AMBA AXI 协议

V1.0

规范

1. 介绍

本章讲述了 AXI 协议的架构，以及协议定义的基本事物。包含以下章节：

- *关于 AXI 协议*
- *架构*
- *基本事物*
- *附加特征*

1.1 关于 AXI 协议

AMBA AXI 协议的目标是高性能、高频的系统设计，其包含了包含了若干特性，使其能够适应一个高速的亚微互联。

最新的 AMBA 接口的目标是：

- 适合高带宽和低延迟的设计
- 提供高频操作，无需使用复杂的桥
- 满足各种组件的接口要求
- 适合具有高初始延迟的存储控制器
- 为互连架构实现提供灵活性
- 向后兼容现有 AHB 和 APB 接口

AXI 协议的关键特征有：

- 分离的地址/控制和数据相位
- 使用字节选通的方式实现非对齐传输
- 采用基于突发的传输，主机只提供起始地址
- 分离的读写数据通道，提供低成本的 DMA 访问
- 支持发送多个 outstanding 地址(注：outstanding 是指，地址和数据传输并没有严格的先后要求，即在处理两个不同的 transaction 时可以不必等待一个 transaction 处理完之后再处理另一个，大大提高系统处理效率)
- 支持乱序(out-of-order transaction)传输(注：out-of-order 是指，数据传输时可以根据不同 ID 而对顺序没有要求，但是相同 ID 的 transaction 必须按顺序传输。所以重要的是通过 ID 来区分，而 outstanding 则与 ID 无关)
- 易于通过添加寄存器达到时序收敛

AXI 协议包含了可选的扩展，该扩展覆盖了用于低功耗操作的信号。

1.2 架构

AXI 协议是基于突发的。每个事物在地址通道上都具有地址和控制信息，来描述要传输的数据特性。在主机和从机之间传输的数据，使用一个到从机的写数据通道，或一个到主机的读地址通道。在写事物中，所有数据流是从主机到从机的，AXI 协议有一个附加的写响应通道，可以让从机通知主机写事物完成。

AXI 协议允许：

- 允许在实际数据传输之前发送地址信息
- 支持多个 outstanding 传输
- 支持乱序(out-of-order)传输

图 A1-1 展示了一个读传输是如何使用读地址和读数据通道的。

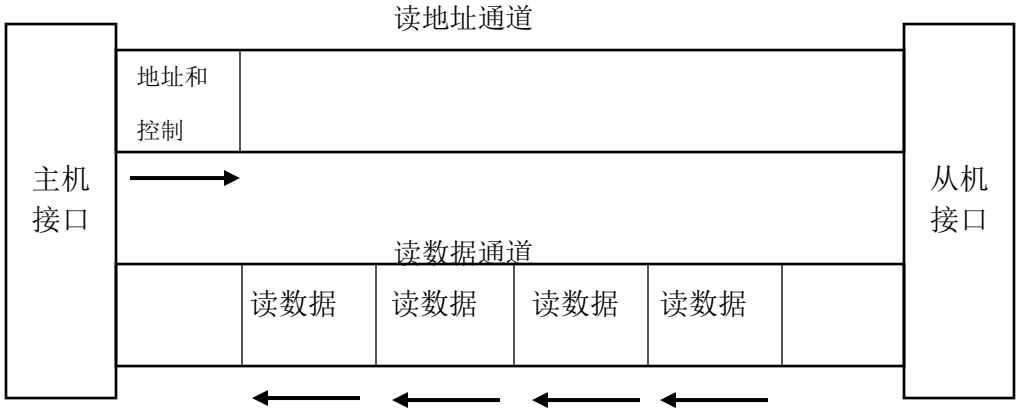


图 1-1 读通道结构

图 1-2 展示了展示了一个写传输是如何使用写地址、写数据以及写响应通道的。

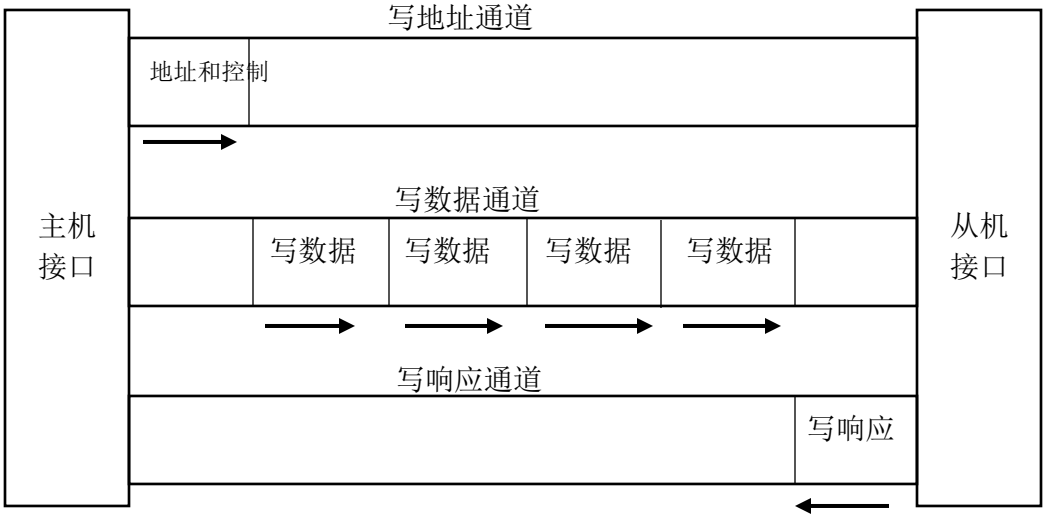


图 1-2 写通道结构

1.2.1 通道定义

每个独立的通道都包含一组信号，并使用一个双向的 **VALID** 和 **READY** 握手机制。

源设备使用 **VALID** 信号来表示通道上的地址、数据或控制信息何时有效。目的设备使用 **READY** 信号来表示其何时可以接收信息。读数据通道和写数据通道都包含一个 **LAST** 信号来表示一个事物中最后一个数据项的传输何时发生。

读和写地址通道

读和写事物都各自有自己的地址通道。地址通道用于传送一次传输所需的所有地址和控制信息。AXI 协议支持以下机制：

- 长度可变的突发，每个突发中的数据传输个数可以从 1 到 16 个
- 突发中一个传输的大小可以是 8-1024 bits
- 回环，增量或固定长度突发

- 使用独占或锁定访问的原子操作
- 系统级高速缓存和缓存控制
- 安全和特权访问

读数据通道

读数据通道用于从从机向主机返回读数据和任何读响应信息。读数据通道包含：

- 数据总线，宽度可以是 8，16，32，64，128，256，512 或 1024 bits
- 一个表示读事物完成状态的读响应信号

写数据通道

写数据通道用于从主机向从机传输写数据，包括：

- 数据总线，宽度可以是 8，16，32，64，128，256，512 或 1024 bits
- 每 8 bits 数据一个字节选通信号，用来表示数据的哪个字节有效

写数据通道的信息通常是被缓存的，以便主机在没有得到从机前一次写事物确认的情况下可以执行新的写事物。

写响应通道

从机使用写响应通道来响应写传输。所有写传输都要求在写响应通道上返回完成信号。

对每个突发，完成信号只产生一次，而不是突发中每个单个的数据传输都回产生一次完成信号。

1.2.2 接口和互联

一个典型的系统由通过某种互联方式连接到一起的若干个主机和从机组成，如图 1-3 所示。

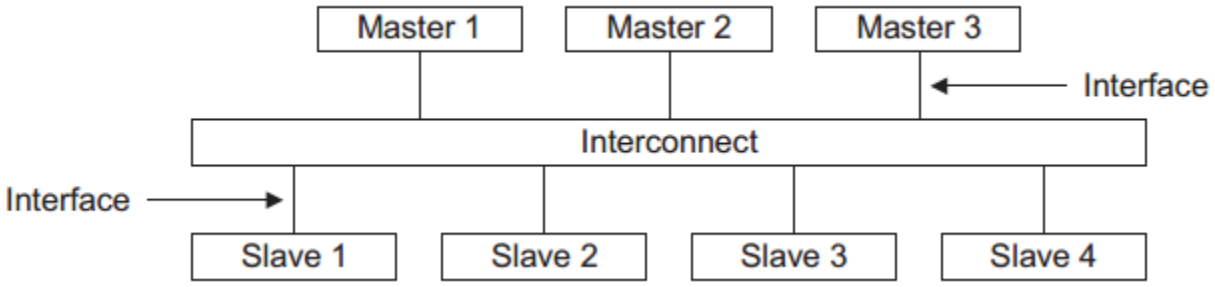


图 1-3 接口和互联

AXI 协议提供了一个单一的接口定义来描述接口：

- 在主机和 interconnect 之间
- 在从机和 interconnect 之间
- 在主机和从机之间

该接口的定义支持多种不同 interconnect 的开发。

设备之间的 interconnect 等效于具有对称的主机和从机端口的一种设备，这种设备可以用来连接真实的主机和从机设备。

大多数系统使用以下三种中的一种 interconnect 拓扑：

- 共用地址和数据总线
- 共用地址总线和多个数据总线
- 具有多个地址和数据总线的多层结构

在大多数系统中，地址通道的带宽要求小于数据通道。这种系统通过多个数据总线共用一个地址总线以便可进行并行数据传输，以此，可以在系统的性能和 interconnect 复杂性之间达到很好的平衡。

1.2.3 寄存器片

每个 AXI 通道只在一个方向上传送信息，并且在各种通道之间不要求有固定的关系。这点很重要，因为这可以在任何通道中插入一个寄存器片。这使得在延迟周期和操作的最大频率之间权衡变得可能。

也可以在一个给定互联中的几乎任何点上使用一个寄存器片。这有利于处理器和高性能 memory 之间直接、快速的连接，但是使用简单的 register slice 可以分离较长的路径 给低性能外设

1.3 基本事物

本章节给出了基本的 AXI 协议事物的例子。每个例子展示了 **VALID** 和 **READY** 握手机制。地址信息和数据的传输都发生在当 **VALID** 和 **READY** 信号都为高时。提供的例子如下：

- *读突发例子*
- *重叠的读突发例子*
- *写突发例子*

本章节也描述了 *事物排序*。

1.3.1 读突发例子

图 1-4 展示了有 4 个传输的一个读突发的例子。在这个例子中，主机驱动地址，从机在一个周期后接收地址。

注意：

主机也会驱动一组控制信号来标示突发长度和类型，但为了简化，该图中省略了这些信号。

在地址出现在地址总线上之后，读数据通道上发生数据传输。从机保持 **VALID** 信号为低，直到读数据有效。在突发中最后一个数据传输，从机断言 **RLAST** 信号来表示最后一个数据项已被传输。

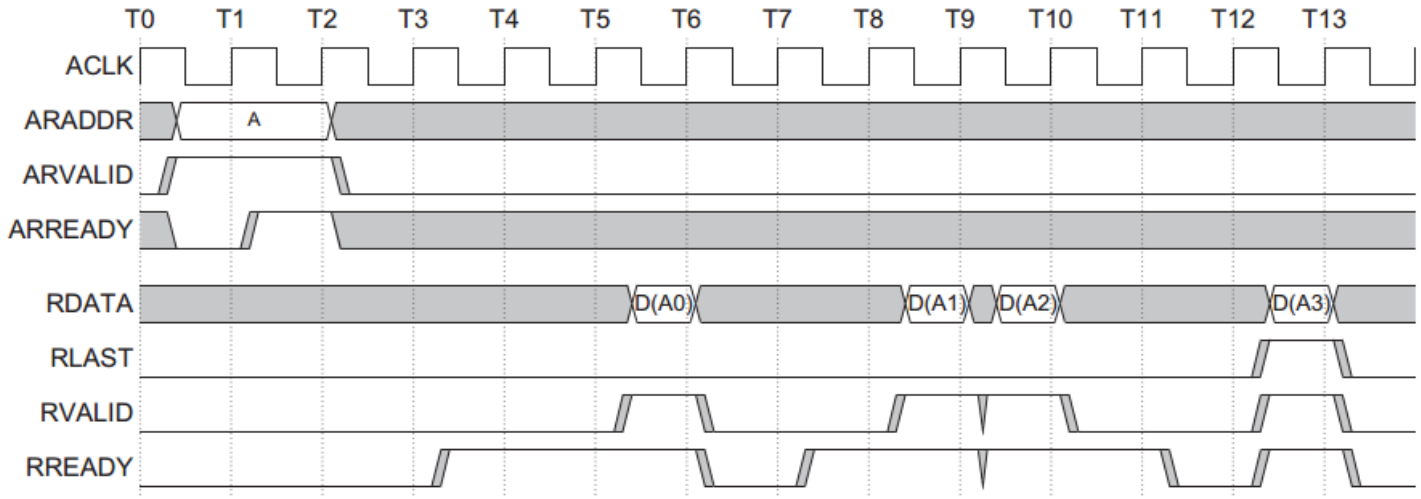


图 1-4 读突发

1.3.2 重叠的读突发例子

图 1-5 展示了从机在接收完第一个地址之后，主机怎样可以驱动另外一个地址。这可以使从机在完成第一突发的同时，并行地处理第二个突发中的数据。

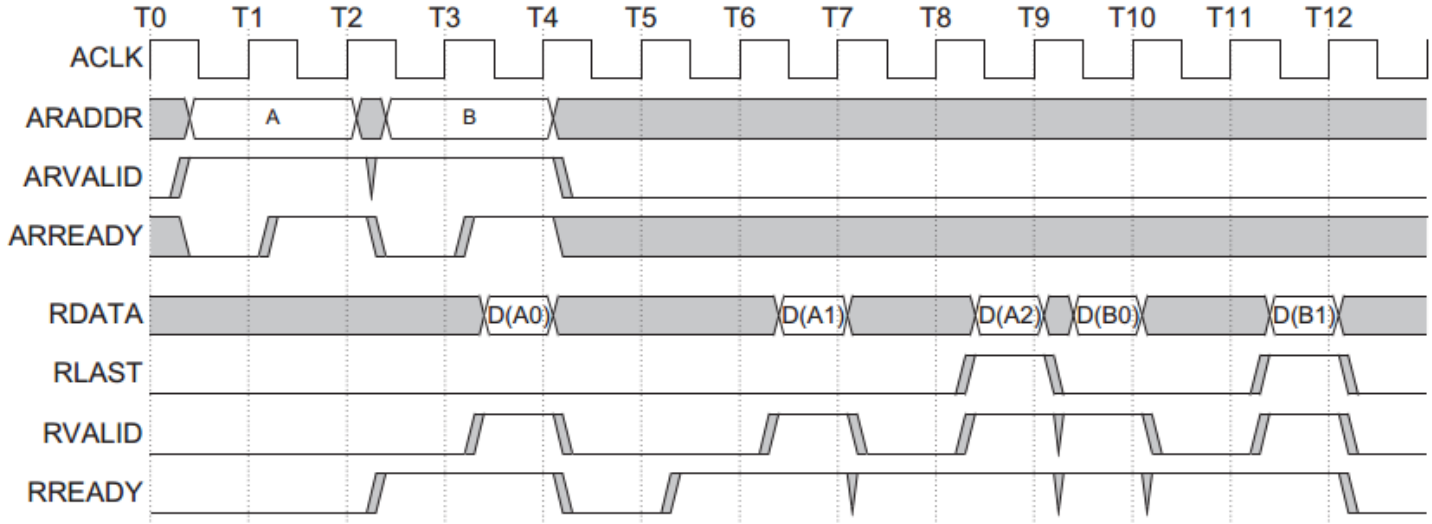


图 1-5 重叠的读突发

1.3.3 写突发例子

图 1-6 展示了一个写突发。当主机在写地址通道上发送了一个地址和控制信息时，突发过程开始。之后，主机通过写数据通道发送每个写数据。当主机发送最后一个数据时，**WLAST** 信号拉高。当从机接收完所有的数据之后，会驱动一个写响应给主机来表示写事物完成。

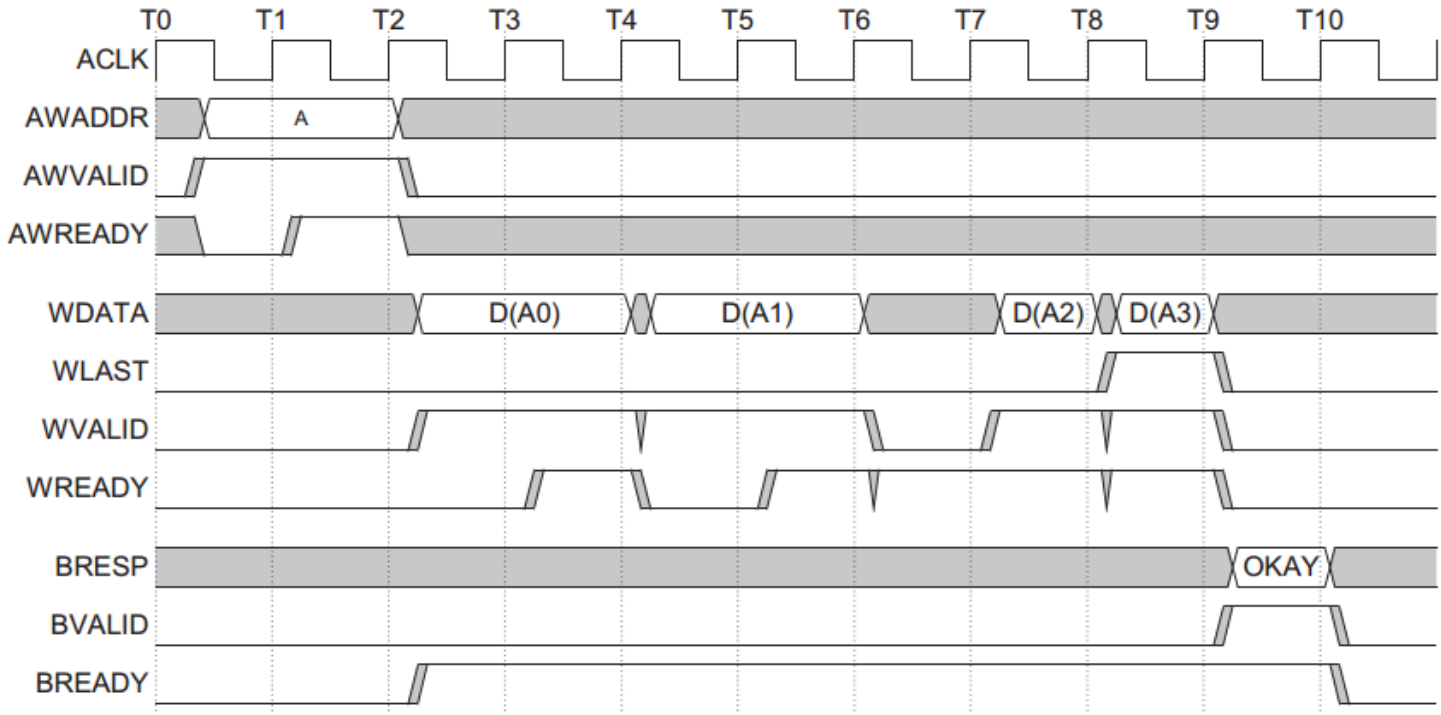


图 1-6 写突发

1.3.4 事物排序

AXI 协议允许完成乱序事物。每个通过接口的事物都会给出一个 ID tag。协议要求相同 ID tag 的事物要按顺序完成，但是不同 ID tags 的事物可以乱序完成。

乱序事物可以以两种方式来提高系统的性能：

- 互联可以允许后发送的、被快速响应从机接收的事物，先于先前发送给较慢从机的事物完成。
- 复杂的从机可以乱序返回读数据。例如，在先访问的数据准备好之前，一个后访问的数据项可能已经可以在一个内部缓存中得到了

如果一个主机要求事物按照其被发送的顺序来完成，则这些事物必须具有相同的 ID tag。但是，如果主机不要求按顺序完成事物，则主机可以使用不同的 ID tags 来发送事物，并允许事物按任何顺序完成。

在一个多主机的系统中，互联负责将额外信息附加到 ID tag，以保证来自所有主机的 ID tags 都是唯一的。ID tag 类似于一个主机号，但被扩展了——每个主机可以通过提供一个 ID tag 来表示虚拟主机号，以此来在同一个端口中实现多个虚拟主机。

尽管复杂的设备可以使用乱序组件，而简单的设备不要求使用。简单的主机可以用相同的 ID tag 来发送每个事物，简单的从机可以按顺序响应每个事物，而不用考虑 ID tag。

1.4 附加特征

AXI 协议也支持以下附加的特征：

突发类型

AXI 协议支持三种突发类型，适用于：

- 正常 memory 访问

- 回环高速缓存线(wrapping cache line)突发
- 访问外设 FIFO 位置的流数据

参见第 4 章*地址选项*。

系统高速缓存支持(System cache support)

AXI 协议的高速缓存支持信号允许一个主机向系统级高速缓存提供一个事物的可缓存、可高速缓存，以及可分配的属性。

参见*高速缓存支持*。

保护单元支持

为了允许特权和安全访问，AXI 协议提供了三种级别的保护单元支持。

参见保护单元支持。

原子操作

AXI 协议为独占访问和锁定访问定义了一种机制。

参见第 6 章*原子访问*。

错误支持

AXI 协议为地址解码错误和从机产生的错误提供了错误支持。

参见第 7 章*响应信号*。

非对齐地址

为了增强一个突发中初始访问的性能，AXI 协议支持非对齐的突发起始地址。

参见第 10 章*非对齐传输*。

2. 信号描述

本章定义了 AXI 的信号。尽管总线宽度和事物 ID 宽度是开发确定的，但本章的表中展示了一个 32-bit 数据总线，一个 4-bit 写数据选通，以及 4-bit 的 ID 字段。本章包含以下章节：

- *全局信号*
- *写地址通道信号*
- *写数据通道信号*
- *写响应通道信号*
- *读地址通道信号*
- *读数据通道信号*
- *低功耗接口信号*

2.1 全局信号

表 2-1 列出了全局的 AXI 信号。

表 2-1 全局信号

信号	源	描述
ACLK	时钟源	全局时钟信号。所有信号都在全局时钟的上升沿采样。
ARESETn	Reset 源	全局复位信号，低有效，参见 A3-Reset

2.2 写地址通道信号

表 2-2 列出了 AXI 写地址通道信号。

表 2-2 写地址通道信号

信号	源	描述
AWID[3:0]	主机	写地址 ID。该信号为写地址组信号的 ID tag。
AWADDR[31:0]	主机	写地址。在一个写突发事物中，写地址总线给出了第一个传输的地址。相应的控制信号用来决定突发中剩余传输的地址。
AWLEN[3:0]	主机	突发长度。突发长度给出了一个突发中准确的传输个数。该信息决定了与地址相对应的数据传送次数。参见表 4-1。
AWSIZE[2:0]	主机	突发大小。该信号表示突发中每个传输的大小。字节选通表示更新哪个字节通道。参见表 4-2。
AWBURST[1:0]	主机	突发类型。突发类型和突发大小决定了突发中每个传输的地址是怎么计算的。参见表 4-3。
AWLOCK[1:0]	主机	锁定类型。该信号为传输的原子特性提供了附加的信息。参见表 6-1。
AWCACHE[3:0]	主机	cache 类型。该信号表示事物的可缓存、可高速缓存、write-through、write-back、以及分配属性。参见表 5-1。
AWPROT[2:0]	主机	保护类型。该信号表示事物的正常、特权或安全保护级别，以及事物是一个数据访问还是指令访问。参见*保护单元支持*。
AWVALID	主机	写地址有效。该信号表示有效写地址和控制信息准备好： 1 = 地址和控制信息有效 0 = 地址和控制信息无效 该信号保持稳定，直到地址确认信号 **AWREADY** 拉高
AWREADY	从机	写地址准备好。该信号表示从机准备好接收地址和相应的控制信号。 1 = 从机准备好 0 = 从机没有准备好

2.3 写数据通道信号

表 2-3 列出了 AXI 写数据通道信号。

表 2-3 写数据通道信号

信号	源	描述
WID[3:0]	主机	写 ID tag。该信号为写数据传输的 ID tag。**WID** 值必须和写事物的 **AWID** 值相同。
WDATA[31:0]	主机	写数据。写数据总线可以是 8、16、32、64、128、256、512 或 1024 bits 宽
WSTRB[3:0]	主机	写选通信号。该信号表示 memory 中哪个字节通道被更新。写数据总线的每 8 位具有一个写选通位。因此，**WSTRB[n]**对应 **WDATA[(8 x n) + 7 : (8 x n)]**。
WLAST	主机	最后一次写。该信号表示一个写突发中的最后一次传输。
WVALID	主机	写有效。该信号表示写数据和选通信号有效： 1 = 写数据和选通信号有效 0 = 写数据和选通信号无效
WREADY	从机	写准备好。该信号表示从机可以接收写数据： 1 = 从机准备好 0 = 从机没有准备好

2.4 写响应通道信号

表 2-4 列出了 AXI 写响应通道信号。

表 2-4 写响应通道信号

信号	源	描述
BID[3:0]	从机	响应 ID tag。该信号是写响应的 ID tag。**BID** 值必须和从机响应的写事物的 **AWID** 值相同。
BRESP[1:0]	从机	写响应。该信号表示写事物的状态。允许的响应有：OKAY,EXOKAY,SLVERR 以及 DECERR。
BVALID	从机	写响应有效。该信号表示一个写响应有效： 1 = 写响应有效 0 = 写响应无效
BREADY	主机	响应准备好。该信号表示主机可以接收一个写响应： 1 = 主机准备好 0 = 主机没有准备好

2.5 读地址通道信号

表 2-5 列出了 AXI 读地址通道信号。

表 A2-5 读地址通道信号

信号	源	描述
ARID[3:0]	主机	读地址 ID。该信号是读地址组信号的 ID tag。
ARADDR[31:0]	主机	读地址。读地址总线给出了一个读突发事物中的第一个传输的地址。主机只提供突发的第一个地址，和地址一起被发送的控制信号决定了突发中剩余传输的地址是怎么计算的。
ARLEN[3:0]	主机	突发长度。该信号给出一个突发中准确的传输个数。该信息决定了与地址相对应的数据传输次数。参见表 4-1。
ARSIZE[2:0]	主机	突发大小。该信号表示突发中每个传输的大小。参见表 4-2。
ARBURST[1:0]	主机	突发类型。突发类型和突发大小决定了突发中每个传输的地址是怎么计算

		的。参见表 4-3。
ARLOCK[1:0]	主机	锁定类型。该信号为传输的原子特性提供附加信息。参见表 6-1。
ARCACHE[3:0]	主机	cache 类型。该信号为传输的高速缓存特性提供附加信息。参见表 5-1。
ARPROT[2:0]	主机	保护类型。该信号为事物提供保护单元信息。参见*保护单元支持*。
ARVALID	主机	读地址有效。该信号表示，当为高时，读地址和控制信息有效，并会保持稳定，直到地址确认信号 **ARREADY** 拉高。 1 = 地址和控制信息有效 0 = 地址和控制信息无效
ARREADY	从机	读地址准备好。该信号表示从机准备好接收地址和相应的控制信号： 1 = 从机准备好 0 = 从机没有准备好

2.6 读数据通道信

表 2-6 列出了 AXI 读数据通道信号。

表 2-6 读数据通道信号

信号	源	描述
RID[3:0]	从机	读 ID tag。该信号是从机产生的读数据组信号的 ID tag。**RID** 的值由从机产生，必须与从机响应的读事物的 **ARID** 值相同。
RDATA[31:0]	从机	读数据。读数据总线可以是 8、16、32、64、128、256、512 或 1024 bits 宽
RRESP[1:0]	从机	读响应。该信号表示读传输的状态。允许的响应有：OKAY,EXOKAY,SLVERR 以及 DECERR。
RLAST	从机	最后一次读。该信号表示一个读突发中的最后一次传输。
RVALID	从机	读有效。该信号表示请求的读数据有效，并且可完成读传输： 1 = 读数据有效 0 = 读数据无效
RREADY	主机	读准备好。该信号表示主机可以接收读数据和响应： 1 = 主机准备好 0 = 主机没有准备好

2.7 低功耗接口信号

表 2-7 列出了可选的低功耗接口信号。

表 2-7 低功耗接口信号

信号	源	描述
CSYSREQ	时钟控制器	系统退出低功耗状态请求。该信号是从系统时钟控制器发出的请求，用于外设进入低功耗状态。
CSYSACK	外设设备	低功耗请求确认。该信号是从外设发出，用来确认低功耗请求。
CACTIVE	外设设备	时钟有效。该信号表示外设请求时钟信号： 1 = 外设时钟被请求 0 = 外设时钟没有被请求

3. 通道握手

本章讲述了主机/从机握手过程，以及大概描述了 READY 和 VALID 握手信号的关系和默认值。包含以下章节：

- 握手过程
- 通道之间的关系
- 通道握手信号之间的依赖

3.1 握手过程

所有的 5 个通道都是用相同的 **VALID/READY** 握手过程来传输地址、数据以及控制信息。这种双向流控制机制意味着主机和从机都可以控制主机和从机之间数据和信息的传输速率。源设备产生 **VALID** 信号来表示地址、数据或控制信息何时有效。目标设备产生 **READY** 信号来表示可以接收源设备的信息。只有当 **VALID** 和 **READY** 同时为高时，才能进行传输。

在主机和从机接口中，输入信号和输出信号之间不能有组合路径。

图 3-1 到图 3-3 展示了握手序列的例子。在图 3-1 中，源设备发出数据或控制信息，并驱动 **VALID** 信号为高。从源设备发出的数据或控制信息保持稳定，直到目标设备驱动 **READY** 信号为高，表示目标设备接收数据或控制信息。箭头标示出了传输发生的时间。

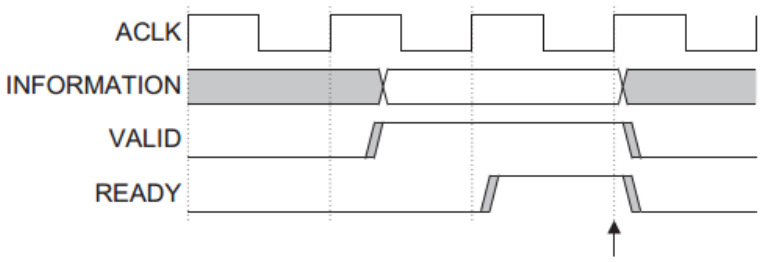


Figure 3-1 VALID before READY handshake

在图 3-2 中，目标设备在数据或控制信息有效之前驱动 **READY** 为高。这表示只要数据或控制信息有效，目标设备就可以在一个单周期内接收数据或控制信息。箭头标示出了传输发生的时间。

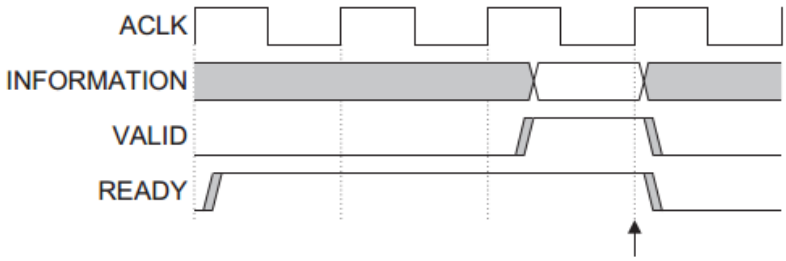


Figure 3-2 READY before VALID handshake

在图 3-3 中，源设备和目标设备在同一个周期内表示源设备和目标设备都可以传输数据或控制信息。在这种情况下，传输立即发生。箭头标示出传输发生的时间。

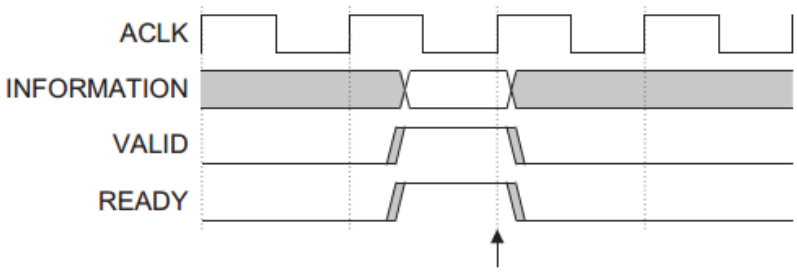


Figure 3-3 VALID with READY handshake

单独的 AXI 协议通道握手机制描述在：

- 写地址通道
- 写数据通道
- 写响应通道
- 读地址通道
- 读数据通道

3.1.1 写地址通道

只有当驱动有效地址和控制信息时，主机才能断言 **AWVALID** 信号。**AWVALID** 必须保持断言，直到从机接收了地址和控制信息，并断言了对应的 **AWREADY** 信号。

AWREADY 信号的默认值可以是高也可以是低，推荐默认值为高，尽管如果 **AWREADY** 为高，则从机必须可以接收发送到从机的任何有效地址。

AWREADY 信号可以默认为低，但不推荐，因为这意味着传输要至少占用两个周期，一个周期断言 **AWVALID**，另一个周期断言 **AWREADY**。

3.1.2 写数据通道

在一个写突发期间，只有当驱动有效写数据时，主机才能断言 **WVALID** 信号。**WVALID** 必须保持断言直到从机接收写数据并断言了 **WREADY** 信号。

只有当从机可以始终在一个单周期内接收写数据时，**WREADY** 的默认值才能为高。

当主机驱动突发中的最后一个写传输时，主机必须断言 **WLAST** 信号。

当 **WVALID** 为低时，**WSTRB[3:0]**信号可以是任何值，虽然这两个信号被推荐为即可以是低，也可以保持为其之前的值。

3.1.3 写响应通道

只有当驱动一个有效写响应时，从机才能断言 **BVALID** 信号。**BVALID** 必须保持断言，直到主机接收写响应并断言 **BREADY**。

只有当主机可以始终在一个单周期内接收写响应时，**BREADY** 才能默认为高。

3.1.4 读地址通道

只有当驱动有效地址和控制信息时，主机才能断言 **ARVALID** 信号。**ARVALID** 信号必须保持断言，直到从机接收地址和控制信息并断言对应的 **ARREADY** 信号。

默认的 **ARREADY** 可以是高，也可以是低。推荐默认为高，虽然如果 **ARREADY** 为高，则从机必须可以接收任何发送到从机的有效地址。

ARREADY 可以默认为低，但不推荐，因为这意味着传输至少要占用两个周期，一个周期断言 **ARVALID**，另一个周期断言 **ARREADY**。

3.1.5 读数据通道

只有当驱动有效读数据时，从机才能断言 **RVALID** 信号。**RVALID** 必须保持断言，直到主机接收数据并断言 **RREADY** 信号。即使一个从机只有一个读数据源，但是当其响应一个读数据请求时必须拉高 **RVALID** 信号。

主机接口使用 **RREADY** 信号来表示其接收数据。只有当主机不论任何时候开始一个读事物时都能立即接收读数据的情况下，**RREADY** 的默认状态才能为高。

当从机驱动突发中最后一个读传输时必须将 **RLAST** 信号拉高。

3.2 通道之间的关系

地址、读、写以及写响应通道之间的关系是灵活的。

例如，写数据可以在相应的写地址之前出现在接口上。当写地址通道包含的寄存器级多于写数据通道时，可能会出现这种情况。写数据也可以和地址出现在同一个周期。

当互联必须决定目标设备地址空间或从机空间时，互联必须重新对齐地址和写数据。这个要求用来保证写数据只会对寻址的从机有效。

有两种关系必须被维持：

- 读数据必须始终紧跟在对应的地址之后
- 一个写响应必须始终紧跟在写事物中和该响应相关的上一个写传输之后

3.3 通道握手信号之间的依赖

为了防止死锁发生，必须遵循存在于握手信号之间的依赖关系。

在任何事物中：

- 一个 AXI 组件的 **VALID** 信号，不能依赖于事物中其他组件的 **READY** 信号
- **READY** 信号可以等待 **VALID** 信号断言

注意：如果在断言 **READY** 之前等待 **VALID** 被断言是可以被接受的，那么根据 **VALID** 默认的断言优先级来断言 **READY** 也同样可以被接受，这可以让设计更有效。

图 3-4 和图 3-5 展示了握手信号依赖。单箭头指向的信号可以在起始信号之前或之后被断言。双箭头指向的信号必须只能在起始信号之后被断言。

图 3-4 展示了在一个读事物中：

- 从机可以先等待 **ARVALID** 被断言，之后从机再断言 **ARREADY**
- 从机必须等待 **ARVALID** 和 **ARREADY** 都被断言之后，从机才能断言 **RVALID** 来返回读数据

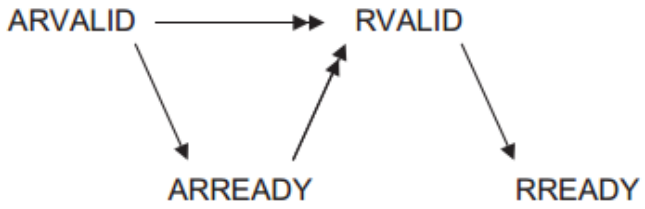


图 3-4 读事物握手依赖

图 3-5 展示了在一个写事物中：

- 主机必须先断言 **AWVALID** 或 **WVALID**，之后再等待从机断言 **AWREADY** 或 **WREADY**
- 从机可以先等待 **AWVALID** 或 **WVALID**、或两个都被断言，之后从机再断言 **AWREADY**
- 从机可以先等待 **AWVALID** 或 **WVALID**、或两个都被断言，之后从机再断言 **WREADY**
- 从机必须先等待 **WVALID** 和 **WREADY** 被断言，之后从机才能断言 **BVALID**

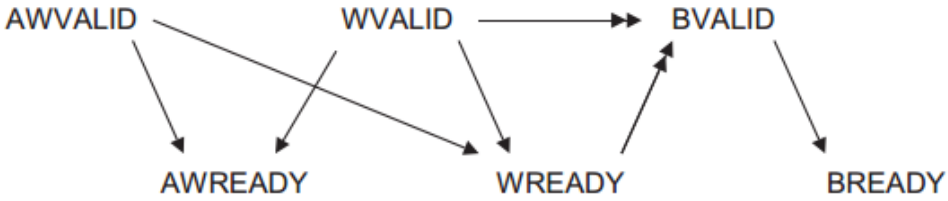


图 3-5 写事物握手依赖

注意：很重要的一点：在一个写事物期间，主机必须先驱动 **WVALID**，然后再等待 **AWREADY** 被断言。如果从机线等待 **WVALID**，然后再断言 **AWREADY**，则可能引发死锁。

---

4. 地址选项

本章讲述了 AXI 突发类型，以及计算突发中传输的地址和字节通道。包含以下章节：

- *关于地址选项*
- *突发长度*
- *突发大小*
- *突发类型*
- *突发地址*

4.1 关于地址选项

AXI 协议是基于突发的，主机通过驱动传输控制信息以及传输中第一个字节的地址来开始每个突发。随着突发的传送，从机负责计算突发中后续传输的地址。

突发不能跨越 4KB 边界，以防止突发跨越从机之间的边界，因此要求从机在其内部限制地址增量的大小。

4.2 突发长度

AWLEN 和 **ARLEN** 信号规定了每个突发中的数据传输次数。如表 4-1 所示，每个突发可以是 1-16 个传输长度。

表 4-1 突发长度编码

ARLEN[3:0] **AWLEN[3:0]**	**数据传输** **次数**
b0000	1
b0001	2
b0010	3
. . .	. . .
b1101	14
b1110	15
b1111	16

对于回环突发，突发长度必须是 2、4、8 或 16 个传输。

每个传输必须通过 **ARLEN** 或 **AWLEN** 确定传输次数。没有组件可以提前终止一个突发来减少数据传输次数。在一个写突发期间，主机可以通过取消断言所有写选通来 disable 进一步的写，但其必须完成突发中剩余的传输。在一个读突发期间，主机可以丢弃进一步的读数据，但是其必须完成突发中剩余的传输。

警告：当访问一个读敏感的设备，如一个 FIFO 时，丢弃没有被请求的读数据可能会导致丢失数据。主机禁止以一个大于请求的突发长度来访问这样的设备。

4.3 突发大小

表 4-2 展示了 **ARSIZE** 或 **AWSIZE** 信号是怎样确定一个突发中，每拍(beat)或每个数据传输中要传输的最大数据字节数的。

表 4-2 突发大小编码

ARSIZE[2:0] AWSIZE[2:0]	传输中的 字节数
b000	1
b001	2
b010	4
b011	8
b100	16
b101	32
b110	64
b111	128

AXI 通过传输地址决定每个传输使用数据总线的哪个字节通道。

对于传输大小窄于数据总线的增量或回环突发，突发中每拍的数据传输都在不同的字节通道上。一个固定长度突发的地址是保持恒定的，并且每个传输都使用相同的字节通道。

任何传输的大小都不能超过事物中组件的数据总线宽度。

4.4 突发类型

AXI 协议定义了三种突发类型，描述如下：

- *固定长度突发*
- *增量突发*
- *回环突发*

表 4-3 展示了 **ARBURST** 或 **AWBURST** 信号是怎样选择突发类型的。

表 4-3 突发类型编码

ARBURST[1:0] AWBURST[1:0]	突发类型	描述	访问
b00	固定长度突发	地址固定的突发	FIFO 类型
b01	增量突发(INCR)	地址递增的突发	正常序列 memory
b10	回环突发(WRAP)	地址递增的突发，在回环边界会卷回到一个较低地址	高速缓存线 (Cache line)
b11	保留	-	-

4.4.1 固定长度突发

在一个固定长度突发中，每个传输的地址都保持相同。这种突发类型用于重复访问相同位置，比如装载和清空一个外设 FIFO。

4.4.2 增量突发

在一个增量突发中，每个传输的地址都是在前一个传输地址的基础上递增的。增量值取决于传输大小。例如，突发中每个传输的大小是 4bytes，则每个传输地址是前一个地址加 4。

4.4.3 回环突发

回环突发类似于增量突发，突发中每个传输的地址按增量递增的。但是在回环突发中，当达到较大的地址边界时，地址会卷回到较小的地址。回环边界是突发中每个传输的大小乘以突发中总共传输的次数。

回环突发需要遵循两个限制：

- 起始地址必须对齐到传输大小
- 突发长度必须是 2、4、8 或 16

4.5 突发地址

本章节给出了一些简单的公式用于决定一个突发中传输的地址和字节通道。公式使用以下变量：

Start_Address	主机发出的起始地址
Number_Bytes	每个数据传输中的最大 byte 数
Data_Bus_Bytes	数据总线中的字节通道数量
Aligned_Address	起始地址的对齐状态
Burst_Length	一个突发中总的数据传输次数
Address_N	一个突发中第 N 个传输的地址。突发中第一个传输对应 N=1。
Wrap_Boundary	一个回环突发中的最低地址
Lower_Byte_Lane	一个传输最低寻址字节的字节通道
Upper_Byte_Lane	一个传输最高寻址字节的字节通道
INT(x)	x 取整操作

使用以下公式来决定一个突发中传输的地址：

- Start_Address = **ADDR**
- Number_Bytes = $2^{\mathbf{SIZE}}$
- Burst_Length = **LEN** + 1
- Alingned_Address = (INT(Start_Address/Number_Bytes)) x Number_Bytes

使用以下公式来决定突发中第一个传输的地址：

- Address_1 = Start_Address

使用以下公式来决定突发中第一个传输之后任意传输的地址：

- Address_N = Aligned_Address + (N – 1) x Number_Bytes

对于回环突发，Wrap_Boundary 变量被扩展用来计算回环边界：

- Wrap_Boundary = (INT(Start_Address / (Number_Bytes x Burst_Length))) x (Number_Bytes x Burst_Length)

如果 Address_N = Wrap_Boundary + (Number_Bytes x Burst_Length)，则使用以下公式：

- Address_N = Wrap_Boundary

以下公式决定了一个突发中的第一个传输使用哪个字节通道：

- Lower_Byte_Lane = Address_N – (INT(Address_N / Data_Bus_Bytes)) x Data_Bus_Bytes
- Upper_Byte_Lane = Lower_Byte_Lane + Number_Bytes – 1

数据传输的字节通道计算如下：

- DATA((8 x Upper_Byte_Lane) + 7 : (8 x Lower_Byte_Lane))

5. 附加控制信息

本章讲述了 AXI 协议对系统级 caches 以及保护单元的支持。包含以下章节：

- *Cache 支持*
- *保护单元支持*

5.1 Cache 支持

通过使用 cache 信息信号 **ARCACHE** 和 **AWCACHE**，可以提供对系统级 caches 以及性能增强组件的支持。这两个信号提供了关于事物怎么被处理的附加信息。

ARCACHE[3:0]或 AWCACHE[3:0]信号通过提供事物的可缓存、可高速缓存以及可分配属性来支持系统级 caches：

***Bufferable*(B) bit，ARCACHE[0]及 AWCACHE[0]**

当该位为高时，意味着 interconnect 或任何组件都可以将达到其最终目标设备的事物延迟任何周期数。该属性只用于写事物

***Cacheable*(C) bit，ARCACHE[1]及 AWCACHE[1]**

当该位为高时，意味着在最终目标设备上的事物其特征不需要与源事物的特征相同。

对于写事物，意味着若干个不同的写事物可以被合并到一起。

对于读事物，这意味着一个位置可以被多个事物预取，或者只能被取一次。

该 bit 应该与 *Read Allocate*(RA)和 *Write Allocate*(WA) bits 一起使用，来判定一个事物是否是可高速缓存的。

***Read Allocate*(RA) bit，ARCACHE[2]及 AWCACHE[2]**

当 RA bit 为高时，意味着如果传输是一个读，并且在 cache 中被错漏，则其应该可被分配。

如果 C bit 为低，则 RA bit 不能为高。

***Write Allocate*(WA) bit，ARCACHE[3]及 AWCACHE[3]**

当 WA bit 为高时，意味着如果传输是一个写，并且在 cache 中被错漏，则其应该可被分配。

如果 C bit 为低，则 WA bit 不能为高。

表 5-1 展示了 **ARCACHE[3:0]**和 **AWCACHE[3:0]**信号编码。

表 5-1 Cache 编码

ARCACHE[3:0] AWCACHE[3:0]				事物属性
WA	**RA**	**C**	**B**	
0	0	0	0	不可高速缓存并且不可缓存
0	0	0	1	只可缓存
0	0	1	0	可高速缓存，但不可分配
0	0	1	1	可高速缓存并且可缓存，但不可分配
0	1	0	0	保留
0	1	0	1	保留
0	1	1	0	可高速缓存 write-through，只可读分配
0	1	1	1	可高速缓存 write-back，只可读分配
1	0	0	0	保留
1	0	0	1	保留
1	0	1	0	可高速缓存 write-through，只可写分配
1	0	1	1	可高速缓存 write-back，只可写分配
1	1	0	0	保留
1	1	0	1	保留
1	1	1	0	可高速缓存 write-through，读写都可分配
1	1	1	1	可高速缓存 write-back，读写都可分配

对于写事物，**AWCACHE** 信号可被用来决定哪个组件提供写响应。如果一个写事物被标示为可缓存的，则可以由一个桥或者系统级 cache 提供写响应。但是，如果事物被标示为不可缓存的，则写响应必须由事物的最终目标设备提供。

AXI 协议没有定义一种让缓存的或者高速缓存的数据到达其目标设备的机制。例如，一个系统级 cache 可以具有一个控制器来管理 cache 入口的清理、刷新以及使无效。另一个例子是，包含一个写缓冲器的桥，这个桥可以具有控制逻辑，如果在接收到一个具有匹配的事物 ID 的不可缓存的写事物时，可用来 drain the buffer。

5.2 保护单元支持

为了支持复杂的系统设计，系统中的互联和其它设备都通常有必要提供保护来防止非法事物。**AWPROT** 和 **ARPROT** 信号给出了三种级别的访问保护：

正常或特权，ARPROT[0]及 AWPROT[0]

- 为低，表示一个正常访问
- 为高，表示一个特权访问

这被某些主机用来表示其处理模式。一个特权处理模式通常在系统中具有更高的访问级别。

安全和非安全，ARPROT[1]及 AWPROT[1]

- 为低，表示一个安全访问
- 为高，表示一个非安全访问

这用于被请求的处理模式具有更大差异度的系统中。

注意：

配置该位，以便当该位为高时，事物被认为是非安全的，当该位为低时，事物被认为是安全的。

指令或数据，ARPROT[2]及 AWPROT[2]

- 为低，表示一个数据访问
- 为高，表示一个指令访问

该位用来表示一个事物是指令访问还是数据访问。

注意：

该位所标示的只是作为一个建议，其并不能在所有情况下精确表示。例如，一个混合了指令和数据的事物。默认情况下，推荐将一个访问标记为一个数据访问，除非明确知道该事物是一个指令访问。

表 5-2 总结了 **ARPROT[2:0]**和 **AWPROT[2:0]**信号的编码。

表 5-2 保护编码

ARPROT[2:0] AWPROT[2:0]	**保护级别**
[0]	1 = 特权访问
	0 = 正常访问
[1]	1 = 非安全访问
	0 = 安全访问
[2]	1 = 指令访问
	0 = 数据访问

6. 原子访问

本章讲述了 AXI 协议怎样开发独占访问和锁定访问机制。包含以下章节：

- *关于原子访问*
- *独占访问*
- *锁定访问*

6.1 关于原子访问

为了使原子访问的开发更简单，使用 **ARLOCK[1:0]**或 **AWLOCK[1:0]**信号提供独占访问和锁定访问。表 6-1 展示了 **ARLOCK[1:0]**和 **AWLOCK[1:0]**信号的编码。

表 6-1 原子访问编码

ARLOCK[1:0] **AWLOCK[1:0]**	**访问类型**
b00	正常访问
b01	独占访问
b10	锁定访问
b11	保留

6.2 独占访问

独占访问机制使得信号量类型操作(semaphore type operations)的开发不需要总线在操作期间保持对某一特定主机的锁定。独占访问的优点是既不会影响关键总线访问延迟，也不会影响最大可达到的带宽。

ARLOCK[1:0]或 AWLOCK[1:0]信号选择独占访问，RRESP[1:0]或 BRESP[1:0]信号(参见表 7-1)标示独占访问成功或失败。

从级必须具有额外的逻辑来支持独占访问。AXI 协议提供了一个自动防故障(fail-self)机制来标示当一个主机尝试对一个不支持独占访问的从机进行独占访问。

6.2.1 独占访问过程

一个独占访问的基本过程如下：

1. 主机执行一个某地址位置的独占读
2. 一段时间之后，主机通过执行一个到相同地址位置的独占写，来尝试结束独占操作
3. 主机独占写访问的结果表示为:
    - 在读和写访问之间，如果没有其它主机对该位置进行写，则独占访问成功
    - 在读和写访问之间，如果任何主机对该位置进行了写，则独占访问失败。在这种情况下地址位置不会被更新

注意：

一个主机可能还没有完成一个独占操作的写操作部分。独占访问对硬件的监视，必须只能监视每个事物 ID 的一个地址。因此，如果一个主机没有完成独占操作的写操作部分，则接下来的一个独占写会改变正在被独占地监视的地址。

6.2.2 从主机视点的独占访问

主机通过执行一个独占读来开始一个独占操作。独占操作通常从从机返回一个 EXOKAY 响应，表示从机记录了要被监视的地址。

注意：

如果主机尝试从一个不支持独占访问的从机执行一个独占读操作，则从机会返回 OKAY 响应而不是 EXOAKY 响应。主机可将这种情况当作错误条件，表示不支持独占访问。推荐主机在这种情况下不执行该独占操作的写操作部分。

在独占读之后一段时间，主机尝试执行一个到相同位置的独占写操作。如果从独占读之后，地址位置没有改变，则独占写操作成功。从机返回 EXOKAY 响应，独占写操作更新 memory 的位置。

如果从独占读之后地址位置发生了改变，则独占写操作失败，从机返回 OKAY 响应代替 EXOKAY 响应。独占写操作不会更新 memory 位置。

一个主机可能没有完成独占访问的写操作部分。如果发生这种情况，则从机继续监视独占的地址，直到另一个独占读操作开始一个新的独占访问。

在独占访问的读操作完成之前，主机不能开始独占访问的写操作部分。

6.2.3 从机视点的独占访问

一个不支持 独占访问的从级可以忽视 **ARLOCK[1:0]**和 **AWLOCK[1:0]**信号。从机必须为正常访问和独占访问提供一个 OKAY 响应。

一个支持独占访问的从机必须具有监控电路。对于这种从机，推荐其具有一个监控单元，用来监控可访问该从机的每个可独占访问的主机 ID。一个单端口从机可以具有一个从机外部的标准独占访问监控器，但是多端口的从机可能要求内部监控器。

独占访问监控器记录任何独占读操作的地址和 **ARID** 值。之后监控器一直监控该位置，直到一个到该位置的写操作发生，或者直到相同 **ARID** 值的另一个独占读操作将监控器复位到一个不同的地址。

当一个具有给定 **AWID** 值的独占写操作发生，则监控器会检查该地址是否正在被独占地监控。如果是，则意味着在独占写操作之前，该位置没有发生过写操作，独占写操作会继续执行，来完成独占访问。从机返回 EXOKAY 响应到主机。

如果在独占写发生时，地址不再被监控，则表示以下一种情况：

- 从独占读之后，位置被更新
- 监控器已经被复位到另一个位置

以上两种情况下，独占写操作都不能更新地址位置，并且从机必须返回 OKAY 响应替代 EXOKAY 响应。

6.2.4 独占访问约束

以下约束适用于独占访问：

- 一个给定 ID 的独占写的大小和长度必须和前一个具有相同 ID 的独占读的大小和长度相同。
- 一个独占访问的地址必须被对齐到事物中总共的字节数。
- 独占读和独占写的地址必须一致
- 独占访问中独操作部分的 **ARID** 必须和写操作部分的 **AWID** 相同
- 独占访问中读和写部分的控制信号必须一致。
- 一个独占访问突发中要被传输的字节数必须是 2 的次幂，即 1、2、4、8、16、32、64 或 128 bytes。
- 一个独占突发中最大可被传输的字节数为 128。

- **ARCACHE[3:0]**或 **AWCACHE[3:0]**的值必须保证监控独占访问的从机要能看见事物。例如，一个正在被从机监控的独占访问不能具有一个表示该事物是可高速缓存的 **ARCACHE[3:0]**或 **AWCACHE[3:0]**值。

不遵循以上约束会导致不可预测的行为。

在一个独占操作期间，被监控的最小字节数由事物的长度和大小定义。监控一个更大的字节数是可接受的，最大为 128，即一个最大的独占访问。但是，这可能会造成混淆：当独占访问实际上成功了，但因为一个邻近的字节被更新，却被标示为失败。

6.2.5 不支持独占访问的从机

响应信号 **BRESP[1:0]**及 **RRESP[1:0]**包含一个用于正常访问成功的 OKAY 响应，以及一个独占访问成功的 EXOKAY 响应。也就是说，一个不支持独占访问的从机，可以提供一个 OKAY 响应来表示独占访问的失败。

注意：

一个到不支持独占访问从机的独占写会始终更新 memory 的位置。

一个到支持独占访问从机的独占写，只有该独占写成功后，才能更新 memory 的位置。

6.3 锁定访问

当一个事物的 **ARLOCK[1:0]**或 **AWLOCK[1:0]**信号表示该事物是一个锁定的传输时，互联必须确保只有发送该事物的主机才能被允许访问从机区域，直到从同一个主机发出的一个非锁定传输完成。互联中的仲裁器用来执行该限制。

当一个主机开始一个锁定的读或写事物序列，主机必须保证没有其他 outstanding 事物正在等待完成。

任何 **ARLOCK[1:0]**或 **AWLOCK[1:0]**表示一个锁定序列的事物，会强制互联锁定接下来的事物。因此，一个锁定的序列，其最后一个事物不能是 **ARLOCK[1:0]**或 **AWLOCK[1:0]**表示为锁定访问的事物，锁定序列必须始终以这样的最后一个事物来结束序列。这个最后的事物被包含进锁定的序列，作用是解除锁定。

当结束一个锁定序列时，主机必须保证所有之前锁定事物都完成后才能发送最后一个非锁定的事物。之后，主机必须确保最后一个非锁定的事物已经完全完成，之后才能开始其他事物。

主机必须确保在一个锁定序列中的所有事物都具有相同的 **ARID** 或 **AWID** 值。

注意：锁定访问要求互联在锁定序列处理期间要阻止任何其他事物的发生，因此会对互联的性能产生影响。推荐锁定访问仅用于支持遗留的设备。

以下约束是推荐但不强制的：

- 将所有锁定事物序列保持在相同的 4KB 地址区域内
- 将锁定事物序列限制为 2 个事物

7. 响应信号

本章讲述了 AXI 独和写事物中的四种从机响应。包含以下章节：

- *关于响应信号*
- *响应类型*

7.1 关于响应信号

AXI 协议允许为读和写事物发出响应。对于读事物，发自从机的响应信息是伴随读数据一起发送的，但是对于写事物，响应信息是通过写响应通道传送的。

AXI 协议的响应有：

- OKAY
- EXOKAY
- SLVERR
- DECERR

表 7-1 展示了 **RRESP[1:0]**和 **BRESP[1:0]**信号编码。

表 7-1 RRESP[1:0]和 BRESP[1:0]编码

RRESP[1:0] **BRESP[1:0]**	**响应**	**含义**
b00	OKAY	正常访问 okay 表示一个正常访问是否成功。也可以表示一个独占访问失败。
b01	EXOKAY	独占访问 okay 表示一个独占访问的读和写部分都成功了。
b10	SLVERR	从机错误，用于当访问成功到达从机，但是从机期望返回一个错误条件给源主机。
b11	DECERR	解码错误，通常由一个互联组件产生，表示没有从机对应事物的地址。

对于一个写事物，只会为整个突发给出一个响应，而不是突发中每个数据传输都有响应。

在一个读事物中，从机可以为一个突发中的不同传输给出不同的响应。例如，在一个 16 次读传输的突发中，从几可以为 15 个传输返回一个 OKAY 响应，并为剩下一个传输给出一个 SLVERR 响应。

协议规定请求的数据传输次数必须被执行，即使报告出一个错误。例如，一个 8 次读传输的事物被从从机请求，但是从机发生一个错误条件，则从机必须执行 8 次数据传输，每个传输返回一个错误响应。如果从机给出一个单一的错误响应，则突发中剩余的传输没有被取消。

本协议为可以发送多个 outstanding 地址的主机设定了限制，这种主机也必须可以支持精确的错误信号。这种主机必须能够前一个传输的一个错误响应，而此时后一个传输已经在进行中。

7.2 响应类型

本章节讲述 AXI 协议的四种响应类型：

- *正常访问成功*
- *独占访问*
- *从机错误*
- *解码错误*

7.2.1 正常访问成功

OKAY 响应表示：

- 一个正常访问成功
- 一个独占访问失败
- 一个独占访问，访问了一个不支持独占访问的从机

OKAY 是大部分事物的响应。

7.2.2 独占访问

EXOKAY 响应表示一个独占访问成功。第 6 章*原子访问*描述了这种响应。

7.2.3 从机错误

SLVERR 响应表示一个不成功的事物。从机错误条件的例子有：

- FIFO/buffer 溢出(overrun)或欠载(underrun)条件
- 尝试不支持的传输大小
- 尝试写访问一个只读的位置
- 从机中超时条件
- 尝试访问一个不具有寄存器的地址
- 尝试访问一个被禁止的或断电的功能

为了简化系统的监控和调试，推荐错误响应只用于错误条件，而不用于正常和期望事件。

7.2.4 解码错误

在一个没有完整解码地址映射的系统中，可能会寻址一个位置，该位置没有对应的从机来响应一个事物。在这样的一个系统中，互联必须提供适当的错误响应来标示访问为非法访问，并且也应该防止系统访问不存在从机的尝试。

当互联不能成功解码一个从机访问，一个有效的做法是将访问路由到一个默认从机，由默认从机返回 DECERR 响应。

一个开发选项是，让默认从机也记录详细的解码错误，以便之后判定错误是怎么发生的。以这种方式，默认从机可以大大简化调试过程。

AXI 协议要求一个事物的所有数据传输都要被完成，即使发生一个错误条件。因此给出 DECERR 响应的组件也必须满足该要求。

8. 顺序模型

本章讲述 AXI 协议是怎样使用事物 ID tags 来发送多个 outstanding 地址，以及乱序事物过程。包含以下章节：

- *关于排序模型*
- *传输 ID 字段*
- *读顺序*
- *正常写顺序*
- *写数据交错*
- *读和写交互*
- *ID 字段的互联使用*
- *推荐的 ID 字段宽度*

8.1 关于顺序模型

AXI 协议允许乱序事物，并可以发送多个 outstanding 地址。这种功能可以开发高性能的互联，使得数据量最大化，并使系统更有效。

ID 信号通过使每个端口作为多点端口来支持乱序事物。具有一个给定 ID 的所有事物都必须进行排序，但是对于具有不同 ID 的事物，并没有顺序的限制。五个事物 ID 分别为：

AWID 写地址组信号的 ID tag

WID 一个写事物的 ID tag。主机与写数据一同传送一个 **WID** 来匹配对应地址的 **AWID**

BID 写响应的 ID tag。从机发送一个 BID 来匹配其要响应的事物的 **AWID** 和 **WID**

ARID 读地址组信号的 ID tag

RID 一个读事物的 ID tag。从机发送一个 RID 来匹配其要响应的事物的 **ARID**。

注意：

并没有关于主机和从机使用这种先进功能的要求。简单的主机和从机只会按照事物发送的顺序一次处理一个事物。

发送多个 outstanding 地址的功能，也就是主机不用等待前一个事物完成就能发送下一个事物的地址。因为这种功能允许并行处理事物，因此可以提高系统性能。

以乱序方式完成事物处理的功能，也就是发送到较快 memory 区域的事物，不用等待前一个发送到较慢 memory 区域事物的完成，就可以提前完成。该功能因为减少了事物延迟的影响，因此也可以提高系统的性能。

注意：

重新排序的概念总是与其他事物相关(也就是事物之间可以重新排序)。一个突发中的数据传输之间不能重新排序。用来定义突发的地址和控制信号控制了突发中传输的顺序。

8.2 传输 ID 字段

AXI 协议提供了一个 ID 字段，来允许一个主机发送若干个分离的事物，每个分离的事物必须按顺序被返回。

主机可以使用事物的 **ARID** 或 **AWID** 字段来提供关于主机排序要求的附加信息。管理事物顺序的规则如下：

- 从不同主机发出的事物没有顺序限制。这些事物可以按任意顺序完成。
- 从同一主机发出、但 ID 值不同的事物没有顺序限制。这些事物可以按任意顺序完成。
- 具有相同 **AWID** 值的写事物序列中的数据，必须按照该主机发送对应地址的顺序完成传输。
- 具有相同 **ARID** 值的读事物序列中的数据，必须按以下顺序返回：
  - 当具有相同 **ARID** 的读事物来自同一个从机时，从机必须保证按照对应地址被接收的顺序返回读数据。
  - 当具有相同 **ARID** 的读事物来自不同从机时，互联必须保证按照主机发送地址的顺序返回读数据。
- **AWID** 和 **ARID** 相同的读事物和写事物之间没有顺序限制。如果一个主机要求一个顺序约束，则该主机必须确保第一个事物已完全结束，之后才能发送第二个事物。

8.3 读顺序

在一个主机接口，具有相同 **ARID** 值的事物中的读数据必须按主机发送地址的顺序被接收。来自不同 **ARID** 值的读事物，其数据可以按任意顺序被接收。具有不同 **ARID** 值的事物中，读数据可以被交错传输。

从机必须将相同 **ARID** 值的事物序列中的读数据按照其接收地址的顺序返回给主机。在一个具有不同 **ARID** 值的读事物序列中，从机可以按任意顺序返回读数据，而不用管事物到达从机的顺序。

从机必须保证任何被返回的数据，其 **RID** 值要和地址的 **ARID** 值相同，该地址是从机发送响应的地址。

互联(interconnect)必须保证来自不同从机的、具有相同 **ARID** 值的事物序列，其读数据按照主机发送地址的顺序被主机接收到。

读数据重新排序的深度，就是从机中还未发出的、可以被重新排序的地址数量。一个按顺序处理所有事物的从机，其读数据重新排序的深度为 1。读数据重新排序深度是一个静态数值，该数值必须由从机的设计者来规定。

8.4 正常写顺序

如果一个从机不支持写数据交错(*参见写数据交错*)，则主机必须按照与发送事物地址相同的顺序来发送写事物的数据。

大部分的从机都被设计成不支持写数据交错，因此，这种类型的从机必须按照其接收地址的顺序来接收写数据。如果互联(interconnect)将来自不同主机的写事物进行组合之后发送给一个从机，则互联必须确保按照与地址相同的顺序组合写数据。

即使写事物具有不同的 **AWID** 值，以上的限制也同样适用。

8.5 写数据交错

写数据交错功能可以使一个从机接口接收具有不同 **AWID** 值的、交错的写数据。从机声明一个写数据交错深度，来表示接口是否可以接收来自源设备的、具有不同 **AWID** 值的交错写数据。写数据交错深度是被静态配置的。任何接口的默认写数据交错深度为 1。

注意：不允许对具有相同 AWID 的不同事物的写数据进行交错。

写数据交错的深度为：从机接口中当前正在等待的、可以为从机提供写数据的不同地址的数目。例如，一个写数据交错深度为 2 的丛机，其接口中有 4 个具有不同 **AWID** 值的不同地址正在等待，则该接口当前可以接收前两个地址的数据。

从机接收每个事物的第一个数据的顺序必须和其接收事物地址的顺序一致。

当互联(interconnect)将多个写数据流组合在一起发送到同一个从机时，写数据交错可以防止传输中断。例如，一个互联(interconnect)可以将一个来自较慢的源设备的写数据流，和另一个来自较快的源设备的写数据流组合在一起。互联(interconnect)通过组合两种写数据流，可以提高系统性能。

注意：

如果两个具有不同 **AWID** 值的写事物访问了相同的、或者重叠的地址，这种情况下，其处理顺序没有定义。一个更高级别的协议必须确保事物处理的正确顺序。

一个主机接口，如果其只能产生只具有一个 **AWID** 值的写数据，则必须按照其发送地址的顺序来产生所有的写数据。但是，如果从机接口的写数据交错深度大于 1，则主机接口可以交错具有不同 **WID** 值的写数据。

对于大部分可以内部控制写数据产生的主机，写数据交错功能是不需要的。这种主机可以按照其产生地址的顺序来产生写数据。但是，一个以不同的速度传送来自多个源的写数据的主机接口，可以交错源来最大限度地使用互联。

为了避免死锁的发生，只有当一个从机接口可以连续接收交错的写数据时，该从机接口才能具有一个大于 1 的写数据交错深度。从机接口绝不能停止接收写数据而尝试改变写数据的顺序。

8.6 读和写交互

读和写事物之间没有顺序限制，可以按任何顺序完成。

如果主机请求在一个读事物和一个写事物之间具有给定的关系，则主机必须保证在发送后一个事物之前，前一个事物已经完成。对于一个读，当上一个读数据被返回给主机时，可以认为前一个事物完成。对于写，只有当主机接收到写响应时，才能认为事物完成，而不能认为所有写数据都被发送了，就认为写事物完成了。

对于外设的地址空间，这通常意味着，当在要求顺序约束的读和写事物之间切换时，需要等待前一个事物完成。

对于 memory 区域，主机可以实现一个检查事物地址的功能，来判定一个新的事物是访问相同的地址空间，还是访问重叠的地址空间。如果事物没有重叠，那么主机可以开始一个新的事物，而不用等待前一个事物完成。

8.7 ID 字段的互联使用

当一个主机被连接到一个互联(interconnect)时，互联会将对该主机端口唯一的额外 bit 附加到 **ARID**、**AWID** 以及 **WID** 字段，这样有两个效果：

- 主机不需要知道其他主机使用什么 ID 值，因为互联(interconnect)通过将主机号加到 ID 字段，来将每个主机使用的 ID 值设为唯一值。
- 从机接口的 ID 字段要宽于主机接口的 ID 字段。

对于读数据，互联使用额外的 **RID** 标识位来判定读数据是去往哪个主机端口的。互联将 **RID** 值传送到正确的主机端口之前，会移除这些 **RID** 标识位。

8.8 推荐的 ID 字段宽度

为了使用 AXI 乱序事物的功能，可使用以下推荐：

- 在主机组件中实现一个最大为 4-bits 的事物 ID
- 实现相对于互联中主机端口号，最大为附加 4-bits 的事物 ID

注意：

对于仅支持一个单一排序接口的主机，可以将事物 ID 字段输出 tie 成一个固定值，例如，tie 成 0。

对于一个不使用排序信息、而会简单地按顺序处理所有事物的从机，可以使用一个标准现成的模块来将 ID 功能添加到从机，因此可以设计一个不带 ID 功能、只具有基本功能的从机。

9. 数据总线

本章讲述在 AXI 读写数据总线上不同大小的传输，以及接口怎么使用字节不变的印第安序(byte-invariant endianness)来处理混合印第安序(mixed-endian)的传输。包含以下章节：

- *关于数据总线*
- *写选通*
- *窄传输*
- *字节不变*

9.1 关于数据总线

AXI 协议有两个独立的数据总线，一个用于读数据，另一个用于写数据。因为这两个数据总线都有各自独立的握手信号，因此有可能数据传输会同时发生在两个总线上。

每个由主机产生的传输必须和数据总线宽度相同，或者窄于数据总线宽度。

9.2 写选通

写选通信号 **WSTRB** 允许在写数据总线上进行稀疏数据传输。每个写选通信号对应写数据总线上的一个字节。当写选通断言时，表示写数据总线上对应的字节通道中包含将被更新到 memory 的有效信息。

写数据总线上每 8 位具有一个写选通位，因此 **WSTRB[n]**对应 **WDATA[(8 x n) + 7 : (8 x n)]**。图 9-1 展示了一个 64-bit 数据总线上的这种关系。

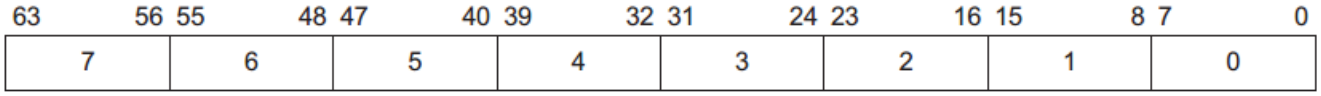


图 9-1 字节通道映射

主机必须确保只有包含有效数据的字节通道，其对应的写选通信号才会被断言，有效数据由事物的控制信息决定。

9.3 窄传输

当主机产生的一个传输，其数据宽度小于数据总线宽度时，由地址和控制信息决定传输使用哪个字节通道。在增量或回环突发中，每拍(beat)使用不同的字节通道。在固定长度突发中，地址保持不变，使用的字节通道也同样保持不变。

图 9-2 和图 9-3 给出了字节通道使用的两个例子。

在图 9-2 中：

- 该突发具有 5 个传输
- 起始地址为 0
- 每个传输为 8-bits
- 传输是在一个 32-bit 总线上

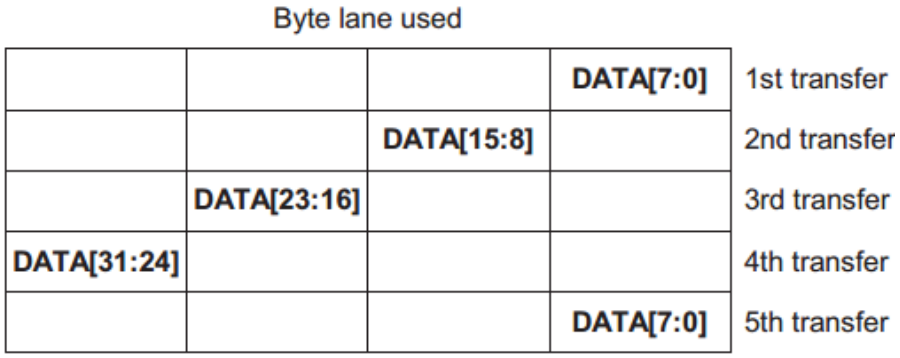


图 9-2 8-bit 传输的窄传输例子

在图 9-3 中：

- 该突发具有 3 个传输
- 起始地址为 4
- 每个传输为 32-bits
- 传输是在一个 64-bit 总线上

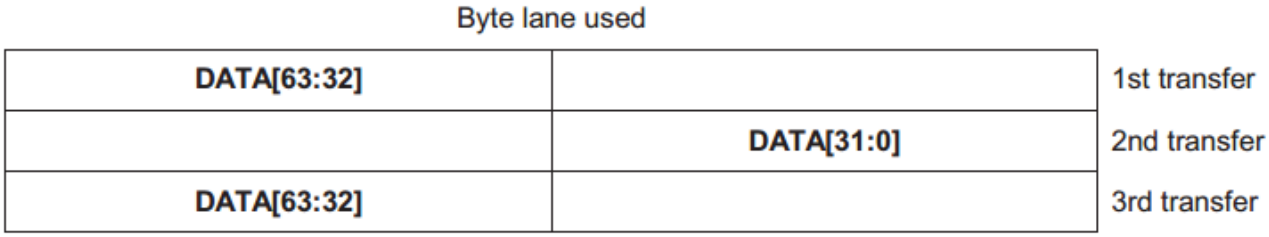


图 9-3 32-bit 传输的窄传输例子

9.4 字节不变

为了访问位于相同 memory 空间中的混合印第安序(mixed-endian)的数据结构，AXI 协议使用一种字节不变的印第安序(byte-invariant endian)的方式。

字节不变的印第安序的意思是：一个传输到给定地址的字节，会在相同的数据总线信号线上，向相同的地址位置传送一个数据的 8-bits。

只具有一种传输宽度的组件必须将其字节通道连接到数据总线中合适的字节通道上。支持多种传输宽度的组件可以具有更复杂的接口来转换非 byte-invariant 的接口。

大部分小端(little-endian)模式的组件可以直接连接到一个 byte-invariant 接口上。仅支持大端(big-endian)传输的组件则要有一个用于 byte-invariant 操作的转换功能。

图 9-4 是一个要求字节不变(byte-invariant)访问的数据结构的例子。有可能 header 信息，比如源和目标标识符，是小端格式，而 payload 是一个大端字节流。

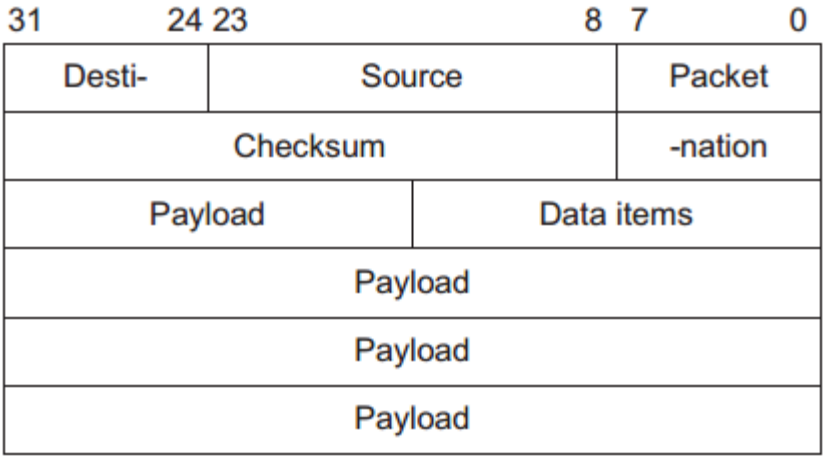


图 9-4 混合印第安数据结构的例子

字节不变(byte-invariant)方式确保小端访问 header 信息不会损坏结构中的其他大端数据。

10. 非对齐传输

本章讲述了 AXI 协议是怎样处理非对齐传输的。包含以下章节：

- *关于非对齐传输*
- *例子*

10.1 关于非对齐传输

AXI 协议使用基于突发的寻址，意思是，每个事物都包含若干个数据传输。通常来说，每个数据传输都会对齐到传输大小。例如，一个 32-bit 宽的传输通常会对齐到 4-byte 边界。但是在某些时候，会期望在非对齐的地址开始一个突发。

对于任何一个由大于 1 byte 宽度的数据传输组成的突发，其第一个 byte 可能不会与自然地址边界对其。例如，起始字节地址为 0x1002 的一个 32-bit 的数据包，就没有与自然的 32-bit 地址边界对齐。

AXI 协议允许一个主机使用低位地址线来发送突发中非对齐的起始地址。低位地址线上的信息必须和字节通道选通的信息一致。

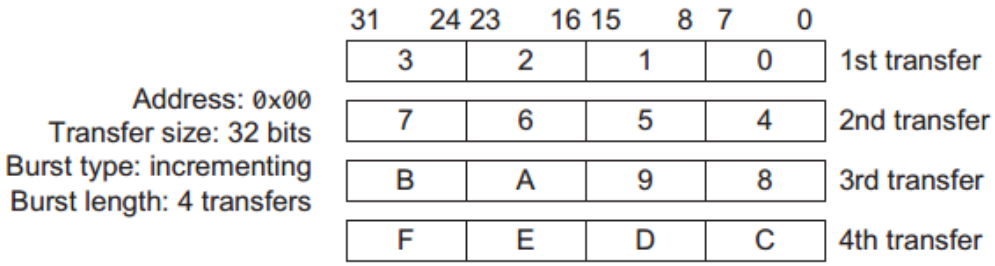
注意：

AXI 协议不要求从机根据从主机上来的任何对齐信息来采取特殊的操作。

主机也可以简单地提供一个对齐的地址，并且在一个写事物中，根据字节通道选通来提供关于数据使用哪个字节通道的信息。

10.2 例子

图 10-1，图 10-2 以及图 10-3 展示了不同宽度的总线上对齐和非对齐的传输。途中每行代表一个传输。深色方格表示根据地址和控制信息，不会被传输的字节。



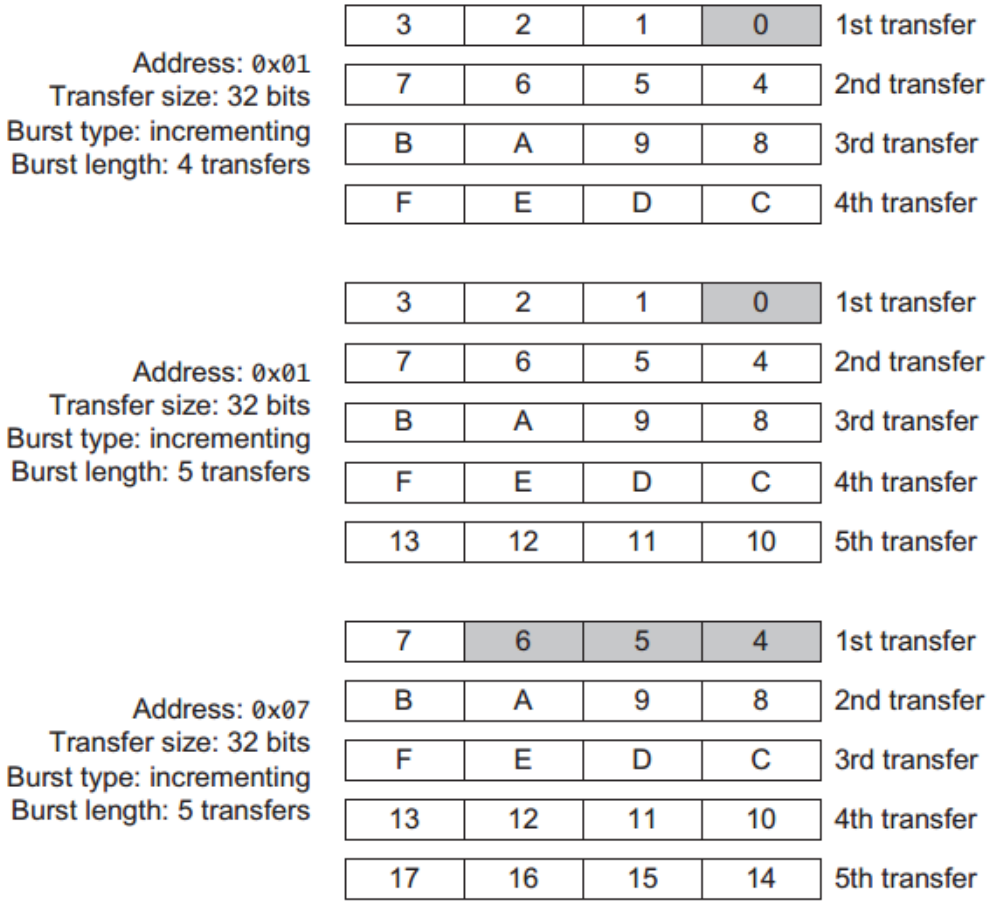


图 10-1 在 32-bit 总线上对齐和非对亲的字传输

图 10-2 展示了一个 64-bit 总线上 32-bit 传输的 3 个突发。

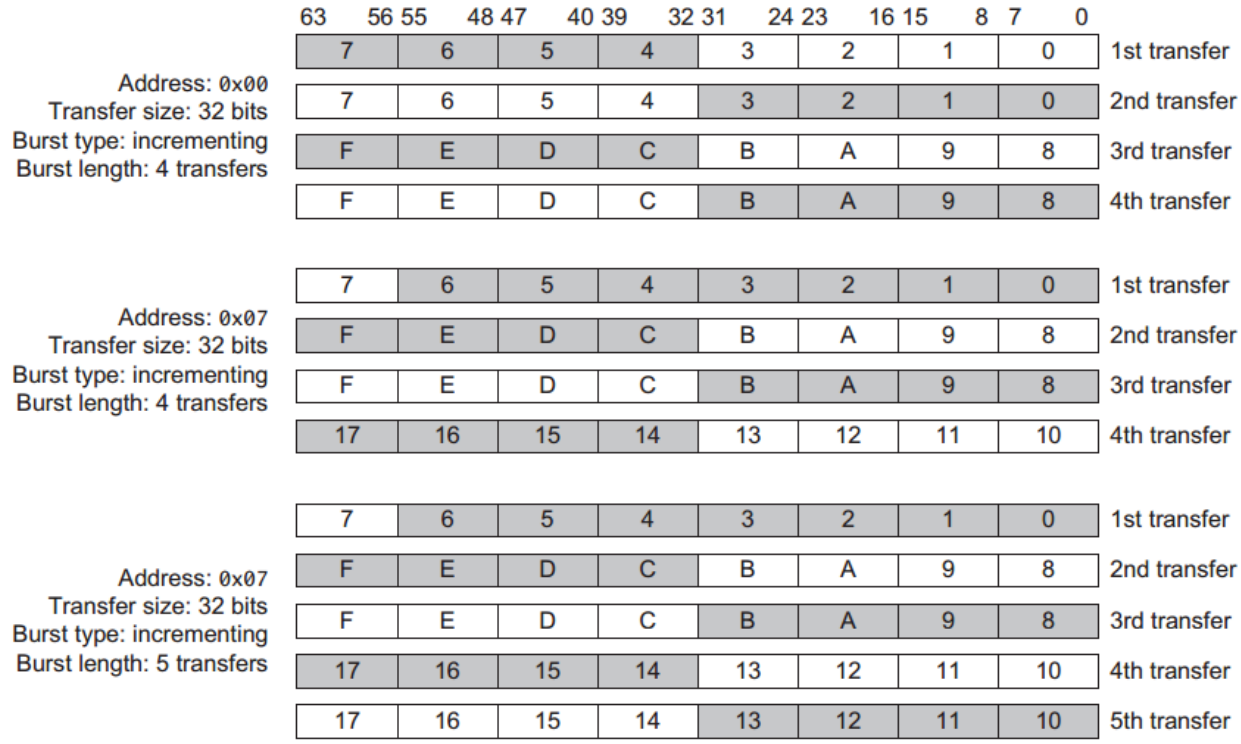


图 10-2　64-bit 总线上对齐和非对齐的 32-bit 传输

图 10-3 展示了一个 64-bit 总线上 32-bit 传输的回环突发。

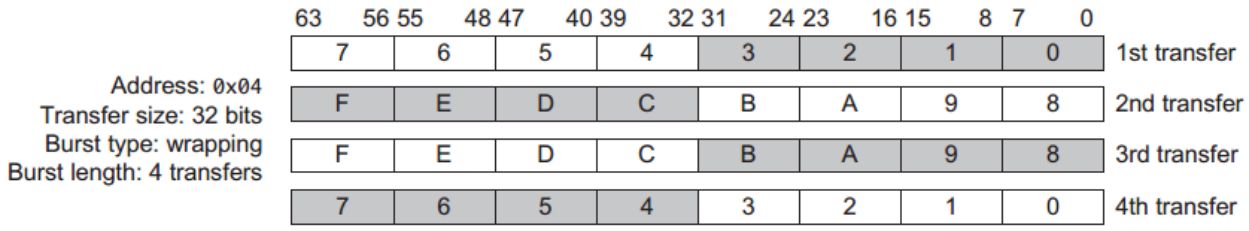


图 10-3　64-bit 总线上对齐的回环字传输

11. 时钟和复位

本章讲述 AXI 时钟和复位信号的时序。包含以下章节：

- *时钟和复位要求*

11.1 时钟和复位要求

本章节给出开发 **ACLK** 和 **ARESETn** 信号的要求。

11.1.1 时钟

每个 AXI 组件使用一个单一的时钟信号——**ACLK**。所有输入信号都在 **ACLK** 的上升沿采样。所有输出信号的改变都必须发生在 **ACLK** 上升沿之后。

主机和从机接口的输入和输出信号之间不能存在组合路径。

11.1.2 复位

AXI 协议包含一个单一的低有效的 reset 信号——**ARESETn**。复位信号可以被异步地断言，但是取消断言必须在 **ACLK** 上升沿之后同步地发生。

在 reset 期间，以下接口要求要满足：

- 主机接口必须驱动 **ARVALID**，**AWVALID** 以及 **WVALID** 为低
- 从机接口必须驱动 **RVALID** 和 **BVALID** 为低

其它素有信号可以被驱动为任何值。

主机只能在 **ARESETn** 为高之后的一个 **ACLK** 上升沿驱动 **ARVALID**，**AWVALID** 或 **WVALID** 为高。图 11-1 展示了 reset 之后 **ARVALID**、**AWVALID** 或 **WVALID** 可以被驱动为高的第一个点。

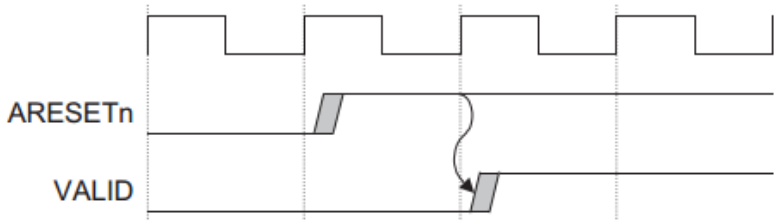


图 11-1 退出 reset

12. 低功耗接口

本章讲述了 AXI 协议在进入及退出低功耗状态时的时钟控制接口。包含以下章节：

- *关于低功耗接口*
- *低功耗时钟控制*

12.1 关于低功耗接口

低功耗接口是数据传输协议的可选扩展，这种数据传输协议目标是两类不同的外设：

- 请求一个断电序列，并且只有当进入一个低功耗状态时才能关闭其时钟的外设。这种外设从一个系统时钟控制器请求一个标识，来判定何时启动一个断电序列。
- 不具有断电序列，并且其可以独立标示何时可以关闭其时钟的外设。

12.2 低功耗时钟控制

低功耗时钟控制接口包含以下信号：

- 一个来自外设的信号，表示其时钟何时可以被使能或禁止
- 两个握手信号，用于系统时钟控制器请求退出或进入一个低功耗状态

时钟控制接口中的主要信号是 **CACTIVE**。外设使用该信号来表示其何时请求时钟被使能。外设断言 **CACTIVE** 来表示请求时钟，并且系统时钟控制器必须立即使能时钟。外设取消断言 **CACTIVE** 来表示没有请求时钟。之后，系统时钟控制器可以判定是否使能或禁止外设时钟。

可以在任何时间使能活禁止其时钟的外设可以驱动 **CACTIVE** 永久为低。必须始终使能其时钟的外设必须驱动 **CACTIVE** 永久为高。

这种简单的系统时钟控制器接口对于一些不需要断电或上电序列的外设已经是足够的了。

对于一个需要断电和上电序列的更复杂的外设，只有从系统时钟控制器发出一个请求之后才能进入一个低功耗状态。AXI 协议提供了一个 2 线的请求/确认握手来实现这个请求：

CSYSREQ 为了请求外设进入一个低功耗状态，系统时钟控制器驱动 **CSYSREQ** 信号为低。在正常操作期间，**CSYSREQ** 为高。

CSYSACK 外设使用 **CSYSACK** 信号来确认低功耗状态请求和退出低功耗状态。

图 12-1 展示了 **CSYSREQ** 和 **CSYSACK** 之间的关系。

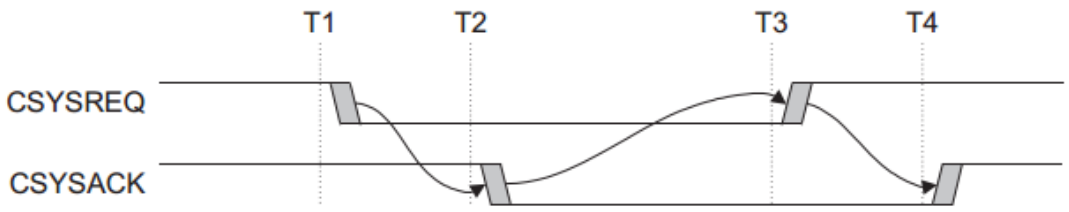


图 12-1 CSYSREQ 和 CSYSACK 握手

图 12-1 中，在序列的开始，**CSYSREQ** 和 **CSYSACK** 都为高用于正常时钟操作。在 T1 时刻，系统时钟控制器取消断言 **CSYSREQ**，表示让外设进入低功耗状态的请求。外设在 T2 时刻通过取消断言 **CSYSACK** 来确认请求。在 T3 时刻，系统时钟控制器断言 **CSYSREQ** 来表示退出低功耗状态，外设在 T4 断言 **CSYSACK** 来确认退出。

这种 **CSYSRE** 和 **CSYSACK** 之间的关系是 AXI 协议的要求。

外设可以接受或拒绝来自系统时钟控制器的低功耗请求。当外设通过取消断言 **CSYSACK** 来确认请求时，**CACTIVE** 信号的电平表明外设接受或拒绝请求。

12.2.1 接受低功耗请求

图 12-2 展示了当外设接受了一个系统低功耗请求时的事件序列。

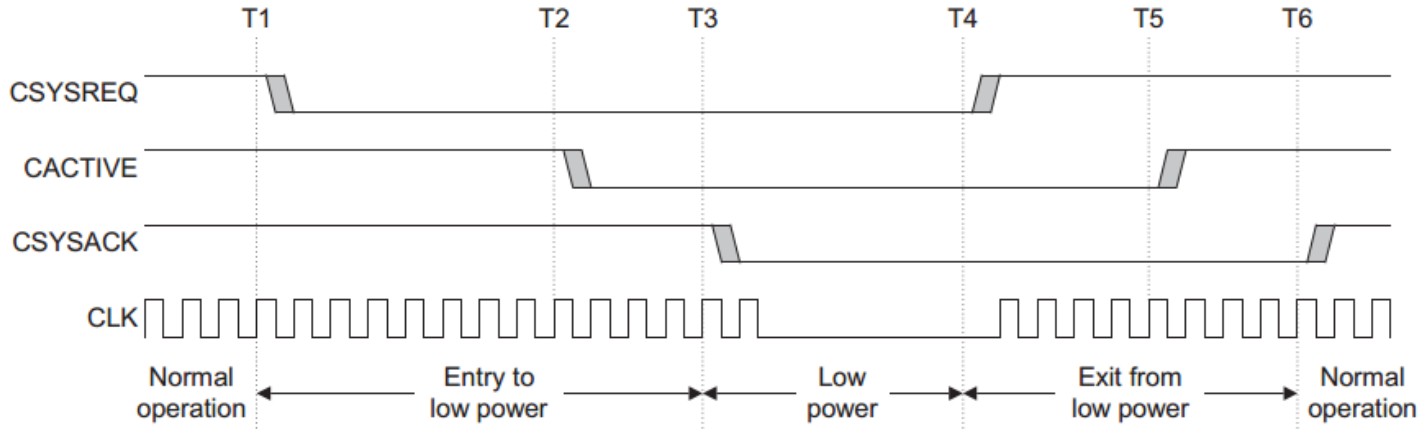


图 12-2 接受低功耗请求

在图 12-1 中，序列始于 T1 时刻，当系统时钟控制器取消断言 **CSYSREQ** 来外设进入低功耗状态时。外设识别到请求之后，就可以开始执行其断电功能，并取消断言 **CACTIVE**。之后，外设在 T3 时刻取消断言 **CSYSACK** 来完成进入低功耗状态的过程。

在 T4 时刻，系统时钟控制器通过断言 **CSYSREQ** 来开始低功耗状态退出序列。之后，外设在 T5 时刻断言 **CACTIVE** 并在 T6 时刻通过断言 **CSYSACK** 来完成退出序列。

12.2.2 拒绝一个低功耗请求

图 12-3 展示了当一个外设拒绝系统低功耗请求时的事件序列。

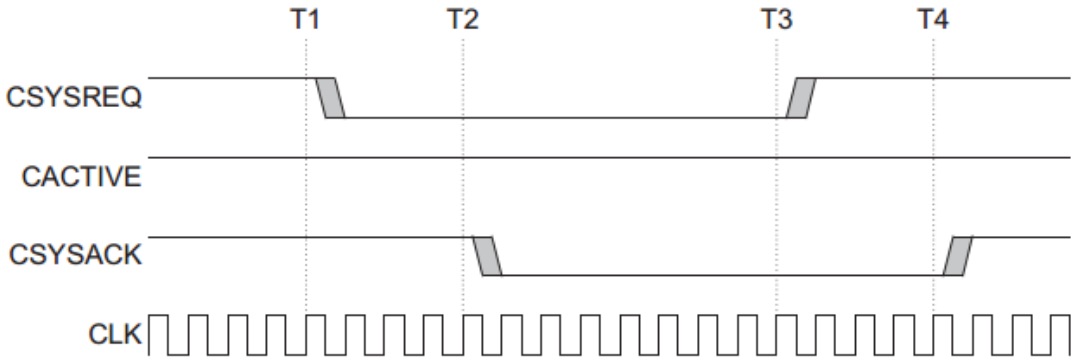


图 12-3 拒绝低功耗请求

在图 12-3 中，外设在确认低功耗请求时，通过保持 **CACTIVE** 为高来拒绝一个低功耗请求。在该点之后，系统时钟控制器在可以开始另一个请求之前，必须通过断言 **CSYSREQ** 来结束低功耗请求。

12.2.3 退出低功耗状态

系统时钟控制器和外设都可以请求退出低功耗状态并重新启动时钟。在低功耗期间，**CACTIVE** 和 **CSYSREQ** 信号都为低，通过驱动这两个信号中的一个为高来启动退出序列。

系统时钟控制器可以通过使能时钟并驱动 **CSYSREQ** 为高来开始退出低功耗状态。之后，外设可以执行一个上电序列，在该上电序列中会驱动 **CACTIVE** 为高。然后通过驱动 **CSYSACK** 为高来完成退出。

外设可以通过驱动 **CACTIVE** 为高来开始退出低功耗状态。之后，系统时钟控制器必须立即恢复时钟。系统时钟控制器还必须驱动 **CSYSREQ** 为高来继续握手序列。然后当退出低功耗时，外设通过驱动 **CSYSACK** 为高来结束序列。外设可以将 **CSYSACK** 为低为任意请求的周期，以完成退出序列。

12.2.4 时钟控制序列总结

图 12-4 展示了进入和退出低功耗状态的典型流程。

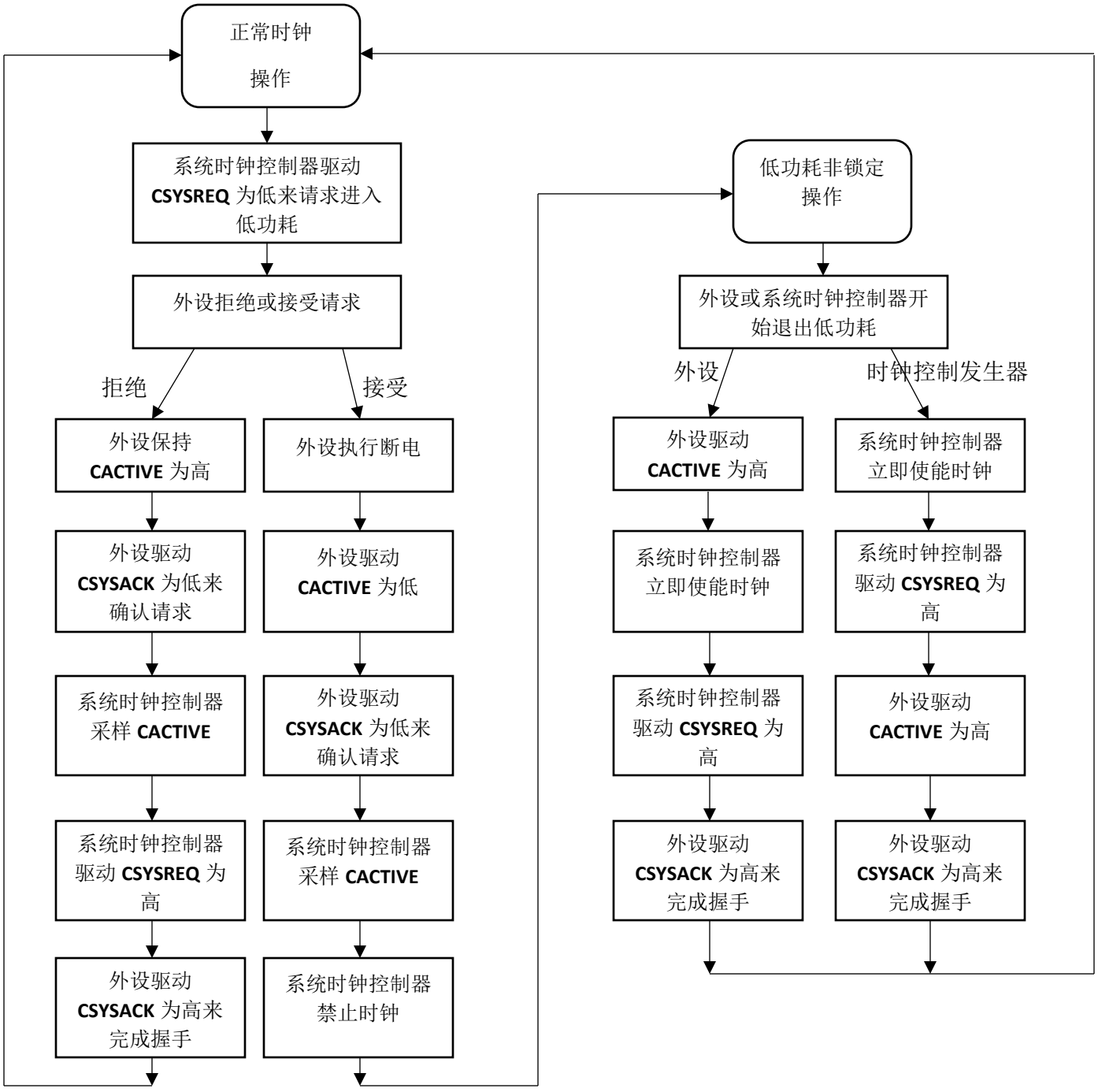


图 12-4 低功耗时钟控制序列

12.2.5 低功耗域中的混合外设

系统时钟控制器可以在同一个低功耗时钟域中混合若干个不同的外设。如果遵循以下规则，则时钟域可以以相同的方式被当作一个单一的外设对待：

- 时钟域 CACTIVE 信号是该时钟域中所有 CACTIVE 信号的逻辑或。这意味着只有当所有外设都表明其可以被 disable 时，系统时钟控制器才能 disable 时钟。
- 系统时钟控制器可以使用一个单一的 CSYSREQ 信号，该信号被路由到时钟域中的所有外设。
- 时钟域 CSYSACK 信号的产生如下：
  - CSYSACK 的下降沿发生在当来自所有外设的最后一个下降沿发生时
  - CSYSACK 的上升沿发生在当来自所有外设的最后一个下降沿发生时