技 术 报 告

名　称：　AMBA 协议 2.0 详细研究

编　号：

版本号：　0.2

作　者　余华新
项　目　Zi209
部　门　产品中心
日　期　2006-09-08

深圳市中兴集成电路设计有限责任公司

中文标题
Title

摘　要：
关键词：

Abstract：
Key Words：

目 录

1 AMBA总线概述 7

1.1 AMBA标准概述 7

1.2 定义AMBA标准的目的 7

1.3 一个典型的基于AMBA总线的SOC系统架构 7

1.4 术语 8

1.5 AHB介绍 8

1.6 ASB介绍(忽略) 9

1.7 APB介绍 9

1.8 选择正确的系统总线 9

1.9 协议注意 10

2 AMBA信号 10

2.1 AHB信号列表 11

2.2 ASB信号列表(忽略) 12

2.3 APB信号列表 13

3 AHB总线 14

3.1 什么是AHB总线？ 14

3.1.1 一个典型的基于AHB总线的微控制器架构 15

3.2 AHB总线互连结构 17

3.3 AHB操作概述 18

3.4 基本传输 19

3.4.1 零等待传输(no wait state transfer) 19

3.4.2 等待传输(transfers with wait states)) 20

3.4.3 多重传送(multiple transfer) 20

3.5 控制信号：传送状态HTRANS[1:0] 22

3.6 控制信号：批量传送HBURST[2:0] 24

3.6.1 提前结束Burst(参 3.11) 24

3.6.2 长度为 4 的回绕地址传送方式（four-beat wrapping burst）。 25

3.6.3 长度为 4 的递增地址传送方式（Incrementing burst of length 4） 25
3.6.4 不限长度的传送方式 26
3.7 控制信号：其它 27
3.7.1 传送方向HWRITE 27
3.7.2 传送大小HSIZE[2:0] 27
3.7.3 保护控制HPROT[3:0] 27
3.8 地址译码 29
3.9 从传输应答 30
3.9.1 传送完成HREADY 30
3.9.2 其它从传输应答HRESP[1:0] 30
3.9.3 两周期应答(Two-cycle response) 31
3.9.4 错误应答Error 31
3.9.5 分段传送与重传的异同 32
3.10 数据总线 34
3.10.1 HWDATA[31:0] 34
3.10.2 HRDATA[31:0] 34
3.10.3 位格式Endianness 34
3.11 总线仲裁机制(重点) 35
3.11.1 信号描述 35
3.11.2 总线访问请求 35
3.11.3 总线访问批准 36
3.11.4 提前结束Burst 39
3.11.5 锁定传送Locked transfers 39
3.11.6 预设总线主设备 40
3.12 分段传送HSPLITx(重点) 41
3.12.1 分段传送顺序 41
3.12.2 多重分段传送(multiple split transfers) 41
3.12.3 防死锁/ 锁住避免(deadlock prevention) 41

3.12.4 分段传送之总线转换 42
3.13 复位机制HRESETn 44
3.14 数据总线位宽HSIZE[2:0] 44
3.15 宽总线-窄从Implementing a narrow S on a wider bus 44
3.16 窄总线-宽从Implementing a wide S on a narrow bus 44
3.16.1 Ms 44
3.17 AHB组件 45
3.18 AHB从(重点) 45
3.18.1 从接口框图 45
3.18.2 从时序 46
3.19 AHB主(重点) 47
3.19.1 主接口框图 47
3.19.2 主时序 48
3.20 仲裁器(arbiter) 50
3.20.1 接口框图 50
3.20.2 时序 50
3.21 译码器(decoder) 52
3.21.1 接口框图 52
3.21.2 时序 52
4 ASB总线(未实用,忽略) 53
5 APB总线 54
5.1 什么是APB总线？ 54
5.1.1 一个典型的基于AMBA总线的系统架构 54
5.2 APB规范 54
5.2.1 APB状态机 55
5.2.2 写操作 55
5.2.3 读操作 56
5.3 APB组件 57

5.4 APB桥 58
5.4.1 接口框图 58
5.4.2 描述 58
5.4.3 时序 59
5.4.4 APB桥（APB主）的时序参数 59
5.5 APB从 61
5.5.1 接口框图 61
5.5.2 描述 61
5.5.3 时序 62
5.5.4 APB从的时序参数 62
5.6 APB到AHB的接口 63
5.6.1 读操作 63
5.6.2 写操作 65
5.6.3 读写交替传送（Back to back） 67
5.6.4 三态数据线的实现 68
5.7 APB到ASB的接口(忽略) 68
5.8 D版本APB到 2.0 版本APB(忽略) 68
6 测试方法学 69
6.1 AMBA测试接口 69
6.2 外部接口 70
6.2.1 TREQA 70
6.2.2 TREQB 70
6.2.3 TACK 70
6.2.4 Test clock 71
6.2.5 Test bus 71
6.3 测试向量类型 71
6.4 测试接口控制器 72
6.4.1 测试操作参数 72

6.4.2 递增寻址 72
6.4.3 进入测试模式 73
6.4.4 地址向量 73
6.4.5 控制向量 74
6.4.6 写操作向量 74
6.4.7 读操作向量 74
6.4.8 批量向量 75
6.4.9 改变操作方向 75
6.4.10 退出测试模式 75
6.5 AHB测试接口控制器 75
6.5.1 Control vector 77
6.6 AHB测试程序举例 78
6.6.1 Entering test mode 78
6.6.2 写操作 79
6.6.3 读操作 79
6.6.4 控制向量 80
6.6.5 批量向量 80
6.6.6 交替读写Read-to-write and write-to-read 81
6.6.7 退出测试模式 81
6.7 ASB测试接口控制器(忽略) 81
6.8 ASB测试程序举例(忽略) 81
7 几个比较 82
7.1 不同总线之比较 82
7.2 AMBA2.0 与AMBA3.0(AXI) 82
7.3 递增与回绕(参 3.6) 82
7.4 重发与分段发送(参 3.9.5) 82
7.5 82

Revision History

Version	Date	Revision	By	Page
0.1	2006-09-08	Initial version(以 AMBA2.0 协议为蓝本,非纯粹协议翻译)	余华新	/
0.2	2006-09-14	补充 APB 部分(From 余讯)	余华新	/

文档阅读指南:

1. 标题　　靛蓝色
2. 接口信号名　　蓝色
3. 强调　　粉红色
4. 主设备　　M
5. 从设备　　S
6. 表号在表上
7. 图号在图下

1 AMBA 总线概述

1.1 AMBA 标准概述

AMBA 的全称是 Advanced Microcontroller Bus Architecture。它的协议包含了 3 个部分，分别阐述了 3 种不同功能的总线结构，它们是：

- AHB -高效能系统总线 Advanced High-performance Bus (AHB)
- ASB - AMBA系统总线 Advanced System Bus (ASB)
- APB -外围总线 Advanced Peripheral Bus (APB)

协议中同时包含了总线的测试方法。

总的来说，这些内容加起来就定义出一套为了高性能 SOC 而设计的片上通信的标准。对于 SOC 设计者来说，认识 AMBA 总线协议是十分必要的。

另外，ASB 总线与 AHB 总线都是系统总线，区别在于 ASB 总线并没有 AHB 总线的高效传送特性。当系统不需要 AHB 总线的高性能的时候，设计者可以选择 ASB 总线。但是对于目前的应用来说，只考虑 AHB 总线。因此在下面的章节中只介绍 AHB 和 APB 总线协议。

1.2 定义 AMBA 标准的目的

- to facilitate the right-first-time development of embedded microcontroller products with one or more CPUs or signal processors
- to be technology-independent and ensure that highly reusable peripheral and system macrocells can be migrated across a diverse range of IC processes and be appropriate for full-custom, standard cell and gate array technologies
- to encourage modular system design to improve processor independence, providing a development road-map for advanced cached CPU cores and the development of peripheral libraries
- to minimize the silicon infrastructure required to support efficient on-chip and off-chip communication for both operation and manufacturing test.

1.3 一个典型的基于 AMBA 总线的 SOC 系统架构

这种SOC架构以AHB或者ASB系统总线为主干，挂上CPU，片上内存，和其它DMA设备。AHB或者ASB系统总线在需要做大量数据传送的模块之间提供了高带宽的接口。同时，外围总线APB在AHB或者ASB和低带宽的外围设备之间提供了通信的桥梁。所以APB是AHB或者ASB的二级扩展总线。

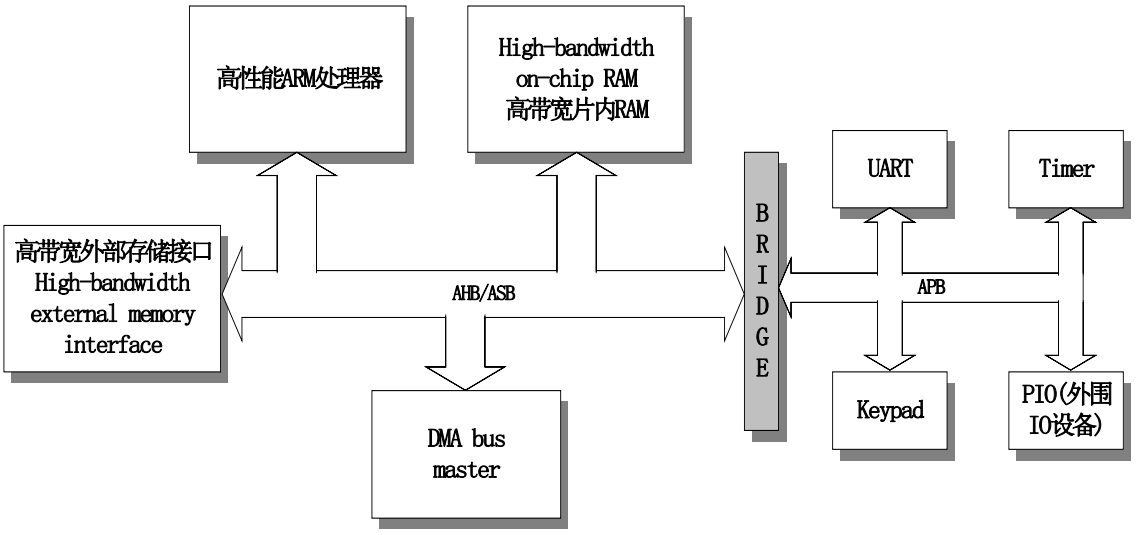

图1-1　一个典型的基于AMBA总线的体系架构

1.4　术语

表 1-1 术语

Bus cycle	总线周期	指一个总线时钟周期: AHB/APB 是时钟正沿,ASB 是时钟负沿.所有总线信号的时序是基于 Bus cycle.
Bus transfer	总线传送	ASB/AHB 总线传送是指对数据目标的读或写操作, 由相应被选中的从应答完成来结束. ASB/AHB 总线传送花 1 个或多个时钟周期(加等待周期) APB 的总线传送必需花 2 个时钟周期
Burst operation	批量操作	指由 **M** 发起的数据传送.　APB 不支持 burst

1.5　AHB 介绍

AHB是高性能高时钟频率的系统模块，是系统的主干，它直接链接了CPU，片上内存，DMA和外部内存接口。AHB分为主和从两端，它支持多个主的链接，例如CPU，DSP和DMA等，并能进行高带宽的操作。

AHB的特性包括:

- Burst 传输
- Split 传输
- 单时钟周期总线Master移交
- 单时钟沿操作
- 非3态电路
- 更宽的数据线宽度（64或128位）

AHB 总线可以包含多个主:　通常有 CPU，测试接口，DMA 或者 DSP。

AHB 的从通常包括：　　外挂内存接口，APB 桥，内部 RAM。其它一些外围设备通常也能作为从。但是低带宽的外围设备通常挂在 APB 上。

AHB 的主要部分的功能为：

AHB Master：　可初始化读写操作，提供地址和控制信号，同一时间只有 1 个主会被激活。

AHB Slave：　对读写操作在一定的地址范围内作出响应，并对主返回成功，失败或者等待等状态。

AHB Arbiter：　仲裁器主要负责让总线上同时只有1个主在工作。仲裁协议是规定的，但是仲裁算法可以根据应用决定，比如highest priority或者fair access算法等。只挂一个主的AHB上可能不带有仲裁器。

AHB Decoder：　负责对地址进行解码，并提供片选信号到各从设备。每个 AHB 总线都需要 1 个中央解码器。

1.6　ASB 介绍(忽略)

1.7　APB 介绍

APB 的全称：Advanced Peripheral Bus

作为 AMBA 总线的一层，APB 总线是为了功耗最小化和减低接口复杂度而设计的。APB 总线应该用于低带宽和不需要高性能流水线总线接口的外设。

APB 总线上的所有信号都在正时钟沿发生变化，这个特性决定了 APB 可以更容易地整合到各种设计流程里。

1.8　选择正确的系统总线

1.8.1 Choice of system bus

Both AMBA AHB and ASB are available for use as the main system bus. Typically the choice of system bus will depend on the interface provided by the system modules required.

The AHB is recommended for all new designs, not only because it provides a higherbandwidth solution, but also because the single-clock-edge protocol results in a smoother integration with design automation tools used during a typical ASIC development.

1.8.2 System bus and peripheral bus

Building all peripherals as fully functional AHB or ASB modules is feasible but may not always be desirable:

? In designs with a large number of peripheral macrocells the increased bus loading may increase power dissipation and sacrifice performance.

? Where timing analysis is required, the slowest element on the bus will limit the maximum performance.

? Many simple peripheral macrocells need latched addresses and control signals as opposed to the high-bandwidth macrocells which benefit from pipelined signalling.
? Many peripheral functions simply require a selection strobe which conveys macrocell selection and read/write bus operation, without the requirement to broadcast the high-frequency clock signal to every peripheral.

1.8.3 When to use AMBA AHB/ASB or APB
A full AHB or ASB interface is used for:
? bus masters
? on-chip memory blocks
? external memory interfaces
? high-bandwidth peripherals with FIFO interfaces
? DMA slave peripherals.
A simple APB interface is recommended for:
? simple register-mapped slave devices
? very low power interfaces where clocks cannot be globally routed
? grouping narrow-bus peripherals to avoid loading the system bus.

1.9 协议注意

The following points should be considered when reading the AMBA specification:
? Technology independence
? Electrical characteristics
? Timing specification.
1.9.1 技术独立性
AMBA is a technology-independent on-chip protocol. The specification only details the bus protocol at the clock cycle level.
1.9.2 电气特性
No information regarding the electrical characteristics is supplied within the AMBA specification as this will be entirely dependent on the manufacturing process technology that is selected for the design.
1.9.3 时序定义
The AMBA protocol defines the behavior of various signals at the cycle level. The exact timing requirements will depend on the process technology used and the frequency of operation.
Because the exact timing requirements are not defined by the AMBA protocol, the system integrator is given maximum flexibility in allocating the signal timing budget amongst the various modules on the bus.

2 AMBA 信号

2.1 AHB 信号列表

AHB 的信号前缀都是 **H**。

表 2- AHB 从设备接口信号表

信号名	含义	IO	源	描述
HCLK	总线时钟		CGU	总线时钟，同步所有传送，所有信号时序与 HCLK 的上升沿相关
HRESETn	复位		RCU	总线复位信号低有效，复位系统与总线。
HADDR[31:0]	地址总线	M2S	主	32 位系统地址总线(一般的系统仲裁可以不用到地址)
HTRANS[1:0]	传送类型	I	主	指明当前传送的类型: NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY(非同步，同步，停止或者忙)
HWRITE	传送方向	I	主	读写操作: 1-写; 0-读
HSIZE[2:0]	传送带宽	I	主	指明当前传送的大小: 8(字节)，16(字)，32(双字)等， 协议支持高达 1024 位的传送
HBURST[2:0]	批量类型:	I	主	0-SINGLE,1-INCR ,2-WRAP4 ,3-INCR4 ,4-WRAP8 ,5-INCR8 ,6-WRAP16 ,7-INCR16
HPROT[3:0]	保护控制	I	主	提供总线访问的一些其它信息，用于一些需要实现保护功能的模块。如果传送是取操作码或数据传送，如果传送是优先访问模式或用户模式访问。是否是只用于 cache? (M 用 MMU 来管理这些信号表示当前访问是否可 cacheable 或 bufferable。)
HWDATA[31:0]	写数据总线	I	主	M2S: 写操作主传给从的数据总线. 推荐最少 32 位数据总线带宽。 可以很方便地扩展到高带宽操作.
HSELx	从选择	I	译码器	S 选择信号： 表示当前哪个 S 被选择在传送. 地址选择就是地址译码出来的 S 选择信号 HSELx
HRDATA[31:0]	读数据总线	O: S2M	从	S2M: 读操作从返回给主的数据总线. 推荐最少 32 位数据总线带宽。 可以很方便地扩展到高带宽操作.
HREADY	传送完成	IO	从	从应答主是否读写操作传输完成: 1-传输完成， 0-传输忙， 可被 驱动为低来延续传输. 注: S 需要 HREADY 是双向信号(HREADY 做为总线上的信号，它是 M 和 S 的输入; 同时每个 S 自己的 HREADY 信号是输出。所以对于 S 会有两个 HREADY 信号，一个来自总线的输入，一个自己给到多路器的输出。)
HRESP[1:0]	传送响应	O	从	从向主应答当前传输状态: OKAY, ERROR, RETRY and SPLIT.
点对点信号				
HBUSREQx	Bus 请求	M2A	主	Mx 向 A 发出的总线使用请求信号。最多 16 个 M (仲裁器接收各个 M 发出的总线请求信号(HBusReq)和所需的总线切换的判断信号，采用一定的总线仲裁算法，确定出可以占据总线的 M，并生成 M2S MUX 的控制信号)

HLOCKx	锁定传送	M2A	主	M 向 A 发出的总线占用请求信号。A 批准后其它 M 要等到 HLOCKx 为 LOW 时才可以获得 A 授权来访问总线: 1-主 x 需要保持对总线的访问，在变成 0 之前其它主都不可以访问总线; 0-不保持 (一般不要随便使用 lock，因为如果设计得不好的话，很容易就把总线锁死了,可以通过采取优先级来控制总线的调度) (一旦 HLOCK 被仲裁器所接受，那么在 LOCK 期间，总线的所有权不会移交给其他的 M,A 必须等这个 M 传送完毕才能给其他 M 权限)
HGRANTx	Bus 同意	A2M	仲裁器	制裁器指示 M: 主 x 的地址/控制信号的控制权优先级最高. 在上一个 M 数据传送完毕 HREADY＝1 时，A 切换 HGRANT 信号。 在 HREADY 与 HGRANTx 都为高时，Mx 获得总线访问权。(所谓 M 就是要主动发起 Transfer 的设备,地址和控制信号当然是 M 自己主动发出的了,条件是 HGRANT 和 HREADY 有效。)
HMASTER[3:0]	主号	A2M	仲裁器	仲裁器指示从: 当前哪个主在进行传输，让从知道那个主需要访问。HMASTER 时序与地址和控制信号对齐，并可用来控制地址多任务器。(split 的 M 的值)(在上一个 M 最后一次传送控制信号被 S 正确 sample 的时候 arbiter 更改 HMASTER，并且用 HMASTER 作为地址和控制总线 mux 的控制信号。)
HMASTLOCK	锁序	I	仲裁器	仲裁指示: 当前的主在锁序传输. 其时序特性与 HMASTER 相同.
HSPLITx[15:0]	分段请求	O:S2A	从	从告知仲裁器:Mx 可以被允许重试分段传输.每位对应相应的 M. (当从准备好数据后,从通过 HSPLITx 信号通知仲裁器取消前面对该主设备的请求屏蔽，使其能够重新请求总线，完成传送过程。)

2.2 ASB 信号列表(忽略)

2.3 APB 信号列表

信号名	含义	IO	源	描述
PCLK	总线时钟		CGU	总线时钟，同步所有传输，正时钟沿触发所有信号,所有信号时序与 HCLK 的上升沿相关
PRESETn	复位		RCU	总线复位信号低有效，复位系统与总线。
PADDR[31:0]	地址总线	I:M2S	主	32 位系统地址总线(一般的系统仲裁可以不用到地址)
PSELx	从选择	I	译码器	S 选择信号： 表示当前哪个 S 被选择在传输. 地址选择就是地址译码出来的 S 选择信号 HSELx
PENABLE	APB 选通	I	主	指示 APB 操作的第 2 个周期.
PWRITE	传输方向	I	主	读写操作: 1-写; 0-读
PRDATA[31:0]	读数据总线	O: S2M	从	S2M: 读操作从返回给主的数据总线. 推荐最少 32 位数据总线带宽。可以很方便地扩展到高带宽操作.
PWDATA[31:0]	写数据总线	I	主	M2S: 写操作主传给从的数据总线. 推荐最少 32 位数据总线带宽。可以很方便地扩展到高带宽操作.

3 AHB 总线

表 3-0 AHB 总线内容索引表

3.1 什么是 AHB 总线		
3.2 AHB 总线互连结构		
3.3 AHB 操作概述		
3.4 基本传送	3.4.1 零等待传送	
	3.4.2 等待传送	
	3.4.3 多重传送	
3.5~3.7 控制信号	3.5 传送状态 HTRANS[1:0]	
	3.6 突发传送 HBURST[2:0]	
	3.7.1 传送方向 HWRITE	
	3.7.2 传送大小 HSIZE[2:0]	
	3.7.3 保护控制 HPROT[3:0]	
3.8 地址译码		
3.9 从传送应答		
3.10 数据总线		
3.11 总线仲裁机制		
	3.11.2 总线访问请求	
	3.11.3 总线访问批准	
	3.11.4 提前结束 Burst	
	3.11.5 锁定传送	
	3.11.6 预设总线主设备	
3.12 分段传送 HSPLITx	3.12.1 分段传送顺序	
	3.12.2 多重分段传送	
	3.12.3 防死锁/ 锁住避免	
	3.12.4 分段传送之总线转换	
3.13 复位机制 HRESETn		
3.14 数据总线位宽 HSIZE[2:0]		
3.15 从带宽适配		
3.16 主带宽适配		
3.17~3.21 AHB 功能单元	3.18 AHB 从	
	3.19AHB 主	
	3.20 仲裁器	
	3.21 译码器	

3.1 什么是 AHB 总线?

AHB 主要用于高性能模块(如 CPU、DMA 和 DSP 等)之间的连接，作为 SoC 的片上系统总线，它包括以下一些特性：

表 3-1

序号	特性
1	单个时钟边沿操作
2	单时钟周期 M 移交
3	非三态的实现方式(以中央多任务器取代三态闸的实现方式)
4	支持批量传送（burst transfers）； // 爆发/突发/猝发

5		支持分段传送(split transaction)； // 分割/分离
6		支持多个主控制器
7		可配置 32 位~128 位总线宽度
8		支持字节、半字节和字的传送

AHB 系统由主模块、从模块和基础结构(Infrastructure)3 部分组成，整个 AHB 总线上的传送都由主模块发出，由从模块负责回应。

表 3-2

序号	组件	描述
1	**AHB 主**	AHB 总线可以包含多个主： 通常有 CPU，测试接口，DMA 或者 DSP
2	**AHB 从**	外挂内存接口，APB 桥，内部 RAM。其它一些外围设备通常也能作为从
3	**APB**	低带宽的外围设备通常挂在 APB 上
4	基础结构	仲裁器(arbiter)、
5		主模块到从模块的多路器、
6		从模块到主模块的多路器、
7		译码器(decoder)、
8		虚拟从模块(dummy S)、
9		虚拟主模块(dummy M)所组成。

3.1.1 一个典型的基于 AHB 总线的微控制器架构

图 3-1 为一典型的 AMBA 系统架构图，其中包含了二个主要的**总线** — 高效能总线 (AHB) 和 外围总线 (APB)。

AHB 主要是针对高效率、高频宽及快速系统模块所设计的总线，它可以连接如微处理器、芯片上或芯片外的内存模块和直接内存存取机制等高效率模块，此总线特别适合用于可合成和自动测试技术上的设计流程中；

AMBA 中另一高效率系统模块为 ARM 系统总线(ASB)，目前已被 AHB 所取代了，因此在本文中将不提及。

AMBA 中的 APB 主要用在低速且低功率的外围，可针对外围作功率消耗及复杂接口的最佳化。由图 3-1 可知，负责 AMBA AHB 和 APB 之间的沟通管道为一桥梁(bridge)，APB bridge 主要负责总线数据的栓取、译码及传送。

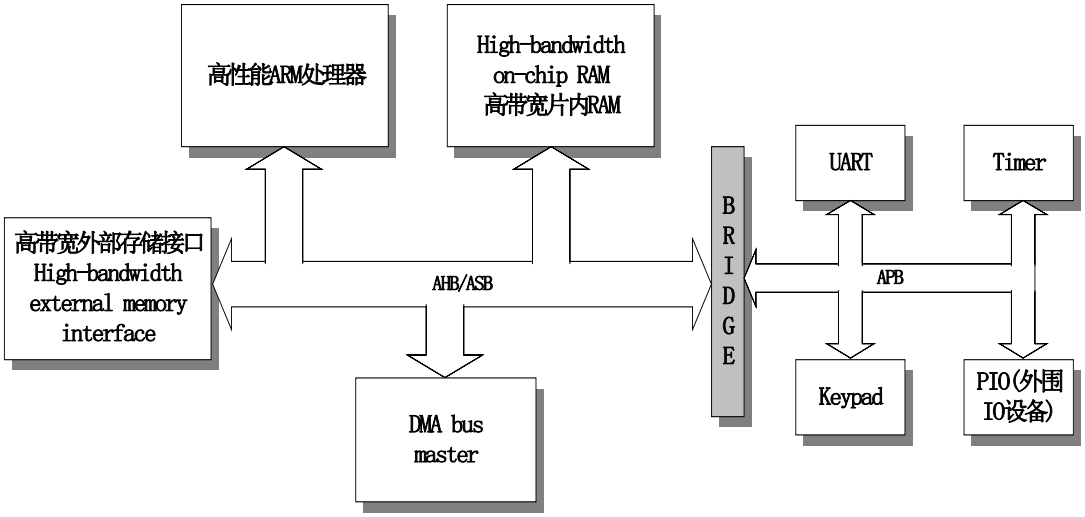

图3-1　一个典型的基于AMBA总线的体系架构

AMBA 的主要目标是要让使用者能在最快的时间内完成一以内嵌式微处理器为平台的系统设计，由于它是一个跟制程无关(Technology Independent)的总线设计，并且可使具重复使用的外围及系统宏组件容易被整合进去无论是全客户、标准组件和逻辑数组为技术的设计流程中；且要让模块化的设计很容易的被整合进去一个以微处理器为平台的系统发展环境。在 AMBA 的规格中亦提供一个测试控制接口环境，使得 AHB 和 APB 可透过外在的系统测试样品来作测试，也因此减轻了功能测试时的工作。

AHB 以仲裁器来控制多任务器以连接在其上的 M 与 S。
在 AHB 总线上，其操作顺序一般如下：

1. M 要求总线使用
2. 仲裁器响应允许**信号**
3. **M** 送出地址与控制信号

基本的 AHB 传送特性:

一次传送包括给出地址、控制信号周期(address, control)与数据周期(data cycle)。地址与控制信号周期最少需要一个周期，但是会因为数据的原因多出几个周期。数据周期可以藉由 HREADY 信号来延迟。拉起 HREADY 代表此次传送已经 OK。

AHB 支持批量式数据传送，可以自动递增地址。递增地址方式分为：持续递增(incrementing without wrapping at address boundary)，另一种为回绕传送 (wrapping at address boundary.)。

在 AHB 上写入总线将数据送到 **S**，由读取总线将数据送到 **M**。
S 以 HRESP[1:0]**信号**表示传送的状态：

- HRESP[1:0] = 00 (OKAY)：当此**信号**再加上拉高的 HREADY（OKAY+高 HREADY）就表示此次的传送已完成。
- HRESP[1:0] = 01 (ERROR)：传送失败。
- HRESP[1:0] = 10 (RETRY) and HRESP[1:0] = 11 (SPLIT)：当数据传送无法立刻完成时，**M** 会继续尝试完成数据的传送。

3.2 AHB 总线互连结构

AHB 的主要部分的功能如下表：

表 3-3 AHB 的主要部分的功能

部件	功能描述
AHB 主	可初始化读写操作，提供地址和控制信号，同一时间只有 1 个主会被激活
AHB 从	对读写操作在一定的地址范围内作出响应，并对主返回成功，失败或者等待等状态
AHB Arbiter	仲裁器主要负责让总线上同时只有 1 个主在工作。仲裁协议是规定的，但是仲裁算法可以根据应用决定，比如 highest priority 或者 fair access 算法等。只挂一个主的 AHB 上可能不带有仲裁器
AHB Decoder	负责对地址进行解码，并提供片选信号到各 **S**。每个 AHB 总线都需要 1 个中央解码器

其互连结构如图 3-2 所示。

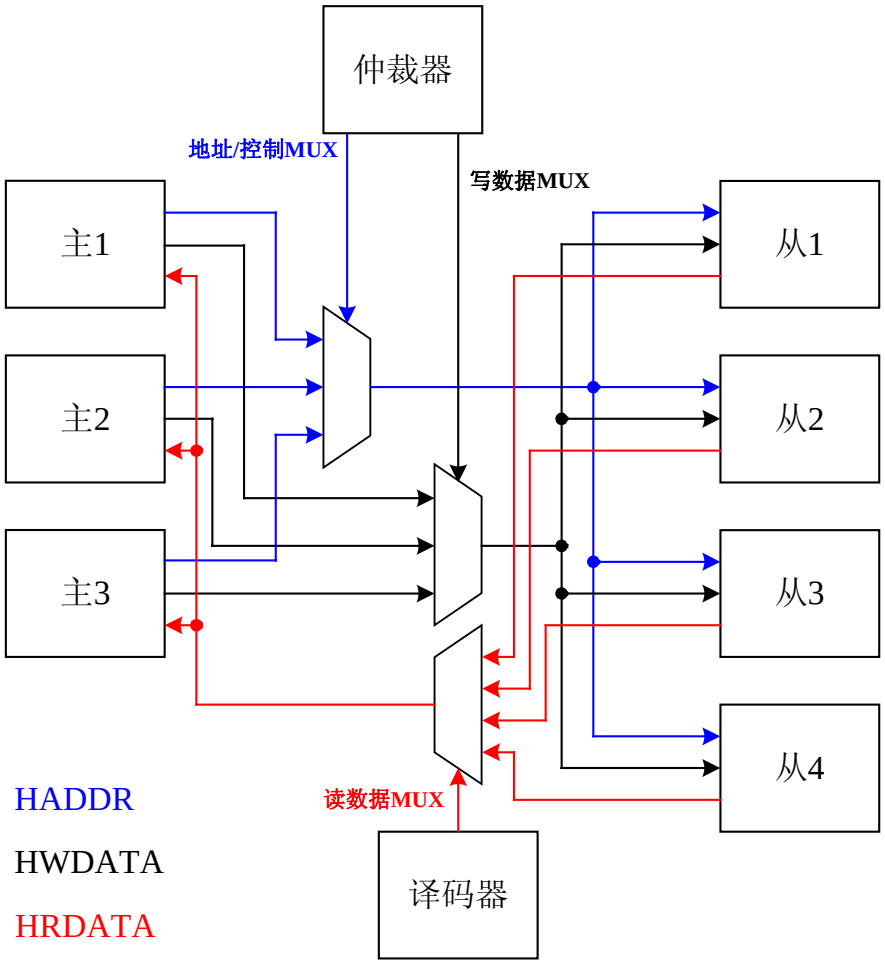

图 3-2 AHB 总线互连结构图

3.3 AHB 操作概述

a. 在 AHB 传送开始之前，**M** 必须得到仲裁器的总线访问授权：**M** 向仲裁器发出总线请求信号，然后仲裁器指示这个 **M** 什么时候可以得到授权

b. **M** 得到总线访问授权后，驱动地址与控制信号，来开始 AHB 传送。这些信号提供传送地址，传送方向，传送带宽，是否这次传送是 burst 传送等信息。

c. 在 AHB 上，一次传送包括给出地址、控制**信号**周期与数据周期。地址与控制**信号**周期最少需要一个周期，但是会因为数据的原因多出几个周期。数据周期可以藉由 HREADY **信号**来延迟。

d. AHB 支持批量式数据传送，可以自动递增地址。递增地址方式分为：持续递增与回绕传送。

e. 当 HREADY 被 **S** 拉低时，会发生等待状态。需要延迟传送的周期时，就会在数据期间将 HREADY 拉低，此时地址与数据都必须延伸。

f. **S** 以 HRESP[1:0]**信号**表示传送的状态为：OKAY，ERROR，RETRY，或 SPLIT。

g. 在 AHB 总线上，**M** 的传送型态可由 HTRANS[1:0]来表示，有 IDLE ，BUSY ，NONSEQ ，SEQ 四种型态。

h. 在 AHB 总线上，HWRITE 表示传送方向，HWRITE 为 HIGH 时，M 做写入动作，数据会由 **M** 放到 HWDATA[31:0] 总线上。

i. HWRITE 为低时，**M** 做读取动作，被寻址到的 **S** 会将数据放到 HRDATA[31:0] 总线上。

j. 传送数据大小由 HSIZE[2:0] 表示每次传送的字节数目。

k. 在 AHB 上可使用大端摆放法或小端摆放法，由系统设计者决定。

l. 仲裁器以 HMASTER 表示当前是哪一个 **M** 在使用 bus，此**信号**也可用来控制地址多任务器。

m. 一个可执行分割传送的 **S** 以 HSPLIT[15:0]**信号**，让仲裁器知道要让哪一个 **M** 完成其未完成的分割传送。

n. **S** 可以用 SPLIT 来解决延迟太长的存取，仲裁器必须观察响应**信号**并且遮掉已被分割传送的 **M**，使其等候 HSPLIT **信号**的通知后才 grant 其总线使用权。

3.4 基本传输

本节介绍 AHB 的几种基本的传送机制。

3.4.1 零等待传输(no wait state transfer)

AHB 基本的运作情形。如下图所示，AHB 的传送模式包含二个阶段(2 个时钟周期完成)，第一阶段为地址的传递，另一阶段为数据的传递，没有额外的等待周期。

其过程如下：

- 地址期间（address phase)： 一个周期
- 数据期间（data phase）： 一个周期 (以 HREADY 信号控制周期数目).

其中，数据传递时序图会根据不同周期数的等待状态而有差异。由图中可知，在第一次 HCLK 正沿触发后会进入地址模式，此时 **M** 会将所要传送的地址及控制信号准备好；

在 HCLK 第二次正沿触发后会进入数据模式，这个模式的时序会依据 **S** 是否能正常接收到数据或准备好数据而有不同周期数的等待时间。数据模式时序的结束与否，取决于 **S** 所送出的 HREADY 信号值，若 **S** 能够正确的接收到 **M** 送来的数据或准备好 **M** 需要的数据，则 **S** 会使 HREADY 信号为 1；

且在第三次 HCLK 正沿触发时，**M** 会去撷取由 **S** 送出的数据或响应信号。假设 **S** 无法完成数据接收或数据准备，则 **S** 会使 HREADY 信号为 0，亦即需要额外的延迟等待时间才能完成数据模式的传送。

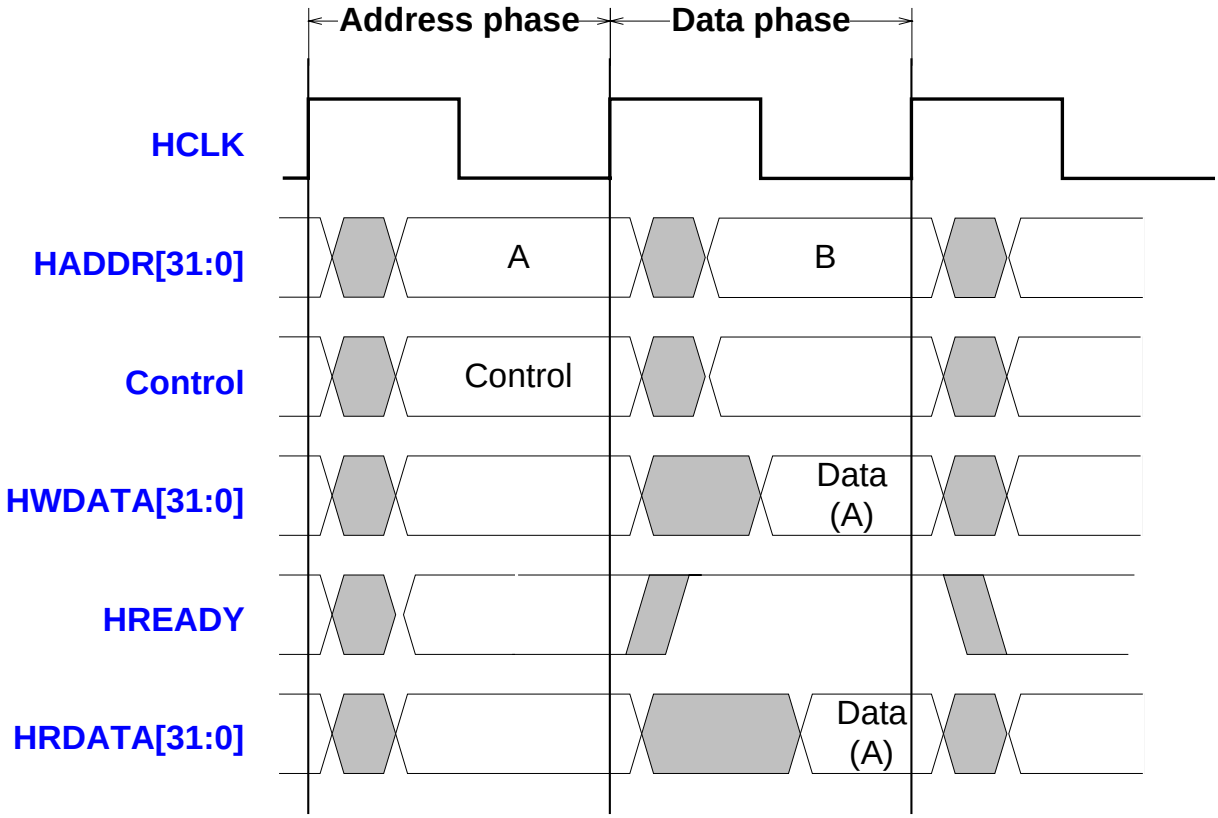

图 3-3 基本传送机制

- 当 HCLK 正沿触发后，**M** 送出地址与控制**信号**，HCLK 由 CLOCK 驱动送至 AHB 接口。
- **S** 在 HCLK 下一个正沿触发时，收到地址与控制**信号**。
- 在 HCLK 的第三个正沿触发时，**M** 收到 **S** 的响应**信号**。

AHB 支持**流水线**的动作，在收上一笔数据的同时，可将下一笔数据的地址送出。

3.4.2 等待传输(transfers with wait states))

图 3-4 的传送完成时间因等待而延迟，这是由于寻址到较慢的 **S** 单元所造成。当 HREADY 被 **S** 拉低时，会发生等待状态。需要延迟传送的周期时，就会在数据期间将 HREADY 拉低，此时地址与数据都必须延伸。

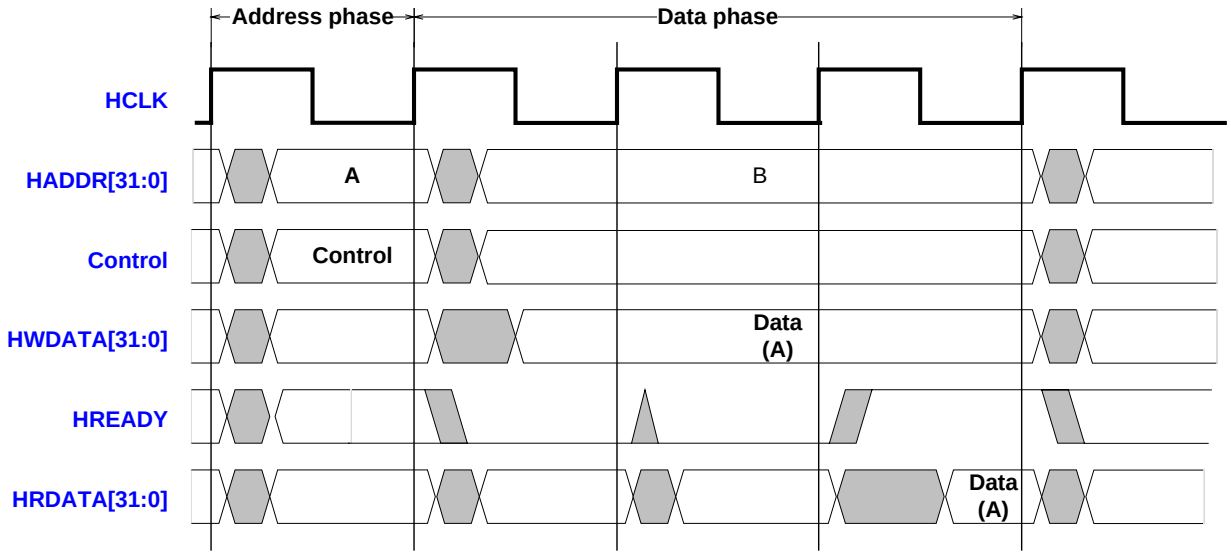

图 3-4 有等待时序的基本 AHB 传递

图 3-4 为一有等待状态的 AHB 传递，由图中可看到在数据模式时，时序必须多花二个周期的延迟等待时间(HREADY 为 0)，**S** 才能正常的接收到 **M** 写的数据或准备好 **M** 欲读取的数据。图 3-3 和图 3-4 为最基本的 AHB 传送方式，如将这两种传送方式结合起来，则可应用在各种多笔数据的传送组合中，在这里就不再举例说明。

3.4.3 多重传送(multiple transfer)

一次完整的数据传送会有多个传送周期，如图 3-5，下一笔地址可与上一笔数据同时在总线上，时序图的说明如下：

- X 与 Y 的存取都是**零等待**状态

若是在存取 X 发生等待状态时，X 的数据相位与 Y 的地址期间都必须延伸

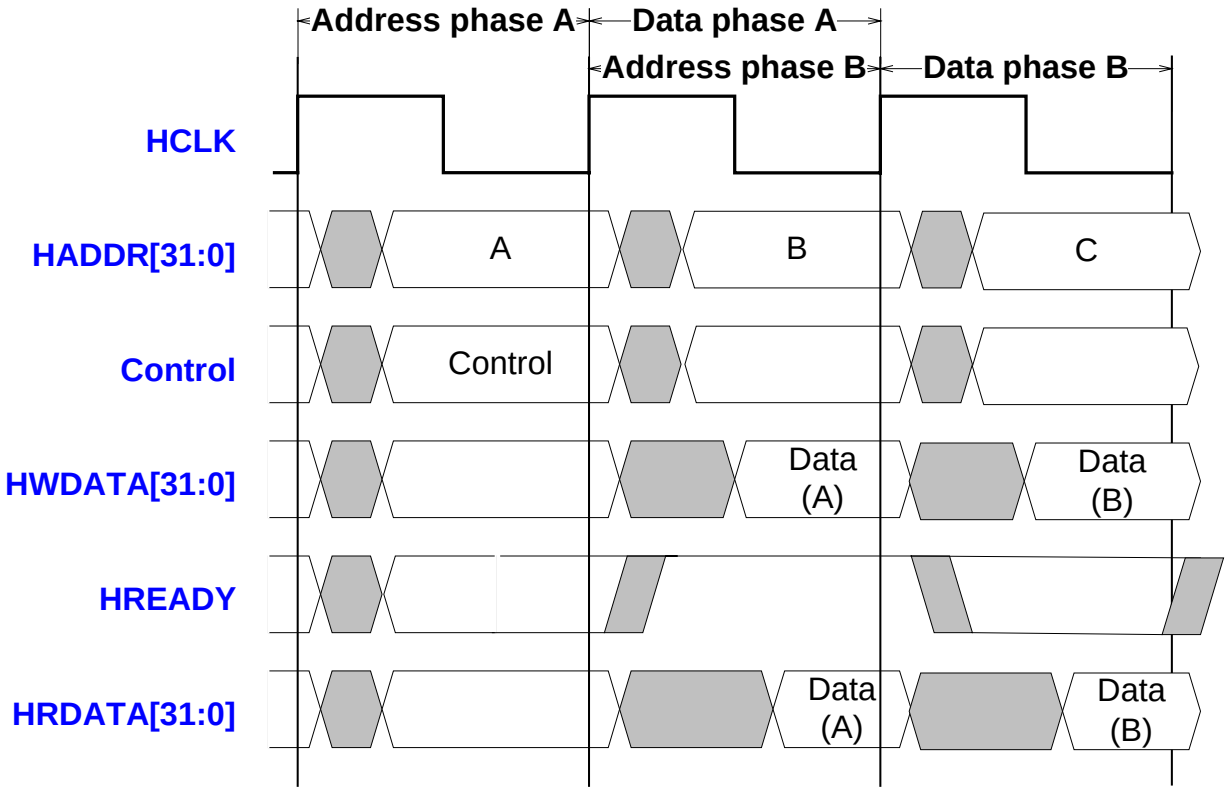

图 3-5 多笔的数据与地址重迭

3.5 控制信号：传送状态 HTRANS[1:0]

在 AHB 总线上，M 的传送状态可由 HTRANS[1:0]来表示，这两位所代表的意义如下：

表 3- 4 HTRANS[1:0]状态类型

HTRANS[1:0]	类型	含义	描述
00	IDLE	空闲状态	此时控制 AHB 总线的 M(M 得到允许信号后)，M 以 IDLE 表示 M 没有数据传送请求。S 必须忽略此时的传送(已收到的地址与其它控制信号)，并立刻以零等待 OKAY 信号来对 M 进行响应。
01	BUSY	忙状态	M 以此信号表示正在处理数据，此时 SLAVE 要响应 OKAY 的信号（零等待状态），此时 SLAVE 会忽略已收到的地址与其它控制信号。 一般在一个批量传送的中间，表示 M 发起一次批量传送，但是下一个传送不能立即进行，这样就使 AHBM 能在传送的中间插入空闲周期，这时候的地址和控制信号必须反映下一个传送数据的情况。作为 AHB 从设备必须忽略此时的传送，并以零等待的 OKAY 状态来响应 M。
10	NONSEQ	非连续状态	表示当前是单笔数据或突发传送的第一笔数据的传送，此时所送出的地址及控制信号与上一笔的传送无关。
11	SEQ	连续状态	表示突发传送中剩余的数据传送是顺序传送，地址及控制信号和前面的操作相关。地址等于上一笔地址加控制信号 HSIZE 字节， 控制信号与上一笔数据相同。。

以图 3-6 说明不同的传送型态:

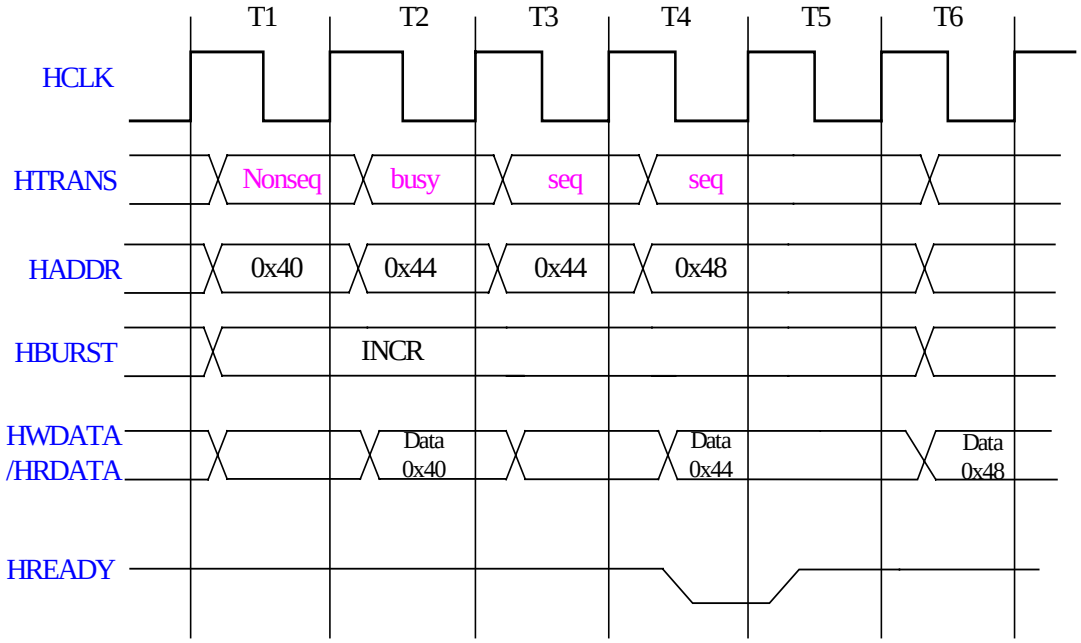

图 3-6 **M**-传送型态

- 在 T1 正沿触发时，传送开始，这是第一笔所以为 NONSEQ。
- 在 T2 正沿触发时，**M** 无法立刻完成传送，**M** 送出 BUSY 信号表示要延迟传送，**M** 可以无限多次的延迟。注意：地址 = 0x44（40 + 4 字节 = 44）。
- 在 T3 正沿触发时，**M** 以 SEQ 表示要继续序向的传送。

S 在 T4 正沿触发时，收到地址 0x 44 并且发现无法完成传送，就将 HREADY 拉低，而 **M** 也会再 T5 正沿时，收到延迟的讯息，如果此次的操作是读取的动作，0x44 地址所在的数据会在 T6 正沿而 HREADY 拉高时被收到。

3.6 控制信号：批量传送 HBURST[2:0]

批量传送是以群组为单位的传送方式，主要依地址的给法来区分：

- 递增传送(incrementing burst)，会依上一笔的地址来递增。
- 回绕传送（wrapping burst），如：回绕长度 = 4；每 4 个字节要对齐在 16 字节的范围内。如果第一个地址 = 0x64，则传送的顺序为 0x68、0x6C、0x60。

每次传送数据的大小由 HBURST[2:0]定义:

表 3-5 HBURST[2:0]操作类型

hburst[2-0]	类型	描述
000	SINGLE	单笔数据传送
001	INCR	**不定长的递增方式的批量传送**
010	WRAP4	4 个数据的回绕方式的批量传送
011	INCR4	4 个数据的递增方式的批量传送
100	WRAP8	8 个数据的回绕方式的批量传送
101	INCR8	8 个数据的递增方式的批量传送
110	WRAP16	16 个数据的回绕方式的批量传送
111	INCR16	16 个数据的递增方式的批量传送

AHB 从设备通常支持 SINGLE，INCR，INCR4，WRAP8，INCR8 和 INCR16。

AHB 对传送范围规定不可超过 1KB，递增传送的大小可为任意数目，但是不可超过 1KB 的范围。

3.6.1 提前结束 Burst(参 3.11)

There are certain circumstances when a burst will not be allowed to complete and therefore it is important that any slave design which makes use of the burst information can take the correct course of action if the burst is terminated early. The slave can determine when a burst has terminated early by monitoring the HTRANS signals and ensuring that after the start of the burst every transfer is labelled as SEQUENTIAL or BUSY. If a NONSEQUENTIAL or IDLE transfer occurs then this indicates that a new burst has started and therefore the previous one must have been terminated.

If a bus master cannot complete a burst because it loses ownership of the bus then it must rebuild the burst appropriately when it next gains access to the bus. For example, if a master has only completed one beat of a four-beat burst then it must use an undefined-length burst to perform the remaining three transfers.

以下分别说明几种**批量**式传送的时序图:

3.6.2 长度为 4 的回绕地址传送方式（four-beat wrapping burst）。

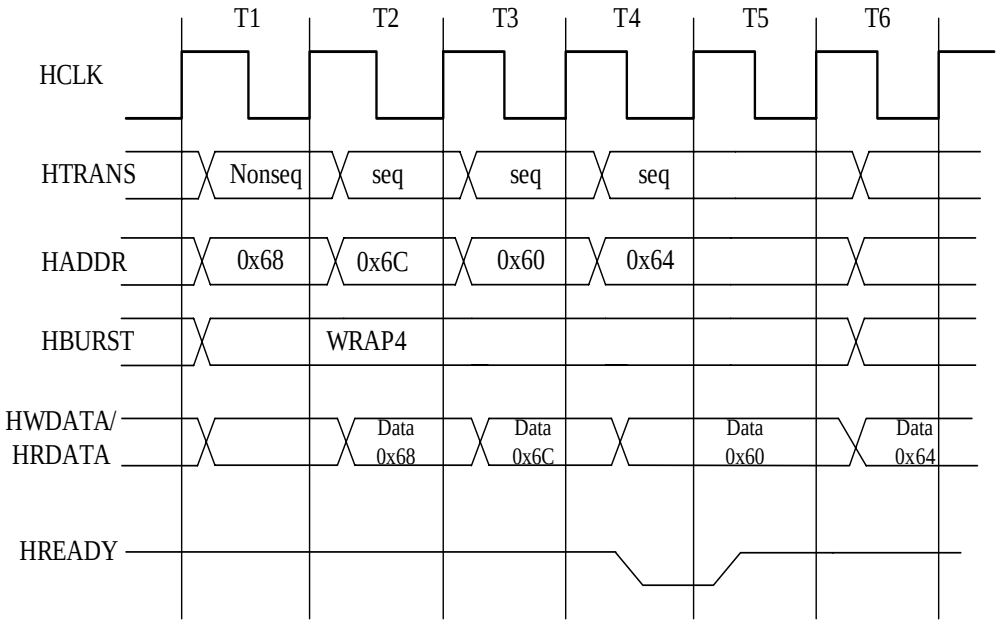

图 3-7 回绕地址传送 (长度 4)

- 在 T1 正沿触发时，第一个传送地址 0x68 送出，NONSEQ 表示这是第一笔的传送；WARP4 则表示这次的单位等于 4，而且要回绕地址。
- **S** 在 T3 时，回传一笔**数据**。注意地址 0x60 为一延伸周期，其数据在 T6 正沿才被接收到 (对读而言)或写入 **S** (对写而言)。

3.6.3 长度为 4 的递增地址传送方式（Incrementing burst of length 4）

以递增的方式计算传送地址分别为，0x68、0x6C、0x70、0x74

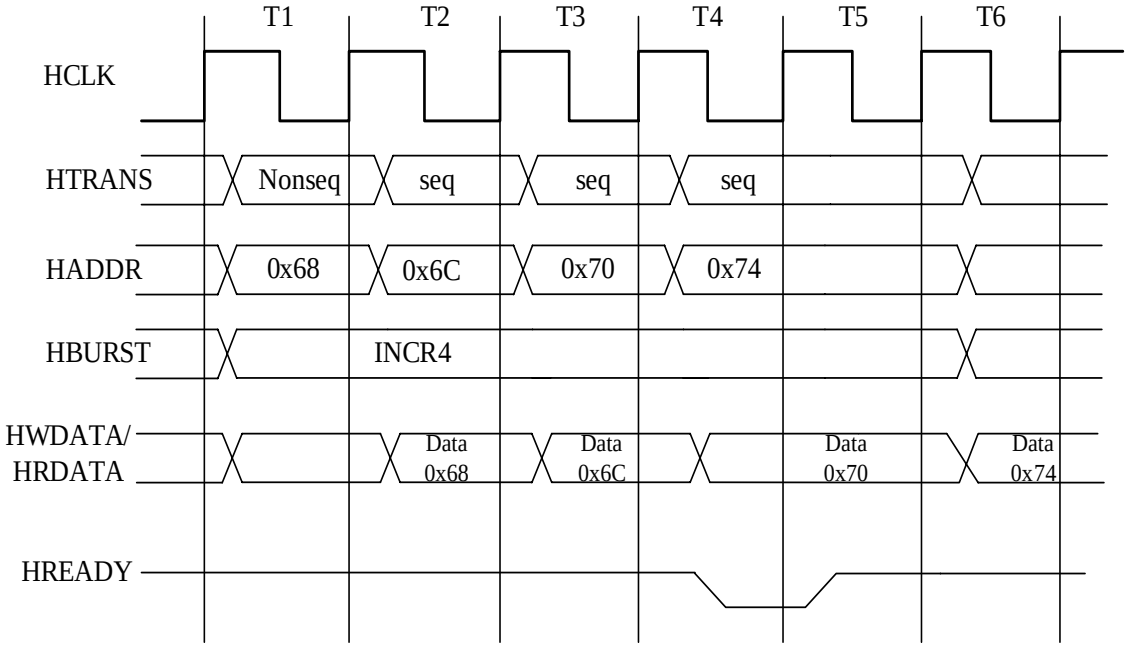

图 3-8 传送地址递增

3.6.4 不限长度的传送方式

当长度没有限定时，不同群组的传送会以 HTRANS[1:0]来表示，NONSEQ 表示另一个**批量**式的第一笔传送，如图 3-10。

- 第一笔 burst 是 16 位传送，由地址 0x68 开始。
- 第二笔 burst 是 32 位传送，由地址 0x108 开始
- 群组长度不限定
- NONSEQ 会出现在任一个群组的第一笔传送

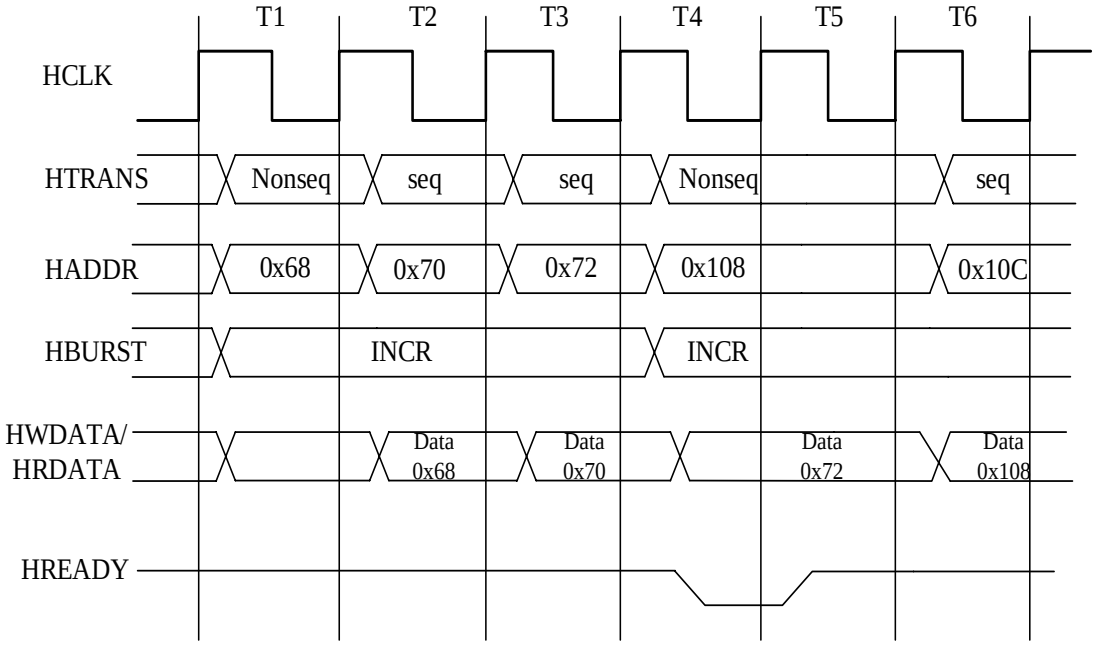

图 3-9 不限定长度的传送

3.7 控制信号：其它

表 3-6 控制信号

HTRANS[1:0]	传送状态
HBURST[2:0]	批量传送
HWRITE	传送方向
HSIZE[2:0]	传送大小
HPROT[3:0]	保护控制

控制信号提供传送信息，控制信号与地址信号相同时序特性,在批量传送过程中必须保持不变.

控制信号必须要在正确的时间点传送到总线上，并且维持到整个数据传送结束才可以释放。在 AHB 总线上的数据总线，将读入与写出总线分开，一般都是 32 位，以下是对各个控制信号的说明。

3.7.1 传送方向 HWRITE

- HWRITE：表示传送方向（依读或写的动作来决定传送**信号**的方向）
  - HWRITE 拉高时，**M** 必须对写入动作初始化，数据会由 **M** 放到 HWDATA[31:0] 总线上。
  - HWRITE 拉低时，**M** 会对读取动作初始化，被寻址到的 **S** 会将**数据**放到 HRDATA[31:0] 总线上。

3.7.2 传送大小 HSIZE[2:0]

- 传送数据大小由 HSIZE[2:0] **信号**控制，表示每次传送的字节数目。

表 3-7 HSIZE[2:0]**定义表**

hsize[2-0]	大小	**描述**
000	8 位	字节
001	16 位	半字
010	32 位	字
011	64 位	-
100	128 位	4-字数据线
101	256 位	8-字数据线
110	512 位	-
111	1024 位	-

HSIZE[2:0]与 HBURST[2:0]两个**信号**可合起来定义回绕地址的范围。
例如: HSIZE[2:0]= 32 位，HBURST[2:0]长度为 4 字节，则回绕地址须对齐在 16 字节。
通常的 AHB 从设备是 32 位数据线，所以它支持 8 位，16 位和 32 位 3 种大小。

3.7.3 保护控制 HPROT[3:0]

HPROT[3:0] 为总线协议保护**信号**用来定义存取的型态与特性：

表 3-8　HPROT[3:0]定义表

HPROT[3:0]	**描述**
[0] = 0	指令提取
[0] = 1	**数据**存取
[1] = 0	使用者存取
[1] = 1	**特权存取**
[2] = 0	不可有缓冲区
[2] = 1	可有缓冲区
[3] = 0	不可有快取
[3] = 1	可有快取

注意：

并非所有的 **M** 都会传送出 HPROT[3:0]，所以除非 **S** 有需要否则不会使用到 HPROT[3:0]信号。

3.8 地址译码

在AHB中，中央译码器会依据地址的高位产生不同的选择信号HSELx如图3-10，而S只有在HREADY拉高时，才会将地址、控制信号与选择信号收下，因为HREADY拉高时，表示当前的数据传送已完成。

AHB有几点特别的规定：

- 每个S至少都有1KB的内存空间。
- 每个M每次存取的空间不可超过1KB。
- 如果在NONSEQ或SEQ型态下存取到不存在的地址时，会有一个预设的S发出ERROR的响应信号。
- 如果在IDLE或BUSY型态下存取到不存在的地址时，会有OKAY的响应信号。
- 预设的S是中央译码器的一部分。
- 根据系统设计，使用地址的高位来产生选择信号。
- 地址的低位送给S以寻址其的内部存储器或缓存器。

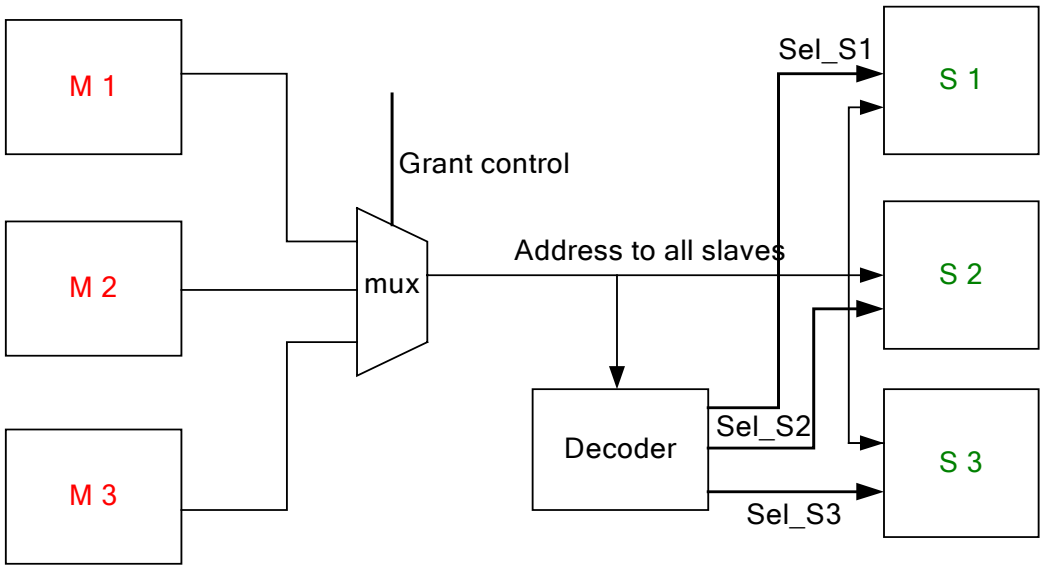

图3-10 S的选择信号

3.9 从传输应答

在 M 开始传送后，从可以选择如何进行传送。
一旦开始传送，协议不允许 M 可以取消传送.

S 使用 HREADY 与 HREPS[1:0]来响应传送的状态，
S 可以用下列几种可能表示**数据**传送的结果：

- **零等待**状态，立即完成传送
- 插入一个或多个等待状态，使有充足时间完成传送
- 因 ERROR 而结束传送
- **分段**传送，可暂时将总线放出，让其它的 **M** 先完成数据传送，再接着传送。

3.9.1 传送完成 HREADY

HREADY: 从应答主是否读写操作传送完成,从拉低此信号用来延续传送。
1- 传送完成,
2- 0-传送忙,

注 1:
S 需要 HREADY 是双向信号(HREADY 做为总线上的信号，它是 M 和 S 的输入；同时每个 S 自己的 HREADY 信号是输出。所以对于 S 会有两个 HREADY 信号，一个来自总线的输入，一个自己给到多路器的输出。)

注 2: 每个从必须预先设定它回到总线而插入的最大等待周期，用来计算访问总线的延时。推荐但不必须的是，等待状态最多只可使用 16 个周期,已防止任何访问长时间霸占总线.

3.9.2 其它从传输应答 HRESP[1:0]

除了 **HREADY**，其它传送结果/**S** 工作状态信号由 **HRESP[1:0]**信号代表：

表 3-9 HRESP[1:0]定义表

HRESP[1:0]	类型	描述	说明
00	OKAY	传送顺利进行。	S 可以用 HREADY（拉高）与 OKAY 的信号响应，代表传送已成功完成。HREADY 被拉低，加上 OKAY 的信号可插入在任何的响应信号（ERROR、RETRY 或 SPLIT）之间，也就是当 S 在未能正确的选择 ERROR、RETRY 或 SPLIT 时，可以插入 HREADY（拉低）与 OKAY 信号，但是等待状态最多只可使用 16 个周期。
01	ERROR	发生传送错误。	此次数据传送有误，此信号必须维持两个周期

10	RETRY	传送不能立即进行，但M必须不断地发起操作。	此信号必须维持两个周期，**M** 必须重提数据传送的要求
11	SPLIT		此信号必须维持两个周期，**S** 响应此信号表示此次的传送无法完成，要以分段传送方式来达成。等到 **S** 可以完成时，会知会仲裁器，让重提数据传送要求的 **M**，完成传送。

AHB 从设备的错误等异常情况可以有专门的错误处理模块，这样 AHB 从设备可只响应 OKAY 状态。

3.9.3 两周期应答(Two-cycle response)

除了 OKEY 之外， ERROR，RETRY 与 SPLIT 信号必须维持两个周期

3.9.4 错误应答 Error

当 M 收到一个 ERROR 的**信号**时，它可以选择结束当前的批量数据传送,继续传送剩下的批量数据也是可接受的,但通常不这样做。在**S**送出 ERROR 的**信号**之前必须先送出OKEY + HREADY (LOW)的**信号**，而 ERROR **信号**至少需要维持两个周期。如图 3-11。

- 在第二个周期 **S** 送出 OKAY+ low HREADY **信号**,让 **S** 有充分的时间决定是否需要发出 ERROR 的**信号**。
- 在 **M** 收到 ERROR **信号**后，立刻结束当前的数据传送。
- 注意：**S** 必须维持 ERROR 至少两个周期

有 ERROR 回应的时序图：

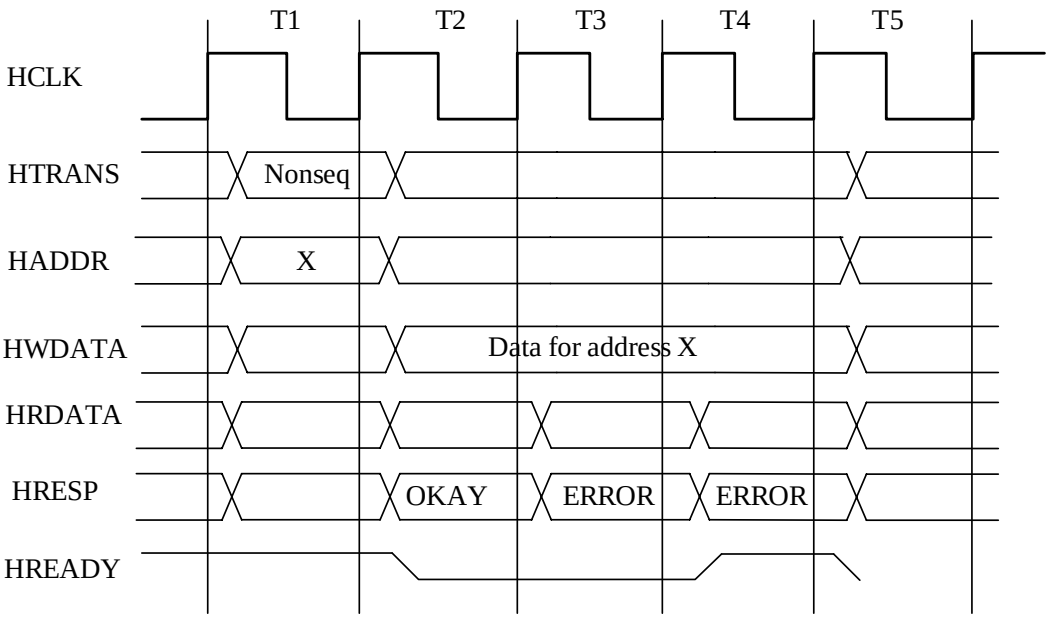

图. 3-11 Error 回应

3.9.5 分段传送与重传的异同

对于 **M** 而言，SPLIT 与 RETRY 可视为相同。SPLIT 与 RETRY 其不同之处在于对仲裁器与 **S**。

RETRY 与 SPLIT 的不同：

- RETRY
  - 仲裁器使用一般的优先权。
  - **S** 只要求 **M** 重提要求。
- SPLIT
  - 仲裁器会调整做**分段**传送 **M** 的优先权，所以即使该 **M** 的优先权比较低，还是可以及早得到总线使用权来完成数据的传送。
  - **S** 的数据准备好时，必须告知仲裁器，让相关 **M** 完成其要求。
  - **分段**传送必须由 **S** 与仲裁器支持，总线可放出让其它的 **M** 使用。

S 回应的时序：

- **S** 可以只给一个周期的 OKAY **信号**，如有需求可维持多个周期。
- ERROR、SPLIT 与 RETRY 至少都要维持两个周期。**S** 必须在上一次的最后一个周期送出 HRESP[1:0]**信号**，并且要维持到此次传送结束，HREADY 在传送的最后一个周期要拉高。
- 如果有需要， **S** 可以在响应的开始即插入多个 OKAY+HREADY (LOW)**信号**。

RETRY 时序图说明：

当 **S** 要求 **M** RETRY 时，会送出至少一个周期的 RETRY 加拉低的 HREADY **信号**，**M** 收到此**信号**时，表示要重新送出地址，如图 3-12。

- 在 T1 时，**M** 送出地址 0x68，而在下一个控制**信号**来之前就送出地址 0x72。
- 在 T2，**S** 送出 RETRY 加低 HREADY **信号**，表示要 **M** 重新传送，则下一个地址 0x72 会先被忽略。
- **S** 在 T2 就送出 HRESP[1:0]并且在 T3 时继续送出， HREADY 在 T3 结束前拉高。
- 当 **M** 收到 RETRTY 时必须以 IDLE 响应。
- 在 T4 时，**M** 重新送出地址 0x68，之后就如同 T1。当然这假设此 **M** 业已获得总线使用权。

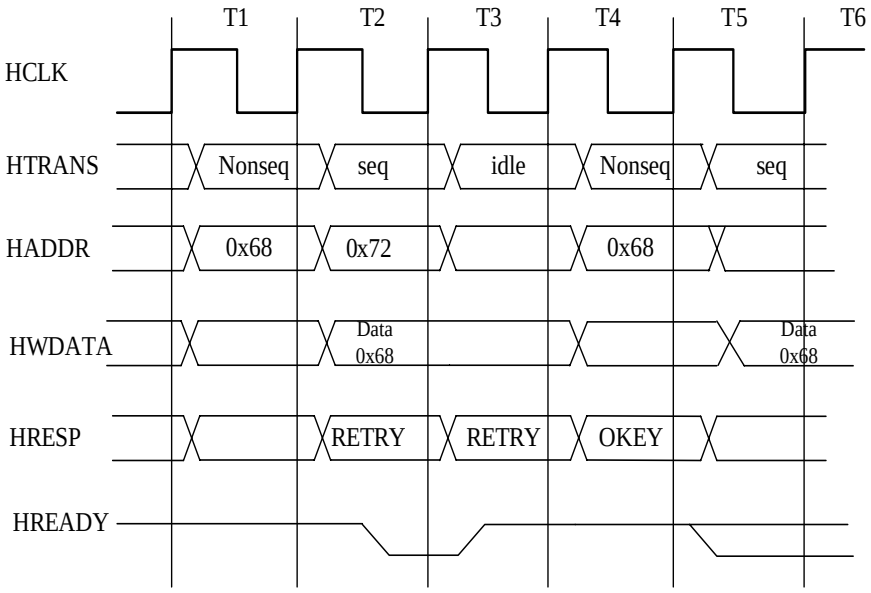

图 3-12 RETRY 传送机制

3.10 数据总线

AHB 将读与写的数据总线分开，避免了三态的使用，最小的宽度为 32 位。

3.10.1 HWDATA[31:0]

写出数据总线为 HWDATA[31:0]

- 写出数据由 **M** 发出。
- **M** 必须将数据总线上的数据维持住，直到 **S** 完成数据的接收 (HREADY HIGH)。
- 所有传送所对应的地址都必须对齐，例如：32 位的传送地址 Addr[1:0] = 00。

大端摆放法(big endian)或小端摆放法(little endian)都可使用

- 小端摆放法(以 32 位为例)
  - 低字节在 HWDATA.的低字节地址
  - Byte 0 (address offset = 0) 在 HWDATA[7:0]
  - Byte 1 (address offset = 1) 在 HWDATA[15:8]
  - Byte 2 (address offset = 2) 在 HWDATA[23:16]
  - Byte 3 (address offset = 3). 在 HWDATA[31:24]
- 大端摆放法
  - 低字节在 HWDATA.的高字节地址
  - Byte 0 (address offset = 0) 在 HWDATA[31:24]
  - Byte 1 (address offset = 1) 在 HWDATA[23:16]
  - Byte 2 (address offset = 2) 在 HWDATA[15:8]
  - Byte 3 (address offset = 3) 在 HWDATA[7:0]

3.10.2 HRDATA[31:0]

读入数据总线为 HRDATA[31:0]

- 在执行读取动作时，数据是由 **S** 所送出。
- 当 **S** 将 HREADY 拉为 HIGH 时，**S** 会在传送的最后一个周期前将有效数据送出。

数据摆法与写的动作相同。

3.10.3 位格式 Endianness

大端摆放法(big endian)或小端摆放法(little endian)都可使用

- 小端摆放法(以 32 位为例)
  - 低字节在 HWDATA.的低字节地址
  - Byte 0 (address offset = 0) 在 HWDATA[7:0]
  - Byte 1 (address offset = 1) 在 HWDATA[15:8]
  - Byte 2 (address offset = 2) 在 HWDATA[23:16]
  - Byte 3 (address offset = 3). 在 HWDATA[31:24]
- 大端摆放法
  - 低字节在 HWDATA.的高字节地址
  - Byte 0 (address offset = 0) 在 HWDATA[31:24]
  - Byte 1 (address offset = 1) 在 HWDATA[23:16]

- Byte 2 (address offset = 2) 在 HWDATA[15:8]
- Byte 3 (address offset = 3) 在 HWDATA[7:0]

3.11 总线仲裁机制(重点)

仲裁机制目的： 确保任何时刻只有一个 **M** 在访问总线.

仲裁器功能:

a.. 通过观察不同的总线请求信号，通过仲裁算法来决定哪个 M 有最高优先级.

b. 接收 S 的完成分段传送的请求

无法完成分段传送的 S 不需要知道仲裁器的处理进程，但需要注意到如果总线占用发生变化， 批量传送可能无法完成.

3.11.1 信号描述

AHB 在系统中最多可容入 16 个 **M**。

仲裁**信号**包括:

- HBUSREQx: 由 bus **M** 到仲裁器，**M** 透过此**信号**对仲裁器提出使用 bus 的要求。
  - 一个 **M** 可在任何时间要求仲裁器。
  - 对于固定长度的传送，只要要求一次就可，仲裁器会依据 HBURST[2:0] 来判断传送的长度。
  - 对于未定的传送长度，**M** 必须持续提出要求，直到完成全部的传送为止。
  - 当所有 **M** 都不需要使用 bus 时，必须在 HTRANS 上送出 IDLE **信号**，仲裁器就会将 bus 的使用权交给预设 **M** (default **M**)

- HLOCKx: 此信号必须与 HBUSREQx 配合，表示一个不可分段的传送动作，HLOCKx 必须再指定地址之前使能一个周期。
- HGRANTx: 由仲裁器输出到 **M**，仲裁器通知某个 **M** 表示此 **M** 得到 bus 的使用权。
- HMASTER[3:0]: 仲裁器以此信号表示当前是哪一个 **M** 在使用 bus，此信号也可用来控制地址多任务器。
- HMASTLOCK: 仲裁器以此信号表示当前的 bus 是被 lock 住的。
- HSPLIT[15:0]:一个可执行分段传送的 **S** 以此信号让仲裁器知道要让哪一个 **M** 完成其未完成的分段传送。

3.11.2 总线访问请求

仲裁器通常会实现在地址译码器内，它会接收每个 **M** 所发出的地址与 bus 要求**信号**，仲裁器也会将 grant 讯后传给要求的 **M**，并且以 HMASTER[3:0]显示当前是哪一个 **M** 可以使用 bus，如下图。

- 译码器 (Decoder with arbiter) 使用 HMASTER[3:0]来控制中央地址与控制**信号**的多任务器.
- 每个 **M** 在被允许之前就可将地址与控制**信号**送出。

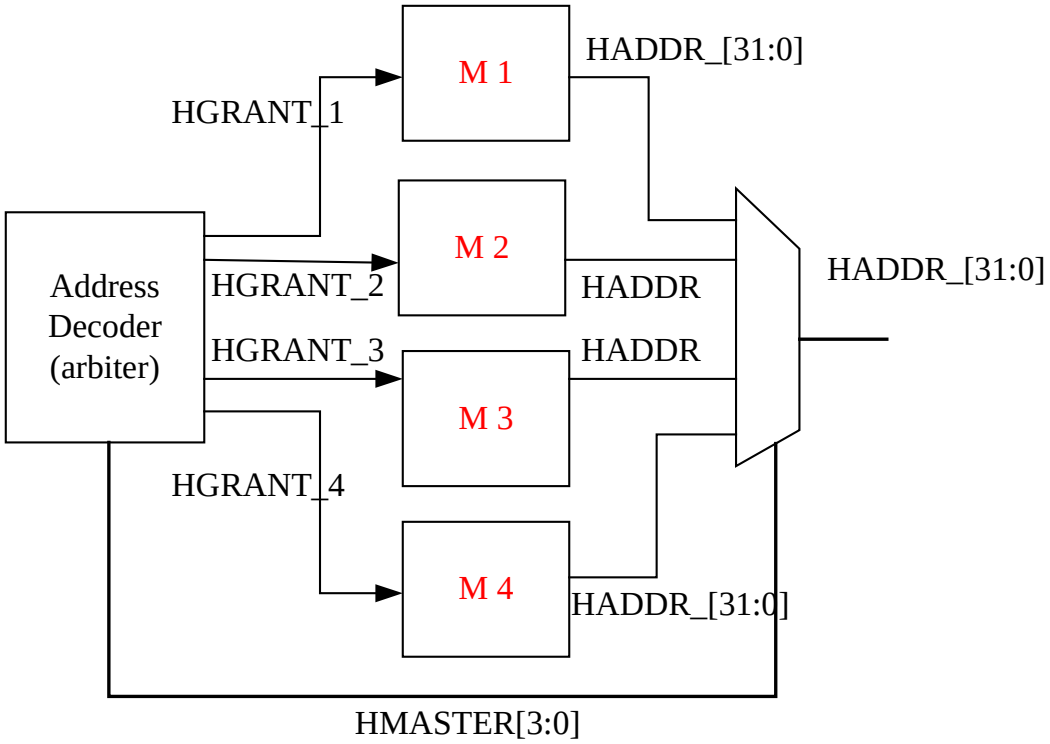

图 3-13 主设备 grant signals

3.11.3 总线访问批准

- 某个总线 M 通过 **HBUSREQx** 信号来向仲裁器请求总线访问权限，并可能在任何周期内都可以请求总线。 仲裁器通过采样时钟的上升沿，然后使用内部的优先算法来决定下一个获得总线的访问权.

- 通常在某个 burst 完成时，仲裁器只批准与之前不同的总线 M.
- 如果需要,仲裁器可以提前结束某个 burst,而允许更高优先级的 M 来访问总线.
- 如果主要锁总线访问，需要声明 **HLOCKx** 给仲裁，这样就不会有其它主可以得到总线授权

- 当主得到总线授权，并且正在执行一个定长的 burst,它不需要继续总线请求去完成这个 burst.

仲裁观察 burst 的进程，并使用 **HBURST[2:0]**来决定主有多少个传送。如果主希望在当前正在进行的 burst 之后继续 1 次 burst，它应在当前 burst 期间重新声明请求信号.

- 如果主在 burst 中间失去了总线访问权，它必须重新声明 **HBUSREQx** 请求信号来重新获得总线访问权限.
- 不定长的 burst,主应该继续声明请求知道它开始最后一个传送。仲裁不能预测不定长的 burst 什么时候结束，并切换授权.

- 主没有请求，但有可能获得了总线控制权。这是因为当没有主去请求总线时,主会授权给一个默

认的主.. 所以当某个主不需要总线访问，它需要把传送类型 **HTRANS** 设置为 IDLE 传送,这点很重要.

3.11.3.1 仲裁时序图 1: 零等待状态

在存取内存之前,**M** 送出 bus 使用要求，只要一收到 grant **信号**，**M** 就可开始传送数据，如图 3-14。

- 在 T1 提出使用要求。
- HGRANTx 在 T3 时拉起，在 T4 正沿时收到.
- HMASTER 表示当前的 **M** 是#3。 **M** #3 在 T4 送出地址**信号**。

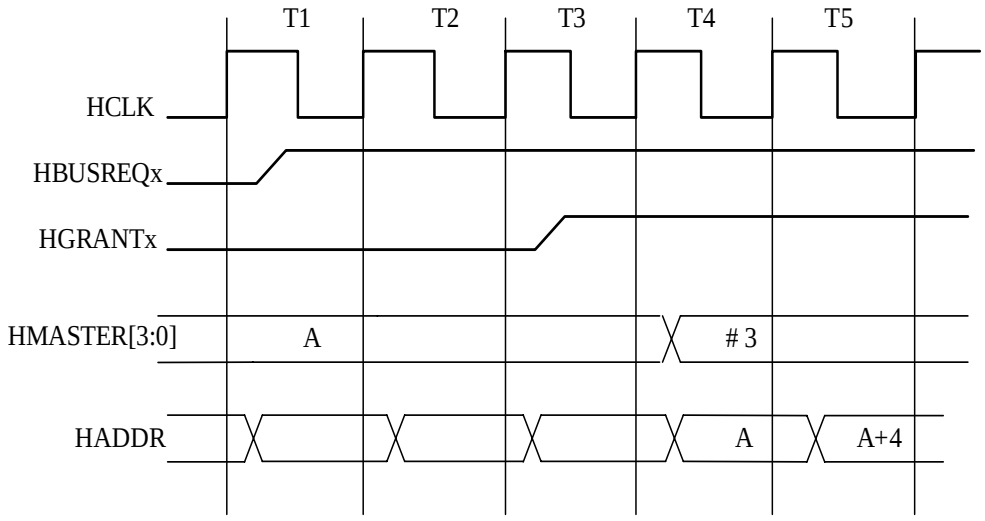

图. 3-14 仲裁时序图: **零等待**状态 (HREADGY = HIGH, not shown)

3.11.3.2 仲裁时序图 2：有等待状态

在存取内存之前，**M** 送出 bus 使用要求，只要一收到 grant **信号**，**M** 就可开始传送数据，如图 3-15.

- 在 T4 正沿, HREADY = LOW 表示需要等待。
- 在 T5 正沿, HREADY = HIGH and HGRANTx = HIGH，此时 **M** 可送出地址**信号**。亦即，HREADY AND HGRANT = true，将存取讯息送到地址与控制总线。

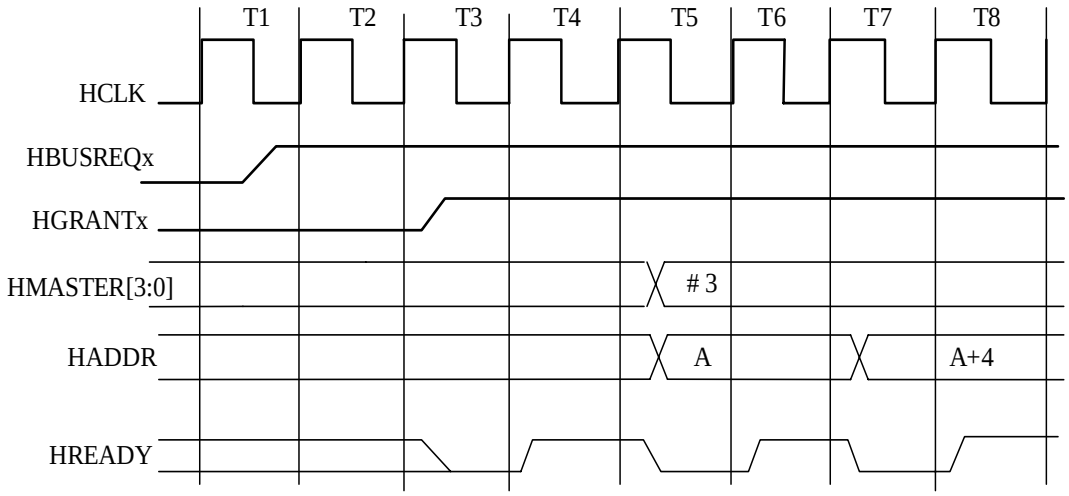

图 3-15 仲裁时序图: 发生等待状态

3.11.3.3 总线使用权的转换 1

数据总线的拥有权必须比地址总线的拥有权晚释放出去，这是因为数据期间发生在地址期间之后，如图 3-16。

- 在 T5 正沿，S 送出 HREADY 信号，表示 M 1 成功传送最后一次存取的地址及控制信号。
- 地址总线在 T5 正沿开始交换使用者，当 HGRANT_2 AND HREADY = HIGH，表示地址总线交换成功。
- 仲裁器必须知道 M 1 还有多少笔数据要传送，才可正确的发出 HGRANT_2 信号。
- 当仲裁器收到最后一个传输地址后，仲裁器就会改变 HGRANTx 信号。
- T5 刚开始时，M 2 送出地址与控制信号。
- 数据总线的交换只有在上一笔数据传送结束后才会开始。
- 在 T7 正沿，开始数据总线的交换。

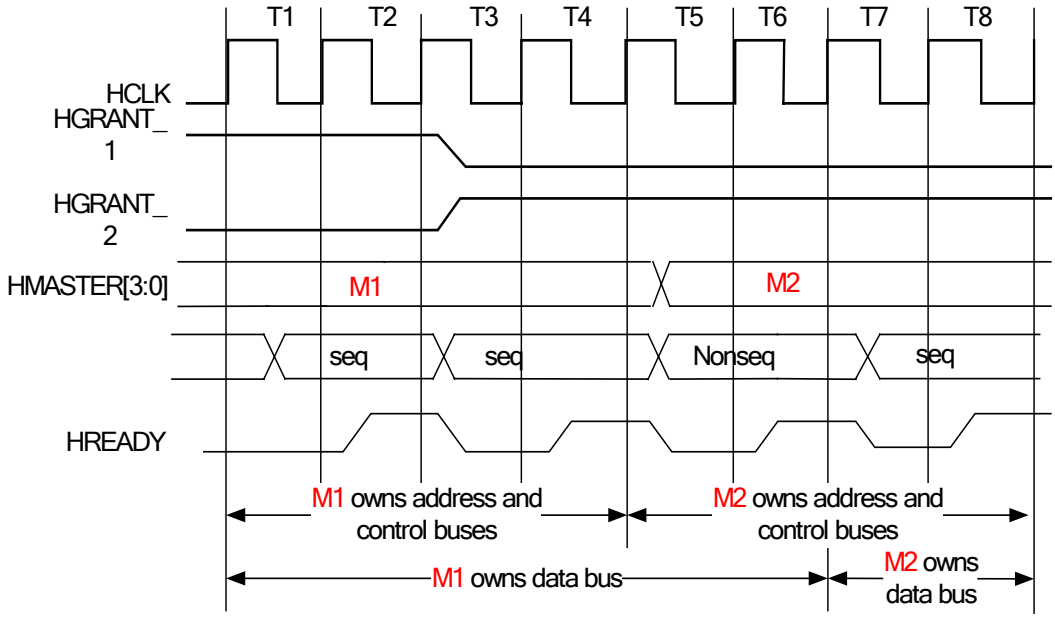

图 3-16 数据总线使用权的转换

3.11.3.4 总线使用权的转换 2

图 3-17 说明另一个总线的交换的范例。在仲裁器收到最后一笔地址后就将 HGRANT_2 信号送出。

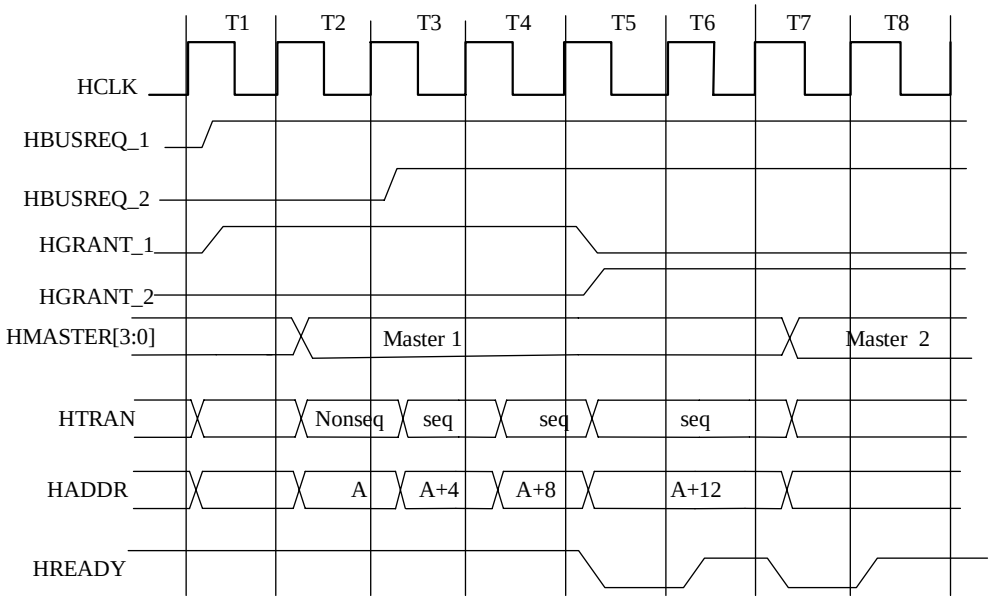

图 3-17 数据总线使用权的转换:**批量**式传送

在 T5 正沿，仲裁器收到地址(A+8)，此地址是倒数第二个传送地址，此时便改变 HGRANTx **信号**，新的 HGRANTx 会在最后一个地址(A+ 12)出现后，在 T6 正沿被收到。

3.11.4 提前结束 Burst

- 通常仲裁器不会移交总线控制权给新的 **M**,除非批量传送完成.
- 但仲裁器为避免存取时间过长, 可提早结束传送，。

M 如果被提早结束传送，如果一定要完成剩余的传送，就必须重新申请总线，**M** 必须让仲裁器知道还有未完成的数据传送，并要确保 HBURST 与 HTRANS 信号的匹配。

如 8 笔数据传了 3 笔，被中断后，HBURST 与 HTRANS 来表示后续进行 5 笔的未定长批量传送，或 4 笔定长批量传送+单笔传送.

3.11.5 锁定传送 Locked transfers

- 仲裁器允许 **M** 执行 locked transfers 以确定最后一笔传送完成而不会有 retry 或 error 的响应**信号**。
- **M 可在锁住传送后插入 IDLE 周期。**

仲裁必须从观察每一个主的 **HLOCKx** 信号，来决定什么时候主希望执行一个锁定顺序传送.仲裁然后负责确保没有其它主被授权，直到这个锁定顺序传送完成.

当一个锁定传送的序列完成后，仲裁将总是还会保持当前的主的授权来传一个附加的传送来保证锁定传送的序列的最后一个传送已经胜利完成，并且收到 SPLIT or RETRY 的应答之一.

所以推荐但并不是强制要求的做法是，主机在任何锁定序列后插入一个 IDLE 传送，在开始其它 burst 传送之前，来提供一个仲裁切换的机会.

3.11.6 预设总线主设备

- 每个系统都有一个**预设 M**。
- 没有 **M** 要求 bus 使用权时，仲裁器会将使用权交给**预设 M**。
- **预设 M** 只会执行 IDLE
- 如果所有 **M** 都还在等待 SPLIT 传送的完成，也会将 bus 使用权交给**预设 M**。

3.12 分段传送 HSPLITx(重点)

注意分段传送时的仲裁相关问题

分段传送

- S 可以用 SPLIT 来解决延迟太长的存取。
- 仲裁器必须观察响应信号并且遮掉(mask)已被分段传送的 M，使其等候 HSPLIT 信号的通知后才 grant 其总线使用权。
- 仲裁器必须在 HMASTER[3:0] 上产生一个 tag 表示哪一个 M 正在执行数据传送。
- 当 S 要执行分段传送的动作时，这个 S 必须存下此信号。
- 当 S 可以完成被分段的传送时会在 HSPLITx[15:0]信号上，送出要完成传送的 M 代号。
- 仲裁器以 HSPLITx[15:0] 来决定哪一个 M 可以重新获得 bus 的使用权。
- 仲裁器会在每个周期去观察 HSPLITx 信号。
- 当有多个 HSPLITx 信号时，会被 Ored 在一起传送给仲裁器。

3.12.1 分段传送顺序

1. 一 M 送出地址与控制信号，开始传送。
2. 被寻址到的 S 立即提供数据，或者送出分段传送应答信号(如果需要等些周期来获得数据)。在每次的传送，仲裁器会广播出当前是哪一个 M 在使用总线。S 需要记录 M 号，以便后面重传使用.
3. 有 SPLIT 信号响应时，仲裁器会将总线使用权交给别的 M 使用,如果所有其它 M 都收到 SPLIT 信号,则移交给预设 M。
4. 当 S 要完成分段传送时，S 会在 HSPLITx 上设定对应的 M ID,来指示哪个 M 可以被允许再次访问总线。
5. 仲裁器会在每个周期观察 HSPLITx 信号，如果有变化，就更改正确的 M 的优先权。
6. 仲裁器将会允许重新要求使用权的 M,但如果有更高优先级的 M 在用总线，可能会延迟。
7. S 会响应 OKAY 信号表示传送完成。

3.12.2 多重分段传送(multiple split transfers)

- 只要一个 M 有额外的 HBUSREQ 与 HGRANT 线， 即可发出超过一个以上的传送要求。
- 仲裁器将一组信号 (REQ and GNT) 视为一个 M。
- 可分段传送的 S 可被设计去处理额外的传送要求而不需纪录额外的地址与控制信号，但要记录 M ID。
- S 可利用 HSPLITx 来表示当前正在处理的是哪一个 M 要完成分段传送。
- 仲裁器可重新仲裁这些 M 的要求，高优先权的 M 可被重新给予使用权。

3.12.3 防死锁/ 锁住避免(deadlock prevention)

Deadlock 可能发生在很多不同 M 对同一个发出 SPLIT 和 RETRY 的 S 身上，而该 S 又无法处理哪么多 SPLIT and RETRY 时，便会发生 deadlock。

- Deadlock 可以透过限制分段传送的数目(最多 16 个)来避免。

- S 并不会记录下每个要求的地址与控制讯息，如果要作 SPLIT 的话，只会记录下当前传送对象的 ID 讯息。当 S 可完成**分段**传送时，才会锁定要求的地址与控制讯息。
- S 会依序地让仲裁器知道 S 要服务哪些要求。
- S 可以任何顺序处理要求的服务。

3.12.3.1 重传

当 S 无法在很短的时间内完成传送的动作时，它可以要求 M 重传。S 一次只可对一个 M 发出 RETRY **信号**，确定所服务的 M 是同一个时，RETRY 的动作才能完成。如果 RETRY 是不同 M 时可以采取下列措施：

- 给一个 error 回应
- 通知仲裁器
- 产生中断
- 重置系统

3.12.4 分段传送之总线转换

当 M 收到 SPLIT 或 RETRY 响应**信号**时，必须先进入 IDLE 状态并且重新要求 bus 使用权，一个新的 M 会在 IDLE 周期结束前由仲裁器决定出来，如图 3-18。M 在收到 SPLIT 或 RETRY **信号**后须立刻进入 IDLE 状态。

- 在 T2 与 T3，S 会传两个周期的 SPILT **信号**。
- 该 M 在 T3 正沿时发现有 SPLIT 响应**信号**，于是进入 IDLE 状态。
- 仲裁器在 T3 正沿时，决定下一个 bus 的使用者。
- 在 T4，新的 M 送出地址与控制**信号**

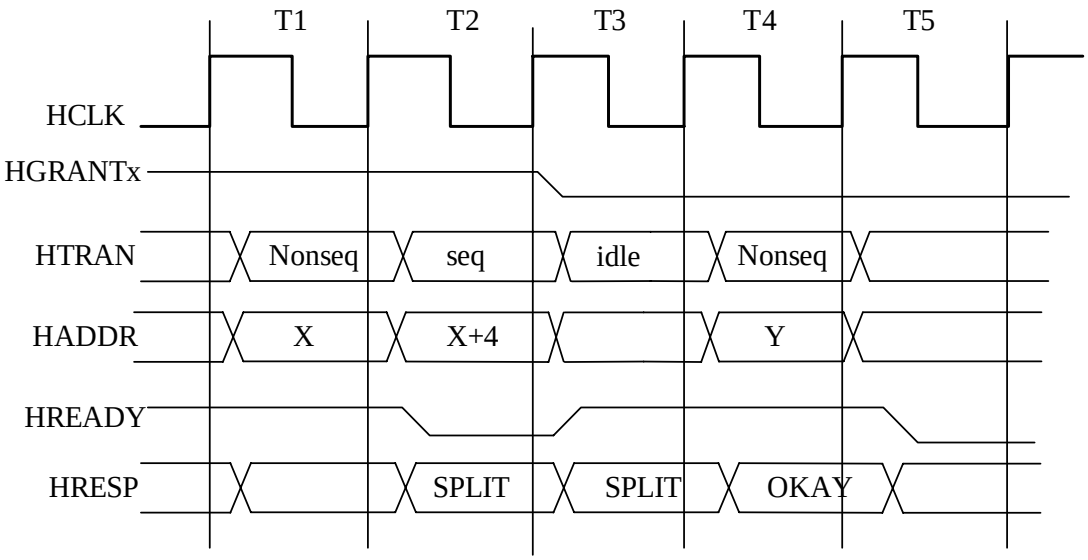

图 3-18 **分段**传送之总线转换

AHB 重置的规定:

- HRESETn 一异步**信号**，被拉低后在 HCLK 正沿后拉为 HIGH 。
- 在 reset 时，所有的 M 都必须为 IDLE。

split 应答是由于 S 当前不能全部准备好 M 所请求的全部数据，从而给出该应答信号，从而使得仲裁器屏蔽掉该 M 的总线请求，使其不能获取总线，以给 S 留出准备数据的时间。

当 S 准备好数据之后，它将通过 HSPLITx 信号通知仲裁器取消前面对该 M 的请求屏蔽，使其能够重新请求总线，完成传送过程。SPLIT 操作需要 S 和仲裁器都提供硬件上的支持，实现起来有一定的复杂度。

3.13 复位机制 HRESETn

复位信号 HRESETn,是低电平有效,并且是所有总线单元的首要复位. 复位可能异步发起，但撤消是与 HCLK 的上升沿同步的.

在复位期间，所有主必须确保地址与控制信号处于有效电平，并且 HTRANS[1:0]指示为 IDLE.

3.14 数据总线位宽 HSIZE[2:0]

可配置 8 位~1024 位数据总线宽度: 8, 16, 32, 64, 128, 256, 512 or 1024 位

推荐最少数据总线带宽 32 位..,

3.15 宽总线-窄从 Implementing a narrow S on a wider bus

从要在宽带宽上操作，需要增加的相应逻辑功能,不同位宽转换电路:

a. 复制数据总线到宽带总线

b. 确保只有一半总线变化。可以省功耗.

S 只能接受与其本接口相同的总线传送，如果 M 是宽带宽，而没有附加逻辑来分离的，S 可应答 ERROR

3.16 窄总线-宽从 Implementing a wide S on a narrow bus

从要在窄带宽上操作，同样需要增加的相应逻辑功能, 不同位宽转换电路:

3.16.1 Ms

Bus Ms can easily be modified to work on a wider bus than originally intended, in the same way that the S is modified to work on a wider bus, by:

? multiplexing the input bus

? replication of the output bus.

However, bus Ms cannot be made to work on a narrower bus than originally intended, unless there is some mechanism included within the M to limit the width of transfers that the bus M attempts. The M must never attempt a transfer where the width (as indicate by HSIZE) is wider than the data bus to which it is connected.

3.17 AHB 组件

3-18~3-21 小节为 AHB 系统的各组件

3.18 AHB 从(重点)

S 应答由系统中的主发起的传送. S 使用从译码器输出的 HSELx 信号来决定它什么时候作应答.r. 其它传送需要的信号，如地址与控制信息由 **M 产生**.

3.18.1 从接口框图

图 3-19 为 AHB **从设备**接口。

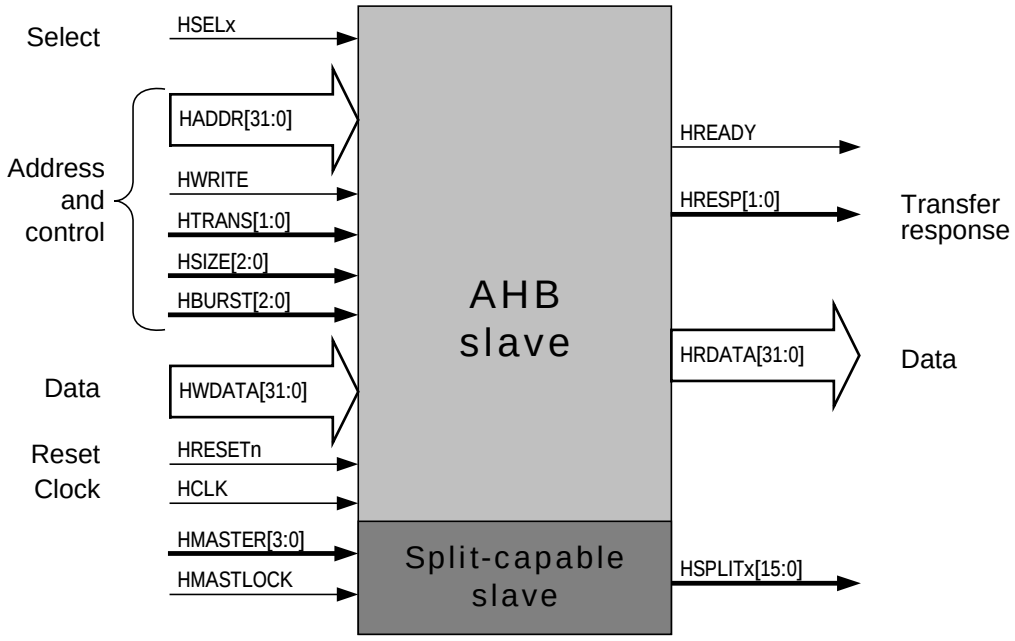

图 3-19 AHB **从设备**接口

仲裁器在决定出哪一个 **M** 拥有总线使用权之后，会将这个 **M** 数据地址、控制**信号**及欲写入 **S** 的数据选出，并且送至每一个 **S**，而所选出的数据地址会再经由 AHB 译码器产生唯一的 HSELx 使能**信号**来启动一 **S** 的数据传送。**M** 启动一个数据传送之后，被使能的 **S**(即 HSELx 为 1 的 **S**) 会发出 HREADY **信号**来决定是否要延长当前数据的传送，若 **S** 响应 HREADY 为 0，表示此笔数据的传递必须被延迟，若 **S** 送出的 HREADY 为 1，则表示 **S** 能够完成此笔数据的传递。

由图中可发现，**S** 除了用 HREADY **信号**来告知此笔数据是否需要额外的延迟时间之外，还会透过 HRESP[1:0]**信号**响应当前数据传送的情形，以下将说明四种 **S** 的响应型态：

第一种响应为 OKAY，当一笔数据可完成传送时，**S** 会响应以 HREADY 为 1，且 HRESP[1:0]为 OKAY；另外 **S** 也会在必须差入额外延迟时间(HREADY 为 0)，但未决定出何种响应方式时，作出 OKAY 的响应；

另一种的响应方式为 ERROR，这种响应会在 **M** 所要求传送的数据发生错误时发生，而这种错误常常出现在 **M** 试图去存取一有读或写的保护机制模块，如 **M** 试图在只读存储器来写入**数据**。

第三种响应方式为 RETRY，此种响应会在 **S** 无法立即完成此笔数据传送，希望 **M** 重新传送此笔数据

时发生；

最后一种 S 响应模式为 SPLIT，这种方式的响应跟 RETRY 相似，都是发生在数据未能完成传递时。最大的不同是仲裁器在这两种信号响应之后，选择 M 时所使用的权位算法不一样。如为 SPLIT 响应，仲裁器只允许其它 M 来对 S 作存取的动作，即使要求数据传送的 M 比当前 M 的优先权位来的低，也就是说仲裁器不会选择当前的 M 来进行数据传送；如果 S 响应的是 RETRY，那么仲裁器会用一般的权位算法去找出当前发出数据传递要求权位最高的 M 来进行数据传递。

AMBA AHB 的 SPLIT 传递方式改善了整个总线的使用效率，我们在这里特别说明 AHB 中 SPLIT 传递的程序。在 M 启动一数据传送之后，假设 S 需要一些延迟时间才能获得数据时，S 会透过 HRESP[1:0] 响应以 SPLIT 传送模式且将当前传送的 M 编号 HMASTER[3:0]记录下来，此时仲裁器将允许其它 M 作新的数据传递。当 S 可继续进行刚才的数据传送时，S 会根据刚才所记录的 HMASTER[3:0]值回应以 HSPLITx[15:0]信号给仲裁器，由于仲裁器每个周期都会观察这个信号，仲裁器此时会回存 HSPLITx[15:0]信号中 M 的优先权值，也就是刚才被记录下来的 M 编号。若 HSPLITx[15:0]信号中的 M 优先权值为当前众多发出数据传送要求的 M 权值最高者，那么 AHB 将继续完成这笔未完成的数据传送；若回存的 M 权位不是最高时，仲裁器将不会优先处理此次的 SPLIT 传送。

3.18.2 从时序

3.19 AHB 主(重点)

分析 AHB 的 M、S、仲裁器和译码器电路的运作情形。图 3-21 为 AHB 中的 M 接口图。M 的接口是整个 AMBA 架构中最复杂的部份，以下通过 M 进行的数据传送来介绍这些接口信号的用途。

3.19.1 主接口框图

AHB 的 M 接口如图 3-20，接口的信号与协议将会在本节中讨论。

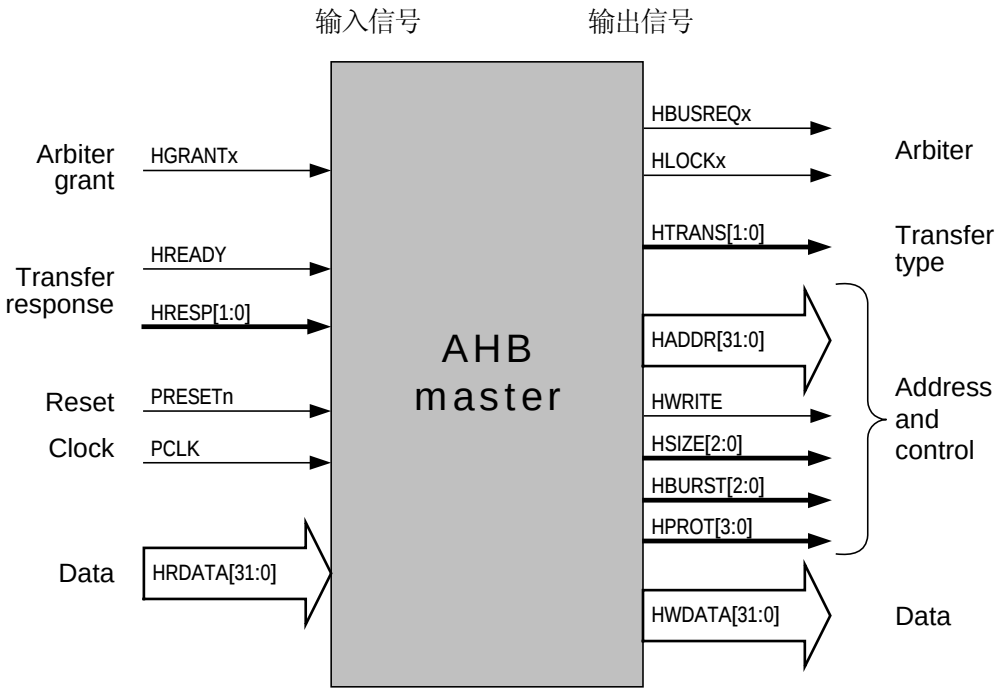

图 3-20 AHB-MASTER 接口

每一次的数据传送可分成四种型态，M 用 HTRANS[1:0] 信号来决定此次传送数据的型态，这四种传送型态分别是 IDLE、BUSY、NONSEQ 和 SEQ。

当 M 传送的数据型态为 IDLE 时，表示 M 这时候想要拥有总线的使用权，但并不需要作数据的传送，此时的 S 必须立即作出 OKAY 的响应；

第二种数据传送型态为 BUSY，当 M 进行一连续笔数据传递期间，若有些数据无法实时在下一个周期作传送，此时 M 会发出 BUSY 信号来延迟此笔数据的传送，S 这时候也会响应一个和 IDLE 传送型态一样的 OKAY 信号，同时忽略这笔数据的传递；

另一种传送型态为 NONSEQ，NONSEQ 的传送型态表示此次的传送为单笔数据传送或一连续笔数据传送中的第一笔，因此这种的传送型态，数据的地址和控制信号跟前一笔数据不具有关联性；

最后一种数据传送型态为 SEQ，在一连续笔数据的传递中，除了第一笔数据之外，其它的数据传递型态为 SEQ (第一笔为 NONSEQ)，这种数据传递的控制信号和前一笔相同，而数据的地址则为前一笔数据地址加上由 HBURST[2:0]和 HSIZE[2:0] 信号所决定出的地址累加值。

3.19.2 主时序

图 3-21 为一数据传送型态范例。

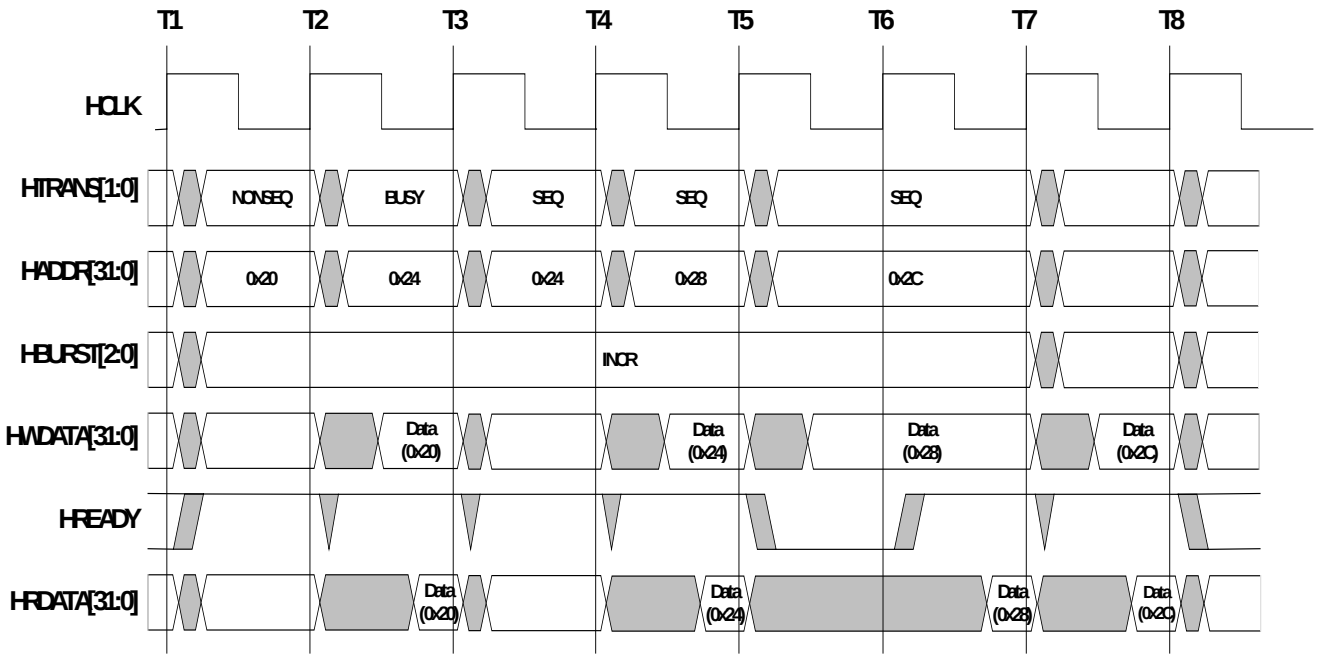

图 3-21 AHB 数据传送型态范例

M 在 T1 后发出一连续笔数据传送的第一笔(HADDR[31:0]=0x20)要求，因此传送的数据型态为 NONSEQ；

在第二笔**数据**(HADDR[31:0]=0x24)的传送时，原本的传送型态应为 SEQ，但因 **M** 无法实时准备好第二笔数据的传送，此时的 **M** 会送出 BUSY 传送型态来延迟第二笔数据的传送；

而在第三笔数据(HADDR[31:0]=0x28)的传送时，由于 **S** 无法在下一周期完成 **M** 欲读取的数据准备，此时 **S** 会透过 HREADY **信号**来增加一个周期的延迟等待时间(HREADY 为 0)；范例中的最后一笔数据(HADDR[31:0]=0x2c) 的传送型态为 SEQ，且 **S** 不需要额外的延迟等待时间。

AHB **M** 每次所传送数据的频宽由 HSIZE[2:0]**信号**线来决定，AMBA 共支持 8、16、32、64、128、256、512、1024 位的数据频宽，当 AMBA AHB 的 **M** 要求作一连续笔数据的传送时，HSIZE[2:0]和 HBURST[2:0]这两个**信号**将决定这些连续笔数据传送地址的计算。表 3-10 为其所支持一连续笔数据传递的型态，以下简单的介绍各种型态之间的差异。由表中可知，AHB 除了支持四笔、八笔、十六笔连续数据传送外，亦支持单笔及一未定长度的数据传送。连续笔数据的传送协议有两种模式，

第一种为循序地址增加模式 (INCR)；另一种为回旋地址增加模式(WRAP)。在循序地址增加模式中，每一笔数据的传送地址将循序增加。假设当前传送的模式为 INCR4，每笔数据为三十二位且作四笔连续的数据传送，若第一笔数据地址为 0x34，那么此次连续传送的数据地址分别为 0x34、0x38、0x3c 及 0x40；

另一种连续笔数据的传送模式为回旋地址增加模式(WRAP)，这种模式和循序模式最大的不同是，若第一笔数据的地址除以此次传送数据的总字节数所得的余数不为 0，则当它存取至数据的地址除上总字节数所得的余数为 0 时，此笔数据地址会等于当前的数据地址减去此次传送全部数据的总字节数。举例来说，假设当前的传送模式为 WRAP4，每笔数据为四个字节且进行四笔连续的传送，则全部的传送总位数为十

六字节，且此次传送的起始地址为 0x34，那么在 0x34、0x38 及 0x3c 地址的数据传送完毕后，下一笔的数据地址会等于原数据地址(0x40)减去此次传送的总字节数(0x10)，也就是说最后一笔的数据地址为 0x30。

虽为连续笔数据的传送，但在传送的过程中(第一笔除外)，**M** 所发出的 HTRANS[1:0]**信号**应为 SEQ 或 BUSY，如传送型态为 NONSEQ 或 IDLE，则表示有另一连续笔数据已开始第一笔数据的传送，之前的数据传送已经被中止。在此笔数据传送完毕后，**M** 可指定传送的数据型态为 INC 并要求继续刚才未完成的数据传送。

表 3-10 连续笔数据传送的型态

HBURST[2:0]	型态	数据传送描述	范例
000	SINGLE	单笔	0x48
001	INCR	未指定笔数	0x48, 0x4c, 0x50
010	WRAP4	四笔回旋式	0x48, 0x4c, 0x40, 0x44
011	INCR4	四笔递增式	0x48, 0x4c, 0x50, 0x54
100	WRAP8	八笔回旋式	0x48, 0x4c,…, 0x5c, 0x40, 0x44
101	INCR8	八笔递增式	0x48, 0x4c,...., 0x5c, 0x60, 0x64
110	WRAP16	十六笔回旋式	0x48, 0x4c,… … …, 0x7c, 0x40, 0x44
111	INCR16	十六笔递增式	0x48, 0x4c,… … …, 0x7c, 0x80, 0x84

AMBA AHB 也支持 **M** 发出 LOCK 数据传送要求，若要启动此种数据型态的传送，**M** 会发出 HLOCK 使能**信号**。在 **M** 得到总线使用权时，仲裁器如发现 **M** 的 HLOCK **信号**使能，则在这次数据传送完成之前将不允许其它 **M** 来使用这个总线。仲裁器同时会使能 HMASTLOCK **信号**，并送至每个 **S** 来通知他们当前正在进行一 LOCK 型态的数据传送。直到完成此次的数据传送，其它的 **M** 将不会得到总线的使用权。

3.20 仲裁器(arbiter)

The role of the arbiter in an AMBA system is to control which **M** has access to the bus. Every bus **M** has a REQUEST/GRANT interface to the arbiter and the arbiter uses a prioritization scheme to decide which bus **M** is currently the highest priority **M** requesting the bus.

Each **M** also generates an HLOCKx signal which is used to indicate that the **M** requires exclusive access to the bus.

The detail of the priority scheme is not specified and is defined for each application. It is acceptable for the arbiter to use other signals, either AMBA or non-AMBA, to influence the priority scheme that is in use.

3.20.1 接口框图

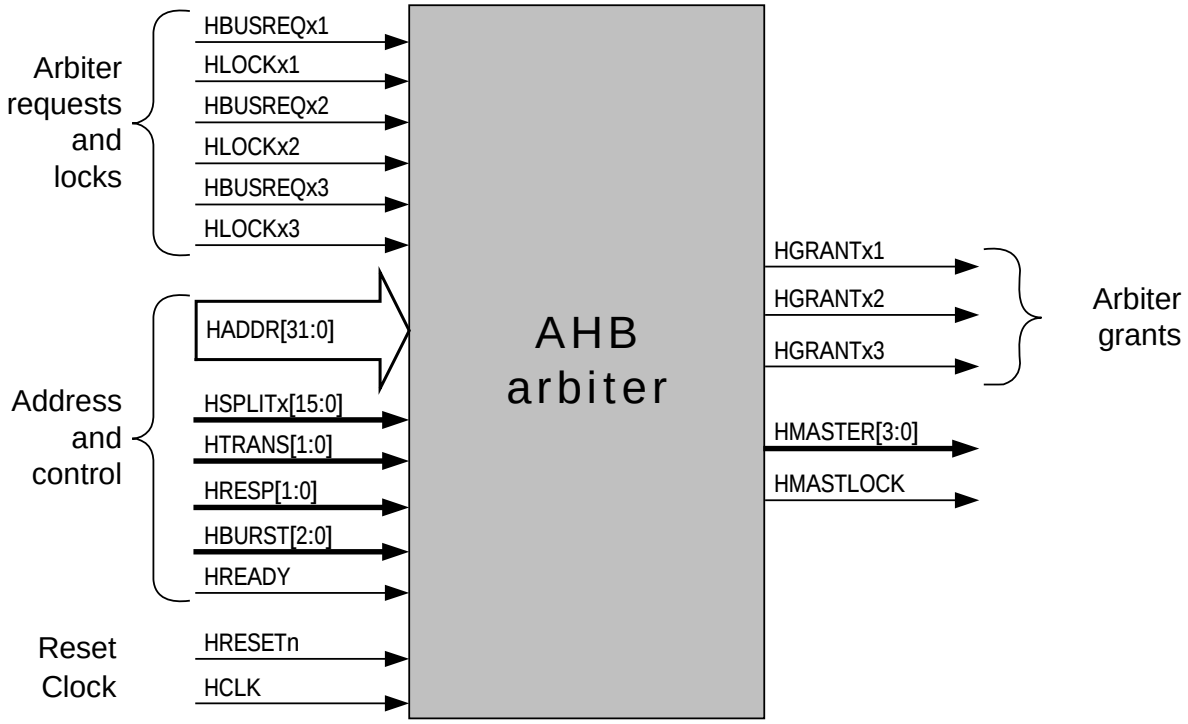

图 3-22AHB 仲裁器接口图

图 3-22 为 AHB 仲裁器的接口图。仲裁器的主要任务为决定哪个 **M** 可拥有总线的存取权。每个 **M** 会有一个 HBUSREQx **信号**接至仲裁器，仲裁器使用其权值算法来决定哪个 **M** 可以拥有总线存取权。

3.20.2 时序

图 3-23 为仲裁器选择一 **M** 可拥有总线存取权的时序范例。

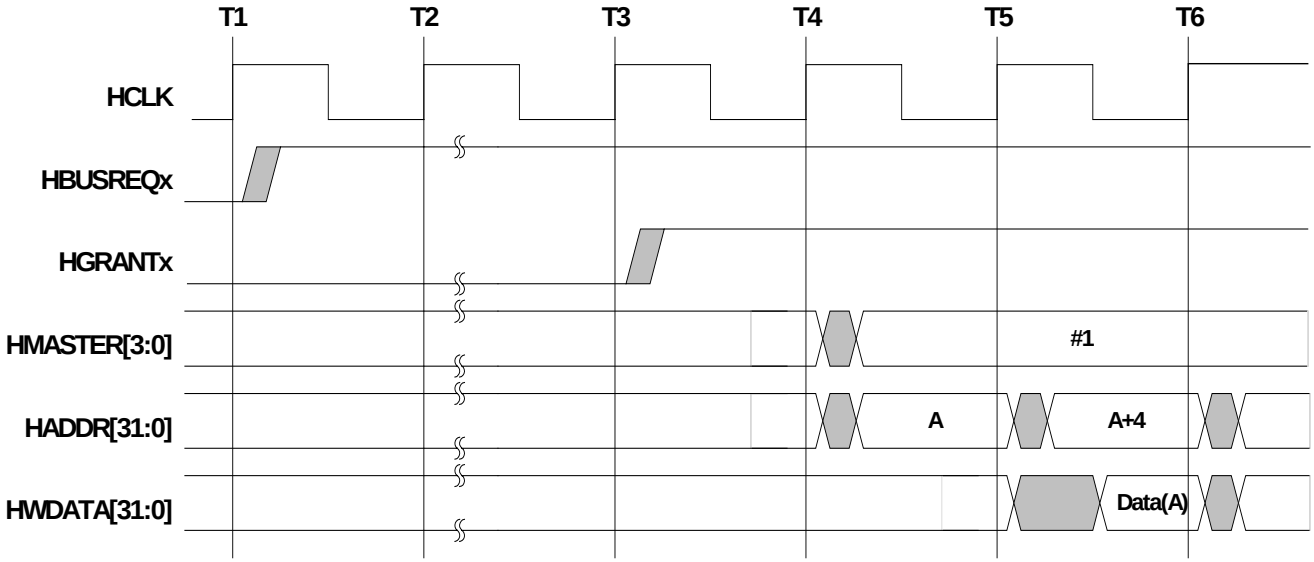

图 3-23 仲裁器的基本运作范例时序图

由图中可知，在 M 发出总线的存取要求后，仲裁器会用它的权值算法来选择一个适合的 M，并对这个 M 发出一 HGRANT 使能信号，赐予它总线的拥有权。而 HMASTER[3:0]即为所选择到的 M 编号，此信号会经过多任务器，然后从众多 M 地址线选择出一地址线(HADDR[31:0])来进行读或写的数据传送。

3.21 译码器(decoder)

译码器: 数据译码功能，通过与系统存储映射的分离，提升外围设备的便携独立性，

图 3-24、3-25 分别为 AHB 译码器接口图和时序图，数据地址由仲裁器选出后会进入 AHB 译码电路，此译码电路会从数据地址高字节中选择并产生唯一的 HSELx 使能**信号**，并将此**信号**送至 **S**；至于 HADDR[31:0]低字节部分，将由 **S** 来作解碼的动作。

3.21.1 接口框图

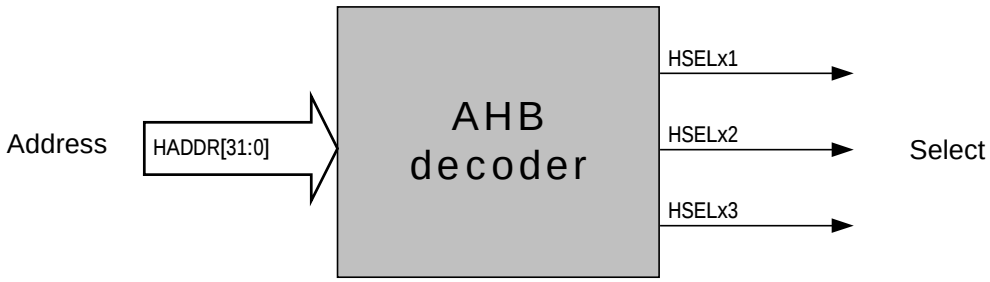

图 3-24 AHB 译码器接口图

3.21.2 时序

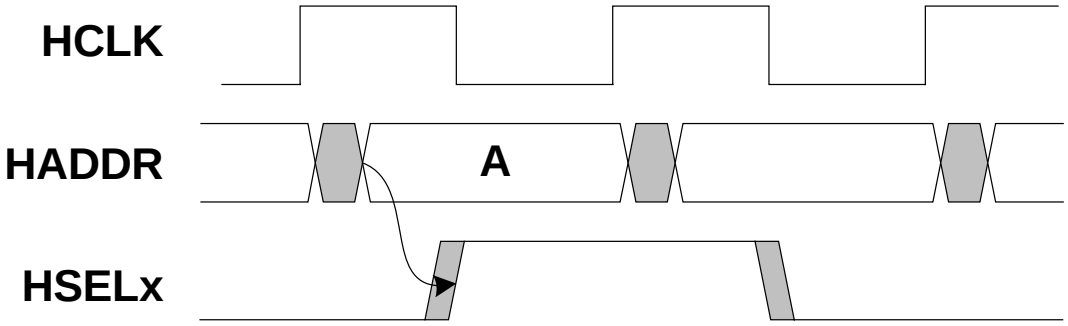

图 3-25 AHB 译码电路时序图

4 ASB 总线(未实用,忽略)

AMBA 中另一高效率系统模块为 ARM 系统总线(ASB)，并未普及，已被 AHB 所取代，因此在本文中将不提及。

5 APB 总线

5.1 什么是 APB 总线?

APB 的全称：Advanced Peripheral Bus

AMBA 中的 APB 总线主要用在低速且低功率消耗的外围，且可针对外围作功率消耗及复杂接口的最佳化。在 APB 总线中，唯一的 **M** 为 APB bridge，其它一些低速和低功率的外围皆为 **S**。因此 APB 总线不需要有一个像 AHB 一样的仲裁器及其它复杂的线路，也就是说 APB 总线的整个架构较 AHB 简单许多。

为了使APB容易被整合进大部分的设计流程中，APB规订所有信号必须在时钟正沿触发时进行传递。

这样的设计方式可改善高速电路的效率且 EDA 对这种设计方式的处理技术较为成熟且容易实现，例如在静态时序分析及可测性设计的考虑。全部的数据和信号皆在频率正沿触发来进行传递的设计方式，在以周期为基底的仿真器中也能有较好的模拟表现；除此之外一般特定应用集成电路链接库(ASIC library)在正沿触发的正反器设计通常有较佳的表现，也就是说使用集成电路链接库的设计者将得到较好的电路效能。

5.1.1 一个典型的基于 AMBA 总线的系统架构

下面复习一下AMBA总线架构：

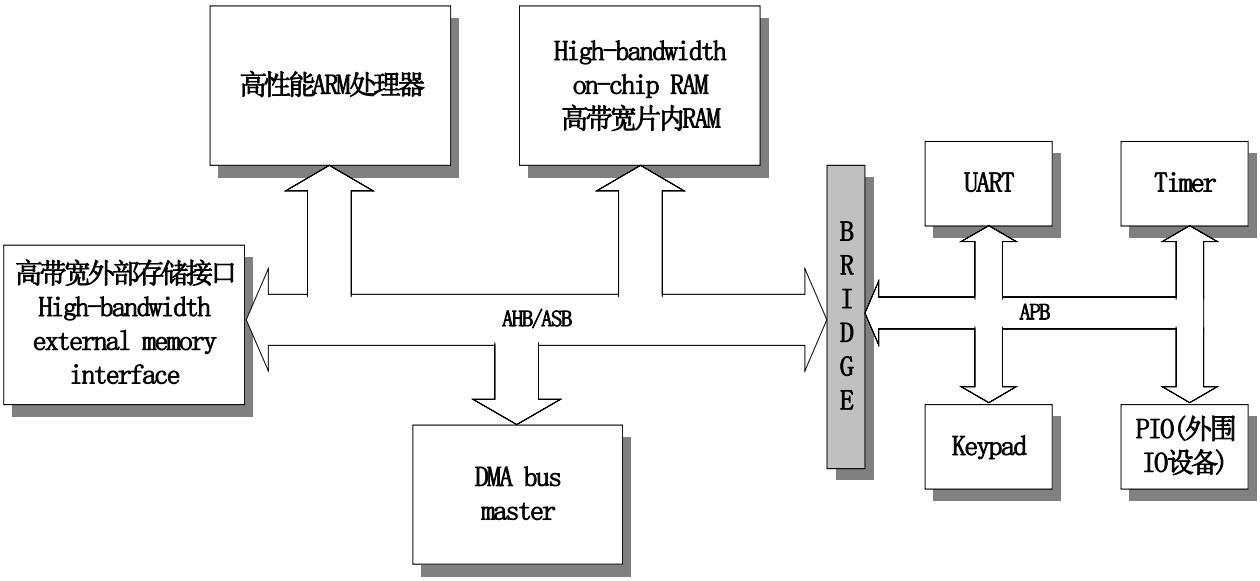

图5-1 一个典型的基于AMBA总线的体系架构

APB 总线是 AHB 或者 ASB 系统总线的扩展，便于外设链接到系统总线上。AHB 和 APB 之间有一条桥来链接。

Test Interface Controller (TIC) 单元用于测试系统总线和 APB 模块。

5.2 APB 规范

5.2.1 APB 状态机

图 5-2 的状态机描述了 APB 外设总线的行为：读或写动作

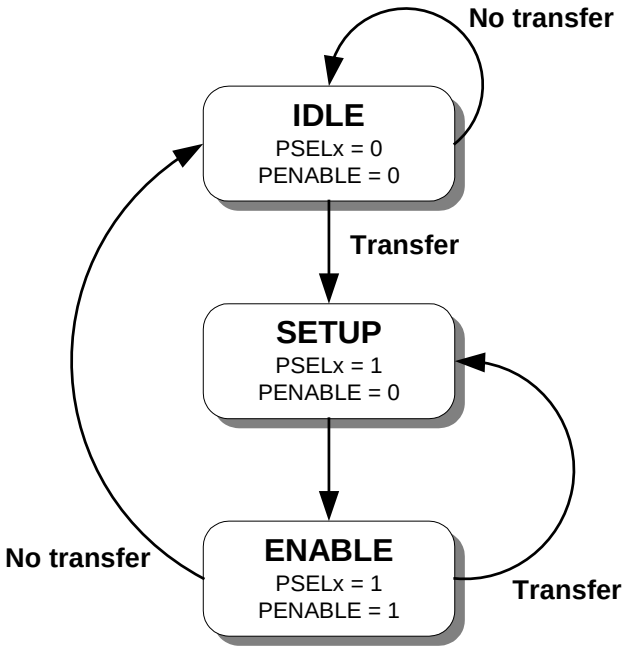

图 5-2 APB 状态机

从状态机看,APB 对每一笔数据的传送，均需花 2 个周期的时间，且 APB 的数据传递不适用在有流水线架构的模块设计中。

表 5-1 APB 状态机跳转表

状态	含义	跳转条件	描述
IDEL	默认状态	No transfer: ENABLE->IDLE IDLE ->IDLE	IDLE 是 APB 状态机的预设状态
SETUP	准备状态	transfer: IDLE -> SETUP ENABLE-> SETUP SETUP-> ENABLE	当一笔数据需要传递时，进入 SETUP. 在此状态中，HADDR[31:0] 地址线会经译码电路产生唯一的 S 使能信号 PSELx。 总线只会在 SETUP 停留一个时钟周期，而在下一个时钟正沿触发时进入 ENABLE 状态。
ENABLE	使能状态	transfer: ENABLE-> SETUP No transfer: ENABLE->IDLE	在该状态下 PENABLE 信号会被置 1。 从 SETUP 转换至 ENABLE 时，数据地址、读写控制和选择信号会保持稳态。该状态也只会持续一个时钟周期，之后如果无传送需求，则在下一个时钟正沿触发时，转回 IDLE； 如果有其它笔数据需要传送，就会转回 SETUP。 在从 ENABLE 转到 SETUP 状态中，地址，写入和选择信号中存在毛刺是可以接受的。

5.2.2 写操作

图 5-3 为 APB 基本的写数据传送时序：

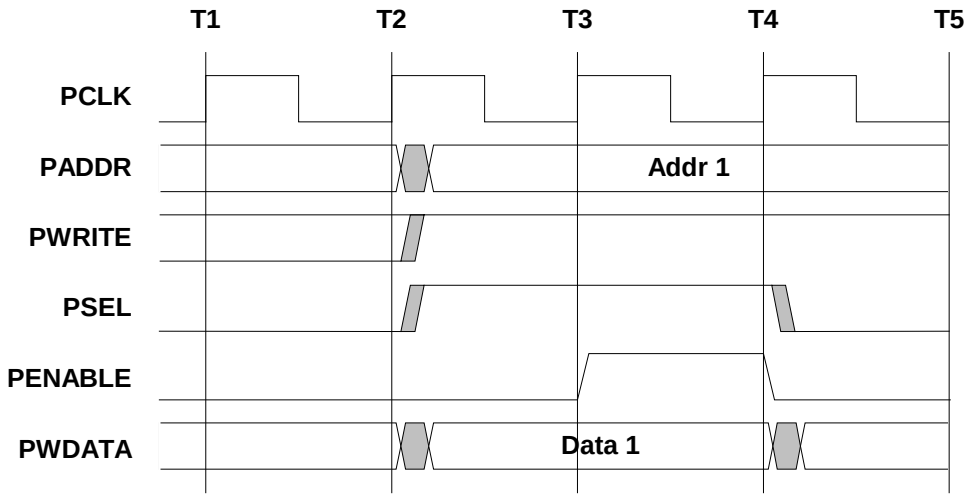

图 5-3 APB 写的时序图

以下简单介绍它的运作情形。在 T1 时，有限状态机进入预设的 IDLE 状态；

在 T2 时，数据地址、读写控制信号和写入的数据会在频率正沿触发时，开始作写的数据传递准备，这个周期也就是刚才所提及的SETUP状态。译码电路在此状态会根据数据地址去译码出所要写入的APBS，此时所对应到 S 的 PSEL 信号将由 0 变 1；

在 T3 时，有限状态机会进入 ENABLE 状态，PENABLE 信号在此状态会被设成 1；

在 T4 频率正沿触发时，PENABLE 信号将由 1 变 0，而 PSEL 信号在若没有其它数据的写入动作时，也将由 1 变 0。为了减少功率的消耗，APB 的数据地址和读写控制信号在下一笔数据传递前，将不会作任何改变。

表 5-2

状态	含义	描述
传送开始	默认状态	在地址，写入数据，写入信号和选择信号全部在正时钟沿改变的时候开始
第一个时钟 T1	即 SETUP 周期	
第二个时钟 T2	即 ENABLE 周期	使能信号 **PENABLE** 置 1。地址，数据和控制信号在该周期全部维持有效。写入传送在这周期末内完成，**PENABLE** 信号也会置 0。选择信号也会立即拉低，除非后续马上有数据传送到同一个外设。为了降低功耗，地址和写入信号在下次传送开始前不会改变。

注意：协议要求使能信号要干净，但是在连续的传送中选择和写入信号可能会有毛刺。

5.2.3 读操作

图 5-4 为 APB 进行读取数据动作的时序图：

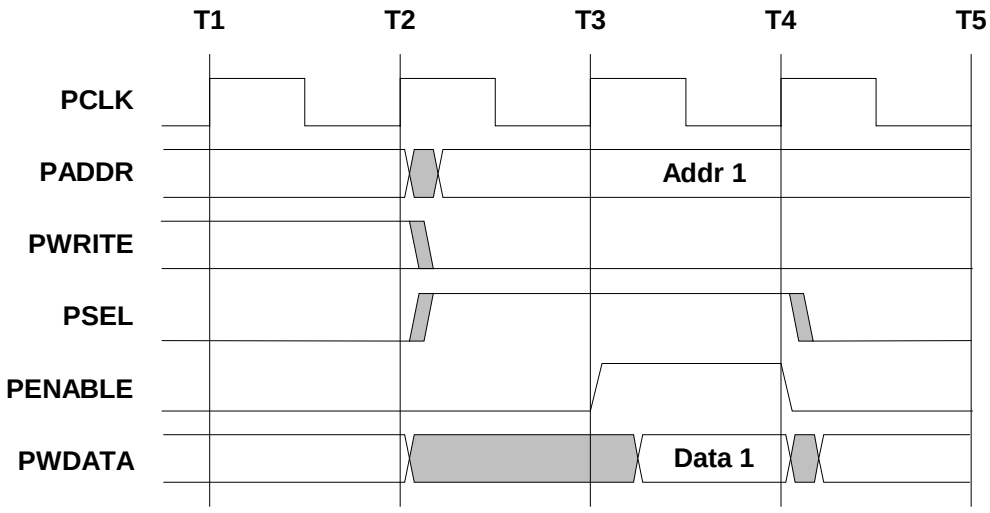

图 5-4　APB 读的时序图

由图中可发现除了写信号是倒过来有效外，APB 读操作时序图和写操作时序图非常相似，在这里我们就不再作详细的解释。

要特别注意的是，在 T3 后，也就是在进入 ENABLE 周期后，APB 从必须要将 **M** 所要读取的**数据**准备好，以便 **M** 可以在 ENABLE 周期末被 T4 正时钟沿触发时正确的将数据读取。

5.3　APB 组件

5.4~5.7 具体描述 APB 组件

5.4 APB 桥

为了让大家有一个较完整的总线系统观念，先介绍连接AHB和APB之间的一桥梁 (APB bridge)。

APB桥为AHB的一个**从设备**，但它在APB中是唯一的**主设备**，而APB中其它低速和低功率消耗的外围皆为APB桥的**从设备**。

5.4.1 接口框图

下图是 APB 桥的信号接口：

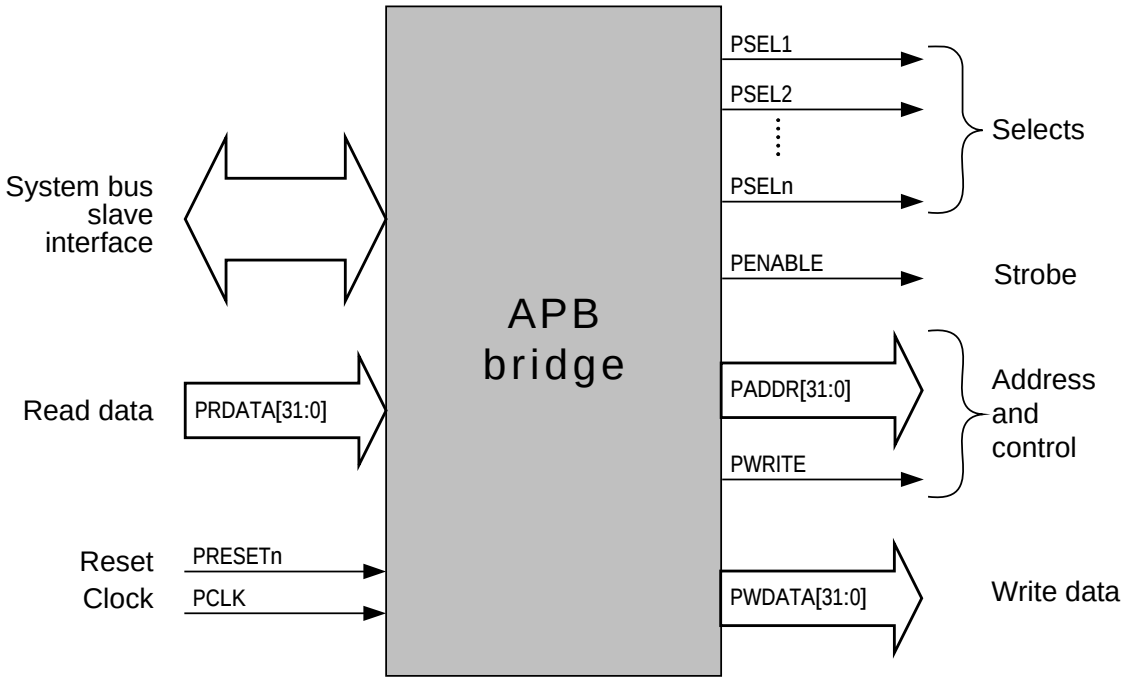

图 5-5 APB bridge 接口图

5.4.2 描述

APB桥将系统总线传送转换成APB方式的传送，它具备一些这些功能：

- 锁存地址，在传送过程中保持地址有效。锁存读写控制信号
- 对锁存的地址进行译码并产生选择信号**PSELx**，在传送过程中只有一个选择信号可以被激活。也就是选择出唯一 一个APB从设备以进行读写动作.
- 写操作时： 负责将AHB送来的数据送上APB总线。
- 读操作时： 负责将APB的数据送上AHB系统总线。
- 产生一时序选通信号PENABLE来作为数据传递时的启动信号

5.4.3 时序

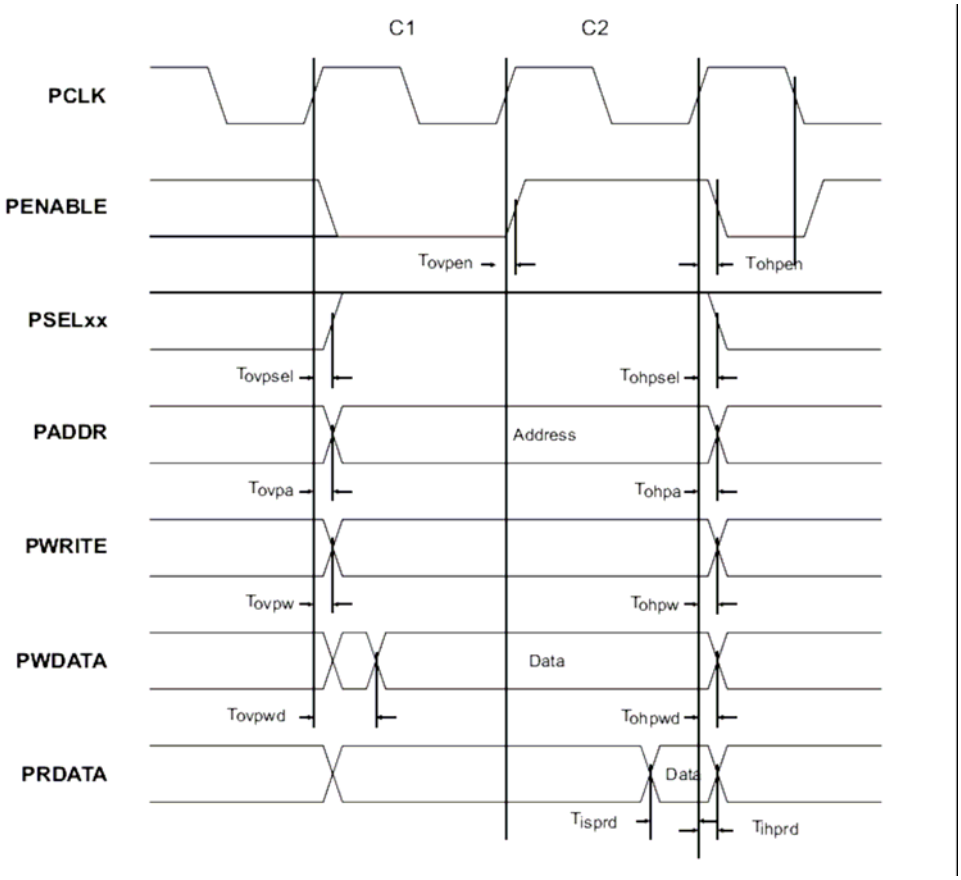

图 5-6 APB 桥时序图

5.4.4 APB 桥（APB 主）的时序参数

- **APB桥输入信号参数：**

Tclkl： **PCLK** LOW time
Tclkh： **PCLK** HIGH time
Tisnres： **PRESETn** de-asserted setup to rising **PCLK**
Tihnres： **PRESETn** de-asserted hold after rising **PCLK**
Tisprd： For read transfers, **PRDATA** setup to rising **PCLK**
Tihprd： For read transfers, **PRDATA** hold after rising **PCLK**

- **APB桥输出信号参数：**

Tovpen： **PENABLE** valid after rising **PCLK**
Tohpen： **PENABLE** hold after rising **PCLK**
Tovpsel： **PSEL** valid after rising **PCLK**
Tohpsel： **PSEL** hold after rising **PCLK**
Tovpa： **PADDR** valid after rising **PCLK**
Tohpa： **PADDR** hold after rising **PCLK**
Tovpw： **PWRITE** valid after rising **PCLK**
Tohpw： **PWRITE** hold after rising **PCLK**
Tovpwd： For write transfers, **PWDATA** valid after rising **PCLK**
Tohpwd： For write transfers, **PWDATA** hold after rising **PCLK**

5.5 APB 从

APB 从的设计是简单但是具有弹性的，可以根据不同的应用进行更改。

5.5.1 接口框图

图 5-7 为 APB 的**从设备**接口图:

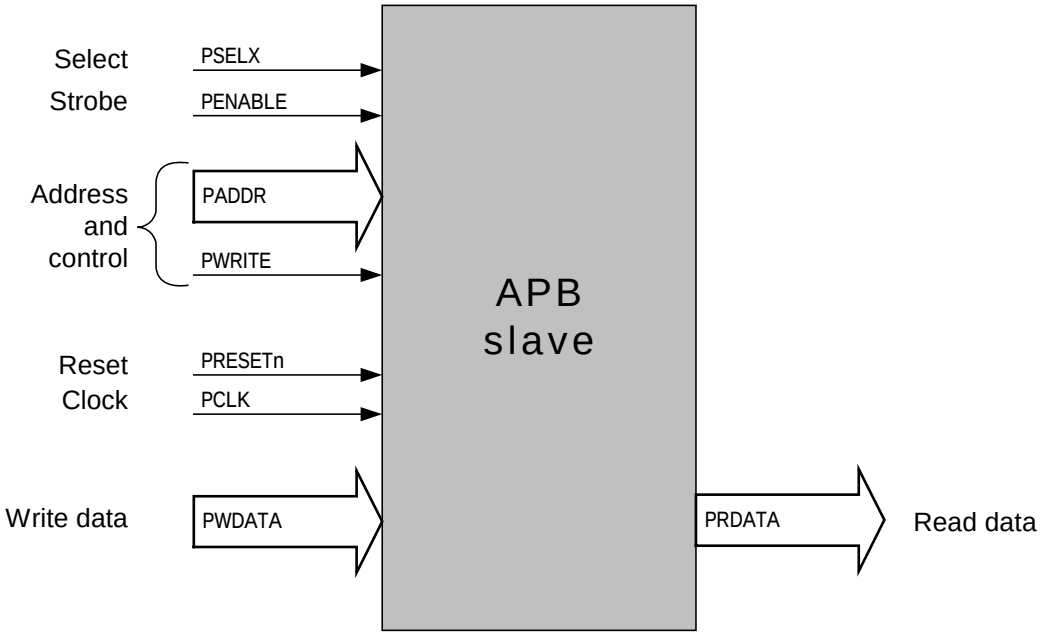

图 5-7 APB **从设备**接口

5.5.2 描述

如前面所提及，APB 总线中除了 APB bridge 为 **M** 外，其它的外围皆为 **S**。因此，APB **从设备**比 AHB **从设备**接口较为简单且非常具弹性:

例如

a. APB 少了仲裁器及复杂的译码电路，APB 进行写操作时，**从设备**可以决定:

- 在 PCLK 上升沿触发， 且 PSEL 为高时锁存数据
- 或在 PENABLE 上升沿，且 PSEL 为高时锁存数据

b. **PSELx**，**PADDR**和**PWRITE**信号的组合可以决定哪个寄存器会被写操作更新。

c. 在读操作的时候，数据可以 **PWRITE** 在＝0，**PSELx** 和 **PENABLE**＝1 的时候被送到总线上，而 **PADDR** 用于决定哪个寄存器会被读。

5.5.3 时序

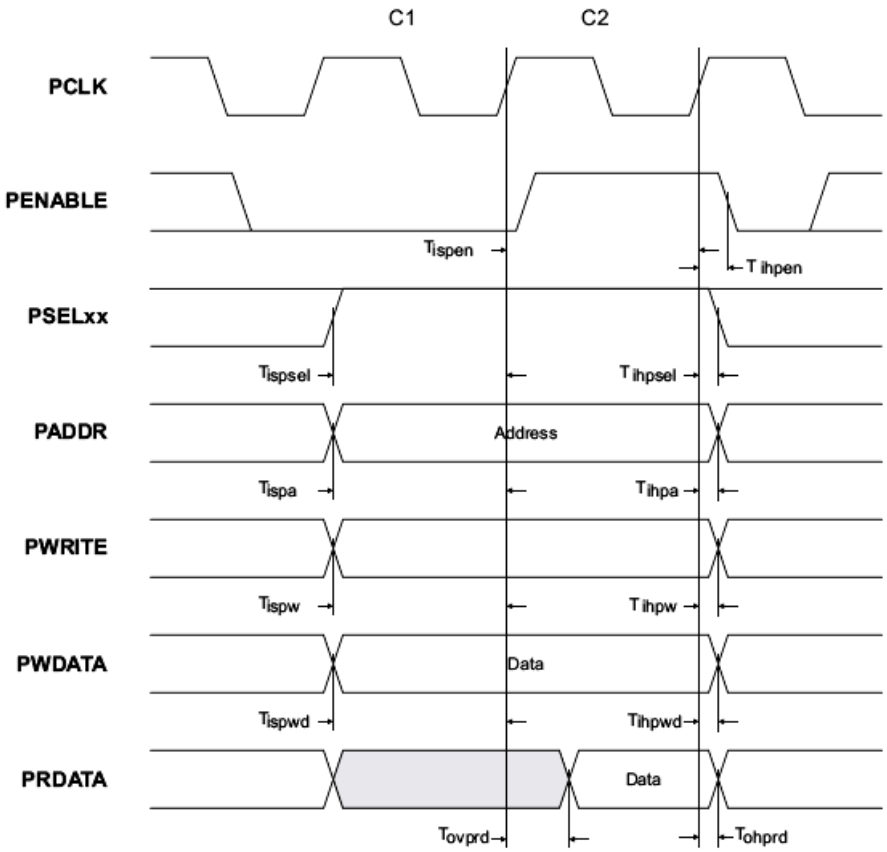

图 5-8 APB 从时序图

5.5.4 APB 从的时序参数

Tclkl： **PCLK** LOW time
Tclkh： **PCLK** HIGH time
Tisnres： **PRESETn** de-asserted setup to rising **PCLK**
Tihnres： **PRESETn** de-asserted hold after falling **PCLK**
Tispen： **PENABLE** setup to rising **PCLK**
Tihpen： **PENABLE** hold after rising **PCLK**
Tispsel： **PSEL** setup to rising **PCLK**
Tihpsel： **PSEL** hold after rising **PCLK**
Tispa： **PADDR** setup to rising **PCLK**
Tihpa： **PADDR** hold after rising **PCLK**
Tispw： **PWRITE** setup to rising **PCLK**
Tihpw： **PWRITE** hold after rising **PCLK**
Tispwd： For write transfers, **PWDATA** setup to rising **PCLK**
Tihpwd： For write transfers, **PWDATA** hold after rising **PCLK**

Tovprd： For read transfers, **PRDATA** valid after rising **PCLK**
Tohprd： For read transfers, **PRDATA** hold after rising **PCLK**

5.6 APB 到 AHB 的接口

5.6.1 读操作

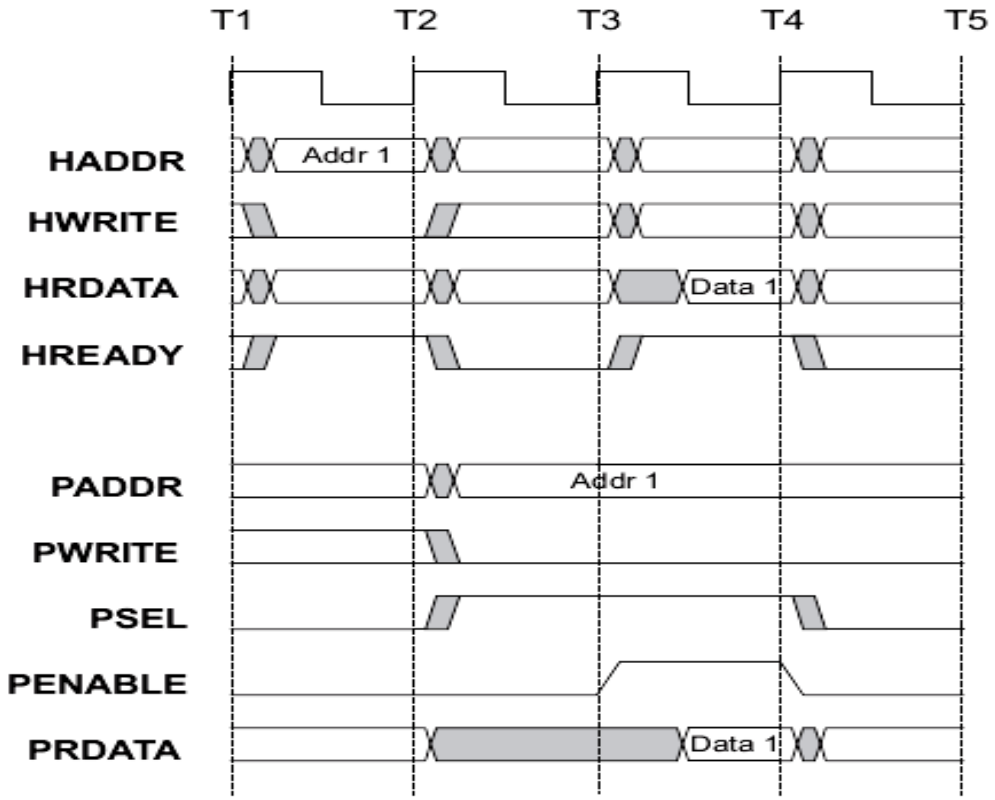

图 5-9　APB 到 AHB 的读操作时序图

在频率很高的情况下，在ENABLE CYCLE中可能数据不能够直接映射到AHB总线，需要在APB桥中在T4的时候打一下锁住，并在T5的时候才被AHB主采样。虽然需要多一个等待周期（一共2个），但是由于频率提升了因此总的性能也提升了。

表 5-3

T1	在 AHB 总线开始传送
T2	地址被 APB 总线采样。如果该传送是针对外设的话，这个地址就会被译码成选择信号发往外设。T2 就是 AHB 的 SETUP CYCLE。
T3	AHB 的 ENALBE CYCLE，**PENABLE** 拉高，数据被读出。
T4	读出的数据直接映射到 AHB 总线上，在上升沿被 AHB 主采样。

下图是批量读操作（非高频），每一组数据都只需要一个等待周期：

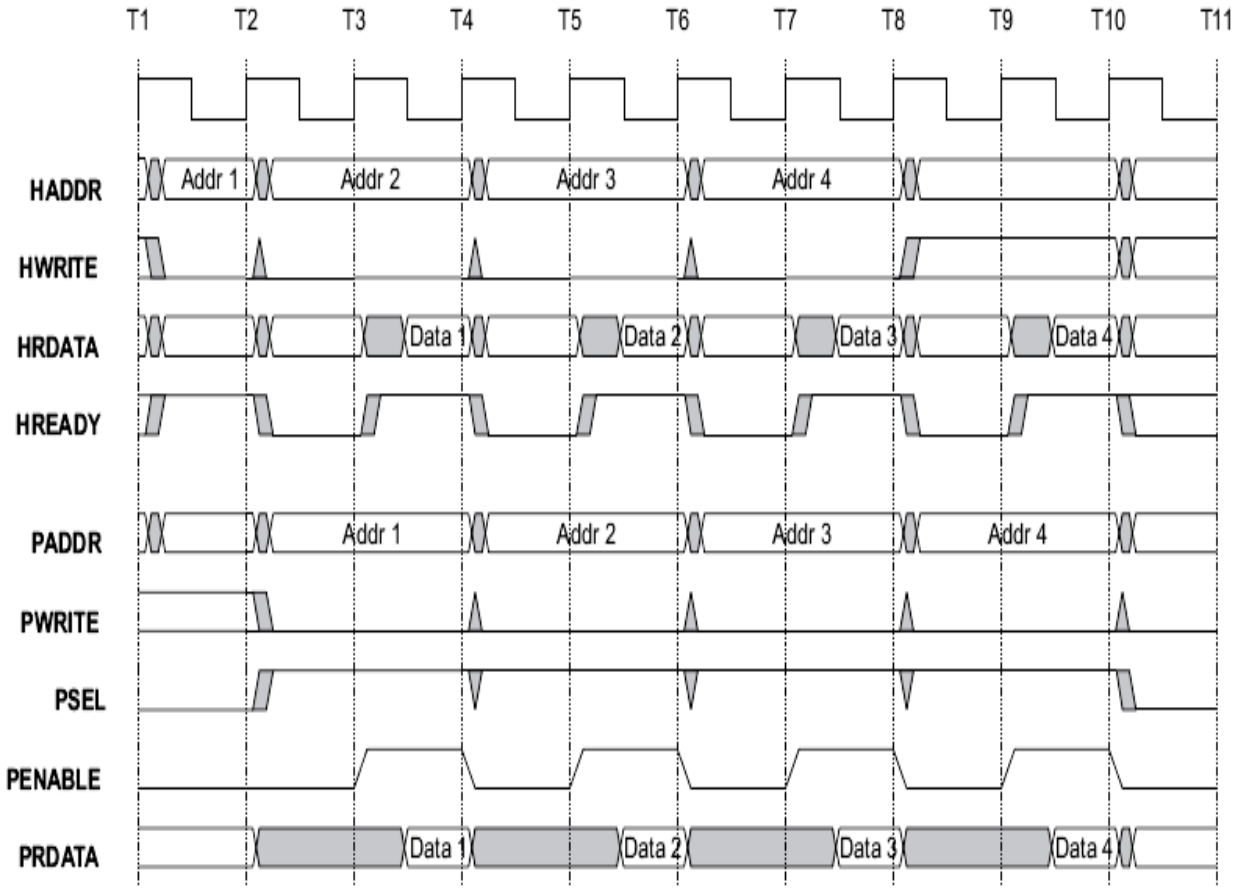

图 5-10　APB 到 AHB 的批量读操作时序图

5.6.2 写操作

以下进一步说明 AHB 和 APB 之间数据传递的情形， 如图 5-11 所示:

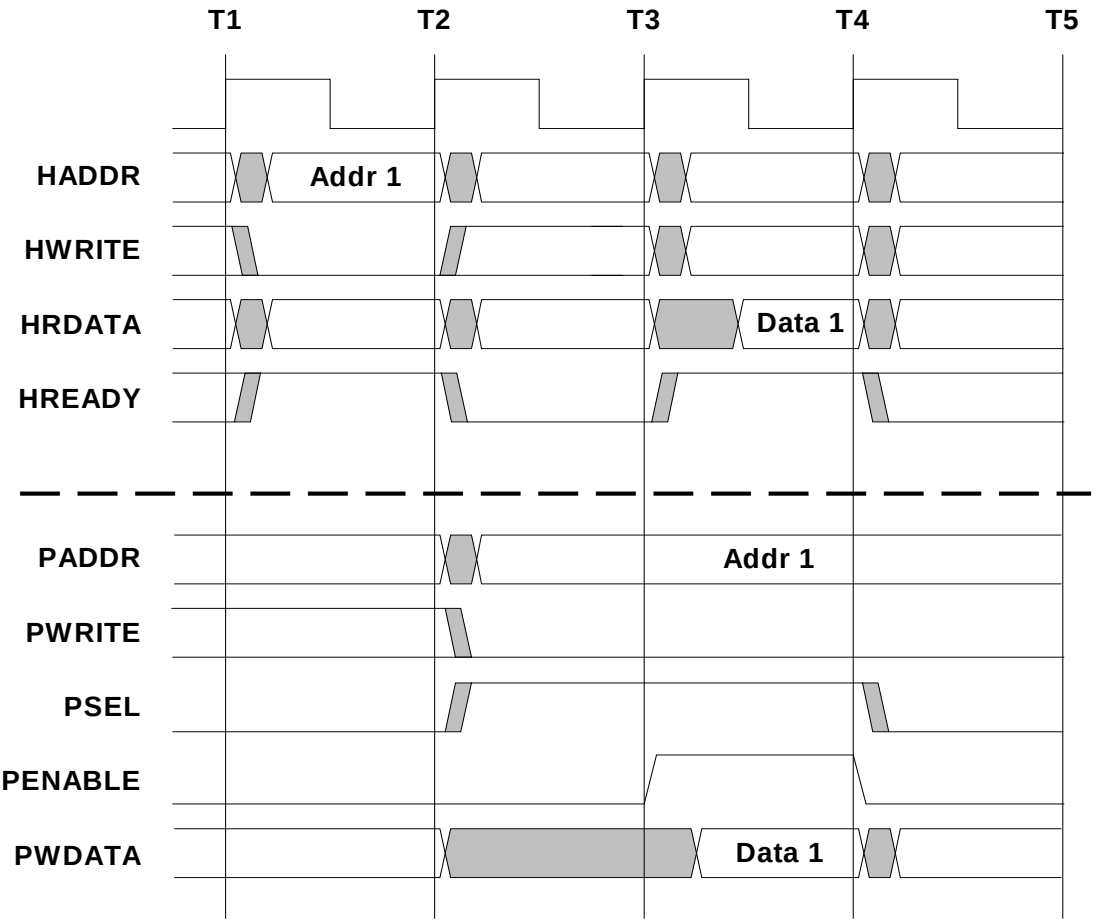

图 5-11 AHB 至 APB 数据写操作时序图

APB总线上的单块数据写操作不需要等待周期。APB桥的责任是对地址和数据进行采样，并在写操作的过程中保持它们的值。

表 5-4

T1	AHB 开始作数据地址和读写控制信号的传递(HADDR 和 HWRITE)
T2	APB bridge 栓取住 AHB 送来的数据地址及读写控制信号，同时进入到 APB 有限状态机的 ENABLE 状态
T3~	其后的读和写动作跟之前所介绍的 APB 读写动作一模一样，在这里我们不再加以详述。

下图是批量写操作的图：

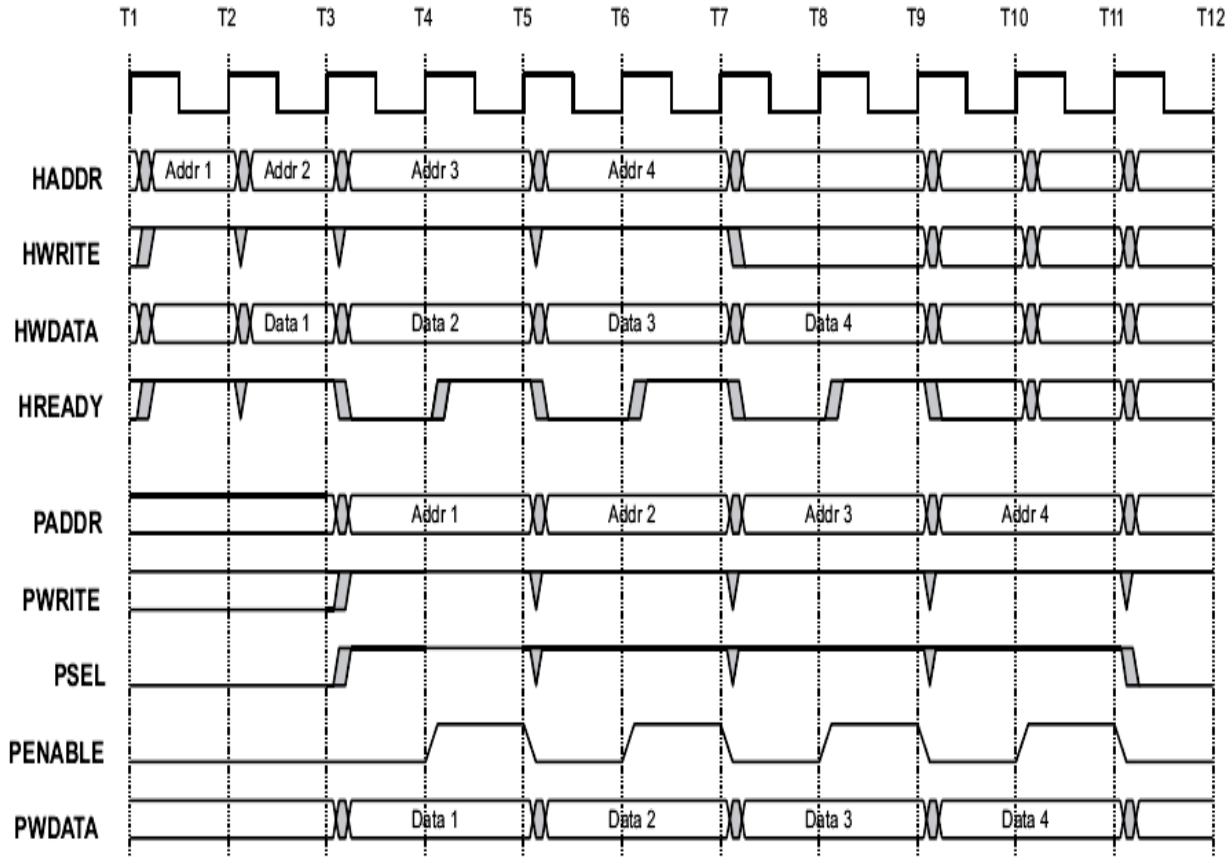

图 5-12　AHB 至 APB 数据批量写操作时序图

当批量写操作的时候，第一块数据不需要等待周期，之后的每一块数据都需要一个等待周期。 APB 桥中需要有 2 个地址寄存器，当处理一个数据块写操作时，可以寄存下一个数据块的地址。

5.6.3 读写交替传送（Back to back）

下图画出了读写交替传送的时序，先是写，再读，再写，再读。

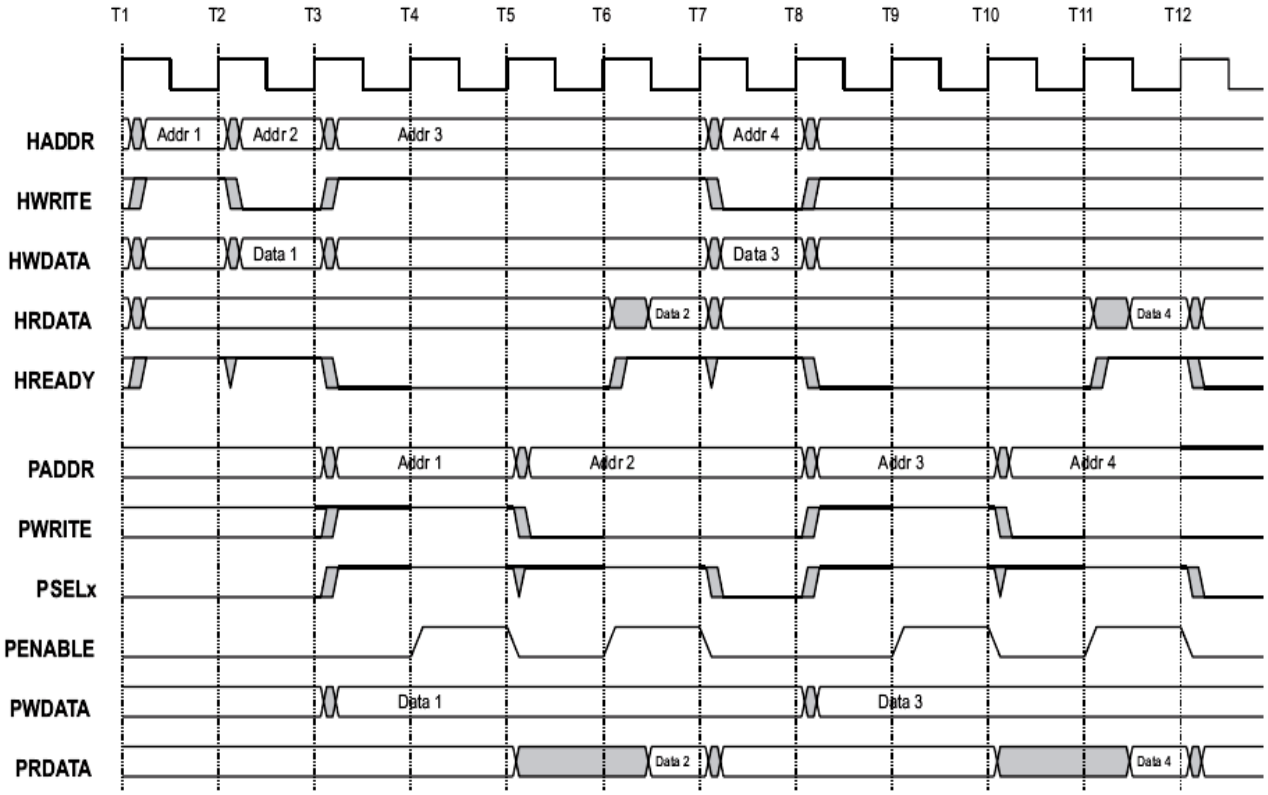

图 5-13　读写交替传送时序图

如果写操作之后跟随着读操作，那么需要 3 个等待周期来完成读操作。通常的情况下，不会有读操作之后紧跟着写操作的发生，因为两者之间 CPU 会进行指令读取。

5.6.4 三态数据线的实现

在 AMBA APB 总线实现中，通常推荐使用分离的读总线和写总线，内部可以使用 MUX 或者 OR 总线来连接不同的 APB 从。如果使用三态总线，读写总线就整合在一条总线上，读写操作绝不可能同事进行。

下图展示出如果使用三态 BUFFER 实现 APB 总线， 并不需要什么特别的需求和考虑。

图中（最后一行）画出了读写数据线整合在一起以后的状态。在批量读的过程中，块与块之间总线需要不断切换驱动方（AHB 或 APB 桥），在批量写的过程中，只有 APB 桥在驱动总线，所以不用切换。

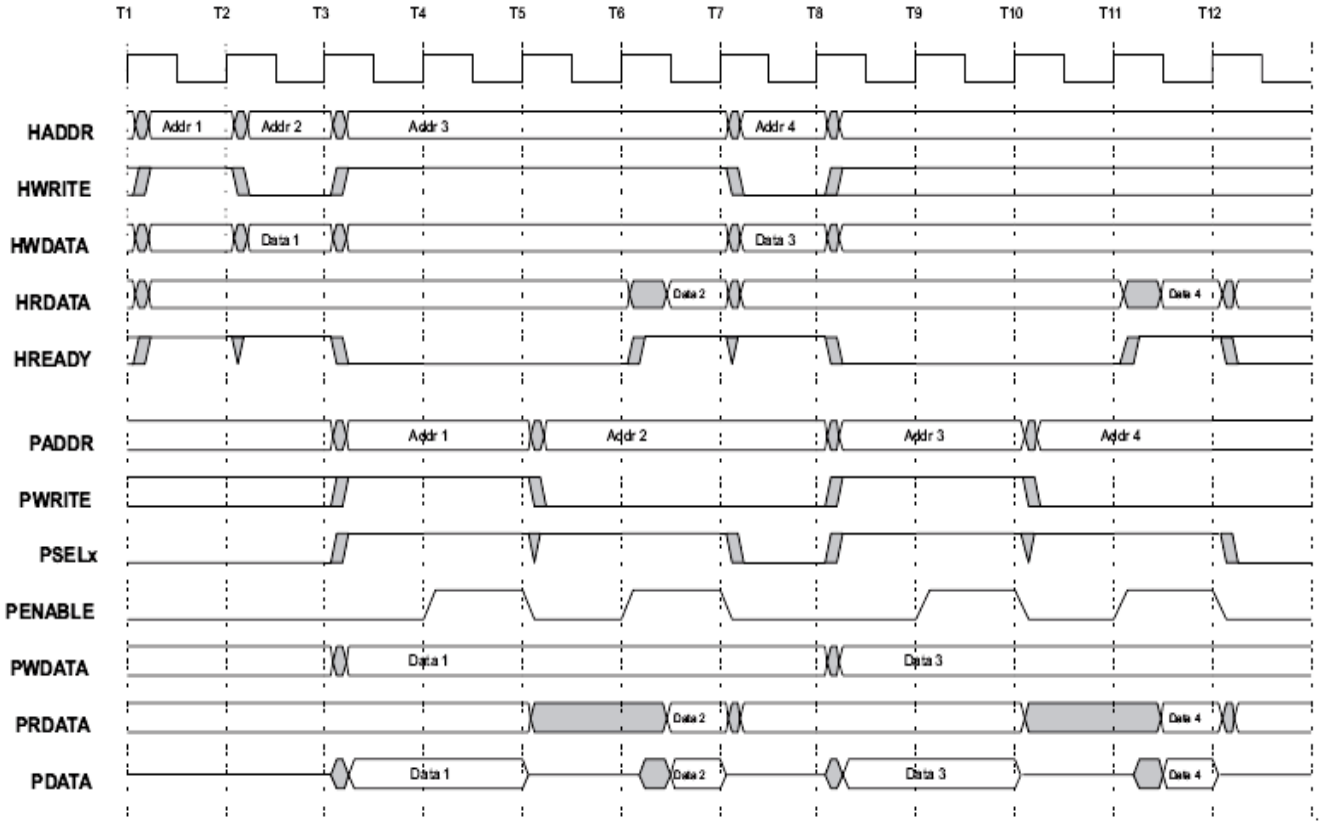

图 5-14 APB 桥时序图

5.7 APB 到 ASB 的接口(忽略)

5.8 D 版本 APB 到 2.0 版本 APB(忽略)

6 测试方法学

This chapter describes the test interface used with AMBA module designs. It contains the following sections:

? About the AMBA test interface on page 6-2
? External interface on page 6-4
? Test vector types on page 6-6
? Test interface controller on page 6-7
? The AHB Test Interface Controller on page 6-12
? Example AMBA AHB test sequences on page 6-17
? The ASB test interface controller on page 6-25
? Example AMBA ASB test sequences on page 6-27.

6.1 AMBA 测试接口

AMBA 提供一个便利的测试方针，它允许系统上的模块独立来作测试，也就是说在测试模式中，每个模块不需要跟系统中其它的模块作数据传送交换。这个特性降低了功能测试时的复杂度，亦即提高了模块的可观察和可控制能力.

图 6-1 为一 AMBA 测试接口示意图。

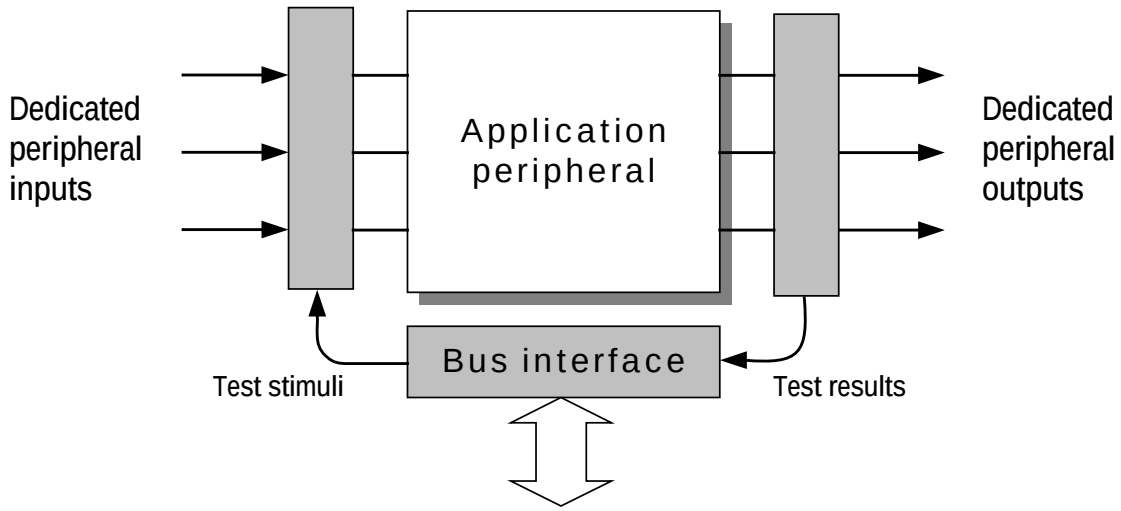

图 6-1 AMBA 测试接口示意图

要达到这个目标，AMBA 系统提供一个测试接口控制器 (TIC) 的总线 **M** 模块。有了这个测试接口控制器后，系统外的测试样品将可顺利的送进去 AMBA 中的内部模块来进行测试工作。这个测试接口控制器使用了 3 个控制**信号**线来作为和模块之间的沟通控制**信号**。除此之外，此控制器也可用来控制测试数据的路径，并提供一个 32 位平行向量接口。为了满足这种的测试方法，测试模式下的这个 32 位总线必须是双向端口。

图 6-2 为一个测试接口控制器和外部总线接口数据互相传递的示意图。

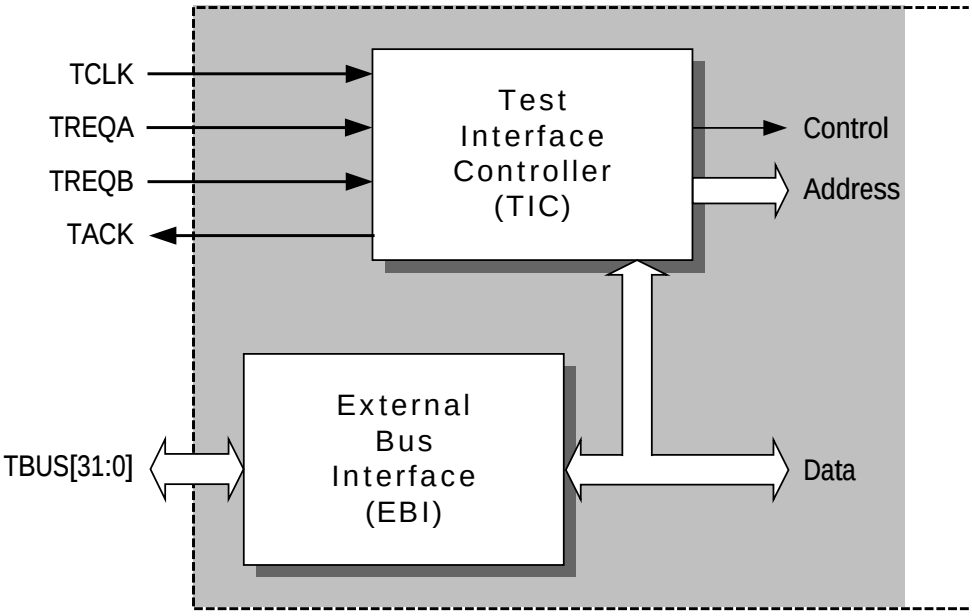

图 6-2 测试接口控制器和外部总线接口

6.2 外部接口

由图 6-2 可知，这个外部总线接口由一测试频率(TCLK)、3 个控制**信号**(TREQA、TREQB 和 TACK)和一个 32 位的测试双向总线所组成。

6.2.1 TREQA

TREQA is the test bus request A input signal and is required as a dedicated device pin.

During normal system operation the TREQA signal is used to request entry into the test mode. This will cause the test bus to become tristated, allowing test vectors to be applied.

During test this signal is used, in combination with TREQB, to indicate the type of test vector that will be applied in the following cycle.

6.2.2 TREQB

TREQB is the test bus request B input signal.

During test this signal is used, in combination with TREQA, to indicate the type of test vector that will be applied in the following cycle.

6.2.3 TACK

表 6-1 和表 6-2 分别为在一般模式和测试模式中，TREQA、TREQB 及 TACK 三个控制**信号**线变化所对应的系统操作情形。

表 6-1 一般模式三控制线的操作情形

TREQA	TREQB	TACK	描述
0	0	0	一般模式
1	0	0	要求系统进入测试模式

0	1	0	保留
—	—	1	进入测试模式

表 6-2 测试模式三控制线的操作情形

TREQA	TREQB	TACK	描述
—	—	0	未完成的存取动作
1	1	1	地址、控制及转向向量
1	0	1	写向量
0	1	1	读向量
0	0	1	离开测试模式

6.2.4 Test clock

TCLK is the test clock input signal.

In test mode, the internal bus clock is driven from the external TCLK source. This pin may be the normal clock oscillator source input or a port replacement signal. The system bus clock must not glitch when switching between normal and test mode. On entry into test mode the TIC indicates that it has switched to the test clock input by asserting the TACK signal.

6.2.5 Test bus

TBUS[31:0] is the 32-bit bidirectional test port.

The test bus is used as an input to apply address, control and write vectors. For read vectors the test bus is used as a device output. The test interface protocol ensures that a turnaround period is always provided when changing the direction of the test bus.

6.3 测试向量类型

在测试模式下支持几种不同的向量传递：

表 6-3

1	address vector	
2	write vector	
3	read vector	
4	control vector	
5	turnaround vector.	

第一种为地址向量(Address Vector)，在进行一个读或写动作前，地址向量必须先被传送。如表四所示，在测试模式中 TREQA 和 TREQB 同时为 1 时，代表下一个周期所传送的数据为地址向量。因此，在下一个周期时，TBUS[31:0]所传送的数据即为地址向量，而 TREQA 和 TREQB 在此时也会被设定为下一次传送向量型态所需要的值；

第二种传递向量为控制向量(Control Vector)，这种向量通常会出现在连续多笔的地址向量之后，控制

向量可用来更新测试接口控制器中的控制信号。举例来说，在一笔地址向量传递完成后，TREQA 和 TREQB 会维持 1，使得控制向量可以在下一个周期来进行传送，而在下一周期时，TBUS[31:0]传送的数据即为控制向量。相同的，TREQA 和 TREQB 这时候也会被设定为下一次传送向量型态所需要的值。

第三种传递向量为写的测试向量(Write Vector)，这种向量可能发生在下列几种向量之后，如单一的地址向量、连续多笔的地址向量或控制向量、连续多笔写的测试向量、单笔或多笔读向量(Read Vector)后的转向向量(Turnaround Vector)。读的测试向量和写的测试向量很相似，它可能发生在下列几种向量之后，如单一的地址向量、连续多笔的地址向量或控制向量，连续多笔读的测试向量、单笔或多笔写向量后的转向向量。

另一个向量为多笔传送模式向量(Burst Vector)，这种向量可由多笔的读和写的测试向量所组成，此类型的向量传递比单笔读或写的向量传递快。

最后一传递向量为转向向量，在测试模式下，允许从一个传送读的向量转向变为一个写的向量，或者由一个写的向量转向成一读的向量，但在进行转向之前必须要插入一个转向向量。

6.4 测试接口控制器

The Test Interface Controller (TIC) is a bus **M** that accepts test vectors from the external test bus, TBUS[31:0], and initiates bus transfers. The TIC latches address vectors and, when required, increments the address to allow read and write bursts of test vectors.

6.4.1 测试操作参数

The default TIC bus **M** operation when entering test mode is:

? 32-bit transfer width

? privileged system access.

This is sufficient for testing many embedded system designs and minimizes the on-chip test support logic. In the case of systems that require the above control signals to be dynamically changed, a control vector mechanism is used to update the control signals within the TIC.

Bit 0 of the control vector is used to indicate if the control vector is valid. Thus, if a control vector is applied with bit 0 LOW, the vector will be ignored and will not update the control information. This mechanism allows address vectors which have bit 0 LOW to be applied for many cycles without updating the control information.

6.4.2 递增寻址

In order to support burst accesses using the test interface the TIC may support incrementing of the bus address. The number of address bits that are incremented is dependent on the maximum burst access length that is required via the test interface. This is system-dependent but a typical implementation would use an 8-bit address incrementer, allowing burst access up to 1kB boundaries using word transfers.

The control vector also provides a mechanism to enable and disable the address incrementer within the TIC. This allows burst accesses to incremental addresses, as

would be used for testing internal RAM. Alternatively, the address increment can be disabled, such that successive accesses of a burst occur to the same address, as would be required to continually read from a single peripheral register.
If the transfer size is changed dynamically then any address incrementer support for burst-mode accesses must be able to support increment by byte, halfword and word offsets, so adaptive address incrementer logic is required.
The address incrementer is disabled by default and must be enabled using a control vector prior to use.

6.4.3 进入测试模式

In normal operating mode TREQA will be LOW, indicating that test access is not required and the test bus will be used as required for normal operation, which will usually be part of the external bus interface. Entering test mode allows test vectors to be applied externally that will cause transfers on the internal bus.
The following sequence is required in order to enter test mode:
1. TREQA is asserted to request test bus access.
2. Test mode is entered when the TIC has been granted the internal bus and this is indicated by the assertion of the TACK signal.
3. At this point TCLK will become the source of the internal clock signal.
4. When test mode has been entered TREQB is asserted to initiate an address vector.
The TIC will not perform any internal transfers until a valid address vector has been applied.
A synchronous tester would not be expected to poll TACK for the bus. Normally the TREQA signal would be asserted for a minimum number of cycles to guarantee to gain access to the bus (completion of the longest wait-state peripheral access or the maximum number of cycles for all bus masters to have completed their current instruction).

6.4.4 地址向量

An address vector must be applied before any read or write operations can occur. The following sequence is required in order to apply an address vector:
1. TREQA and TREQB are both asserted HIGH indicating an address vector next cycle.
2. In the next cycle the address is applied to TBUS[31:0], while TREQA and TREQB change to reflect the type of test vector that will follow.
In some high-speed systems it may be necessary to apply more than one address vector in succession, to allow sufficient time for the address to propagate from the external test bus through to the internal address bus. In such a case the TIC can negate TACK for the first cycle of the address vector, forcing a second cycle of address vector to be applied.

6.4.5 控制向量

A control vector is always the last in a sequence of address vectors and is used to update control information within the TIC. The sequence is as follows:

1. TREQA and TREQB are asserted HIGH indicating an address vector next cycle.
2. In the next cycle the address is applied to TBUS[31:0]. TREQA and TREQB both remain HIGH as the control vector will occur in the following cycle.
3. In the next cycle the control information is applied to TBUS[31:0], while TREQA and TREQB change to reflect the type of test vector that will follow.
4. Finally the transfer occurs on the internal bus.

It is possible to apply an invalid control vector, by setting bit 0 of the control vector LOW. This will not change the control information within the TIC.

6.4.6 写操作向量

Once test mode has successfully been entered, read and write operations may be performed through the test interface. In order to perform a write operation internally it is necessary to supply an address followed by the write data.

The address used for the write transfer will depend on the preceding vectors and a write vector may occur after any of the following:

? a single address vector

? an address/control vector sequence

? another write test vector, forming a burst of writes

? a turnaround vector after a single read or burst of reads.

When an internal bus transfer is extended by the insertion of wait states this is indicated externally by the TACK signal going LOW. During the waited condition the TREQA and TREQB should change to indicate the vector type that will follow when the current vector has completed. However, it is important to note that in the case of a write vector the data should continue to be applied to TBUS[31:0].

6.4.7 读操作向量

In a similar manner to write test vectors, read test vectors may follow a number of different vectors, as listed below, and the address used for the transfer will depend on the preceding vectors:

? a single address vector

? an address/control vector sequence

? another read test vector, forming a burst of reads

? a single write or burst of writes.

A read, or burst of reads, must always be followed by a turnaround vector to prevent bus clash on the external TBUS signals. As for a write vector, if the external transfer is extended then this is indicated externally by the TACK signal going LOW. The read data should not be sampled externally until the internal transfer has completed.

6.4.8 批量向量

Multiple write vectors or read vectors may be joined together to form bursts of vectors. This enables test vectors to be applied at a much faster rate by removing the need for an address vector to be associated with each read or write vector.
Burst transfers may use either incrementing addresses or static addresses, depending on whether or not the TIC contains an address incrementer which is enabled. With no address incrementer the TIC will perform non-sequential transfers to a constant address. If the TIC does contain an enabled address incrementer then the address used for each successive transfer will be incremented by the appropriate amount, which is dictated by the transfer size.

6.4.9 改变操作方向

It is possible to change the transfer direction of a burst, from read to write or write to read.
If changing from read to write it is still necessary to insert a turnaround vector. This will not load a new address but will internally cause a new burst to be started allowing internal slaves to observe that the direction of the burst has altered.

6.4.10 退出测试模式

Test mode is exited using the following sequence:
1. Apply a single cycle of address vector, which ensures any internal transfers have been completed.
2. TREQA and TREQB are both driven LOW to indicate that test mode is to be exited.
3. When the test interface has been configured for normal system operation TACK will go LOW to indicate that test mode has been exited.
It is important that test mode can be entered and exited cleanly so that diagnostic testing may be performed during system operation.

6.5 AHB 测试接口控制器

The following state diagram illustrates the operation of the TIC.

The following points describe the TIC state diagram operation:

? At reset the TIC is in the IDLE state and will not be requesting use of the AHB. When in the IDLE state TACK is driven LOW to indicate that the test interface cannot be used.
? The TACK signal is used to control all transactions around the state machine,

except for the transition from IDLE to START. In all other cases the state machine remains in the same state if the TACK signal is low.

? The TREQA signal is used to move from the IDLE state to the START state. This has been changed from the previous specification, which required TREQA to be high and TREQB to be low, and has the advantage that it is possible to use just TREQA to move from normal operation into test mode.

? In some system implementations it will be necessary to switch from an internal clock source to an external clock TCLK which is used during test mode. When TREQA first goes high this can be used as an indication that the clock source should be changed and a return signal that indicates when the clock switch has occurred successfully can be used to prevent the move into the START state until the test clock is in use.

? If clock switching is being used then it is possible that TREQA is asynchronous to the on-chip clock before test mode is entered and therefore a synchronizer is used to generate a synchronized version of TREQA to control the movement from the IDLE state to the START state.

? The START state is used to ensure that the first vector applied is an address vector to prevent read and write vectors occurring before the address has been initialized. The START state is only exited when TREQA/B indicate an address vector and the following state is ADDRVEC.

? In the ADDRVEC state the TIC registers the address on the TBUS. The ADDRVEC state is used for both address and control vectors, so additional logic is required to determine whether the value on TBUS should be considered as an address or as a control vector. If the previous cycle was an address vector and the following cycle (as indicated by TREQA/B) is not an address vector then the current cycle is a control vector.

? It is possible to stay in the ADDRVEC state for a number of cycles, but usually an address vector will be followed by either read or write transfers.

? If a write transfer is being performed the TIC moves into the WRITEVEC state at the same time that it initiates the transfer on the bus and multiple write transfers can be performed by remaining in the WRITEVEC state. Usually the WRITEVEC will be followed by an address vector, however it is also possible to move directly to read transfer by moving to the READVEC state.

? When a read, or a burst of reads is performed the TIC enters the READVEC state. This state indicates that the TIC is starting a read transfer on the bus and it is not until the following cycle that the read data will appear. When the READVEC state is first entered the TBUS will be tristate, but will become driven for further cycles in the READVEC state.

? All read vectors must be followed by two turnaround vectors. For the first of these cycles the TIC will move into the LASTREAD state, during which the last read of the transfer will complete and will be driven out on to the external TBUS. During the LASTREAD state no internal transfers will be started and the

TIC will perform IDLE transfers on the bus.

? Following the LASTREAD state the TIC moves into the TURNAROUND state, during which time the external TBUS will be tristate. The TURNAROUND state will usually be followed by an address vector, but it is also possible to go immediately to a write vector or another read.

? The usual method to exit from test is to return to the ADDRVEC state and then set TREQA/TREQB both LOW to return to IDLE and effectively exit from test. In fact, at any point the test mode can be exited by setting both TREQA and TREQB LOW and eventually this will cause the TIC to exit from test.

Note

When applying TIC vectors it is theoretically possible to assert the HLOCK output and then exit from the test. If this happens and then the TIC is granted the bus under normal operation it will effectively lock up the bus. No protection is provided within the TIC to prevent this occurrence.

6.5.1 Control vector

A control vector is included within the TIC to determine the types of transfer it can perform. The control vector is used to set the values of HSIZE, HPROT and HLOCK. The default TIC bus **M** operation when entering test mode is:

? 32-bit transfer width - HSIZE[1:0] signifies word transfer

? privileged system access - HPROT[3:0] signifies privileged data access, uncacheable and unbufferable.

Bit 0 of the control vector is used to indicate if the control vector is valid. Thus, if a control vector is applied with bit 0 LOW, the vector will be ignored and will not update the control information. This mechanism allows address vectors which have bit 0 LOW to be applied for many cycles without updating the control information.

Although the default settings will be sufficient for testing many embedded system designs, the control vector can be used both to change the control signals of the transfer and also to determine whether the TIC should generate fixed addresses or incrementing addresses.

Table 6-3 defines the bit positions of the control vector. The control vector bit definitions are designed to be backwards compatible with earlier versions of the TIC and therefore not all of the control bits are in obvious positions.

There is no mechanism to control the types of burst that the TIC can perform and only incrementing bursts of an undefined length are supported. The TIC only supports 8-bit, 16-bit and 32-bit transfers and therefore HSIZE[2] cannot be altered and will always be low.

In order to support burst accesses using the test interface the Test Interface Controller may support incrementing of the bus address. The TIC increments 8 address bits and the address range that can be covered by this incrementer is dependent on the size of the transfers being performed.

The control vector provides a mechanism to enable and disable the address incrementer within the TIC. This allows burst accesses to incremental addresses, as would be used for testing internal RAM. Alternatively, the address increment can be disabled such that successive accesses of a burst occur to the same address, as would be required to continually read from a single peripheral register.

If HSIZE[1:0] is changed dynamically then any address incrementer support for burstmode accesses must be able to support increment by byte, halfword and word offsets, so adaptive address incrementer logic is required.

The address incrementer is disabled by default and must be enabled using a control vector prior to use.

Note

The control vector is primarily used to change signals which have the same timing as the address bus. However the control vector also allows the lock signal to be changed, which is actually required before the locked transfer commences. If the HLOCK signal is used during testing it should be set a transfer before it is required. This difference in timing on the HLOCK signal may in some cases cause an additional transfer to be locked both before and after the sequence that should in fact be locked.

6.6 AHB 测试程序举例

6.6.1 Entering test mode

图 6- 为 AMBA 系统由一般模式进入测试模式时的处理程序。以此图为例，假设系统在开始时处于一般模式中，当测试控制器送出一个要求信号来要求系统进入测试模式后(TREQA 由 0 变 1)，系统会响应以一允许信号(TACK 由 0 变 1)来进入测试模式。由图中可知在测试模式中，当 TREQB 由 0 变 1 时，系统会进行地址向量的传递。

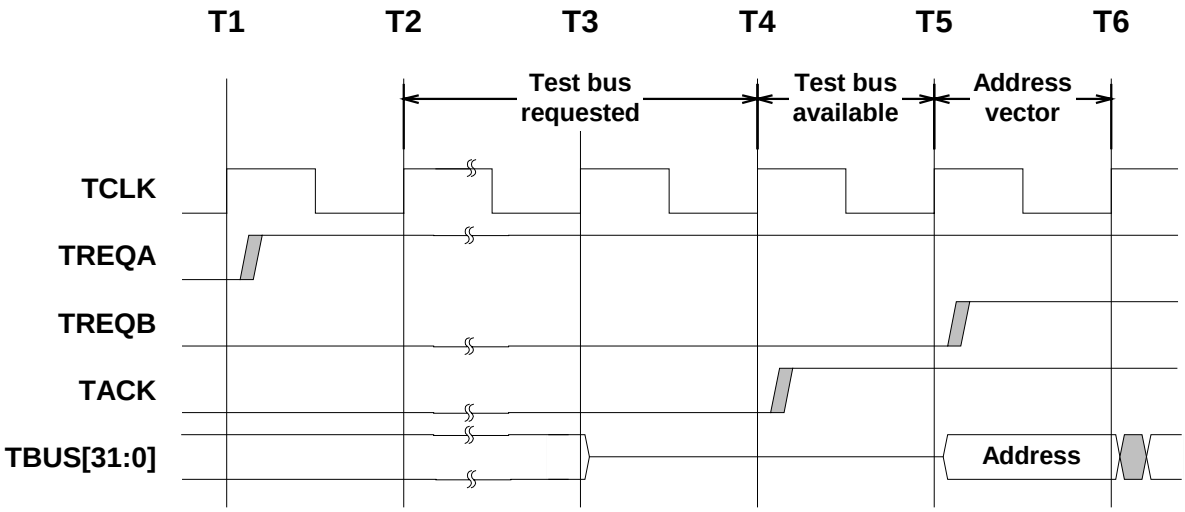

图 6-3　启动测试模式的程序时序图

In normal operating mode TREQA will be LOW, indicating that test access is not required and the test bus will be used as required for normal operation, which will usually be part of the external bus interface. Entering test mode allows test vectors to be applied externally that will cause transfers on the internal bus.
The following sequence, as illustrated in Figure 6-4, is required in order to enter test mode:
1. TREQA is asserted to request test bus access.
2. Test mode is entered when the TIC has been granted the internal bus and this is indicated by the assertion of the TACK signal.
3. At this point TCLK will become the source of the internal HCLK signal.
4. When test mode has been entered TREQB is asserted to initiate an address vector.
5. The TIC will not perform any internal transfers until a valid address vector has been applied.

A synchronous tester is not expected to poll TACK for the bus.
Normally the TREQA signal is asserted for a minimum number of cycles to guarantee access to the bus (completion of the longest wait-state peripheral access or the maximum number of cycles for all bus masters to have completed their current instruction).

6.6.2 写操作

Figure 6-5 shows the sequence of events when applying a set of write test vectors. Initially an address vector is applied and this is followed by a write test vector.

The following points apply when writing test vectors:
? The TREQA and TREQB signals are pipelined and are used to indicate what type of vector will be applied in the following cycle. Figure 6-5 shows an example of a number of write transfers being performed.
? The TIC samples the address and TREQA/B signals at time T3. Following this it can initiate the appropriate transfer on the AHB.
? In the following cycle the write data is driven on to the TBUS and it is then sampled on the following clock edge, T4, and driven on to the internal bus.
? If the internal transfer is not able to complete then the TACK signal is driven low and this indicates that the external test vector must be applied for another cycle.

6.6.3 读操作

Read transfers are more complex because they require the TBUS to be driven in the opposite direction and therefore additional cycles are required to prevent bus clash when changing between different drivers of TBUS. Figure 6-6 shows a typical test sequence for reads.

The following points apply when reading test vectors:
? The TREQA and TREQB signals are used in the same way as for a write transfer. Initially, TREQA/B are used to apply an address vector, in the following cycle they are used to indicate that a read transfer is required. For the first cycle of a read the TBUS must be tristate, which ensures that the external equipment driving TBUS has an entire cycle to tristate its buffers before the TIC will enable the on-chip buffers to drive out the read data.
? At the end of a burst of reads it is also necessary to allow time for bus turnaround. In this case the TIC must turn off the internal buffers and an entire cycle is allowed before the external test equipment starts to drive.
? The end of a burst of reads is indicated by both TREQA and TREQB being HIGH, as for an address vector. In fact they must indicate address vector for two cycles, which allows for both the turnaround cycle at the start of the burst and also the turnaround cycle at the end of the burst.

6.6.4 控制向量

The operation of the TIC may be modified by the use of a control vector. Whenever more than one address vector is applied in succession then the last vector is considered to be a control vector and is not latched as the address. Bit 0 of the control vector is used to determine whether or not the control vector should be considered valid, which allows multiple address vectors to be applied without changing the control information,

At time T4 the TIC can determine that the TBUS contains a control vector. This is because the previous cycle was an address vector and TREQA/B are indicating that the following cycle is either a read or a write and therefore the current cycle must be a control vector.

6.6.5 批量向量

The examples of read and write transfers in Figure 6-5 on page 6-19 and Figure 6-6 on page 6-20 also show how additional transfers can be used to form burst transfers on the bus. The TIC has limited capabilities for burst transfers and can only perform undefined-length incrementing bursts.
The TIC contains an 8-bit incrementer and if an attempt is made to perform a burst which crosses the incrementer boundary then the address will wrap and the TIC will signal the transfer as NONSEQUENTIAL. The exact boundary at which this will occur is dependent on the size of the transfer. For word transfers the incrementer will overflow at 1kB boundaries, for halfword transfers it will overflow at 512-byte boundaries and for byte transfers the overflow will occur at 256-byte boundaries.

6.6.6 交替读写 Read-to-write and write-to-read

It is possible to switch between read transfers and write transfers without applying a new address vector. Usually this would be done with the address incrementer disabled, so that both the read transfers and the write transfers would be to the same address. It is also possible to do this with the incrementer enabled if the test circumstances require it.

When moving from a read transfer to a write transfer it is also necessary to allow the two cycles for bus handover and therefore TREQA and TREQB should signal address vector for two cycles after the read. This will not cause the address to be changed unless it is followed by a third address vector. Figure 6-8 illustrates the sequence of events.

6.6.7 退出测试模式

在完成测试后，系统必须跳出测试模式回到一般模式，以下描述从测试模式跳回一般模式的程序。

首先必须传送一个地址向量来确保模块内部传送的数据已经完成，

再来 TREQA 和 TREQB 这两个信号将被设为 0，指示当前已经离开测试模式，TACK 也会被设成 0 来表示当前为一般模式。

6.7 ASB 测试接口控制器(忽略)

6.8 ASB 测试程序举例(忽略)

7 几个比较

7.1 不同总线之比较

表 7-1 不同总线之比较

	APB	AHB	ASB	PLI	SFR
支持频宽(位)	32	2^n (n=3~10)	8, 16, 32		
最高频宽(每周期)	4 字节	128 字节	4 字节		
适用模块特性	低速低功率	高速高效能	高速高效能		
时序准则	时序图	时序图	时序图		
频率同步	是(正沿)	是(正沿)	是(负缘)		
数据总线实现	多任务器	多任务器	三态电路		
中断	否	否	否		
分离式数据传送	否	是	否		
仲裁器	否	是	是		
连续笔数据传送	否	是	是		
主	AHB2APB 桥	CPU,DMA 主等			
从	PIO/ UART 等	MEM,AHB2APB 桥等			

7.2 AMBA2.0 与 AMBA3.0(AXI)

表 7-2 AMBA2.0 与 AMBA3.0(AXI)比较

	AMBA2.0	**AMBA3.0-AXI**
单双工	单工	双工
读写同时	同时只能写或只能读	同时读写
性能	1	2
发布时间	1999	
其它		

7.3 递增与回绕(参 3.6)

7.4 重发与分段发送(参 3.9.5)

7.5