时钟, STOP# 必须无效。

c. 在传送的最后一个数据段期间 (FRAME# 无效而 IRDY# 有效), STOP# 或 TRDY# 之一有效的任何时钟沿就成为本次传送的最后时钟, 且在下一个时钟沿 IRDY# 无效 (从此开始一个 IDLE 周期, 并规定了传送的结束)。

d. 一旦目标已使 TRDY# 或 STOP# 有效, 在当前数据段完成之前, 它就不能再改变 DEVSEL#, TRDY# 或 STOP#。一旦总线主控或目标开始了数据传送, 它就不能改变状态。

13. 总线主控和目标之间每个 IRDY# 和 TRDY# 都有效的时钟沿都在传送数

据。总线主控和目标分别可用 IRDY#无效或 TRDY#无效将等待周期插入到数据段中。

14. 当数据有效时, 数据源应无条件地使其 XRDY#有效(写传送中 IRDY#, 读传送中 TRDY#)。接收单元应按其选择而有效的驱动它的 XRDY#。

15. 当前主控由目标终止时(STOP#有效), 总线主控必须至少在两个时钟内时其 REQ#无效, 一个是总线回到空闲状态, 另一个在它之前或之后。如果总线主控想完成操作, 它必须用下一个未传送数据的地址在晚些来重试被目标终止的传送。

16. 单元通过使其 DEVSEL#有效来确认作为操作的目标。

17. 只有 DEVSEL#有效, 目标才能驱动 TRDY#。

18. 一旦 DEVSEL#有效, 就不能无效, 直到最后数据段完成。除非发出目标失败信号。

#### § 4.1.3 闭锁操作

19. LOCK#只能由一个单元拥有并驱动, 在总线释放时可能保持不变。

20. 支持 PCI 上的 LOCK#的目标必须遵守下列规则:

a. 当 LOCK#在地址段期间无效时, 锁定操作的目标自我锁定。

b. 一旦锁定建立, 该目标保持锁定直到它采样到 FRAME#及 LOCK#均无效或它发出目标失败信号。

c. 保证给 LOCK#的拥有者(一旦锁定建立)至少 16 字节的资源。对多口设备, 这包括不在 PCI 上进行的操作。

21. LOCK#由当前的总线主控驱动, 在 PCI 上使用 LOCK#的总线主控必须遵守下列规则:

a. 在锁定操作期间, 一个总线主控只能操作一个资源。

b. 锁定不能跨越设备的界限。

c. 16 字节(线性排列)是总线主控在锁定操作期间能计数的最大资源范围。对这 16 个字节的任一部分操作都要锁定全部 16 个字节。

d. 锁定操作的第一个传送必须是读传送。

e. LOCK#必须在地址段后那个时钟有效, 并保持有效以维持控制。

f. 如果在数据段完成和锁定未建立之前发重试信号, LOCK#必须释放。

g. 无论何时, 操作被目标失败或总线主控失败终止时 LOCK#必须释放。

h. 在连续锁定操作之间, LOCK#必须释放至少一个空闲周期。

#### § 4.1.4 仲裁:

22. 总线仲裁器可以在任何周期使某单元的 GNT#无效。任何单元必须保证在开始传送的时钟沿 GNT#是有效的。如果 GNT#无效, 传送就不能进行。

23. 一旦 GNT#有效, 根据下列规则可使其无效:

a. 如果 GNT#无效而 FRAME#有效, 总线传送有效且继续进行。

b. 如果总线不是处于空闲状态, 一个 GNT#能在另一个 GNT#有效的同时无效。否则, 使某个 GNT#无效和使下一个 GNT#有效之间必须有一个时钟的延迟, 不然在 AD 线和 PAR 线上就可能会有冲突。

c. FRAME#无效时, GNT#可以在任何时间无效。以便为高优先级的总线主控服务, 或作为对相应 REQ#无效的应答。

24. 若仲裁器使某个单元的 GNT#有效而总线处于空闲状态, 该单元就必须在 8 个 PCI 时钟内(要求值)开放其 AD[31..00]、C/BE[3..0]#及 PAR 输出缓存, 推荐值是 2~3 个时钟。

#### § 4.1.5 奇偶校验:

25. 在每个地址段和数据段, 所有 AD 线(如总线主控支持 64 位数据通道, 还包 AD[63..32])

都必须驱动到稳定的值。甚至未参与当前数据传送的字节通道读应有稳定的数据在其上, 以便进行奇偶校验。奇偶校验根据下列规则产生:

a. 无论传送类型、格式, 所有 PCI 传送的奇偶校验的计算是相同的。

b. AD[31..00]、C/BE[3..0]#和 PAR 上“1”的数量是偶数。

c. AD[63..32]、C/BE[7..4]#和 PAR64 上“1”的数量是偶数。

d. 产生奇偶校验是不可选择的, 所有 PCI 兼容设备都必须做奇偶校验。

#### § 4.2 寻址

PCI 上地址解码是分散的, 即每个单元负责自己的地址解码, 这样避免了采用中央解码逻辑及在使用配置的设备之外的设备选择信号。PCI 支持两种类型的设备地址解码: 正解码和反解码。正解码的设备只在分



配给它的地址范围内进行解码，解码速度相对较快。反解码只能被总线上的一个设备使用，因为它接收所有不被其它单元解码的操作，这种解码对于诸如相应于高端地址空间的标准扩展总线这样的单元是很有效的，该单元常常就是与一个标准总线相连的桥路。能完成正的或反的解码的目标，对保留总线命令编码不能作出反应(驱动 DEVSEL#有效)。

包含于低两位地址(AD[1..0])上的信息随地址空间而改变。在 I/O 地址空间，所有 32 位都用来提供完整的字节地址。这样就使得要求地址分解到字节一级的单元能完成地址解码，不必因为等待字节允许而增加等待状态的周期(这会将反解码推迟一个额外时钟周期)。AD[1..0]仅用于地址解码并说明参与当前传送的最低字节。AD[1..0]的编码组合如下表所示，任何不在表中的组合都是非法的，并由目标失败来终止。

AD1	AD0	C/BE3#	C/BE2#	C/BE1#	C/BE0#
0	0	×	×	×	0
0	1	×	×	0	1
1	0	×	0	1	1
1	1	0	1	1	1

一旦一个目标确认了一个 I/O 操作，那么它就要决定它能否完成如字节允许所说明的全部操作。如果所选中的字节都不在选中的目标地址范围内，整个操作不能完成，此时，目标不传送任何数据，用目标失败来终止传送。

在存储器命令期间，AD[1..0]决定了猝发顺序，有如下意义：

AD1	AD0	猝发顺序
0	0	线性增加
0	1	高速缓存线触发器模式
1	×	保留

在存储器命令传送期间，所有目标都应检查 AD[1..0]，并且提供期上所要求的猝发顺序，或在第一个数据段后让目标脱离总线。所有支持猝发的设备都要求线性触发顺序的采用。对采用高速缓存线触发器则不要求。在存储器地址空间，是对由 AD[31..02]解码得到的双字地址进行操作的。在线性增加模式下，在每个数据段后，认为地址增加 4 个字节，直到传送终止。

当目前的读传送是针对可高速缓存存储器的时候，无论字节允许情况如何，都必须返回全部字节的数据。这就要求决定可高速缓存性能的


单元保证目标返回全部字节的数据。如果可高速缓存性由提出传送要求的一方决定，那么它必须使所有字节允许都有效(低)，以使目标能返回全部所要求的数据。如果可高速缓存性由目标决定，它就必须忽略字节允许而返回全部双字。可高速缓存的目标要么就返回整条高速缓存线上的数据，要么只返回所要求数据的第一个。

如果目标不支持高速缓存但支持预取，它也必须不管字节允许情况而返回全部字节数据。如果没有别的影响，目标可以只工作于这种模式。

在配置地址空间，是对由 AD[7..2]解码得到的双字地址进行操作的。当一条配置命令被解码，IDSEL 有效且 AD[1..0]是 00 时，某个单元就判断出它是本次操作的目标。否则该单元不理睬当前传送。通过解码配置命令及桥路号，且 AD[1..0]是 01，桥路可以判别出某个操作是针对挂在它边上的设备的。

PCI 允许任何连续或非连续的字节允许的组合。如果没有字节允许有效(全为 1)，目标必须用 TRDY#有效来完成这次传送并在读请求时提供奇偶校验。无字节允许有效的传送的目标必须在无任何永久性改变的情况下结束当前数据段。在读传送中，这意味着数据和状态都没有改变。如果完成该操作对数据和状态都没有影响，那么目标可以在该操作中提供数据，也可以不提供。在读操作中，无论字节允许的状态怎样，目标都必须提供 AD[31..00]及 C/BE[3..0]#上所有数据的奇偶校验。在写传送中，数据不存储且 PAR 有效。

### § 4.3 总线传送

由一个以上单元驱动的信号必须有一个转换周期。该转换周期用以避免在一个单元停止驱动该信号而另一个单元开始驱动它时可能发生的冲突。在时序图上这种转换用  来表示。对不同的信号，转换周期发生的时间不一样。例如：IRDY#、TRDY#、DEVSEL#、STOP#和 ACK64#用地址段作为它们的转换周期。FRAME#、REQ64#、C/BE[3..0]#、C/BE[7..4]#、AD[31..00]及 AD[63..32]用传送之间的 IDLE 周期作为它们的转换周期。LOCK#的转换周期是当前拥有者释放它之后的那个时钟。PERR#的转换周期是最后的数据段之后的第四个时钟，此时 AD 线转换完成已有三个时钟。

下列时序图中示出参与 32 位传送的各种重要信号之间的关系。实线表示正被当前总线主控或目标驱动的信号。虚线表示没有单元驱动的信号，然而，如果虚线为高，则应假定它仍保持稳定值。当虚线为高、低

之间,说明三态信号是不确定值(例如 AD 线和 C/BE#线)。实线变为点划线,说明信号被驱动,现在是三态。当一条实线由低变高,然后变为点划线,这表明信号被驱动到高,以给总线预充电(s/t/s 信号),然后变为三态。

#### § 4.3.1 读传送

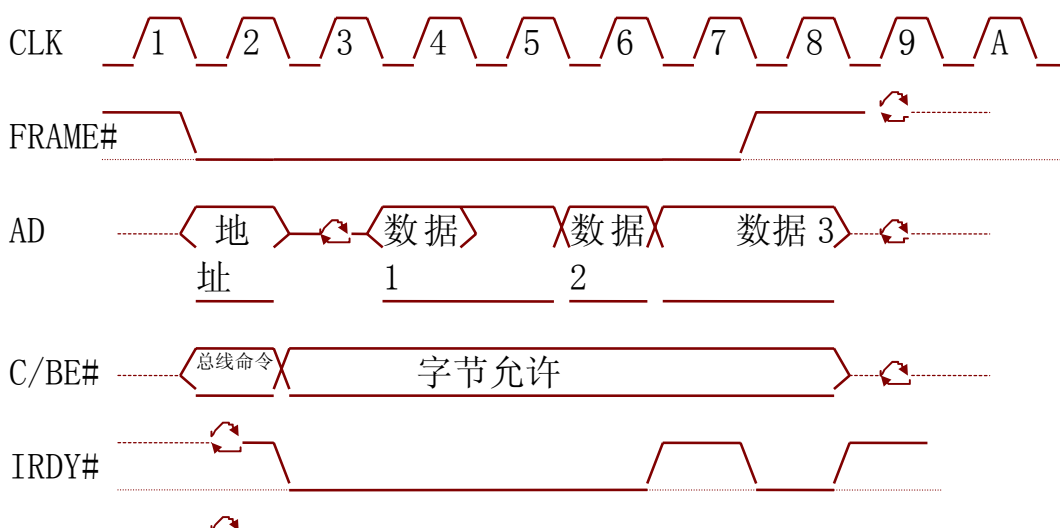
图 4.1 的读传送发生在 clock2, 由 FRAME#第一次有效时的地址段开始, 在地址段期间, AD[31..00]包含一个有效的地址, C/BE[3..0]#包含一个有效的总线命令。

第一个数据段的第一个时钟是 clock3。在数据段期间, C/BE#说明了哪些字节通道参与了当前数据段的传送。C/BE#输出缓冲器在从数据段的第一个时钟到传送结束这段时间内都必须保持输出允许。这样就能确保 C/BE#不长时间浮空。

读传送的第一个数据段要求有一个转换周期(由目标通过 TRDY#设置)。在这种情况下, 地址在 clock2 有效后, 总线主控停止驱动 AD 线。目标可以提供有效数据的最早时间是 clock4。当 DEVSEL#有效时, 在转换周期后, 目标必须驱动 AD 线。一旦允许输出, 输出缓冲器必须保持输出允许直到传送结束。

图中, 在 clock3、5、7, IRDY#和 TRDY#其中之一无效, 就插入等待周期。在 clock4、6、8, IRDY#和 TRDY#都有效时, 完成数据传送。只有 DEVSEL#有效, 才能驱动 TRDY#。

在最后一个数据段, 只有 IRDY#有效时, FRAME#才能无效。





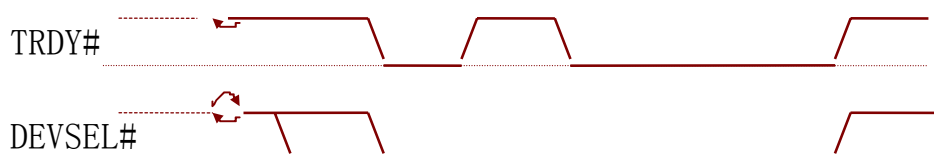


图 4.1 基本的读操作

### § 4.3.2 写传送

图 4.2 所示为写传送，地址和数据都由总线主控提供，因而在地址段之后不要求有转换周期。第一、第二个数据段是按零等待周期完成的。在 clock5，因 IRDY#无效，传送由总线主控延迟。虽然这样做允许总线主控延迟数据，但不允许延迟字节允许。

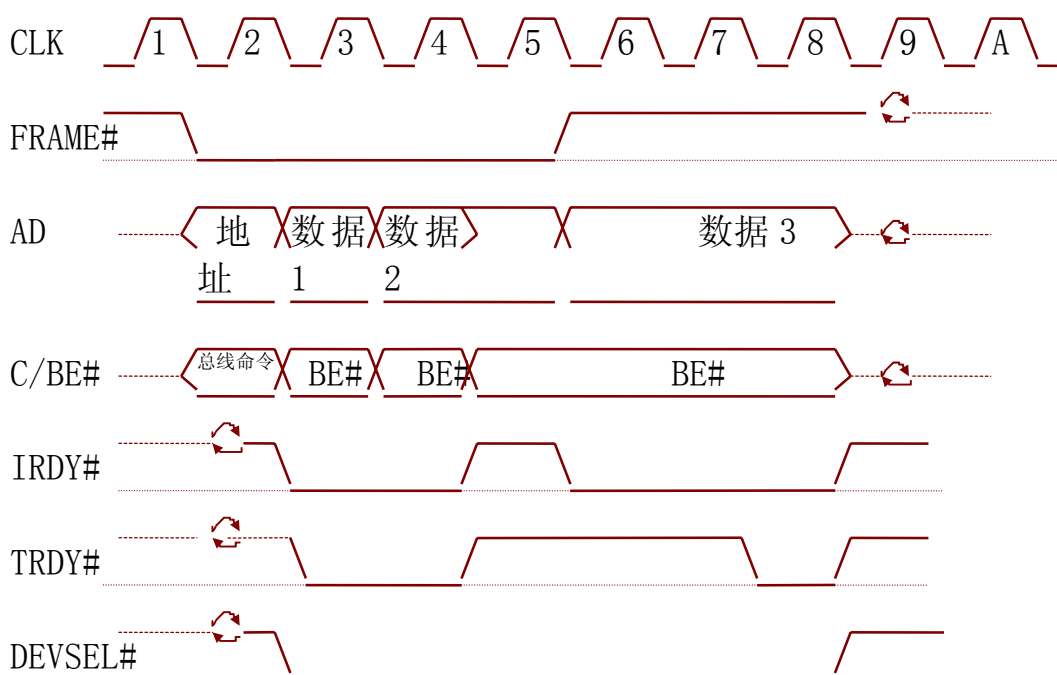


图 4.2 基本写操作

### § 4.3.3 传送终止

总线主控和目标都可以终止 PCI 传送。在总线主控和目标都不能单独有效地终止传送时，总线主控保持主要的控制，无论终止是由什么引

起的，都将带给传送一个有效的、系统的结论。

#### § 4.3.3.1 总线主控引起的终止

总线主控用来引起终止的机制是当 IRDY#有效时，FRAME#无效。这就对目标表明最后一个数据段正在进行。当 IRDY#和 TRDY#都有效时，最后一个数据传送发生。当 FRAME#及 IRDY#都无效时，传送完成，总线回到 IDLE 状态。

总线主控可以因下列两种原因之一而用这种机制去终止传送：  
完成：相应于总线主控已完成它想要进行的传送。这是终止的最常见原因。

时间溢出：总线主控的 GNT#已无效且其内部延迟计时器已满(目标引起操作延迟或是要进行的操作太长)。高速缓存存储器写传送不被延迟计时器所控制，它只能在高速缓存边界上被停止。用高速缓存存储器写命令启动传送的总线主控的在写满一条高速缓存线之前不会理会延迟计时器，当传送达到一条高速缓存线的边界且延迟计时器已计满(并且 GNT#无效)时，总线主控必须终止这次传送。如果是由目标来终止高速缓存存储器写传送，则总线主控用存储器写命令尽快完成这个传送(因为不再存在高速缓存存储器写的条件)。也应该注意到，除非 GNT#无效，否则在计时器计满后也不需终止传送。

图 4.3 是两个正常完成的例子，在 clock3，FRAME#无效，IRDY#有效向目标表明是最后一个数据传送，主控保持 IRDY#有效，待目标发出 TRDY#有效，最后一个数据传送传送完成，总线回到 IDLE 状态。

图 4.3 的两种情况也可能是由时间溢出引起正常终止。左边的情况，因计时器满，FRAME#在 clock3 无效，GNT#无效，且总线主控已准备好传送最后一个数据(IRDY#有效)。因为在计时器满时 GNT#已无效，故不允许继续使用总线，除非是使用高速缓存存储器写命令，因为它只能在高速缓存线边界上被停止。右边的情况，在 clock1 计时器满。IRDY#在 clock2 无效，总线主控未准备好传送数据，这就要求 FRAME#仍维持有效。在 clock3，总线主控已准备好完成这次传送(IRDY#有效)，故 FRAME#无效。这种终止延迟最多不能超过 2—3 个时钟周期。



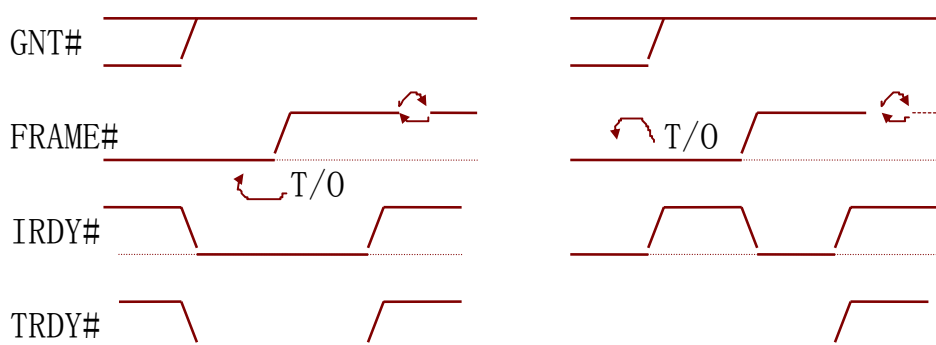


图 4.3 总线主控引起的终止

在这种机制的改进型中，允许在没有目标应答的情况下总线主控终止传送。这种异常终止称为**总线主控失败**。尽管这样可能会在要求这次传送的应用中引起严重错误，但传送能很好的完成，以保护别的单元的正常 PCI 操作。

由总线主控失败引起终止的例子如图 4.4 所示。这是由总线主控启动的终止的一种不正常情况(除配置和特殊命令周期外)。如果 DEVSEL# 直到 clock6 仍无效，则总线主控可判断这次传送没有应答。总线主控假定这次操作的目标没有能力完成所要求的传送或地址错误，在 clock7 使 FRAME#无效，在 clock8 使 IRDY#无效。总线主控用总线主控失败来终止一次传送的最快情况是在第一次采样到 FRAME#有效后五个周期，总线主控作一个单数据传送时情况就是这样，如果 DEVSEL#在 clock3、4、5、或 6 已有效，这就说明传送请求已由一个单元获知，就不再允许总线主控失败。

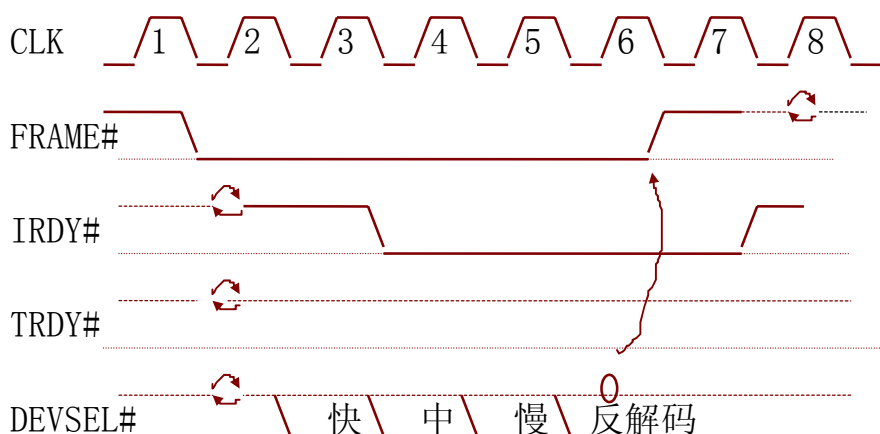


图 4.4 总线主控失败终止时序

在 PC 兼容系统中，主总线桥路在用总线主控失败终止时，读传送必须全部返回“1”，写传送必须放弃数据。桥路必须在状态寄存器中设立总线主控失败判断位。在总线主控不能通过其设备驱动器反映错误时，别的总线主控设备可以将这一位看作错误标志而发出 SERR#。一个 PCI 对 PCI 桥路必须支持 PC 兼容性，就象主总线桥路那样。当这个 PCI 对 PCI 桥路在别的系统中应用时，桥路必须象别的总线主控一样工作。桥路所作的读数据的预取对系统必须是透明的。这意味着当一个预取传送由总线主控失败终止时，桥路必须简单地停止传送并继续进行无响应的正常操作。当传送不是由目标申请时，这种情况发生。

#### § 4.3.3.2 目标启动的终止

目标启动的终止中所用的机制是 STOP#信号。目标发出 STOP#信号去要求总线主控终止传送，一旦发出，STOP#就保持有效直到 FRAME#无效。总线主控用下一个未传送数据的地址在晚些时候重新开始由重试或解除连接终止的传送。当当前传送由目标终止时，总线主控必须使其 REQ#信号无效。总线主控至少应使 REQ#保持两个 PCI 时钟无效，一个是在总线主控回到 IDLE 状态时，另一个是在 IDLE 状态之前或之后。如果总线主控想完成传送，在其使 REQ#无效两个时钟之后或某些可能的不能进行传送的状态发生之后立即重使 REQ#有效。否则，单元只是在它重新需要使用接口时使 REQ#有效。

目标可由下列两个原因之一而启动使用这种机制的终止：

重试(Retry)：相应于目标正处于不能处理传送的状态的终止。这种情形包括死锁的可能性、某些非 PCI 资源忙状态或一种闭锁操作的锁定状态。重试意味着目标终止该传送且没有传送数据。

解除连接(Disconnect)：相应于目标在 PCI 导线延迟时间内不能作出应答而要求终止(PCI 导线延迟时间是 8 个 PCI 时钟)。注意在第一个数据段时这种情况通常不会发生。解除连接意味着目标在数据传送时或在数据已被传送之后终止传送。

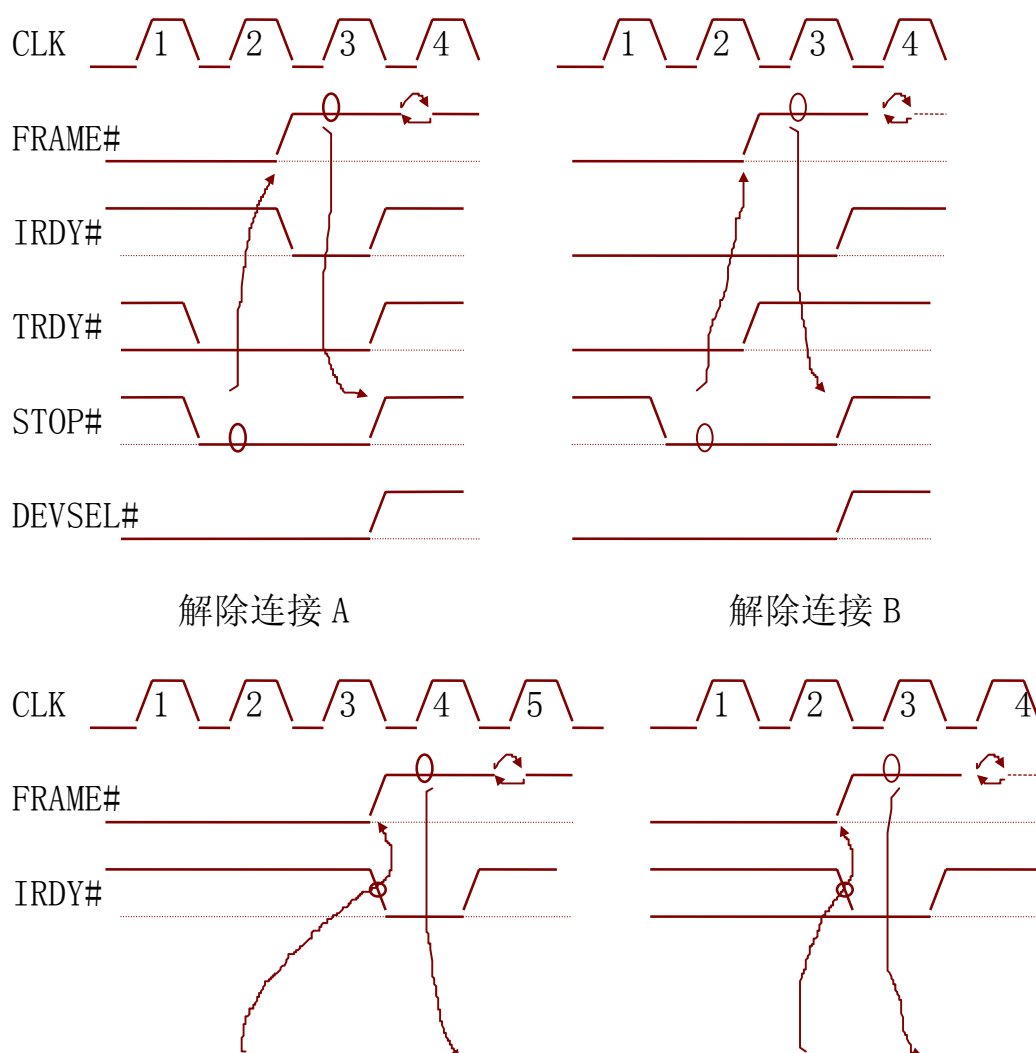
可高速缓存的目标除非到了一条高速缓存线的边界，否则不能解除与高速缓存线存储器的连接，无论高速缓存当前是否被允许。所以当操作是针对一个可高速缓存的存储器范围时，“监视”单元常假设一条高速缓存存储器写命令将完成而不被解除连接。

在这种机制的改进型中，目标可终止有重大错误的传送，或是不能作出应答的传送。这种异常终止称为**目标失败**。尽管这样可能会在要求这次传送的应用中引起严重错误，但传送能较好完成，以保护别的单元的正常 PCI 操作。

许多目标将至少被要求具有重试能力，但目标也可以选择其它的目标启动终止的版本。总线主控对它们全部都要能准确处理。象下面这些简单的目标，重试能力也是可选用的：(1)不支持锁定操作；(2)不能判断死锁和活锁的可能性；(3)不进入一种可能需要的已拒绝操作的状态。

图 4.5 示出三种解除连接的例子，STOP#与 FRAME#之间有如下关系：

- 当 STOP#有效时给出解除连接信号(这时 DEVSEL#必须有效，否则就说明目标失败)，一旦 STOP#有效，它必须保持有效直到 FRAME#无效。
- 在 STOP#有效后，FRAME#应立即无效。例 C 中多花了一个时钟周期是因为在 STOP#有效后 IRDY#不能立即有效。
- 在 FRAME#无效的下一个时钟 STOP#立即无效。



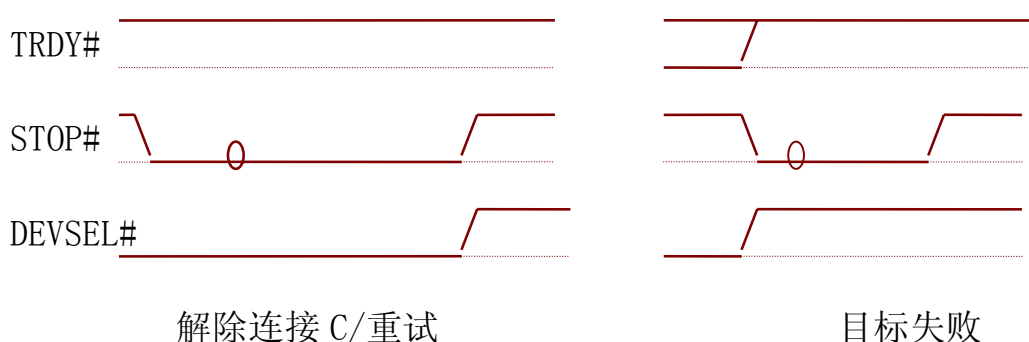


图 4.5 目标启动的终止

IRDY#与 TRDY#之间的关系独立于 STOP#和 FRAME#之间的关系。所以，在目标要求终止期间，数据可以被传送，也可以不被传送，这仅取决于 IRDY#和 TRDY#的状态。然而，当 STOP#有效且 TRDY#无效，就说明目标将不再传送数据，且总线主控也不再象在一个完整传送中那样去等待最后的数据传送。

在例 A 和 B 中，目标想在 STOP#有效后，还传送一个数据，然后停止，因而在 STOP#有效时使 TRDY#有效，但只能进行一个数据传送，完成后 TRDY#必须无效。例 A 中，因总线主控未准备好，数据在 FRAME#无效后传送。例 B 中数据在 FRAME#无效前传送。如果目标在最后数据段中要求一个等待周期，它必须延迟 STOP#的有效直到它准备好完成这次传送。

例 C 是解除连接的一种全无数据传送的特殊情况 (TRDY#无效因而在 STOP#有效后无数据传送)，注意，FRAME#的无效延迟到 IRDY#有效后才进行。这也是一个重试的例子。重试的一个普通的例子是当目标因进行闭锁操作而被另一个总线主控锁定时。另外的例子是目标在允许传送进行之前，先要对别的非 PCI 资源进行操作。

图 4.5 右下角是一个当 STOP#有效时 DEVSEL#无效时目标终止的例子。这说明目标要求终止传送且不再重做该传送。此外，如果在当前传送中已有数据被传送，它有可能不可靠。在发出目标失败信号前，DEVSEL#必须有效一个或多个时钟周期有效，且 TRDY#必须无效。\*\*\* 仔细分析 PCI. TDF 采用的是哪一种目标终止的策略

#### § 4.4 仲裁

为了减少操作延迟，PCI 仲裁更近于基于操作而不是基于时间槽。就是说总线主控必须就它在总线上完成的每一个操作作出仲裁。PCI 使用了一个中央仲裁电路，在其中每个总线主控单元都有一个唯一的请求 (REQ#) 和允许 (GNT#) 信号。这种简单的请求——允许联络用来获得对总



线的操作权。仲裁是“隐蔽”的，意为它发生在上一个操作期间，所以不会因仲裁而花费 PCI 周期，除非总线处于 IDLE 状态。

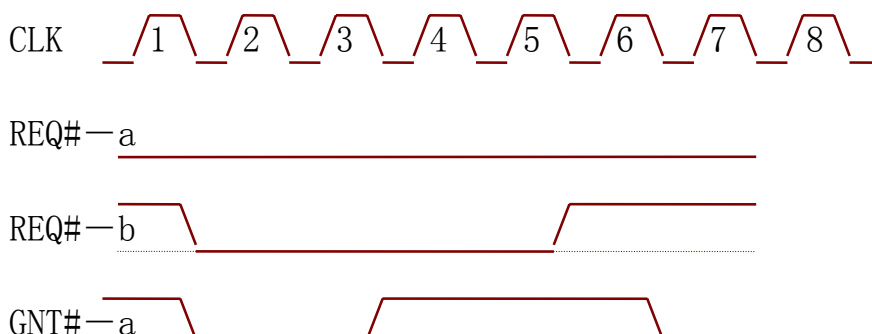
中央仲裁采用了特别的仲裁算法，例如循环、优先、公正等等。必须定义一种仲裁算法作为产生错误情况延时的基础。然而，因为仲裁算法基本上不属于总线规范的范围，系统设计者可以改变它，但必须提供它们所选用的 I/O 控制器和外插卡的延迟要求。总线允许同一单元的背对背(back to back)传送，也允许仲裁对优先的和重要的请求的灵活性。在任何周期，只要 GNT#信号有效，仲裁就提供给某一单元总线操作权。

某一单元通过使其 REQ#有效而请求总线。单元只能使用 REQ#来发出真正的总线使用需求信号。单元绝不能用 REQ#将自己“停放(park)”在总线上，如果应用了总线停放，正是仲裁器指派了约定的拥有者。当仲裁器判断某一单元可以使用总线时，它使该单元的 GNT#有效。

仲裁信号协议详见 4.1 操作规则，图 4.6 说明了基本仲裁，用了两个单元来说明仲裁器如何交换总线操作。REQ#-a 在 clock1 之前或 clock1 时有效，以申请使用接口。因为 GNT#-a 在 clock2 有效，所以单元 A 可以操作总线。单元 A 可以在 clock2 开始传送，因为此时 FRAME#和 IRDY#无效而 GNT#-a 有效。在 clock3，当 FRAME#有效时，单元 A 开始传送。因为单元 A 想作另一次传送，所以保持 REQ-a 一直有效。当 FRAME#在 clock3 有效时，仲裁器决定单元 B 是下一个总线使用者，且在 clock4 使 GNT#-b 有效，GNT#-b 无效。

在 clock4，当单元 A 完成它的传送时，放弃总线。当 FRAME#和 IRDY#都无效时，所有 PCI 单元都能判断当前传送的结束。在 clock5，单元 B 拥有总线操作权(因 FRAME#和 IRDY#均无效)，并在时钟 clock7 完成其传送。

注意在 clock6，REQ#-b 无效且 FRAME#有效使，说明单元 B 只要求一次传送。因为 REQ#-a 仍然有效，故仲裁器同意单元 A 作下一个总线操作。



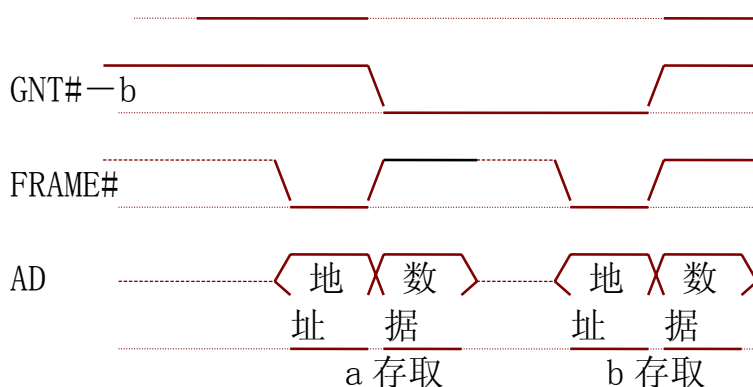


图 4.6 基本仲裁

当前总线拥有者在它要求额外(外加)传送时,应保持 REQ#有效。如果没有别的总线请求有效,或当前总线主控具有最高优先级,仲裁器让当前总线主控继续操作总线。

GNT#让单元作一次总线传送,如果某一单元只想作一次传送,它将在 FRAME#有效的同一个时钟使其 REQ#无效,如果一个单元想作另一个操作,它将继续使其 REQ#有效。单元可在任何时候使其 REQ#无效,但仲裁器可以将其理解为该单元不再延长它对总线的请求而且可以使其 GNT#无效。

当目标终止某一传送使(STOP#有效),总线主控必须使其 REQ#至少在两个 PCI 时钟内无效,一个是总线回到 IDLE 状态,另一个在此之前或之后。如果总线主控想完成该传送,它必须在 REQ#无效或一个潜在的需求条件可能发生之后重新使其 REQ#有效。如果总线主控不想完成该传送(因为它正在预取或一个高优先级的内部需求要求服务),那么该单元在下次它又需要使用接口时使 REQ#有效。这样,当上一个目标准备下次传送时,允许别的单元使用接口。

如果当前总线主控的 GNT#已有效后它没有开始操作(其 REQ#也有效),并且有 16 个 PCI 时钟总线处于 IDLE 状态,则仲裁器可以假定当前总线主控“断路”。然而,仲裁器可以在任何时候改变 GNT#而为更高优先级的单元服务。

#### § 4.5 仲裁放置(PARKING)

“放置”指的是当前无单元使用总线或申请总线时,允许仲裁器使某一被选定单元的 GNT#信号有效。仲裁器可以按其所愿方式选择缺省总线拥有者或选择完全不放置。当仲裁器使某一被选定单元的 GNT#信号有效而



总线处于 IDLE 状态时, 该单元必须在 8 个 PCI 时钟(要求这样)内使其 AD[31..00]、C/BE[3..0]#和(一个时钟后)PAR 的输出缓冲器, 推荐值是 2—3 个时钟。不强迫该单元在一个时钟内开放所有缓存器。这种要求保证仲裁器能安全地将总线放置给某些单元并知道总线将不会浮空。如果仲裁器不放置总线, 嵌有仲裁器的中央资源设备就驱动总线。

在所有情况下, 如果总线处于 IDLE 状态而仲裁器使一个单元的 GNT#无效, 该单元就失去了对总线操作的权力。唯一的例外情况是仲裁器使 GNT#无效的同时, 该单元使 FRAME#有效, 这时总线主控将继续进行传送。否则, 该单元就必须使 AD[31..00], C/BE[3..0]#和(一个时钟后)PAR 处于三态以脱离总线。与上述情况不同, 该单元必须在一个时钟关闭所有缓存器以避免和下一个总线拥有者发生冲突。

如上所述, 从总线 IDLE 状态起到实现总线仲裁的最小延迟如下:

- 放置: 已放置单元零个时钟周期, 其它单元两个时钟周期。
- 不放置: 每个单元都是一个时钟周期。

当总线已被放置到某一单元上时, 可以允许该单元不用 REQ#有效就开始传送。当总线处于 IDLE 状态且 GNT#有效, 总线主控就能开始一次传送。当总线主控只要求一次传送时, 它不能使 REQ#有效; 否则, 在它不要求使用总线时, 仲裁器也可以继续使其 GNT#有效。当该单元需要作多个传送时, 它应使 REQ#有效以告知仲裁器它想作多个传送。

## § 4.6 延迟

PCI 是一种低延迟、高通过率的 I/O 总线。本节叙述帮助预测和控制错误情况延迟的 PCI 机制。给出这些机制, 对单一 PCI 环境和高效主存储器接口的等待时间能很准确地预测出来。扩展标准总线(ISA、EISA 或 MC)的加入将使延迟预算更加困难。在带有扩展总线的情况下, 错误延迟方案可以由扩展总线或表现不佳的接口属性而不是 PCI 总线决定。

### § 4.6.1 PCI 上的延迟

操作延迟由三部分组成:

- **仲裁延迟**——从总线主控发出 REQ#有效到收到 GNT#有效的的时间。它由仲裁算法、设备的优先级要求和系统使用情况决定。对高优先级设备, 该时间的典型值为 2 个 PCI 时钟。低优先级设备的延迟则根据高优先级的设备的数量和设备使用情况来决定。在没有别的总线主控请求时, 仲

裁延迟的典型值是 2 个时钟

- **总线获取延迟**——从总线主控收到 GNT#有效到它使 FRAME#有效的的时间，是设备等待总线变空共等待的时间。它要么是系统中第一个数据段的最大延迟时间(从 FRAME# 有效到 TRDY#有效的的时间)。要么就是总线主控延迟计时器的值(如果所有第一个数据段延迟都较低的话)。当总线处于 IDLE 状态时总线获取延迟是零周期。否则要等待当前总线主控完成其数据传送或其延迟计时器满。

- **目标延迟**——从总线主控使 FRAME#有效到目标为第一次数据传送使 TRDY#有效所花费的时间。其典型值正好是第一个数据的延迟。但作为缓存的 PCI 桥路设备能使目标 延迟更长。例如，如果一个设备正在通过 CPU 桥路缓存对主存储器继续操作，该桥 路在允许操作完成前可能不得不刷新其缓存器。该桥路将发出重试信号给提出要求的设备，刷新缓存器，然后准备好应答请求。根据要刷新什么地方和缓存器的大小，这样做会大大增加目标延迟时间。

在无别的单元请求总线的情况下，PCI 总线主控能猝发传送目标所能接收长度的数据。然而，PCI 规定了两种机制，在有别的请求存在时，覆盖总线主控的总线占有状态。这样，可预计的总线获取延迟就能实现。这两种机制定义如下：

总线主控延迟计时器(LT)：在总线主控未使 FRAME#有效时，每个总线主控的 LT 都被清除且被挂起。当总线主控使 FRAME#有效时，它就允许其 LT 计数。如果该总线主控在计数计满之前使 FRAME#无效，则 LT 无意义。否则，一旦计数满(计数 T 时钟，时间溢出)，如果总线主控之 GNT#改变，它必须立即放弃总线占用(目标准备好后就开始有效的终止)。大致上，T 代表了分配给总线主控的最小时间片段(单位是 PCI 时钟)，它是一种通过量(高值)和延迟(低值)之间的折衷。例如，假设第一个数据段要 8 个时钟周期的延迟，T=40 时在总线主控和目标读能做零等待猝发的情况下提供了一个 32 个数据段的猝发，将 T 减少到 20 则每次猝发都只传送 12~14 个数据段，但最大传送就限制在 28 个时钟内。

目标开始的终止(特别解除连接)：如果到数据段“N+1”所要增加的延迟超过 8 个时钟，目标就要根据数据段 N(此时 N=1, 2, 3...)的完成按结束传送方式处理 TRDY#和 STOP#。例如假设一个 PCI 总线主控从一个扩展总线读取数据要花费至少 15 个时钟。根据这一规则，N=1，到数据段 2 的延迟是 15 个时钟周期，所以目标必须在数据段 1 完成时终止传送。 \*\*\* 目标要有相应的动作 Time\_tar[].rst = !IRDY & !TRDY

注意，这两种机制都没有限制第一个数据段的延迟。例如，假设在一特定系统中，没有目标慢到完成第一个数据段要延迟 TRDY# 超过  $T+8$  个以上的时钟。根据给定假设和上述机制，总线主控将做的最长的传送是  $T+8$  个时钟 (假设总线主控之 GNT# 在其 LT 计满之前变化)。

不同系统所用实际操作延迟往往有所不同，这是由于各系统所用设备的延迟特性引起的。但 PCI 设备制造商必须做一些典型操作延迟的测量，所以它们能提供足够的片内缓存来减少超载和欠载。

#### § 4.6.2 延迟指导原则

在许多的 PCI 系统中，操作延迟的典型值都较短 (大约 2us) 内并且易于测量，然而，错误情况延迟不仅会增大，而且在一些情况下，很难预测。例如，通过桥路的扩展标准总线适配器 (ISA/EISA/MC) 的延迟通常是适配器的性能，而不是 PCI 的性能。作为补偿，要求保证错误情况下操作延迟的总线主控必须为 30ms 而提供足够的缓存。如果某些错误是可以忽略的，并能够判断缓存器的超载/欠载，以起动数据包的传送与重发，可将错误延迟时间做到 10ms 左右以降低成本。

某些应用 (如嵌入控制，多媒体等) 可能会要求比典型主系统 (通用目的) 的原始 PCI 所提供的延迟更加严格的延迟控制。这种类型的应用，要求目标具有快速、可预计、有限延迟等特性。建议采用如下原则来设计目标接口。其中“单层”意指对选定资源进行立刻操作的 PCI 目标，例如单口 DRAM 和帧缓存 VRAM。“多层”相应于这样的目标：为给所选定资源提供操作而必须取得一个独立的仲裁资源。如 PCI 到总线的桥路和双口 DRAM。

(1) 单层目标将第一个数据段的延迟限制在 16 个时钟。如果一个暂时的内部状态 (如 DRAM 刷新) 将要操作，则该操作的延迟会超过 16 个时钟，并要立刻重试。

——Timer1[].rst = !FRAME

(2) 多层目标将重试与忙资源发生冲突的操作。例如，对 EISA 的从属设备进行操作，而此时，EISA 被别的总线主控所占用，该操作就要立刻重试。同样，一个对正在扫描刷新的帧缓存 DRAM 的操作也要重试。

(3) 多层目标 (尤其是总线对总线的桥路) 更加注重写缓存的方案。写缓存使得限制延迟更加困难，因为它所要求的较强的顺序性常常是在允许别的方式的操作进行之前要完成所有的序列。特别要推荐的是，在主 CPU 到 PCI 的桥路能分解到固定的 PCI 帧缓存写的同时，也支持分解到

ISA/EISA/MC 目标的写。

最后，为了易于做到可靠的系统级折衷，设备销售商应该清楚地标明设备的延迟特性。总线主控设备必须明确标明延迟的预计值及违法延迟出现的结果。目标设备则规定错误状态的反应，最好是所有可能引起重试和解除连接的事情都有说明，如果目标的延迟由外部因素而决定，也要清楚地表明。

#### § 4.7 快速背对背传送

快速背对背传送是总线传送之间没有 IDLE 状态的传送。当 IRDY#和 TRDY#都有效，FRAME#无效时，完成最后的数据段。在上一个传送的最后数据传送的同一个时钟，当前总线主控不经过 IDLE 状态而直接开始下一个传送。当避开 TRDY#、DEVSEL#和 STOP#上的争用时，在 PCI 上就可作快速背对背传送。快速背对背传送有两种类型：

第一种类型的快速背对背传送支持由总线主控承担避免争用的任务。当总线主控能保证无争用发生时，它可以取消传送之间的 IDLE 周期。如果总线主控的第一个操作是写操作而第二个操作的目标与第一个相同时，这种情况就可能实现。这种类型的快速背对背传送要求总线主控能知道潜在的目标地址范围，否则争用就可能发生。这种类型的快速背对背传送对总线主控是任选的，但必须能被目标解码。

第二种类型的快速背对背传送支持由所有可能的目标共同承担无争用的负担。状态寄存器中的快速背对背能力位，当且仅当该设备作为总线的目标且满足下面条件时，可以被硬件置为逻辑“1”（高电平）。

(1) 目标在传送前没有 IDLE 状态的情况下开始传送，不能错过总线主控的开始信号和地址。换句话说，就是在连续的时钟周期中，目标必须能跟上总线状态直接由最后数据传送 (FRAME#高，IRDY#低) 转换到地址段 (FRAME#低，IRDY#高)。注意在这两个传送中，或在其中之一中，目标可能被选中，也可以不被选中，但必须要无遗漏地跟踪总线状态。

(2) 目标必须避免 TRDY#、DEVSEL#和 STOP#信号发生冲突。如果目标不是使用可能的最快 DEVSEL#有效时间，基本上就可避免这种冲突发生。对于做零等待解码的目标，除下列两种情形之一外，该目标必须延迟这三个信号有效一个时钟周期。

a. 当前总线传送前有一个总线 IDLE 状态。这就是说该传送不是一个快速背对背传送。

b. 在上一次总线传送时，当前目标已驱动 DEVSEL#有效。这就是说该传送是一个与上次传送目标相同的快速背对背传送。

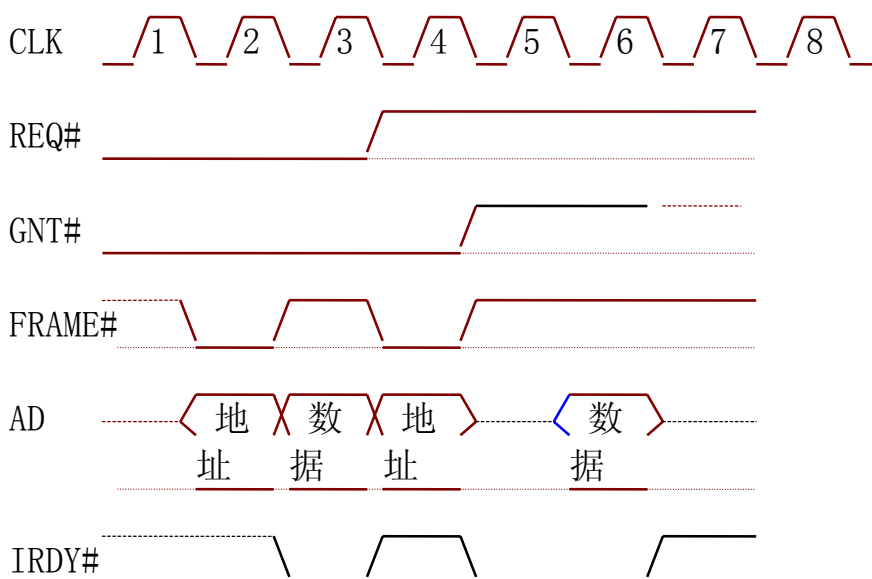


如果某目标不能提供上述两种保证，它就不能使用这一位，且在读状态寄存器时，自动返回零。

对于想完成由目标机制支持的快速背对背传送的总线主控，在其配置空间的命令寄存器中有一个快速背对背允许位(该可读/写位只对作为总线主控的设备有意义，而且是可选的)。当该位设置为“1”(高)，如果当前总线主控所作的上一次传送是写传送，总线主控可以无视哪个目标被寻址而开始快速背对背传送。如果该位被设置成“0”(低)或没用它，只有当总线主控能保证新的传送的目标与上一次传送的目标相同时，它才能做快速背对背传送。在确认同一总线上所有目标的快速背对背传送置位后，由系统配置程序来设置这一位。注意，基于总线主控的快速背对背传送机制不能象基于目标的机制那样，对不同的目标进行快速周期操作。

在所有其它条件下，总线主控必须至少插入一个 IDLE 总线状态。(在不同总线主控所进行的传送之间也总有至少一个 IDLE 总线状态)。注意，多口目标在快速背对背传送期间，当其被真正锁住时，将只锁它们自己。

注意，未加入快速背对背传送的单元不能仅用 IRDY#和 FRAME#就立刻区分出传送的界限(无 IDLE 状态)，在快速背对背传送期间，只有参与传送的目标和总线主控才需要区分这些界限。当最后传送结束时，所有的单元都能观察到一个 IDLE 周期。然而，支持基于目标的机制的单元必须能区分所有 PCI 传送的完成并能识别所有的地址段。



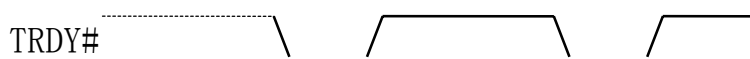


图 4.7 快速背对背操作的仲裁

在图 4.7 中，总线主控在 clock3 完成第一次写操作，在 clock4 产生下一个传送的地址段。目标必须在当前数据传送完成后的那个时钟开始采用 FRAME#。在总线主控可以选择支持这种功能时，目标必须能解码背对背操作。在使 DEVSEL#有效而确认对总线的拥有关系后，目标可以自由地重试这种要求。

提供该特性的最重要的好处就是对使用 PCI 总线进行的“低端”系统配置具有很好的运行性能而代价很小。建议所有新的目标或这种配置中可能使用这种性能的目标都使用这一性能。\*\*\* 在调试过程中在目标中设置，要仔细分析

#### § 4.8 闭锁操作

PCI 提供一种闭锁操作，这种闭锁操作相应于**资源锁定**。当使用资源锁定时，即使有别的总线主控拥有 LOCK#，也能对进行非闭锁操作的单元进行处理，即它允许在锁定状态有非闭锁操作的存在。这就允许将来的处理器作一种贯穿几个操作的、对非锁定操作无干扰的硬件锁定实时的传送诸如视频信号这样的数据。这种机制基于只锁定作为初始锁定操作的目标的 PCI 资源。该机制与现存用于闭锁的软件完全兼容。

在资源锁定中，一个操作的闭锁性由操作的目标来保证，而不是把其它单元从正在操作的总线上排斥掉。锁定的量被定义成对齐的 16 字节。对这个 16 字节块中的任何一个字节的闭锁操作都将要锁定这个 16 字节块。总线主控不能依赖锁定的 16 字节之外的任何地址。对目标的所定，最小是 16 字节，最大到整个资源。 —— **在什么地方设置**

每个提供系统存储器的设备都必须有 LOCK#信号。特别是，如果一个设备采用了可闭锁的存储器，那么它就必须用 LOCK#信号，并且保证完成存储器中的闭锁操作。连接系统存储器的主桥也要用 LOCK#。

LOCK#信号说明一个闭锁操作正在进行。GNT#的有效并不能保证对 LOCK#的控制。LOCK#的控制只能由它自己的协议与 GNT#相互配合后才能获得。对于兼容指令，仲裁器能有选择地将资源锁定转换成“总线所定”，其方式是允许拥有 LOCK#的单元对总线进行闭锁操作，直到 LOCK#释放。

LOCK#的规则对总线主控和目标都适用，PCI 上支持 LOCK#的目标必

须遵循以下规则：

1. 所有支持闭锁操作的 PCI 目标必须按地址去采样 LOCK#信号。当 LOCK#在地址段期间无效时，一个操作的目标锁定它自己。当 LOCK#在地址段期间有效时，当前被锁定的目标对所有传送都应之以 STOP#有效而 TRDY#无效。

2. 一旦锁定建立，该目标保持锁定直到它采样到 FRAME#及 LOCK#均无效或它发出目标失败信号时，就不再锁定自己了。

3. (一旦锁定建立) LOCK#的拥有者就要保证至少锁定资源中的 16 字节。对多口设备，这还要包括不在 PCI 上进行的操作。

如果一个操作的目标只能进行中速和低速解码，它就必须在地地址段期间锁住 LOCK#信号，以便在解码完成之后再去决定该操作是否是闭锁操作。如果目标采样 LOCK#直到它使 DEVSEL#有效，它就不能确定当前操作是一个闭锁操作还是一个和闭锁操作同时发生的操作。一个单元可以用存储“状态”来确定操作是锁定的，但这这就要求在连续时钟中去锁住 LOCK#并加以比较才能决定是否闭锁操作。

为允许对多口设备的别的操作，目标可以在地址段后的那个时钟再采样 LOCK#以确定设备是否真被锁定。如果 LOCK#在地址段期间无效而在地址段后的那个时钟又有效，那么该多口设备已被锁定并要对 PCI 总线主控确保锁定。如果在地址段及其后的那个时钟，LOCK#都无效，该目标就可以对别的要求作出应答，且不锁定。当 LOCK#在地址段期间无效时，当前被锁定的目标才能接受请求。

注意，当 LOCK#有效时，仲裁器必须处于“公正”规则的几种状态下，否则就有可能发生活锁(Livelock)。

现存的不支持 PCI 锁定使用规则的软件有不能正常工作的可能性。推荐对支持 LOCK#并且能向后兼容现存软件的 PCI 驻留存储器(原来的系统存储器)实现资源锁定。

LOCK#由总线主控驱动，在 PCI 上使用 LOCK#的总线主控必须遵循以下规则：

1. 在一个锁定操作期间，一个总线主控只能锁定一个唯一的资源。
2. 锁定不能跨越设备界限。

3. 在一个锁定操作期间，(对齐的)16 个字节是一个总线主控所能计数的作为闭锁的最大资源范围。对这 16 个字节中的任何部分的操作都必须锁定全部 16 字节。

4. 锁定操作的第一个传送必须是读传送。

5. LOCK#在地址段后的那个时钟必须有效并维持有效以保持控制。

6. 如果在数据段完成之前且锁定尚未建立，重试信号已发出，则 LOCK# 必须释放。

7. 无论何时，如果传送由目标失败或由总线主控失败终止，LOCK# 必须释放。

8. 在连续锁定操作之间的至少一个 IDLE 状态中，LOCK# 必须释放。

#### § 4.8.1 开始闭锁操作

无论何时，LOCK# 有效则总线主控就标注 LOCK# 忙；LOCK# 和 FRAME# 都无效时就标注 LOCK# 不忙。

当某一单元需要闭锁操作时，在发出 REQ# 之前，它要检查内部 LOCK# 的跟踪状态。如果 LOCK# 忙，该单元延迟使 REQ# 有效，直到 LOCK# 不忙。在等待 GNT# 时，总线主控继续监视 LOCK#，如果 LOCK# 又变成忙，总线主控就使 REQ# 无效，因为别的总线主控已获得 LOCK# 的控制权。

当允许总线主控操作总线而 LOCK# 又不忙时，对 LOCK# 的拥有关系就成立。在当前传送已完成，且在总线上只有一个单元能驱动 LOCK# 的情况下，该总线主控就能顺利实现一个闭锁操作。甚至当它们是当前总线主控时，所有别的单元也不能驱动 LOCK#。

图 4.8 说明了一个闭锁操作的开始。在地址段 LOCK# 无效，以要求一个由读命令为开始。在地址段后的那个时钟，LOCK# 必须有效即 clock3 以使目标保持在被锁定状态，而这种状态又允许当前的总线主控保留 LOCK# 的拥有关系直到当前传送结束。

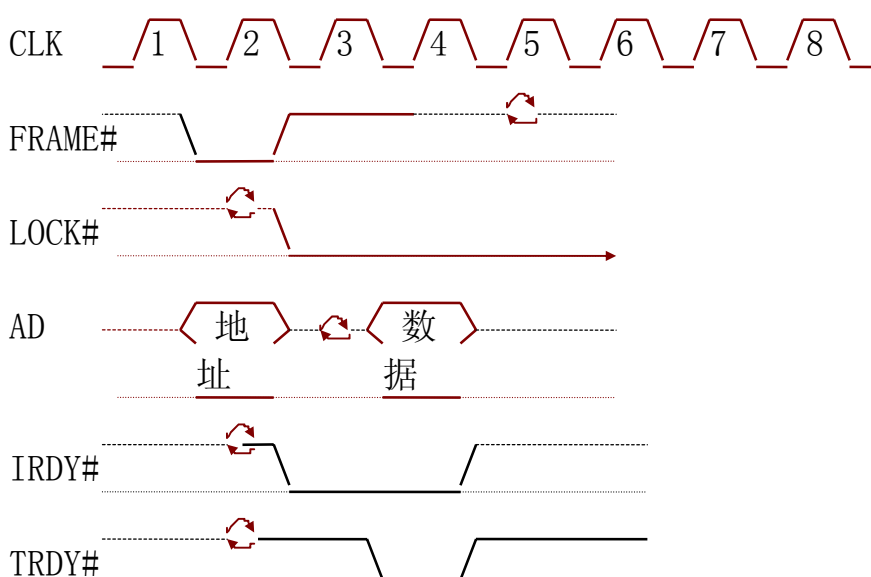


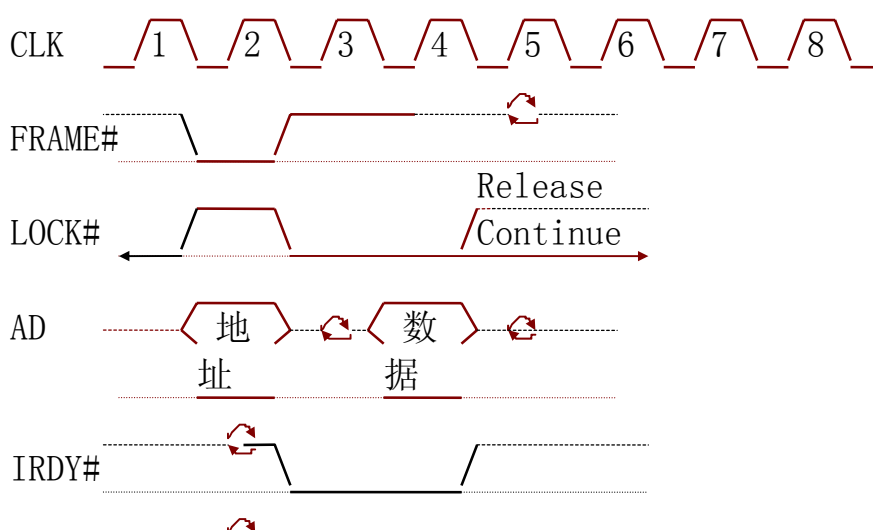


图 4.8 闭锁操作开始

如果目标在没有完成第一个数据段时重试该传送，总线主控不仅必须终止该传送，而且它还必须释放 LOCK#。只有在第一个传送的第一个数据段完成之后(读传送，IRDY#TRDY#有效)，一个锁定操作才在总线上建立。并且总线主控要保持 LOCK#有效，直到锁定操作完成或一个错误(总线主控或目标失败)引起的一个提前终止。即使在锁定操作已建立的情况下，目标的重试或解除连接也属于正常终止操作。如果总线主控被目标用解除连接或重试终止，则说明目标正处于忙状态，没有能力完成所要求的数据段。在目标处于不忙状态且继续通过拒绝别的操作来遵守锁定规则时，它将接受该操作。总线主控继续控制 LOCK#。当 LOCK#有效时，对 PCI 上非锁定目标的非锁定操作也是可以进行的。当闭锁操作完成后，LOCK#就变成无效，别的总线主控就可以争取该信号的拥有关系。

#### § 4.8.2 继续闭锁操作

图 4.9 为总线主控继续一个闭锁操作，然而，这个闭锁操作可能完成所示闭锁操作，也可能完不成。当该总线主控获得对总线的操作权后，它对预先锁定的目标开始作另一个闭锁操作。LOCK#在地址段期间无效以重建锁定。已被锁定的设备接受并应答这个请求。在 clock3，LOCK#有效以保持目标处于锁定状态并允许当前的总线主控对 LOCK#的拥有关系直到当前传送结束。



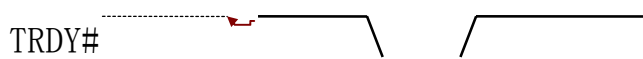


图 4.9 继续闭锁操作

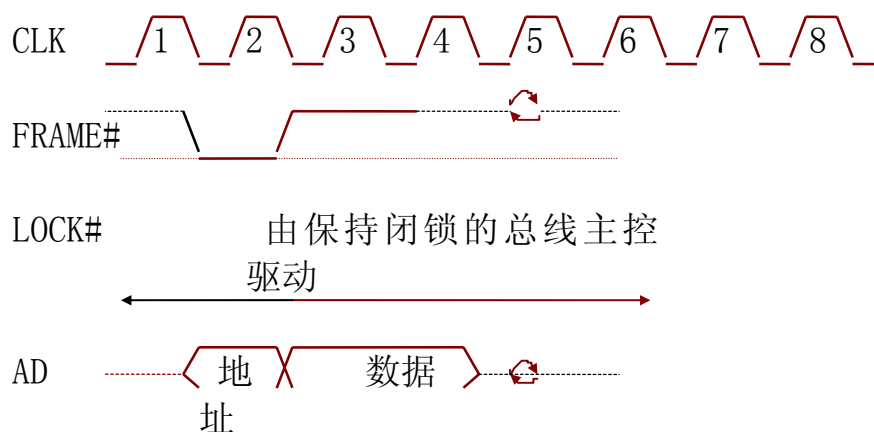
当总线主控继续作锁定操作时，它维持使 LOCK#有效。当该总线主控完成该闭锁操作时，在最后一个数据段之后(clock5)，它使 LOCK#无效。

#### § 4.8.3 对锁定操作进行非闭锁操作

图 4.10 示出一个总线主控试图对锁定目标作非锁定操作。当 LOCK#在地址段有效，且目标被锁定，这就说明了当前被锁定的目标对所有传送都应之以 STOP#有效而 TRDY#无效，因而对锁定目标作非锁定操作没有结果。一个被锁定的目标在决定它是否作出应答时忽略 LOCK#。由于 LOCK#和 FRAME#在地址期间有效，未锁定的目标不会进入锁定状态。

#### § 4.8.4 完成闭锁操作

在闭锁操作的最后传送期间，LOCK#无效，于是目标将接受请求，然后又重新有效直到闭锁操作成功地终止。在闭锁操作完成后，总线主控可以在任何时候使 LOCK#无效。然而，建议(但不是要求)在锁定操作的最后数据段完成后，LOCK#和 IRDY#一起无效。在别的时候释放 LOCK#可能会引起一个顺序传送被不必要的重试终止。只要 LOCK#和 FRAME#都无效，已锁定的单元就不会再锁定自己。如果某一总线主控想在总线上作两个独立的闭锁操作，那么它必须确保在 FRAME#和 LOCK#均无效的两个操作之间，至少有一个时钟。这样就能保证任何被第一个锁定的目标在第二个操作开始之前释放。



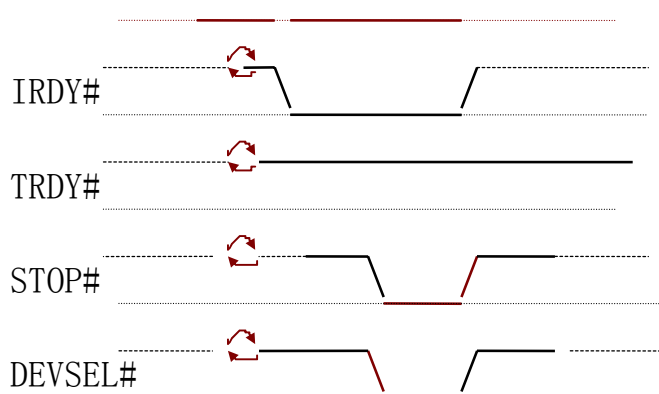


图 4.10 对锁定单元的非闭锁操作

#### § 4.8.5 对 LOCK#和高速缓存器的连续回写的支持

如前所述的资源锁定在使用高速缓存器的连续回写时有发生死锁的可能性。在支持高速缓存回写并且使用了一个已完成锁定的资源时，如果软件允许锁定跨越高速缓存线界限，则死锁就可能要发生。这种可能性的一个例子是锁定跨越高速缓存线  $n$  和  $n+1$  时。缓存线  $n+1$  已被高速缓存器改变。某一总线主控通过读高速缓存线  $n$  而建立了一个锁定。该锁定操作通过读高速缓存线  $n+1$  而继续。对  $n+1$  的监视导致了 HITM，它说明已监测到一条被改变的高速缓存线。因该目标只能接受从 LOCK#拥有者处来的操作，改变后的高速缓存线的回写就要失败。这就会导致死锁，因为只有改变后的高速缓存线之回写完成之后才能进行读，而只有 LOCK#释放后才能进行回写。

这种死锁可以通过要求支持可高速缓存回写存储器的目标在被锁定后仍允许回写而得以避免。

目标可以通过地址段期间(或在地址排队情况下 SDONE 有效后的那个时钟)SDONE 和 SBO#的状态来区分回写和别的写传送。在地址段期间，如已说明 CLEAN，则当前传送要么是 CLEAN，要么就是回写。在地址段期间，从 STANDBY 到 CLEAN 的变化说明某条高速缓存线的复原。在地址段期间，从 HITM 到 CLEAN 的变化说明回写由监视所引起，由监视引起的回写的目标就算已被锁定，也要接受该回写。该目标可以有选择地接受高速缓存线的复原，但如它已被锁定，则不要求这样做。当目标锁定后，所有别的操作都被它用重试终止(主意，由监视引起的回写的目标不能终止这种传送，直到高速缓存线已传送，这意味着对监视引起的回写不能作重试和解除连接)。

#### § 4.8.6 完整的总线锁定

通过在 LOCK#有效时，仲裁器不允许别的单元操作总线，可将 PCI 资源锁定转换成完整的总线锁定。在完整的总线锁定情况下，当一个闭锁操作正在进行时，别的总线锁定不能进行。若锁定顺序中第一个操作要重试，那么总线主控就必须使它的 REQ#和 LOCK#无效。若第一个操作正常完成，那么完整的总线锁定就建立了，并且仲裁器就不会再允许别的单元操作总线。如果完整的总线锁定建立后，仲裁器又允许别的单元操作总线，那么仲裁器就必须改变别的单元以便近似获得完整总线锁定。完整的总线锁定对系统性能有很大影响，尤其是视频子系统。

与完整的资源锁定及回写高速缓存存储器类似，完整的总线锁定也存在着死锁的可能性。支持完整的总线锁定的仲裁器，在一个锁定已进行后，必须允许高速缓存器对总线进行操作，以完成由于对一改变的高速缓存线的监视而引起的回写(要求目标在锁定后接受回写，因为它不能说出正在使用的是完整的总线锁定还是资源锁定)。

### § 4.9 PCI 协议对 Cache 的支持

#### § 4.9.1 Cache 的作用

近年来，由于 VLSI、RSIC 和高速逻辑设计技术的飞速发展，微处理器(MPU)的工作主频和运算速度成倍增长。工作主频处于 100~400MHz(时钟周期为 10~2.5ns)，指令执行速度高达 12 亿次。而 DRAM 存储器读/写周期约 60ns，如果微处理器直接访问 DRAM，由于两者速度差异较大，降低微处理器的速度。解决的办法是在微处理器的寄存器与 DRAM 存储器间设置高速缓存 Cache，以形成三个层次。三部分的速度水平大致如下：MPU 10~2.5ns；Cache 10ns 左右；DRAM 60ns 左右。Cache 由 SRAM 组成，可在 MPU 内部或片外实现。PCI 高速缓存支持能力在 PCI 存储器单元和桥路(或高速缓存单元)之间提供了一种标准接口，这种接口允许使用对高速缓存连接的监视机制。它改进了性能，同时也引入了下述问题：1). MPU 读/写数据的源或目的是在 Cache 还是在 DRAM 中？ 2). 访问共享存储器时，其中的内容与 Cache 中的内容是否一致，即 Cache

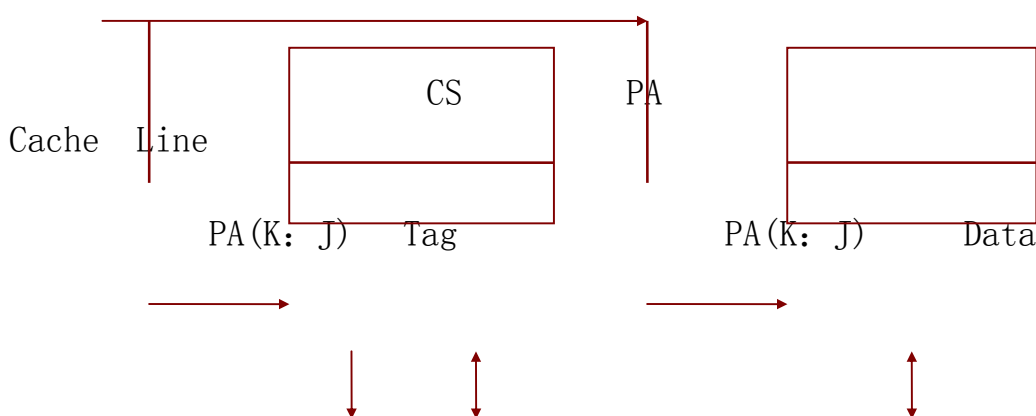
Coherence 问题。PCI 协议通过提供对 Cache 的支持解决了上述问题。

Cache 数据一致性的基本含义是：在 MPU—Cache—Memory (简记为 Mem, 下同) 的系统中, Cache 将保留 Mem 的部分副本, 并依据某种策略, 如 Write Back, 对副本进行修改。由于 Mem 是共享的, 必须保证 Cache 以外的任何访问源 R/W 共享 Cacheable 存储器时, 能读到 Cache 与 Mem 两者中最新的数据; 在写操作时, 能使两者的数据相同或只保留 Mem 一个副本。

在 X86 微机系统中, 一级 Cache (L1—Cache) 在 MPU 片内, CACHE/BRIDGE 是二级 Cache (L2—Cache) 和系统桥路控制器的组合体。Mem 可分为两类: 一类为系统 Mem, 它不直接连在 PCI 总线上, 总线主控经过桥路访问它。在访问过程中, 桥路自动完成对 Cache 数据一致性的检查与处理, 其过程对总线主控透明。简要过程如下: 桥路接收到总线主控的 Read Mem 命令后, 同时启动 Mem Read 和检查 MPU 内部的 L1—Cache (假定 L2—Cache 不存在) 是否有相同的副本。如果命中且该副本已被修改, 则将 Cache 的副本回写到 Mem 中, 同时改变 Cache 副本的状态为非修改状态。桥路从回写的数据中取出所需的部分送给总线主控, 到此读命令完成。对写命令, 桥路将数据直接写入 Mem 同时作废或修改 Cache 中的相应副本。第二类为 PCI—Mem 它独立地直接连在 PCI 总线上, 可被 MPU 和一个或多个总线主控共享。当 PCI—Mem 的内容是可被高速缓存时, 称其为 Cacheable, 否则为 non—Cacheable.。

#### § 4.9.2 Cache 的组织和访问

Cache 的组织如图 4.11 所示;



CS PA Data

图 4.11 Cache 的组织

Cache 以 Cache Line 为单位进行组织。每个 Line 通常有  $8 \times n\text{Byte}$  ( $n=1, 2, \dots$ )。除数据存储单元外, 还配有相应的 Tag, 它由此 Line 对应的物理地址 (PA) 和状态 (CS) 组成。地址位  $K$  由 Cache 容量确定, 如  $2^K$ ,  $J$  由 Line 的大小决定, 例如  $J=4$ , 则 Line 为 16 字节。每个 Line 或者是有效的, 或者是无效的, 不存在部分有效的 Cache Line, 它有如下状态:

- Invalid: 数据无效;
- Clean: 数据有效但未被修改;
- Dirty: 数据有效, 但已被修改, 与 Mem 中相应的数据不一致。

对 Cache 的访问是先用物理地址 PA ( $K: J$ ) (或虚地址) 访问 Data 和 Tag。然后, 比较 PA 和检查 CS, 以确定所需的 Cache Line 是否在 Cache 中。如果 CS 为 Invalid 或 PA 不匹配则称为未命中 Miss, 反之, 为命中 Hit。

由于 Cache 中的内容是 Mem 内容的副本, 依据 MPU 对 Cache 写操作是否修改相应 Mem 中的副本, 提出了两个广泛使用的 Cache 访问策略:

#### 1. 通写 (Write-through)

MPU 写 Cache 时, 同时修改相应的 Mem 单元, 使两者相对应的 Line 数据相同。当这样以来, 写 Cache 时间与写 Mem 时间相同, 无加速作用。

#### 2 回写 (Write-back)

MPU 写 Cache 时, 仅修改 Cache, 此时 Cache Line 的内容与 Mem 中相应 Line 的不一致。这时, 读/写 Cache 的时间都比读/写 Mem 的时间短, 有较高的加速作用。

两种策略下, MPU、PCI 总线主控访问共享 Cacheable PCI-Mem 的操作比较如表 4.1 所示:

表 4.1

操作	Write-through	Write-back
MPU-Write-Mem	Hit: 修改 Cache 和 Mem Miss: 修改 Mem	Hit: 修改 Cache Miss: 修改 Mem
MPU-Read-Mem	Hit: 从 Cache 中读数据 Miss: 从 Mem 中读一个 Cache Line, 写入 Cache, 同时将 MPU 所	Hit: 从 Cache 中读数据 Miss: 从 Mem 中读一个 Cache Line, 写入 Cache, 同时将 MPU 所



	需数据 送回	需数据 送回
PCI 总线 主 控 Write-Mem	写 Mem, 检查 Cache, 若 Hit, 则该 Cache Line 将被更新	桥路检查 Cache 若 Miss 则直接写 Mem; 若 Hit-Clean 则写 Mem 并失效 Cache Line; 若 Hit-Dirty, 桥路强制 PCI -Mem 终止此次写操作, 然后 回写已修改的 Line 到 Mem, 失效此 Line, 回写完成后, 允 许 PCI 总线主控重新启动被 终止的写操作。
PCI 总线 主 控 Read-Mem	直接从 Mem 读数。	桥路检查 Cache 若 Miss 或 Hit-Clean 则直接从 Mem 读数; 若 Hit-Dirty, 桥路强 制 PCI-Mem 终止此次读操 作, 然后回写已修改的 Line 到 Mem, 改变此 Line 的状 态为 Clean, 回写完成后, 允 许 PCI 总线主控重新启动被 终止的读操作。

可见, 在 Write-through 策略下, PCI 总线主控访问共享 Cacheable PCI-Mem 时, 按正常的存储器操作完成, 对 Cache 的处理由桥路完成, 无需总线主控感知, 为了使总线主控对存储器的操作和桥路对高速缓存的处理同步完成, 桥路需要提供一条信号线通知存储器, 桥路何时完成 Cache 操作, 以便作为目标的 PCI 存储器在 Cache 操作完成后执行存储器读写/操作。

而在 Write-back 策略下, 由于在 Hit-Dirty 条件下, 需将 Cache 中被修改的 Cache Line 回写到 PCI 存储器中, 所以, 除了有与 Write-through 中相同的指示信号外, 还需增加指示 Cache 回写的信号线, 这便是 PCI 协议 Cache 的两个信号线 SDONE 和 SB0#。

## § 4.9.3 PCI 协议下 Cache 的状态

SDONE 为 Cache 检查完成标志, SBO#为是否命中了已被修改的 Line 标志。它们在桥路/高速缓存和所要求的存储器目标之间传递高速缓存状态信息。任何支持可高速缓存存储器的 PCI 目标都必须监视高速缓存支持引脚并作出适当的应答。不可高速缓存的目标可以不理睬 SDONE 和 SBO#, 这可以节省一点操作延迟。

因为 PCI 允许长度不定的猝发传送, 作这种操作的可高速缓存的目标, 在对超过一条高速缓存线界线的存储器范围进行操作时, 必须解除连接。这意味这任何可高速缓存的目标必须知道高速缓存线范围, 或是利用高速缓存线寄存器, 或是由硬件设置此参数。如果允许作跨越高速缓存线界线的猝发传送, 高速缓存相关性就可能“崩溃”(相应地, 桥路/高速缓存能监视该传送并为监视产生下一个高速缓存线地址)。

在此定义的 Cache 状态是桥路依据 PCI 总线主控启动的存储器读/写操作, 对 Cache 进行 Hit 及状态检查后在 PCI 总线上报告的检查结果(只作为输出)。

SDONE	SBO#	状 态
0	×	STANDBY
1	1	CLEAN
1	0	HITM

在 Write-back 策略下, 各状态的具体含义如下:

STANDBY 表示下述三个状态之一:

- (1) 高速缓存器当前未监视地址但已做好监视;
- (2) 正对接收(锁存)的第一个 Mem 访问地址作 Cache 检查。同时, 可接收第二个地址;
- (3) 已接收到两个地址, 正对第一个地址作 Cache 检查, 完成后, 如果第二个地址仍有效, 则对第二个地址作 Cache 检查。

CLEAN 说明没有高速缓存冲突且存储器操作可正常完成:

- (1) 读/写操作没有命中;
- (2) 读/写操作命中了未修改的 Line;
- (3) Memory Write and Invalidate 命令操作命中了已修改的 Line。

由高速缓存存储器写命令或存储器写命令所引起的对一条未变化的高速缓存线的回写是不要求的，当高速缓存是当前总线主控且正在回写一条变化的高速缓存线时，它将在地址段期间发出 CLEAN 信号。

HITM 说明监视碰到了一条变化了的高速缓存线，且要求高速缓存把回写这条变化的高速缓存线作为它的下一个操作。高速缓存将保持这种状态直到回写发生。当 HITM 在总线上出现时，所有其它可高速缓存的操作都由存储控制器以重试来终止这次传送，允许回写产生，然后由重试终止的那个单元重新请求传送。在对变化的高速缓存线的回写过程中，高速缓存在地址段由 HTIM 转到 CLEAN。

为能有效地使用 PCI 总线，可高速缓存存储器控制器(相应于存储器控制器)和高速缓存/桥路都要跟踪总线操作。为减少高速缓存传送的重试终止，参与可高速缓存传送的单元要能锁存两个地址，连在 PCI 总线上的 Mem 都不知道其它 Mem 是不是可高速缓存的，这便要求每个 Mem 控制器实时监控 SDONE 和 SBO#信号，以掌握 Cache 检查的结果，特别是自身已被 PCI 总线主控访问时，确定本次读写操作是终止(HITM)还是继续完成。由于不可高速缓存的存储控制器不关心 SDONE 和 SBO#，一旦被启动，便义无反顾地执行到完成。另外，若只在桥路和可高速缓存的存储控制器中只设一个用于 Cache 检查(Bridge)和监视(Mem)的地址锁存，则在不可高速缓存的存储器和可高速缓存的存储器访问交替进行使，将使对可高速缓存的存储器的访问无法完成，其基本过程如下：

首先，一个不可高速缓存的存储器读/写被启动，桥路锁存此次读/写的地址，并开始对 Cache 的检查。由于可不高速缓存的存储器对 Cache 检查结果不关心，在结果出现前，完成了读/写操作，释放 PCI 总线。随后，一个可高速缓存的存储器读/写被启动，因只能锁存单个地址，且桥路正对前一个地址作 Cache 检查，所以刚启动的可高速缓存的存储器读/写被强行终止。假定在终止过程中，检查结果报出，地址锁存器控。若此时，PCI 仲裁器准许启动另一个不可高速缓存的存储器操作，则将重复上述过程。最终将使可高速缓存的存储器读/写操作无法完成。解决的办法是可高速缓存的存储器控制器和桥路可分别锁存两个访问地址。

#### § 4.9.4 Cache 检查状态的转换关系：

##### 1. Write-back 策略下的状态转换

### (1) STANDBY→CLEAN→[CLEAN]→STANDY

这是正常情况，高速缓存保持 STANDBY 状态，直到监视完成，然后发出 CLEAN 信号说明传送将正常完成。当对第一个传送的监视已完成而第二个地址尚未到达时，高速缓存转换到 STANDBY 状态。如果第二个地址已锁存且高速缓存是第二个传送的总线主控时，高速缓存将在两个连续时钟内发出 CLEAN 信号，条件是传送是对高速缓存线的回写，或知道监视是 CLEAN，否则高速缓存转换到 STANDBY。

### (2) STANDBY→HITM→CLEAN→[CLEAN]→STANDY

这是在监视中监测到了一条变化了的高速缓存线的情况，一旦高速缓存发出 HITM 信号，它将保持这种状态直到变化的高速缓存线被回写。高速缓存变换到 CLEAN 说明它正在进行回写。在 CLEAN 之后，高速缓存将发出 STANDBY 信号，说明它已准备好监视一个新的地址。如果高速缓存是第二次传送的总线主控，而这次传送又是对高速缓存的回写，或是知道监视是 CLEAN 时，它将继续发出 CLEAN 信号，否则，它转换到 STANDBY。

## 2. Write-through 策略下的状态转换

### STANDBY→CLEAN→[CLEAN]→STANDY

这与回写高速缓存大致一样，只是不用 SB0#信号。存储器控制器监视总线，每当 SDONE 有效，存储器就能允许别的可高速缓存的传送完成。如果高速缓存是第二次传送的总线主控，而这次传送又是对高速缓存的回写，或是知道监视是 CLEAN 时，高速缓存将继续发出 CLEAN 信号，否则它转换到 STANDBY。建议能高速缓存的目标采用 SB0#和 SDONE。

对于每次在总线上出现的 FRAME#，高速缓存在它已监视地址后，使 SDONE 有效。若 FRAME#有效两次而 SDONE 无一次有效，如果第二个传送是可高速缓存的，它就不能完成。如果第二个操作是可高速缓存的存储器控制器必须插入等待状态直到上一个监视完成(SDONE 有效)。如果第二个操作是不可高速缓存的，该操作能完成并且高速缓存将不监视地址。这时，只有一个地址未完成。

## § 4.9.5 时序关系说明

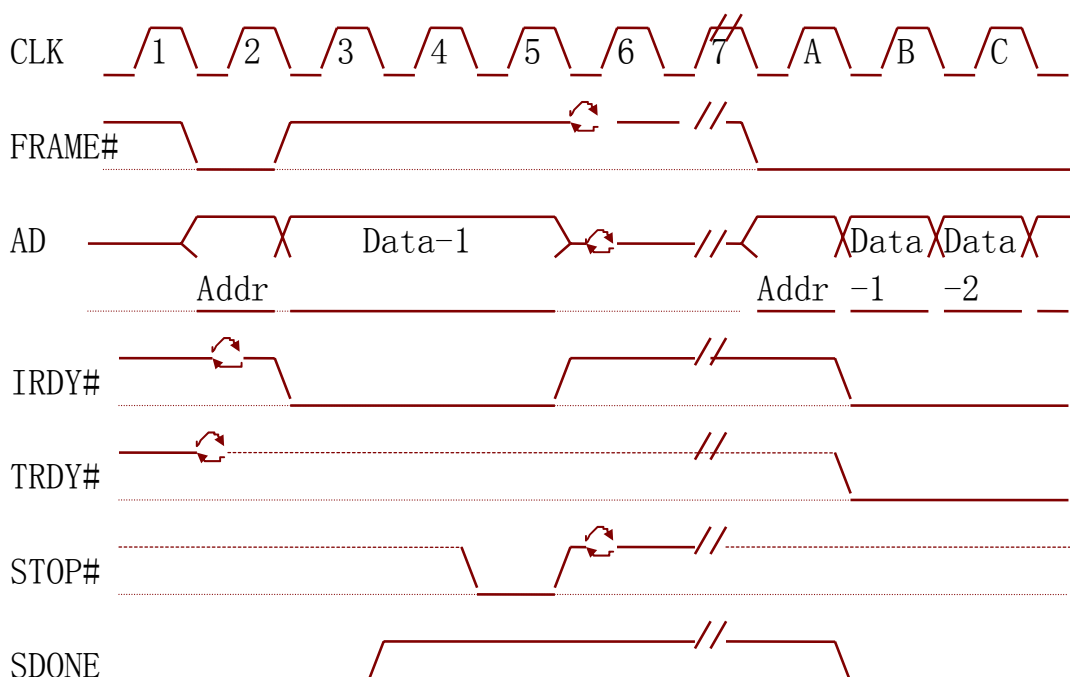
可高速缓存的存储器读/写被启动后，目标总是发出 TRDY#无效，等待桥路发出 Cache 检查结果。据此来决定是终止还是完成此次操作。而不可高速缓存的存储器读/写操作则不管 SDONE 和 SB0#，按正常时序完

成操作。以下各图中假定时钟 1 时总线从 IDLE 状态开始。

碰到变化的高速缓存线并随后进行回写的时序关系如图 4.12 所示

— 在时钟 2 地址锁存，开始第一个传送，目标保持 TRDY#(插入等待状态)直到监视完成，高速缓存在时钟 4 使 SDONE 和 SBO#都有效，说明监视碰到了变化的高速缓存线(一旦 SBO#有效，就必须保持有效直到 SDONE 有效)。因为传送的目标是可高速缓存的，所以在时钟 5，它使 STOP#有效，从而终止传送。这样就允许发出 HITM 信号的高速缓存回写变化的高速缓存线到存储器。在读传送中，如 HITM 出现，储存控制器必须使 AD 线三态。当总线上出现 HITM 信号时，所有对可高速缓存的目标的传送都要由重试终止。

虚线说明自监视在第一次传送中被标志以来，不可高速缓存传送有可能完成，可高速缓存传送也可能会开始，但因 HITM 信号已发出，故必须用重试来终止。回写传送发生在时钟 A，在地址段高速缓存由 HITM 到 CLEAN 的转变向存储器表明监视回写已开始，且要求它接收这条高速缓存线上的所有数据(如果存储控制器不能完成这次传送，它必须插入等待状态直到它能完成。这种情况只有在可高速缓存的目标有内部冲突时才会发生)。在回写期间，高速缓存和存储控制器都可以插入等待状态。在时钟 B 高速缓存由 CLEAN 到 STANDBY 的转变表明高速缓存可接收下一个地址。回写完成后，总线返回到正常操作，可高速缓存传送将继续进行。

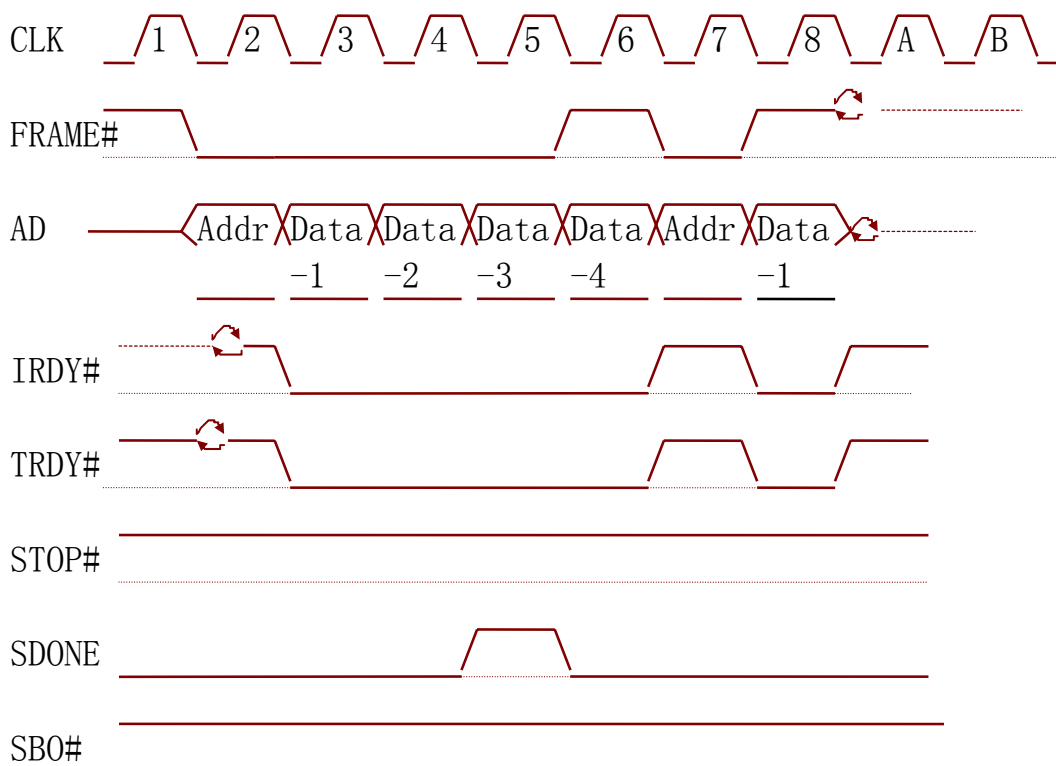






图—4.12 紧随回写操作的时序关系

对于高速缓存存储器写 (Memory Write and Invalidate) 命令，高速缓存在处理该命令上有几种选择。因为总线主控保证高速缓存线上的每个字节都将要改变，Mem 不等检查结果，按正常写操作时序完成。桥路失效相应 Cache Line，高速缓存只是简单地发出 CLEAN 信号，甚至在碰到变化的高速缓存线时也是这样。

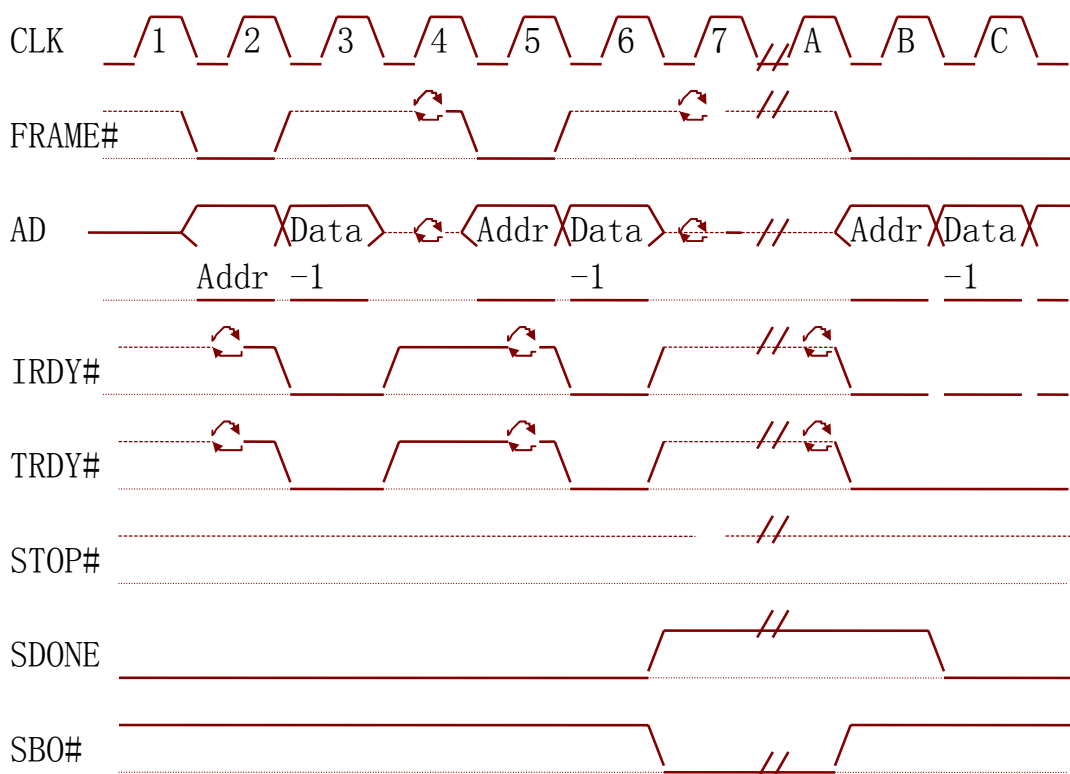


图—4.13 高速缓存存储器写命令

在图 4.13 中，高速缓存在时钟 5 发出 CLEAN 信号，说明要么是监视错过或碰到一条变化的高速缓存线。一旦高速缓存表明了 CLEAN，它就要准备监视出现在总线上的下一个地址。如果 SB0# 在时钟 5 有效，就表明监视碰到了一条变化的高速缓存线并将回写。高速缓存可以象对待别的命令一样对待高速缓存存储器写命令，并允许总线上出现 HITM 条



件(这个回写在总线上引起了一个没要求的额外传送)。高速缓存会在说明监视结果之前花费固定数量的时钟去对待 TRDY#有效。如果在结果出现之前 TRDY#有效, 建议高速缓存放弃这条高速缓存线并发出 CLEAN 信号。如果 TRDY#仍未有效, 高速缓存就继续提供监视结果。然而, 等待 TRDY#的时间必须是恒定的, 因为存储控制器在继续传送之前, 通常要等 SDONE 有效。



图—4.14 数据传送—碰到变化的高速缓存线并随即回写

在图—4.14 中, 第一个传送开始于时钟 2 并且结束于时钟 3。当对第一个传送的监视正在进行时, 另一个传送在时钟 5 开始于时钟 6 结束。如果第二个传送完成时对第一个传送的监视仍在进行, 那么第二个传送就是不可高速缓存的传送。在时钟 7, 第一个传送监视完成。一旦 FRAME# 有效, 且一个监视正在进行, SDONE 和 SBO# 的状态只对第一个地址有意义, 直到 SDONE 有效。一旦 SDONE 有效, 那么下一次再有效时就只适应于第二个传送了, 如果在上图中 SDONE 是在时钟 5 有效而不是在时钟 7 有效, 那么监视的结果对第二次传送没有影响, 即便它在第二次传送期间发出。

为减少可高速缓存的存储器写操作等待 Cache 检查结果的时间,改善性能,允许短数据写(如单个 32 位数据)在结果出现前完成。这具有“赌”的性质,要求存储控制器能保留刚完成的写操作的地址和数据,以便在 HITM 时,对回写的数据依据刚完成的写操作的数据进行修改。时序关系略。

当总线上呈现 HITM 时,仲裁器就要遵循一定的算法顺序,否则就会发生活锁,当两个高优先级的单元正在操作可高速缓存的存储器,而使具有变化的高速缓存线的高速缓存无法完成回写时,就会发生活锁。当总线上呈现 HITM 时,所有可高速缓存的传送都要由重试来终止。

建议当系统中有高速缓存时,仲裁器可选择将高速缓存的 REQ# 接到固定输入,以便当总线呈现 HITM 时,它的优先级能得到提高。这样就确保回写时,被重试终止的可高速缓存的传送保持最小且延迟也会缩小。

当系统中使用了高速缓存时(尤其是回写高速缓存),有目标所造成的延迟必须增加到它对变化的高速缓存线所作的回写的时间中去。这个增加的值依赖何时回写能操作总线的仲裁算法。

#### § 4.10 其它总线锁定

##### § 4.10.1 设备选择

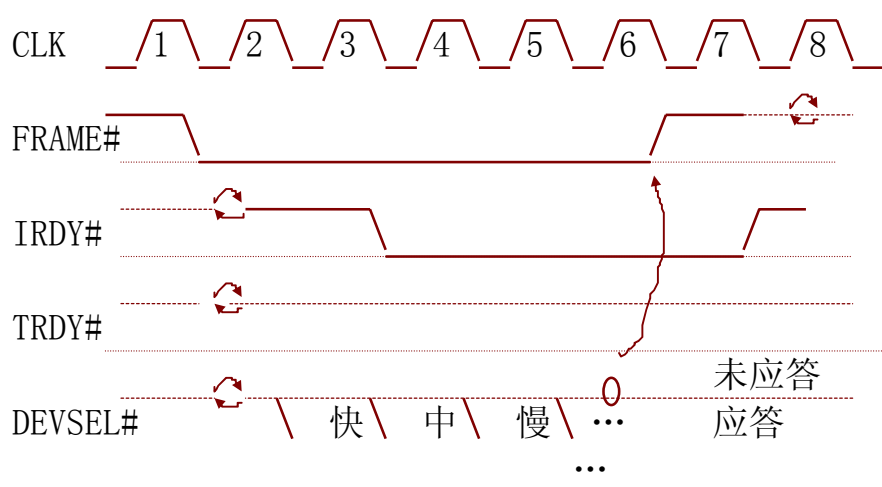


图 4.15 DEVSEL#有效

如图 4.15, DEVSEL#由当前传送的目标驱动。DEVSEL#可以在地址段后 1、2、3 个时钟内驱动, 并且要和 PCI 配置空间状态寄存器中的说明一致。如果在 FRAME#有效后 3 个时钟内没有单元驱动 DEVSEL#有效, 则做反解码的单元就响应请求并使其 DEVSEL#有效。如果该系统没有反解码单元, 总线主控就不能知道有效的 DEVSEL#, 也无法利用总线主控失败机制去终止这次传送。目标设备最好能在 FRAME#有效后 1 到 2 个时钟内完成解码并使 DEVSEL#有效。

目标必须在开放其 TRDY#、STOP#或数据(读)时或在此之前使 DEVSEL#有效。除目标失败外, 在所有其它情况下, 一旦 DEVSEL#有效, 它必须保持 DEVSEL#有效直到 FRAME#无效(IRDY#无效)且完成最后的数据段。对正常的总线主控终止, DEVSEL#必须与 TRDY#同时有效。

目标在驱动/有效 DEVSEL#或别的任何目标应答信号之前, 必须做完全解码。在完全解码之前驱动 DEVSEL#, 然后再在别的总线周期去解码是错误的。在非配置命令中, 目标在 DEVSEL#有效之前必须用 FRAME#去开放 AD 线。在配置命令中, 目标在 DEVSEL#有效之前, 必须用 FRAME#去开放 IDSEL 和 AD[1..0]。——AD[].oe=!FRAME# | !IRDY#

如果第一个操作映射到某一目标地址范围, 它就使 DEVSEL#有效以确认该操作。但如果总线主控想作跨越资源界限的猝发, 该目标就要申请解除连接。

当某一目标确认一个 I/O 操作, 并且字节允许表明有一个或几个要操作的字节位于该目标的地址范围之外, 它就必须发出目标失败信号。为了解决这类 I/O 问题, 反解码设备(扩展总线桥)应作下列事情之一:

- 对不同设备公用的公共双字的地址做正极性解码, 并用字节允许去检测此问题, 然后发出目标失败信号。
- 将全部操作移到扩展总线上, 放弃在该总线上不能进行的那部分操作(这种情况仅发生在第一个寻址的目标在扩展总线上而其它则在 PCI 上)。

#### § 4.10.2 特殊周期

特殊周期命令在 PCI 总线上提供了一种简单的信息传播机制, 它可以利用 PCI 单元之间的逻辑边带信号传送, 条件是这样的信号传送不要求物理信号的精确时间和同步。

特殊周期命令有时可包含可选的、基于数据的信息，PCI 定序器 (Sequencer) 本身并不对它进行译码，只是在必要时让它通过到与 PCI 定序器有联络的硬件应用环境中去。

特殊周期命令也和其它别的命令一样包含地址段和数据段，地址段以 FRAME#有效来起始，在 FRAME#和 IRDY#都无效时地址段完成。地址段除命令域 C/BE[3..0]#=0001 (特殊周期) 外，不包含别的信息。没有明确的目标地址，但 AD[31..00]都要驱动到稳定电平并保证奇偶校验正确，PCI 单元将不使 DEVSEL#有效来应答它，因而要传播到所有单元。每个接收单元都要检测这些信息对它是否可用。这意味着在这种传送中没有任何形式的目标联络，并且反解码桥路不得将这种总线操作传到它的二级总线上去。特殊周期命令将不会通过桥路。

在数据段期间，C/BE[3..0]#有效，AD[15..00]包含信息编码，AD[31..16]为可选的数据域编码，特殊周期命令的总线主控能插入等待状态但目标不能 (因为没有目标)。信息和其所依赖的数据仅在 IRDY#有效的第一个时钟有效。其中包含的信息和顺序数据段的时间与信息有关。

PCI 总线定序器象别的命令一样起始这种命令，并用总线主控失败来终止它。硬件应用提供象其它任何别的命令一样的全部信息并启动总线定序器。当定序器反映该操作已由总线主控失败终止时，硬件应用环境就知道该操作已完成。在这种情况下，配置状态寄存器中“收到总线主控失败”位不能被置位。特殊周期命令最快可在 5 个时钟能完成，在下一个操作前，要求增加一个转换周期。所以，从一个特殊周期开始到别的周期开始共需要 6 个 PCI 时钟。

当前的特殊周期信息编码如下表所示：

信 息 编 码 (AD[15..00])	信息类 型	含 义
0000H	SHUTDOWN	是一种广播信息，说明该处理器正处于关闭模式
0001H	HALT	是从处理器来的关闭信息，说明它已执行了一条“停机”指令
0002H	X86 特殊 配置	是一种 X86 处理器和芯片集所用的通用编码。 AD[31..16]规定特殊周期信息的特别意义，这些特别意义有 Intel 公司定义并可在产品特别文件中找到。

0003H~FFFFH	保留	
-------------	----	--

### § 4.10.3 地址/数据分步

某单元使适当的信号在几个时钟内分别有效的能力称为分步，这种概念允许带有“弱”

输出缓存的单元驱动一组信号，在几个时钟内到有效状态，由此而减少由每个缓存器所引起的地电流负载。另一种方法允许带有“强”输出缓存的单元在几个时钟的每个时钟沿驱动它的子系统直到这些附属部分全部都被驱动，以此来减少必须同时切换的信号的数量。

任一应用都允许某一单元因价格而综合考虑性能(减少电源/地引脚)。当使用连续分步方法时，要注意避免在每个时钟沿都要采样的关键控制信号和在每个时钟沿可能传送的分步信号之间的相互耦合。对性能要求苛刻的外设，应小心地使用这种“承诺”。

分步只允许在 AD[31..00]、AD[63..32]、PAR、PAR64#和 IDSEL 引脚上进行，因为它们总是由控制信号选中，这些信号只有在选中的时钟沿才被认为有效。在地址段，AD 线有 FRAME#使其选中，在数据段有 IRDY#或 TRDY#(根据数据被传送的方向)。PAR 在 AD 选中后每个时钟沿都隐含地选中，IDSEL 由 FRAME#和配置命令解码的组合去选中。

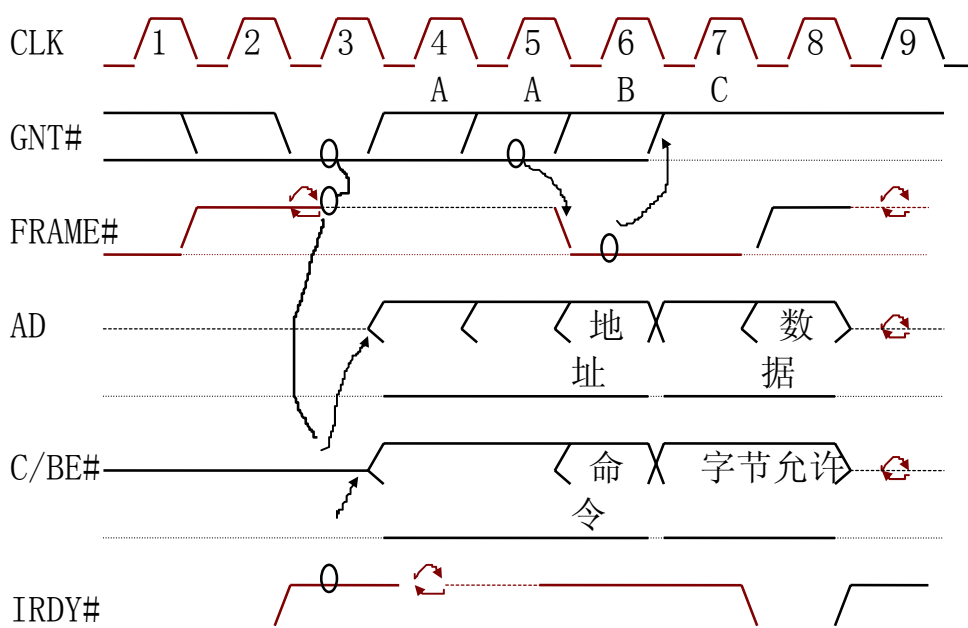




图 4.16 地址分步

图 4.16 所示为总线主控延迟发出 FRAME#直到它成功地驱动了全部 AD 线，一旦总线主控拥有了总线，而且此时总线处于 IDLE 状态，它就能且应该驱动 AD 线和 C/BE#线。但它也可以在使 FRAME#有效之前花费多个时钟去驱动有效地址。然而，延迟发出 FRAME#有效总线主控有失去总线拥有权的危险。如果在注有 A 的那个时钟沿 GNT#无效，该总线主控就要立刻使它的信号处于三态，因为仲裁器已允许别的单元操作总线（这个新的总线主控应该是优先级较高的）。如果在标有 B 或 C 的那些时钟沿 GNT#无效，FRAME#将有效而且传送继续进行。

在配置地址空间操作时，要求额外作设备选择解码，并通过 IDSEL 引脚发出信号到 PCI 设备，IDSEL 引脚如何准确地驱动由主/存储器桥路或系统设计者实现，设计上已允许该信号连接到在配置操作中无别的用途的高 21 位地址线的任何一条上，就能选中 21 个不同的设备，这种方法增加了 AD 线上的额外负担，可以通过适当的阻性耦合来减轻它，这又使得 IDSEL 线上的转换率非常慢，因而需要在 FRAME#之前预驱动地址总线几个时钟。以保证 IDSEL 被采用时它稳定。

#### § 4.10.4 中断应答

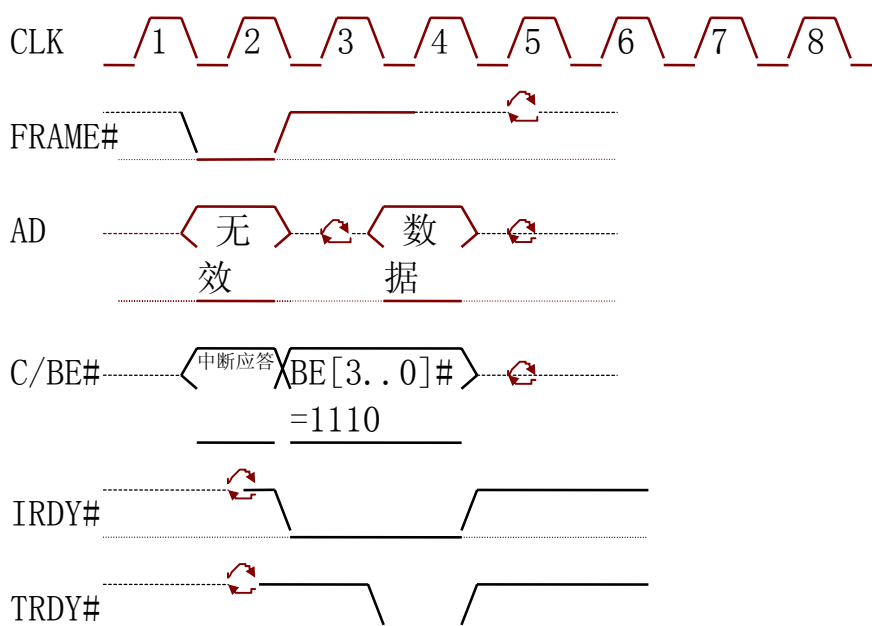


图 4.17 中断应答

PCI 总线支持一种如图 4.16 所示的中断应答周期。此图所示是 PCI 上的一次 X86 中断应答周期的例子。在地址段，AD[31..00]不包含有有效的地址，但必须用有效数据驱动，PAR 有效，并且奇偶性可以检测，在数据段，只有一个字节允许有效。只有一个单元能对中断应答作出反应。做此事的设备必须使其 DEVSEL#有效，否则此周期将被负解码。当 TRDY#有效时，必须返回中断向量。和别的周期一样，中断应答周期也能插入等待状态，并且这种请求可以被目标终止。——？中断矢量由谁提供

与 8259 的双周期应答不一样，PCI 执行的是一种单周期应答。通过放弃处理器第一个中断应答请求，桥路将处理器的双周期格式就轻易地转换成 PCI 单周期格式。

#### § 4.10.5 错误功能

PCI 提供奇偶校验和其它系统错误的检测并发出有错信号。PCI 错误服务区可以包括从对错误(尤其是奇偶校验错误)不感兴趣的设备到检测、发生信号，并修复错误单元。这样就可以使出错的单元得以修复而不影响到未出错的单元。为达到如此灵活性，所有单元对所有传送都要作奇偶校验。

##### § 4.10.5.1 奇偶校验

PCI 上的奇偶校验提供了一种机制，以逐个检查总线主控是否成功地寻址所希望的目标，或它们之间的数据传送是否正确。在地址段和数据段期间，无论是否所有 AD[31..00]及 C/BE[3..0]#线都带有有意义的信息，它们都要参与奇偶校验。未传送数据的字节通道也要求驱动到稳定并包括在奇偶校验中。在配置周期，特殊周期，或中断应答命令中，一些(或全部)地址线未定义，但都要求驱动到稳定数据并包括到奇偶校验计算中去。

所有总线主控和目标都要对从 PCI 总线上获得的地址和数据做奇偶校验并报告奇偶校验错误，为进行奇偶校验，设备要执行下面功能：

1. 设备内部通过锁存 AD[31..00] 和 C/BE[3..0]# 并进行异或产生偶校验。
2. 在下一个时钟脉冲，设备将它的内部计算结果和产生偶校验设备的 PAR 进行异或。
3. 如果这两个值一致，偶校验没错，否则，在下一个时钟脉冲设备或通过使 PERR# 有效 报告数据错误，或通过使 SERR# 有效报告地址错误。

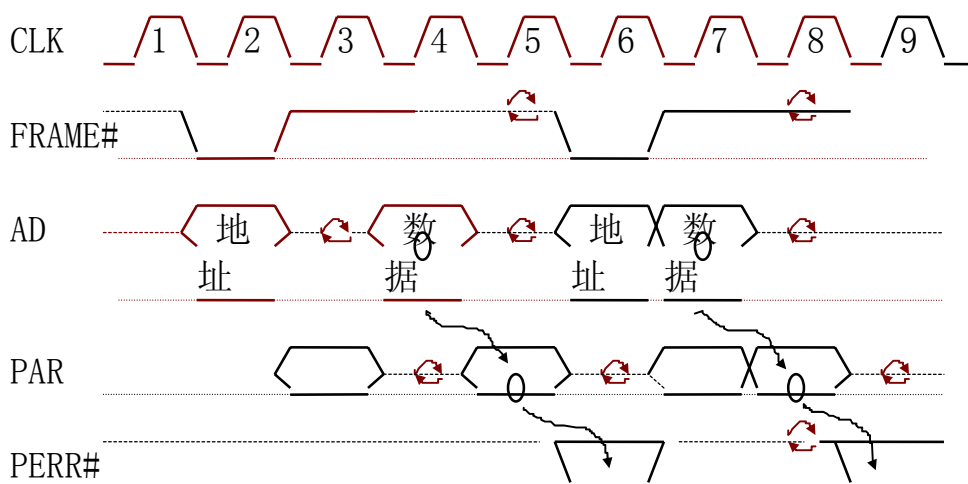


图 4.18 奇偶校验操作

在任何给定的总线段，驱动 AD[31..00] 的设备也必须驱动 PAR 信号，且比相应的地址或数据段 滞后一个时钟。图 4.18 说明了一个带有奇偶校验的读和写传送，在时钟 3 和 7，总线主控驱动地址段的 PAR。读传送中目标驱动数据段的 PAR(时钟 5)，写传送中总线主控驱动数据段的 PAR(时钟 8)。注意，除了一个时钟的滞后外，PAR 和 AD[31..00] 一样也要包括等待状态和转换周期。

在发生了错误的数据传送之后，单元必须在两个时钟内使 PERR# 有效，正在接收数据的单元在检测到奇偶错误之后可以不受约束地使 PERR# 有效(可能在数据传送开始之前就发生了)。一旦 PERR# 有效，它就要保持有效直到实际传送完之后两个时钟。只要 PERR# 有效，总线主控就知道发生了数据奇偶错误，但只有在传送之后两个时钟才知道传送出了错误。

在未插入等待状态的多数据传送情况下，PERR# 将在多个连续时钟内有意义，并可能在其中部分或全部时钟内有效。因为 PERR# 是一个连续三态信号，所以在每个有意义的时钟沿它都应该驱动到正确的电压值。

为使它在每次总线操作结束后返回标称状态，它必须在 AD 线转换周期之后两个时钟驱动到高，历时一个时钟周期(图 4.14 的时钟 7)，之后一个时钟是 PERR#的转换时钟周期(图 4.14 的时钟 8)。PERR#在当前周期不被驱动直到地址段后至少三个时钟周期。

#### § 4.10.5.2 错误反应

当 PCI 检测到奇偶校验或其它系统错误时，可以预计，只要可能，无论何时奇偶错误都会通过这种操作和设备驱动程序链而被返回。从目标到总线主控、设备驱动程序、设备管理器及操作系统的错误反应链都是为了在任何一级上都允许错误弥补。因为通常情况下不可能用特殊错误链来消除系统错误，所以它们就直接反应到系统一级。

在 PCI 错误反应设计中，使用了两个信号 PERR#和 SERR#(引脚)，PERR#专用于反应除特殊周期命令外的所有传送中的数据奇偶错误。它是一个持续三态信号并连到索引 PCI 单元中。总线协议保证 PERR#不会同时被多个总线设备驱动，并且适当的信号转换时间也避免了任何驱动器争用。只有总线主控才能反映读数据奇偶错误，只有选中的目标才能反应写数据的奇偶错误。

必须要检测奇偶性以确定总线主控是否成功地寻址了所希望的目标，数据传送是否正确。在所有情况下，支持奇偶性检测的单元在检测到奇偶错误时，必须设置配置空间状态寄存器的奇偶错误检测到位(Parity Error Detected)。对奇偶校验错误的信号发生和应答都由奇偶错误应答(Parity Error Response)位来控制。除了后面两种专用设备外，所有设备都要用到这一位。如果此位清除，该单元就忽略所有奇偶错误，并认为奇偶正确而完成传送。如果这一位设置，则该单元在检测到奇偶错误时，要使 PERR#有效，其它错误应答根据设备而定。

当一个总线主控检测到数据奇偶错误并使 PERR#有效(在读传送中)或采样到 PERR#有效(在写传送中)时，它必须设置数据奇偶反应位(状态寄存器的第 8 位)，并可继续该传送或终止它。一个检测到奇偶错误的传送的目标可以继续该操作或通过目标终止来结束它。目标绝不能设置数据奇偶错误反应位。当 PERR#有效时，建议让总线主控和目标完成这次传送，对目标而言，PERR#只是一个输出信号，而对总线主控，它可以是输入和输出信号。

当一个操作的总线主控发现在它的传送中发生了奇偶校验错误，它

就应该向系统说明。建议采样中断的方法让总线主控通知其设备驱动程序(或调整状态寄存器,或标志),如果这些方法都无效,作为最后弥补,使 SERR#有效,以便可靠地将错误信息送给操作系统。注意,系统设计人员可能会在中央资源中将所有 PERR#错误系统转换成 SERR#错误信号以便将奇偶错误信息传送给操作系统。

SERR#用于发出所有地址奇偶校验错误和特殊周期命令中的数据奇偶校验错误,并且可能有选择地用于别的非奇偶性或系统错误中。它是漏极开路输出,并可以与所有 PCI 单元之 SERR#线或所以可能同时被多个单元驱动。因为漏极开路信号在每个时钟沿不能产生稳定的信号,一旦 SERR#有效,它的逻辑值必须看成是未定的,直到该信号至少在两个连续的时钟沿被采样到无效。

任何单元,无论是总线主控或是目标,都可以在 SERR#上检查到或发出地址奇偶错误。无论是何种类型的错误,只有命令寄存器中的 SERR#允许位设置成逻辑 1 时, SERR#才可能有效。无论是何种类型的错误,只要某一单元使 SERR#有效时,该单元都要设置配置空间状态寄存器中的系统错误信号位。此外,如果错误类型是奇偶错误时(如地址奇偶错误),在各种情况下,奇偶错误检测到这一位必须设置,但在 SERR#上的信息反应则由命令寄存器中的奇偶错误应答位来制约。

检测到地址奇偶错误的被选中单元将做下列事情之一:

确认周期并象地址正常一样终止;确认周期并用目标失败终止;或是不确认周期,任由传送被总线主控失败终止。该目标不允许用重试或解除连接终止传送,因为已检测到地址奇偶错误。

SERR#与任何 PCI 传送没有时序上的联系(CLK 之外)。然而,错误将尽快地传送出去,最好就在检测的两个时钟内。只有将低脉冲转换成给处理器的信号的中央资源才会在意 SERR#(作为输入)。中央资源如何给处理器发信号要根据系统而定,但应包括产生 NMI,高优先级中断,设置状态位或标志。然而,使 SERR#有效的单元必须使中央资源产生一个 NMI,否则,错误信息就会通过别的机制发出(即中断状态寄存器或标志)。

当奇偶错误应答位处于允许状态时,而且 SERR#允许位也允许时,在下列条件之下单元将使 SERR#有效:

- 地址奇偶错误或特殊周期中的数据奇偶错误被检测到。
- 奇偶错误的检测结果未通过别的机制传送出去(仅对总线主控)。

当 SERR#允许位允许时,在下列条件下单元将使 SERR#有效:

- 总线主控(无驱动程序)参与了被异常终止的传送。



- 重大错误使得该单元丧失正常操作的能力。

注意，所有单元都要求产生奇偶校验(对此要求无例外情况)。用于非奇偶校验错误的 SERR#信号是可选的。然而，必须注意到 SERR#上发出信号将产生 NMI，所以，在使用 SERR#时要很小心。

对于配置周期命令和特殊周期命令，总线主控失败对于桥路来说不属于异常条件。在这种情况下或是能正常弥补的情况下，不能用 SERR#。目标失败通常是目标异常终止，并且在总线主控不能通过自己的设备驱动程序反应错误时，可能将它当作一种错误而由 SERR#来反应出去(近由总线主控)。

因为在 PCI 上要求发出奇偶校验错误信号，故需要 PERR#和 SERR#引脚，然而对以下这两类设备可以不考虑这种要求：

1. 为母板或平面设计的设备，如芯片集。因为它不能用于扩展板，系统商要控制这类设备的使用。
2. 不涉及、不包含、不处理任何代表固定的或有后遗症的系统或应用状态的数据的设备。例如人机界面和视频/音频设备。这类设备只涉及故有的或有后效的系统或应用状态的暂时出现的数据(如像素点)，所以，即使不检测错误也不会产生系统综合性问题。

## 第五章配置空间

除了常规的存储器空间和 I/O 空间外，PCI 规范还规定了配置空间以满足现在及将来系统配置机制的要求。这种配置机制反映了设备的功能和状态，提供了无用户参与的安装、配置和引导，全部设备重定位，由独立于设备的软件设置系统地址映机构，包括中断汇集。

配置空间是为配置、初始化和灾难性错误除了功能而设计的。它只限于初始化软件和错误处理软件使用。所有应用软件必须继续用 I/O 或存储器空间操作来操作设备寄存器。

设备的配置空间必须是任何时候都可操作的，不仅是在系统引导期间。系统配置软件可能需要扫描 PCI 总线以确定当前总线上存在哪些设备，0FFFFH 是无效的销售商标识符，因而，在回答读不存在设备的销售商标识符时，主总线到 PCI 的桥路返回一个全“1”的值，并由总线主控失败终止该操作。

所有 PCI 设备必须把对保留寄存器的配置空间的写操作当成无效操作，即在总线上这些操作正常完成而放弃数据，对保留寄存器或未用寄

寄存器的读操作也要正常地完成但返回的数据是 0。注意，对一些保留作今后用的位软件要能正确处理。在读操作中，软件必须适当的屏蔽提取出已定义的位，并且不能依赖于任何有特定值的保留位。在写操作中，软件要确保保留位上的值得到保护，即保留位位置上的数先必须被读出，与别的位的新值组合后在写回去。

### § 5.1 配置空间的组织

PCI 总线规范规定了 256 字节的配置空间，这个空间分为 64 字节的预定义首区和 192 字节的设备相关区。在每个区中，设备只要必要的和相关的寄存器。预定义首区的前 16 个字节对各类设备而言读是相同的。所有 PCI 兼容设备必须支持首区中的销售商标识符、设备标识符及命令区和状态区，版本号，类别号和首区类型，而对其它寄存器的使用则可根据设备功能来选择(比如作为保留寄存器)。由于配置空间要受主系统的 BIOS 程序管理，因而预定义首区十分重要，如果一个设备支持某个寄存器所关联的功能，该设备必须在已定义的布置中使用它，并利用它的功能。剩下的 48 个字节根据设备所支持的基本功能可以有标题的布置。目前只定义了一种首区类型(00H)如表 5.1 所示

表 5.1 配置空间首区

偏移	名称	属性	功能描述
00H	销售商标识符	只读	由 PCI SIG 分配的、有效的销售商标识符。 0FFFFH 是无效的销售商标识符。
02H	设备标识符	只读	说明特定的设备，由销售商分配。
			提供对设备产生和应答 PCI 周期能力的粗略控制，解码如下： bit0: 设备对 I/O 空间操作的应答。0 不应答，1 应答。 bit1: 设备对存储器空间操作的应答。 0 不允许，1 允许。

04H	命令	rst	bit2: 设备的总线主控能力。0 不能, 1 允许作为主控操作总线。 bit3: 设备对特殊周期的监视。 0 忽略特殊周期, 1 监视特殊周期 bit4: 总线主控产生高速缓存存储器写的的能力。0 用存储器写命令
偏移	名称	rst	代替该命令, 1 总线主控可以产生这种命令。带复位。
	命令	mas	bit5: 控制对 VGA 调色板寄存器的操作。VGA 兼容设备都要用。 0 设备象对待别的操作一样对待调色板操作, 1 允许作特殊调色板监视。
	属性	读/写	bit6: 控制设备对奇偶校验错误的应答。0 忽略奇偶错误并继续正常操作, 1 监测到奇偶错误后必须作出正常的反应。带复位。
		prn	功能描述
		读/写	bit7: 控制设备是否做地址/数据分步。 0 不做地址/数据分步, 常做地址/数据分步的设备必须将该位置 1, 可做可不做的设
		mas	备应使这一位可读/写, 并在 RST#后初始化为 1。
		POST	bit8: SERR#驱动器的允许位, 0 禁止 SERR#输出, 1 允许
			<del>SERR#输出。这一位及第 6 位必须有以便应答</del>
			地址奇偶错误
			bit9: 控制总线主控能否与不同的设备作快速背对背传输。 0 意味着总线主控只能同一个单元作快速背对背传输。 如果目标具有快速背对背能力, 初始化软件可将该位置 1。
			bit10~15: 保留。
06H			用于从与 PCI 总线有联系的事件中解码状态。

	状态	只读	<p>bit0~6: 保留。</p> <p>bit7: 当前目标能否接受快速背对背传送。如果该设备能接受这个传送，这位能置成 1，否则必须为 0。</p> <p>bit8: 这一位只有总线主控能用。当符合三种条件时，该位置成 1</p> <p>1). 总线单元自己使 PERR#有效或知道 PERR#有效，</p> <p>2). 在发生错误的传送中，由总线主控设置该单元，</p> <p>3). 命令寄存器中的奇偶错误应答位已设置成 1。</p> <p>bit10~9: 这两个只读位说明对于除配置读/写外的任何总线命令，</p> <p>某一设备使 DEVSEL#有效的最慢时间。</p> <p>00b 代表快，01b 代表中，10b 代表慢，11b 保留。</p> <p>bit11: 当一个目标以目标失败终止一个传送时必须由该目标置为 1</p> <p>不发出目标失败的设备不需要用到这一位。</p> <p>bit12: 在一个传送被目标失败终止时，总线主控必须设置该位为 1</p> <p>所有总线主控设备都要用这一位。</p> <p>bit13: 在一个传送由总线主控失败终止时，总线主控设置该位为 1</p> <p>所有总线主控设备都要用这一位。</p> <p>bit14: 使用 SERR#允许的设备使 SERR#有效使就要设置这一位。</p> <p>bit15: 无论奇偶错误处理是否被禁止，在该设备监测到一个奇偶错误时，必须设置这一位。</p>
08H	版本号	只读	由销售商定义版本号
			说明设备的通用功能及特殊寄存器级编程接口 基本类别子类别 编程接口 意义





0CH	高速缓存线范围	读/写 mas rst	能产生高速缓存存储器写的总线主控必须用该寄存器以 32 位为单位规定高速缓存线范围,以便参加高速缓存协议的设备知道何时在高速缓存线边界上作重试猝发操作。该寄存器为 0 时,设备就忽略 PCI 高速缓存支持线 (SDONE 和 SB0#),复位时寄存器为 0。
0DH	延迟计时器	mas rst	以 PCI 时钟为单位规定总线主控的延时计时器的值。代表了分配给总线主控的最小时间片段。这个寄存器必须被任何可猝发两个以上数据段的总线主控所写。对于猝发两个或更少数据段的设备,这个寄存器是只读的,且值不大于 16。一般做成高 5 位可写,低 3 位只读。带复位。
0EH	首区类型	只读	bit7: 说明该设备是不是多功能设备,0 单功能,1 多功能。 bit6~0: 说明配置空间中 10H~3FH 的布置,01H 对应于 PCI-PCI 的桥路,00H 对应于目前其它的 PCI 设备。所有别的编码均保留。
0FH	内装自测试 BIST	读/写	bit7: 是否支持 BIST。若支持则返回 1,如果不支持,返回 0。 bit6: 写入 1,产生 BIST。当 BIST 完成后,设备复位这一位。 如果 BIST 在两秒后不能完成,软件使设备失败。 bit5,4: 保留 bit3~0: 0 意味着设备已通过测试,非 0 值为设备失败编码。
10H ~ 偏移 27H	基本地址名称	属性	上电自举程序要判断系统中有多少存储器,系统 I/O 控制器要求多少地址空间,以建立一种稳定的地址映射。bit0 是只读的。 bit0=0: 设备映射到存储器空间。此时 bit3~1 也是只读的。 bit2,1=00 基址寄存器宽 32 位,可映射于 32 位空间任意位置 bit2,1=01 基址寄存器宽 32 位,必须映射于低 1M 空间。

			功能描述 bit2,1=10 基址寄存器宽 64 位, 可映射于 64 位空间任意位置 <del>bit2,1=11 保留 bit3. 设备在某个范围预取时无副作用, 则为</del> 1, 否则应为 0 bit0=1: 设备映射到 I/O 空间。此时 bit1 是保留的。
			28H~2FH, 34H~3BH 均保留,
			30H~33H 为扩展 ROM 基地址。
3CH	中断线	读/写	任何使用中断引脚的设备必须使用它, 由 POST 软件初始化和配置系统时置入数值, 说明中断引脚连到系统中断控制器的哪个中断号上。设备驱动器和操作系统用它来判断优先级和矢量信息。
3DH	中断引脚	只读	说明设备用的是哪一个中断引脚, 1 对应 INTA#, 2 对应 INTB#, 3 对应 INTC#, 4 对应 INTD#。不用中断引脚的设备要将该寄存器置为 0。
3EH	Min_Gnt	只读	Min_Gnt 和 Max_Gnt 以 1/4ms 为单位规定出一段时间反映设备所期望的延迟计时器的值的设置, 为 0 说明无大的要求。 Min_Gnt 规定在 33MHz 时钟下设备所需猝发期间有多长。
3FH	Max_Gnt	只读	Max_Gnt 用来规定该设备需要获得总线操作权有多频繁。
			40H~FFH 是设备相关区, 由设备自己定义。

## § 5.2 补充说明:

命令寄存器决定准备对不同总线命令的响应, 配置软件要设置相应的位, 当命令寄存器写成 0 时, 除对配置操作以外的所有操作, 该设备从逻辑上看便脱离了 PCI 总线。

状态寄存器记录总线事件和错误, 还包含目标设备能力信息, 设备状态寄存器通常用于读操作, 但对那些可以复位, 但还没有设置的位, 写动作稍有不同。如果寄存器被写入, 这样的位就被复位, 在相应位位

置上的数据为 1。例如为了清除第 14 位而不影响其它任何位，将数据 0100 0000 0000 0000 写入寄存器。

设置 PCI 配置空间的一个最重要的功能之一是在地址空间上能重定位 PCI 设备。上电自举软件要判断存在那些设备，是否带有扩展 ROM，系统中有多少存储器，系统 I/O 控制器要求多少地址空间，以在一种设备无关的方式上建立协调的地址映射。

基地址寄存器告诉 PCI 目标哪些地址在终端设备的地址空间内，这些寄存器必须位于目标内，而不是终端设备内，从而使目标能执行地址解码。

一个设备采用高位的数量由这个设备将应答多少地址空间而定。应答 1M 存储器空间的设备(用 32 位基地址寄存器)将设置地址的高 12 位，而其它位将由硬件置为 0。上电即运行的软件可以通过写全 1 到这个寄存器然后再读回这个值来判断这个设备要求多少地址空间。设备将在所有不关心的位上返回 0，有效地规定出所要求的地址空间。

这种设计隐含所用的全部地址空间是 2 的幂，并且是自然排列的，设备可以自由占用比其所需要还多的地址空间，但对于要求少于 4K 字节存储器空间和 256 字节的 I/O 空间的设备，解码会限制为 4K 字节存储器空间和 256 字节的 I/O 空间。占用超过其所用地址空间的设备不要求对未用地址空间作出应答。

在配置空间首区的 10H~27H 这 6 个双字位置用来布置基本地址寄存器。第一个基本地址寄存器总是开始于 10H 处，第二个寄存器根据第一个寄存器的大小可能在偏移量 14H 或 18H 处。后面的基地址寄存器的偏移量由前一个基地址寄存器的大小决定。

典型设备将为其控制功能而要求一个存储器范围。某些图形设备可能要用两个范围，一个给控制功能另一个给帧缓存。要求同时映射控制功能到存储器和 I/O 空间的设备必须采用两个寄存器(一个给存储器，一个给 I/O)。这类设备的驱动器可能只用一个空间，在这种情况下，另一个空间将不用。设备通常允许控制功能映射到存储器空间，这就使得没有 I/O 空间的设备也能实现其控制功能。

### § 5.3 扩展 ROM 的组织

某些 PCI 设备，特别是那些在 PC 体系结构下的插入模块，为扩展 ROM 而要求有局部的 EPROM。始于 PCI 配置空间首区 30H 出的 4 字节寄存器

就是用来控制这些扩展 ROM 的地址和大小信息的, bit0 决定该设备是否支持对其扩展 ROM 的操作, 为 0 时禁止。bit0 为 1 时, 允许使用这个寄存器的别的部分作地址解码。命令寄存器中的存储器空间允许位优先于扩展 ROM 允许位。只有当存储器空间允许位和扩展 ROM 基本地址允许位都设置成 1 时, 设备才必须应答对其扩展 ROM 的操作。

bit0=1 时, 设备支持扩展 ROM, bit10~1 为保留位, bit31~11 位为扩展 ROM 基本地址的高 21 位。设备有效的使用的位的数量由这个设备支持怎样的地址排列而定。例如允许其扩展 ROM 映射到任何 64K 范围的设备将利用这个寄存器中的高 16 位, 剩下的低五位由硬件置为 0。设备无关的配置软件可以通过写全 1 到这个寄存器的地址部分然后再读回这个值来判断这个设备支持怎样的地址排列。设备将在所有不关心的位上返回 0, 有效地规定出它所支持的排列方式。

为了使设备所支持的扩展 ROM 编码在系统自举或特定设备初始化时能被执行, PCI 规范规定了扩展 ROM 中的编码映像机制以适应不同的机器和处理器配置。这些 ROM 编码不是在原有的位置上执行, 而是复制到 RAM 中去执行。

PCI 设备的扩展 ROM 可能包含适应于不同的系统和处理器配置的许多编码映像, 每个映像都必须开始于 512 字节的边界。每个 ROM 映像由两部分信息组成, 其一是 ROM 首区, 要求布置在 ROM 映像的开始处, 其二是 PCI 数据配置区, 必须布置在映像的第一个 64K 字节范围内。

PCI 扩展 ROM 首区的格式如表 5.2 所示:

表 5.2 PCI 扩展 ROM 首区格式

偏移量	长度(字节)	值	说 明
00H	1	55H	ROM 标记, 字节 1
01H	1	AAH	ROM 标记, 字节 2
02H~17H	16H	VV	保留
18H~19H	2	VV	PCI 数据配置指针(参考点为该 ROM 映像开始点)

PCI 数据配置区的格式如表 5.3 所示:

表 5.3 PCI 数据配置区格式

偏移量	长度(字节)	说 明
-----	--------	-----

0	4	字符串“PCIR”。数据配置区标志，P 在偏移量 0 处。
4	2	销售商标识符。和配置空间首区的销售商标识符一致。
6	2	设备标识符。和配置空间首区的设备标识符一致。
8	2	必须的产品数据指针 (VPD) 说明从 ROM 映像开始到 VPD 点的偏移量。这个区域是小尾格式的。VPD 必须在该 ROM 映像的前 64K 字节内。0 值说明 ROM 映像中没有必须的产品数据，VPD 配置的内容在这个规范的未来版本中说明。
A	2	从 PCI 数据配置区的第一个字节开始，以字节为单位说明的 PCI 数据配置长度。
C	1	PCI 数据配置版本级别，这个版本的级别是 0。
偏移量	长度 (字节)	说 明
D	3	类别码，和配置空间首区的类别码一致。
10	2	映像长度，该区域是小尾格式并以 512 字节为单位。
12	2	编码/数据的版本级别，包含 ROM 映像中编码的版本级别。
14	1	编码类型，可以是能被特定的处理器和系统配置执行的二进制编码，也可能是可解释编码。0 IntelX86PC-AT 兼容，1 PCI 的 OPENBOOT 标准，2~FF 保留。
15	1	指示器说明该映像是不是最后一个，bit7=1 最后一个，
15	1	bit7=0, 不是最后一个，bit6~0 保留。
16	2	保留。

在大多数情况下，系统上电自测试 (POST) 编码对待外插 PCI 设备和对待焊接在主板上的 PCI 设备是相同的。例外情况是对扩展 ROM 的处理。POST 码通过两个步骤检查可选 ROM 的存在，首先检查这个设备在配置空间中是否使用了扩展 ROM 基本地址寄存器，如果使用了这个寄存器，POST



必须映射和允许 ROM 到地址空间中的一个未用部分，并检查最前的两个字节是否为 AA55H，如果发现它，则说明 ROM 存在。POST 还要在 ROM 中查找有正确的编码类型的映像，并且，它的销售商标识符和设备标识符对应该设备的相应区域。在找到这个正确的映像之后，POST 复制适当数量的数据到 RAM 中，然后执行这个初始化编码。根据这个区域的编码类型来决定复制的数据量和怎样执行设备的初始化编码。

#### § 5.4 Intel X86, PC-AT 兼容的扩展 ROMS 的进一步说明

Intel X86, PC-AT 兼容的扩展 ROMS 的 PCI 扩展 ROM 首区格式如表—5 所示

表—5 Intel X86, PC-AT 兼容的 PCI 扩展 ROM 首区格式

偏移量	长度(字节)	值	说 明
00H	1	55H	ROM 标记, 字节 1
01H	1	AAH	ROM 标记, 字节 2
02H	1	VV	以 512 字节为单位的初始化范围
03H	4	VV	INIT 功能入口点。POST 对此功能作一个 FAR CALL
07H~17H	11H	VV	保留
18H~19H	2	VV	PCI 数据配置指针(参考点为该 ROM 映像开始点)

在处理每个扩展 ROM 时系统的 POST 编码首先要从 ROM 中复制由初始化范围决定的字节数到 RAM 中，在保持这些 RAM 区域可写情况下调用入口地址在 03H 的中断功能。从而允许中断码存储一些静态数据到 RAM 中去，并调整这个编码的运行时间，使系统在运行时消耗更少的空间。中断返回后这些 RAM 区只读，在系统引导时才不会被改写。

POST 编码必须对带扩展 ROM 的 VGA 设备作特殊处理。VGA 设备的扩展 BIOS 要复制到 0C0000H 处。VGA 设备可以通过检测设备的配置空间的类别码而识别出来。

PCI 兼容的扩展 ROMS 包括一个 INIT 功能，它负责初始化 I/O 设备，并准备运行时间的操作。因为放置编码的 RAM 区域在 INIT 功能执行时

是可写的，在 INIT 功能期间，它能在其 RAM 中存储静态参数。这些数据可以被运行时间 BIOS 或设备驱动器利用。在运行期间，RAM 的这些区域是不可写的。

INIT 功能也能调整运行时间所占用的 RAM 量。可以通过调整映像中偏移量 02H 处的范围字节来做到这一点。这样能帮助在扩展 ROM 范围内 (0C0000H~0D0000H) 节省存储器资源。例如一个设备的扩展 ROM 可能因其初始化和运行时间代码要求 24K 字节，但运行时间代码只有 8K 字节。ROM 中的这个映像将示出一个 24K 字节的范围，所以 POST 编码将全部 24K 字节都复制到 RAM 中。然后在运行 INIT 功能时，它可以将字节范围调整到 8K 字节。当 INIT 功能返回时 POST 看到的运行时间范围是 8KB，它就能复制下一个扩展 BIOS 到最优位置。

INIT 功能负责保证贯穿映像范围的检测是正确的。如果 INIT 功能通过任何途径改变了 RAM 区域，那么就必须计算一个新的检查并且存到映像中。

在入口处，INIT 功能传输三个参数：总线号、设备号、和支持扩展 ROM 的设备的功能号。这些参数可用来操作已初始化了的设备。它们被传到 X86 的寄存器，[AH] 包含总线号，[AL] 高五位包含设备号，[AL] 第三位包含功能号。

典型的映像布置如图 5.1 所示

映像长度是该映像的总长度，  
区  
必须大于等于初始化长度。初始

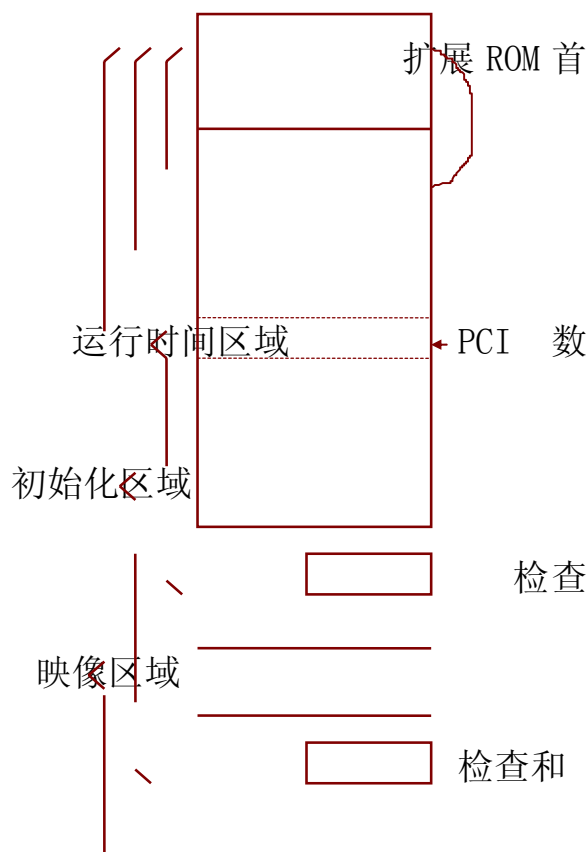
化长度规定了包含初始化和运行

时间编码的数据长度，这是在执  
据结构

行初始化程序之前 POST 编码要  
复制到 RAM 中的数据量，初始

化长度必须大于等于运行时间长  
和

度，复制到 RAM 中的初始化数  
的检查必须是 0。运行时间  
长度规定了包含运行时间码的



映像长度。这也是系统运行时  
POST 编码要保留在 RAM 中的

数据量。这个映像的数据检查和

也必须是 0。

图 5.1 映像布置

PCI 数据配置必须包含于映像的运行时间部分, 否则就要包含于初始化部分。

PCI 设备有两个特点, 使得其驱动器与标准的或现存的设备驱动器有差别, 其一是 PCI 设备在地址空间可重置(没有硬件设置)。PCI 设备驱动器(或别的配置软件)要用存储在设备的配置空间中的映射信息来决定该设备映射于何处, 这也用来确定中断线的使用。

其二是 PCI 中断可以共享。由于设备通常把不至一个设备联接到同一条中断引脚上, 所以 PCI 设备驱动器要支持中断共享。中断共享的准确方式由具体的操作系统决定。

某些系统可能不保证在中断送达 CPU 之前数据传送到主存储器。如果不能正确处理, 可能会引起数据丢失。这种状态通常和在 PCI 总线和别的总线之间的桥路采用邮局式缓存器有关系用下述三种方法可以保证数据和中断的稳定性。设备驱动器必须做到方法 3, 除非它隐含地知道它的设备能完成方法 2 或它被告知系统硬件能做到方法 1。

1). 系统硬件能保证在中断送达处理器之前刷新邮局式缓存器。

2). 设备在发出中断信号之前发出一个该中断能完成对刚写入数据的读信号, 这样就引起邮局式缓存器刷新。

3). 在设备操作写数据之前, 设备驱动器能完成对设备中任何寄存器的读。这个读操作使得邮局式缓存器得以刷新。

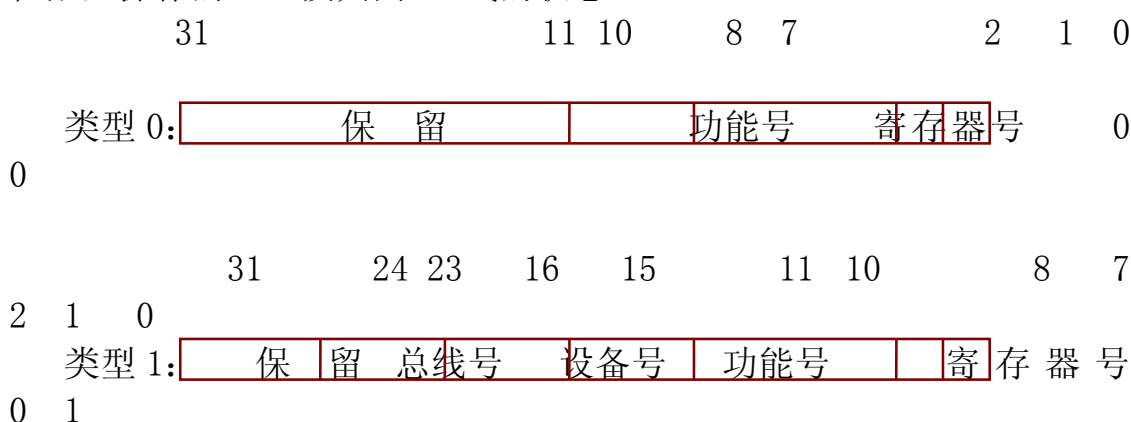
#### § 5.5 配置周期操作:

在地址段期间, 当 IDSEL 信号有效且 AD[1..0] 是 00 时, 就选中该设备为配置命令的目标, AD[7..2] 寻址每个设备配置空间 64 个双字寄存器之一, 字节允许寻址每个双字中的字节, 且 AD[31..11] 上的逻辑是不必关心的, AD[10..8] 说明寻址多功能单元的哪个设备。配置命令和其

它命令一样，允许作字节、字、双字和猝发操作。如果没有目标作出应答，这个要求就有总线主控失败终止。

IDSEL 引脚如何准确地驱动由主/存储器桥路或系统设计者实现，设计上已允许该信号连接到在配置操作中无别的用途的高 21 位地址线的任何一条上，就能选中 21 个不同的设备，这种方法增加了 AD 线上的额外负担，可以通过适当的阻性耦合来减轻它，这又使得 IDSEL 线上的转换率非常慢，因而需要在 FRAME# 之前预驱动地址总线几个时钟。以保证 IDSEL 被采用时它稳定。

为了支持分级 PCI 总线，使用了两种类型的配置操作，下面所示是在配置操作的地址段期间 AD 线的状态：



寄存器号： 用来索引所需目标的配置空间的 64 个双字之一。

功能号： 用来选择某一设备的 8 种可能的功能之一。

设备号： 用来选择某一给定上的 32 个设备之一(将 AD[31..11]中的一条连到 IDSEL，则只能选择 21 种设备之一)。

总线号： 用于选择系统中的 256 条总线之一。

类型 1 配置周期用于对别的 PCI 总线提出配置请求。如果某一配置操作的目标挂于别的总线(非局部 PCI 总线)，就必须使用类型 1 配置操作。类型 1 操作被除 PCI—PCI 桥路之外的所有目标所忽略。这些桥路设备解码总线号以判断配置操作的目标是否挂在该桥路上。如果总线号所表明的总线未挂在该桥路上，就忽略该操作。如果是对挂于本桥路上的总线的操作，则确认该操作，如果总线号不是该桥路的二级总线，这个操作不作改变，只简单的通过该桥路。如果总线号符合二级总线，该桥路就将此操作转换成类型 0 配置操作。

类型 0 配置周期用于选择当前正在操作的 PCI 总线上的设备。该操

作必须由本地设备申请,并由总线主控失败终止。该操作不能扩散到 PCI 局部总线之外。配置编码将按设备顺序号检查总线(即从功能号 0 开始)。如果检查到一个单功能设备,就不在该设备上检查更多的功能。如果检查到一个多功能设备,那么就要检查所有的功能号。多功能设备要对 AD[10..08]作全解码,并且只有在它已为所选功能准备了配置空间寄存器时才对配置周期作出应答。例如,一个双功能设备必须对功能 0 作出应答,它可以选择其它功能号(1~7)作为第二功能。

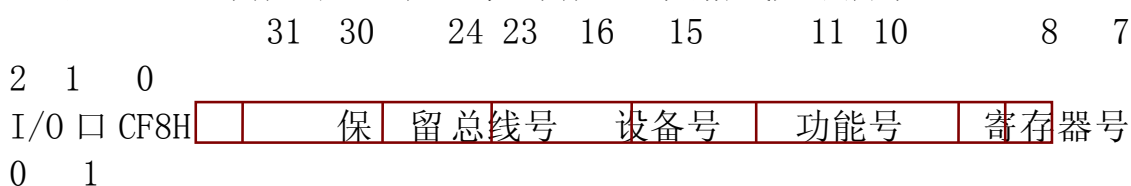
## § 5.6 配置机制

系统必须提供由软件产生 PCI 配置周期的机制,这种机制一般放置于主桥路中,对 PC-AT 兼容系统,定义了两种配置机制,建议采用配置机制 1#,且所有以后的主桥路都要提供这种机制。配置机制 2#是为后向兼容而定义,并且在新的设计中不能再采用。这两种配置机制都可用汇编语言实现。

### § 5.6.1 配置机制 1#

配置机制 1#用了两个双字 I/O 端口,CF8H 为配置地址寄存器端口,CFCH 为配置数据寄存器端口,这种机制是先写一个指定的 PCI 总线,总线上的设备及该设备中的寄存器的值到 CF8H,之后对 CFCH 的读写就会有桥路转换为 PCI 配置周期。

配置地址寄存器是一个 32 位寄存器,其格式如下所示:



↑ 位使能 (1=允许, 0=禁止)

位 31 是允许标志,位 30 到 24 是只读的,而且在读时必须全返回 0,位 23 到 16 选择了系统中特定的 PCI 总线,位 15 到 11 选择了该总线上的某一特定设备,位 10 到 8 选择了多 1 功能设备的一个特定功能,位 7 到 2 选择了该设备配置空间的一个双字,位 1 和位 0 是只读的,而且在读时必须返回全 0。



无论何时,主桥路只把对 CF8H 的全双字 I/O 写锁存到配置地址寄存器中,在对 CF8H 的全双字读时返回配置地址寄存器中的数据。而对这个地址的所有非双字读写必须象正常的 I/O 操作那样对待,使用 BYTE 和 WORD 寄存器的 I/O 设备不会受其影响,它们将不作改变而通过桥路。

当桥路查看到一个对以 CFCH 开始的地址范围的 I/O 操作时,它要检查配置地址寄存器的允许位和总线号如果允许配置周期传送,并且总线号配上了此桥路的总线号或是挂在该桥路另一边的任何总线号时,那么就必须完成一个配置周期传送。

主桥路和 PCI-PCI 桥路通常要求双配置空间寄存器,一个寄存器(总线号)规定了直接挂在该桥路上的 PCI 总线号,另一个寄存器(辅助总线号)规定了通过该桥路进行操作的最后一级总线号。POST 编码负责使这些寄存器初始化到合适的值。

如果被寻址的设备挂于这个桥路的另一边的总线上,则发生类型 1 传送,在配置周期的地址段,桥路直接将配置地址寄存器的内容拷贝到 PCI 的 AD 线上,且 AD[1..0]为 01。

如果被寻址的设备挂在该桥路的 PCI 总线上,则发生类型 0 传送,这时,桥路对设备号进行解码,以时适当的 IDSEL 线有效,并在 PCI 总线上完成一个 AD[1..0]=00 的配置周期。

在这两种情况下,数据传送的字节允许必须直接从处理器总线上复制过来。

采用配置机制 1#的主桥路也允许软件产生特殊周期。如果写入配置地址寄存器的总线号与桥路总线号相匹配;设备号全为 1;功能号全为 1;并且寄存器号为 0,那么在下次对配置数据寄存器的写操作将产生特殊周期。如果写入配置地址寄存器的总线号与桥路总线号不匹配,则桥路把对配置数据寄存器的写操作作为类型 1 配置周期直接传递给 PCI,就象其它总线号不匹配是一样。

### § 5.6.2 配置机制 2#

配置机制 2#是一种将 PCI 配置空间映射到 CPU 的 I/O 空间中的 4K 字节中去的模式,它使用了两个寄存器。

I/O 端口地址为 CF8H 的配置空间允许(CSE)寄存器的格式如下:



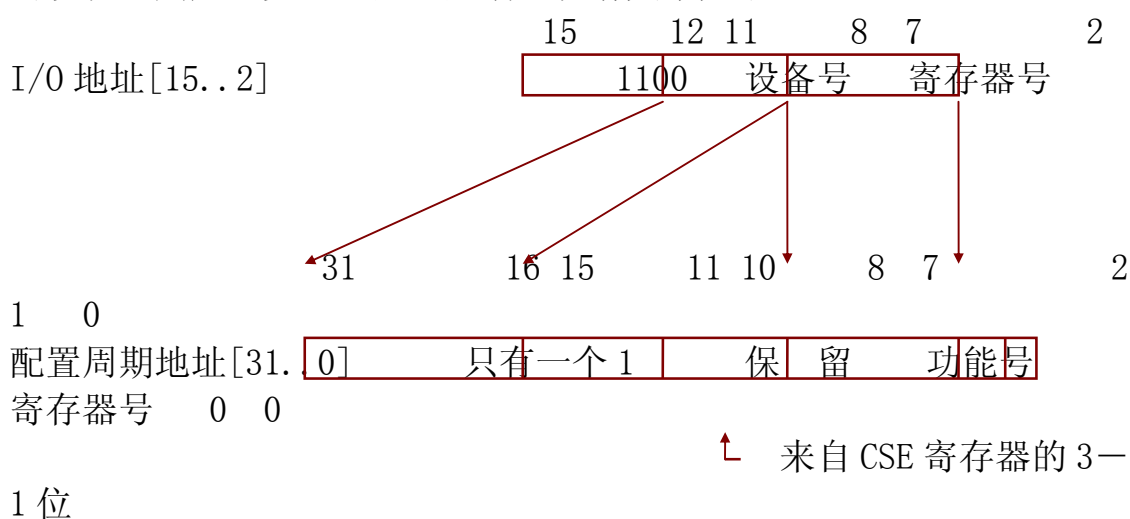
↑
↑  
 0000 = 正常模式                  特殊周期允许  
 其它 = 配置映射允许

该寄存器是一个可读/写的 I/O 口，逻辑上挂在主桥路上，位 0 为 0 产生配置周期，位 0 为 1 产生特殊周期，位 3 到 1 的功能号传递给 AD[10..8]。关键字部分保证将 I/O 读写映射到配置空间读写，对 CSE 寄存器的操作必须是单字节操作。复位后 CSE 寄存器被清零，主桥路处于正常 I/O 操作的约定状态。

I/O 端口地址为 CFAH 的前向寄存器用于确定哪一条 PCI 总线正在被操作，该寄存器可读/写，在复位时初始化为 0，在读时返回最后写入的数据。当前向寄存器是 00 时，则直接挂在桥路上的总线便是被操作的总线，产生类型 0 配置操作。当前向寄存器不是 0 时产生类型 1 配置操作，并将前向寄存器的内容在配置周期的地址段映射到 AD[32..16]。

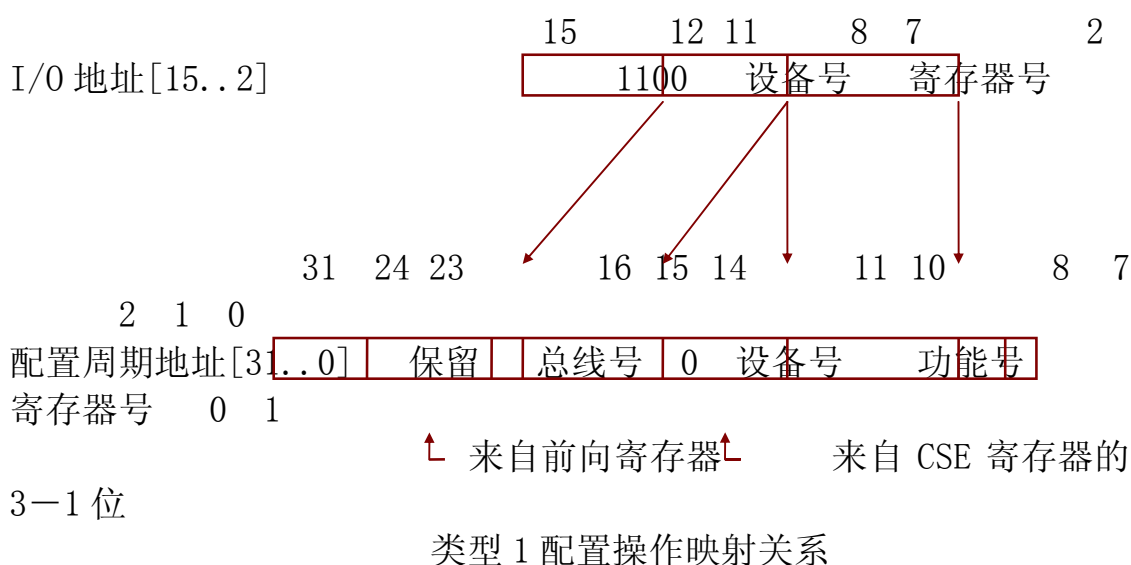
当 CSE 寄存器关键字不为 0 时，允许做配置空间映射，桥路将对地址范围 C000H~CFFFH 的所有 I/O 操作转换成 PCI 配置周期。用 I/O 地址的位 11~8 可以寻址 16 个 PCI 设备(每条总线)。I/O 地址的位 7~2 选择该设备配置空间中的一个特殊双字。

当前向寄存器为 0 时，说明要操作直接挂在桥路上的 PCI 总线 0 上的设备，则产生类型 0 配置周期，映射关系如下：



类型 0 配置操作映射关系

当前向寄存器不为 0 时，说明操作的不是直接挂在桥路上的 PCI 总线上的设备，该桥路产生类型 1 配置周期并将前向寄存器映射到 PCI 总线的 AD[23..16]上，映射关系如下：



采用配置机制 2#对配置空间进行操作的主桥路也允许软件产生配置周期，当 CSE 寄存器位 0 设置成 1，功能号域全为 1，关键字域非零，那么桥路就允许当 CPU 的下次 I/O 写操作是对 I/O 地址 CF00H 时进行 PCI 总线上的一次特殊周期或类型 1 配置周期。此时，如果前向寄存器的内容是 00，那么主桥路在 PCI 总线上产生 PCI 特殊周期，在该周期的地址段，桥路在 C/BE[3..0]#上产生特殊周期编码，并在该周期第一个数据段期间将写入 I/O 地址 CF00H 的数据驱动到 AD[31..00]上。如果前向寄存器的内容不是 00，那么主桥路就产生一个设备号和功能号全为 1 的类型 1 配置周期，且在 PCI 配置写周期的地址段期间寄存器号为 00H。

## § 5.7 PCI BIOS 对配置空间的支持:

PCI BIOS 提供了一种独立于硬件的操作 PCI 设备的方法，通过对中断 INT 1AH 的功能调用，可以判断 PCI BIOS 是否存在，使用何种配置机制，寻找 PCI 设备，对 PCI 配置空间进行操作，及产生特殊周期。

### 1. PCI BIOS 功能组是否存在

输入: AH=B1H

AL=01H

输出 EDX[31..00]= “ ICP”。

AH=00H 说明 PCI BIOS 存在 (如果 EDX[31..00]=“ ICP”)

AL=操作配置空间的硬件机制。

BH=接口水平主要版本。 BL=接口水平次要版本。

CL=系统中最后一个 PCI 总线号

CF=1 说明 BIOS 不存在,

CF=0 说明 PCI BIOS 存在 (如果 EDX[31..00]=" ICP")

## 2. 寻找 PCI 设备

输入: AH=B1H AL=02H

CX=Device\_ID(0..65535) DX=Vendor\_ID(0..65534)

SI = 设备/功能索引(0..N)

输出: AH=00H 表示成功

AH=81H 表示设备没找到 AH=83H 表示错误的销售商标识

符

BH=总线号(0..255)

BL=设备号在高 5 位, 功能号在低 3 位。

CF=完成状态: CF=0 成功, CF=1 错误

## 3. 寻找 PCI 类别码

输入: AH=B1H AL=03H

ECX=类别码(在低 3 字节) SI =设备/功能索引(0..N)

输出: AH=00H 表示成功 AH=86H 表示设备没找到

BH=总线号(0..255)

BL=设备号在高 5 位, 功能号在低 3 位。

CF=完成状态: CF=0 成功, CF=1 错误

## 4. 在系统中特定的 PCI 总线上产生特殊周期

输入: AH=B1H AL=06H

BH=总线号(0..255) EDX=特殊周期数据

输出: AH=00H 表示成功 AH=81H 表示不支持该功能

CF=完成状态: CF=0 成功, CF=1 错误

## 5. 读一个字节

输入: AH=B1H AL=08H

BH=总线号(0..255) BL=设备号在高 5 位, 功能号

在低 3 位

DI=寄存器偏移地址(0..255)

输出: AH=00H 表示成功 AH=87H 表示错误的寄存器地址

CL=读取的字节

CF=完成状态: CF=0 成功, CF=1 错误

6. 读一个字

输入: AH=B1H

AL=09H

BH=总线号(0..255)

BL=设备号在高 5 位, 功能号

在低 3 位

DI=寄存器偏移地址(0, 2, 4, ..254)

输出: AH=00H 表示成功

AH=87H 表示错误的寄存器地址

CX=读取的字

CF=完成状态: CF=0 成功, CF=1 错误

7. 读一个双字

输入: AH=B1H

AL=0AH

BH=总线号(0..255)

BL=设备号在高 5 位, 功能号

在低 3 位

DI=寄存器偏移地址(0, 4, 8, ..252)

输出: AH=00H 表示成功

AH=87H 表示错误的寄存器地址

ECX=读取的双字

CF=完成状态: CF=0 成功, CF=1 错误

8. 写一个字节

输入: AH=B1H

AL=0BH

BH=总线号(0..255)

BL=设备号在高 5 位, 功能号

在低 3 位

DI=寄存器偏移地址(0..255)

CL=要写的字节

输出: AH=00H 表示成功

AH=87H 表示错误的寄存器地址

CF=完成状态: CF=0 成功, CF=1 错误

9. 读一个字

输入: AH=B1H

AL=09H

BH=总线号(0..255)

BL=设备号在高 5 位, 功能号

在低 3 位

DI=寄存器偏移地址(0, 2, 4, ..254)

CX=要写的字

输出: AH=00H 表示成功

AH=87H 表示错误的寄存器地址



CF=完成状态: CF=0 成功, CF=1 错误

10. 保留

输入: AH=08H~B0H 或 B2H~FFH

输出: CY=1      无效功能