AXI 总线协议资料整理

第一部分：

1、AXI 简介：AXI （Advanced eXtensible Interface）是一种总线协议，该协议是 ARM 公司提出的 AMBA （Advanced Microcontroller Bus Architecture）3.0 协议中最重要的部分，是一种面向高性能、高带宽、低延迟的片内总线。它的地址 /控制和数据相位是分离的， 支持不对齐的数据传输， 同时在突发传输中， 只需要首地址，同时分离的读写数据通道、 并支持显著传输访问和乱序访问， 并更加容易就行时序收敛。 AXI 是 AMBA 中一个新的高性能协议。 AXI 技术丰富了现有的 AMBA 标准内容，满足超高性能和复杂的片上系统（ SoC）设计的需求。

2、 AXI 特点： 单向通道体系结构。信息流只以单方向传输，简化时钟域间的桥接，减少门数量。当信号经过复杂的片上系统时，减少延时。

支持多项数据交换。通过并行执行猝发操作，极大地提高了数据吞吐能力，可在更短的时间内完成任务，在满足高性能要求的同时，又减少了功耗。

独立的地址和数据通道。 地址和数据通道分开， 能对每一个通道进行单独优化，可以根据需要控制时序通道，将时钟频率提到最高，并将延时降到最低。

第二部分：

本部分对 AXI1.0 协议的各章进行整理。

第一章

本章主要介绍 AXI 协议和 AXI 协议定义的基础事务。

1、 AXI 总线共有 5 个通道分别是 read address channel 、write address channel、 read data channel、 write data channel、 write response channel。每一个 AXI 传输通道都是单方向的。

2、 每一个事务都有地址和控制信息在地址通道（ address channel）中，用来描述被传输数据的性质。

3、 读事务的结构图如下：

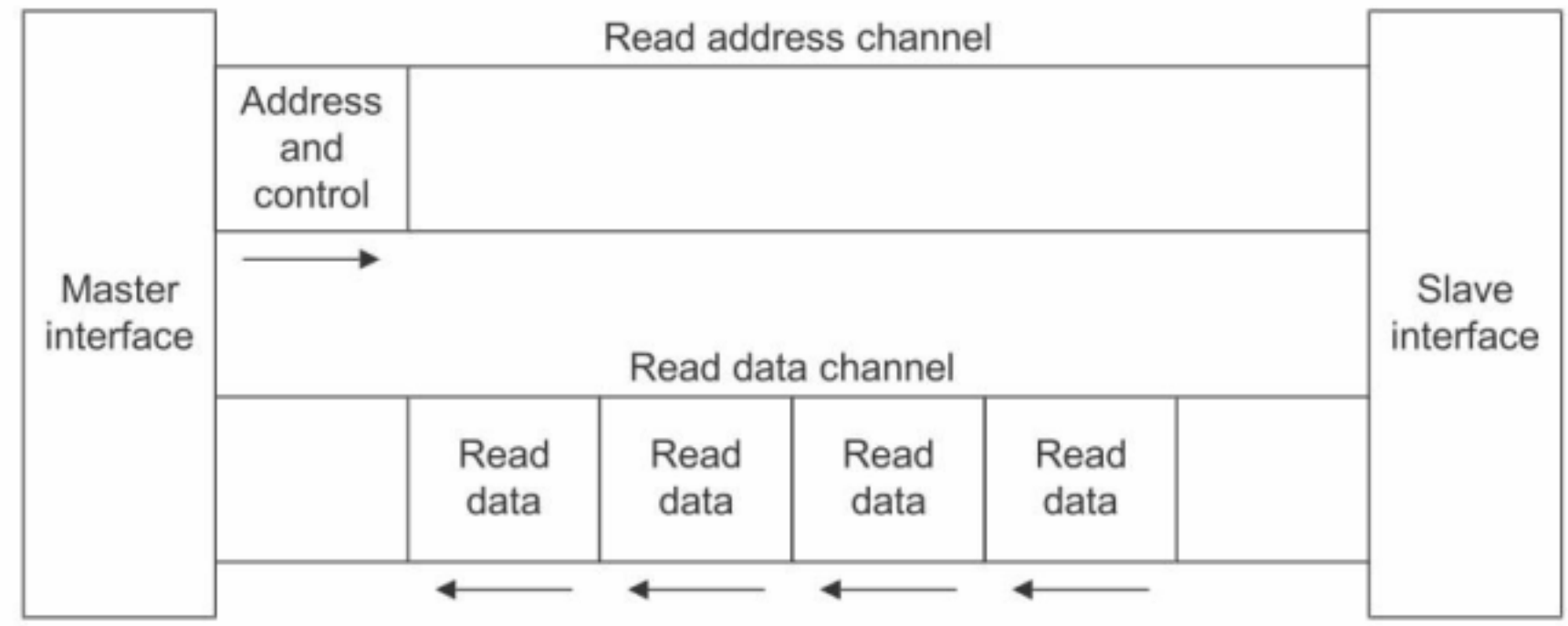


Figure 1-1 Channel architecture of reads

4、 写事务的结构图如下：

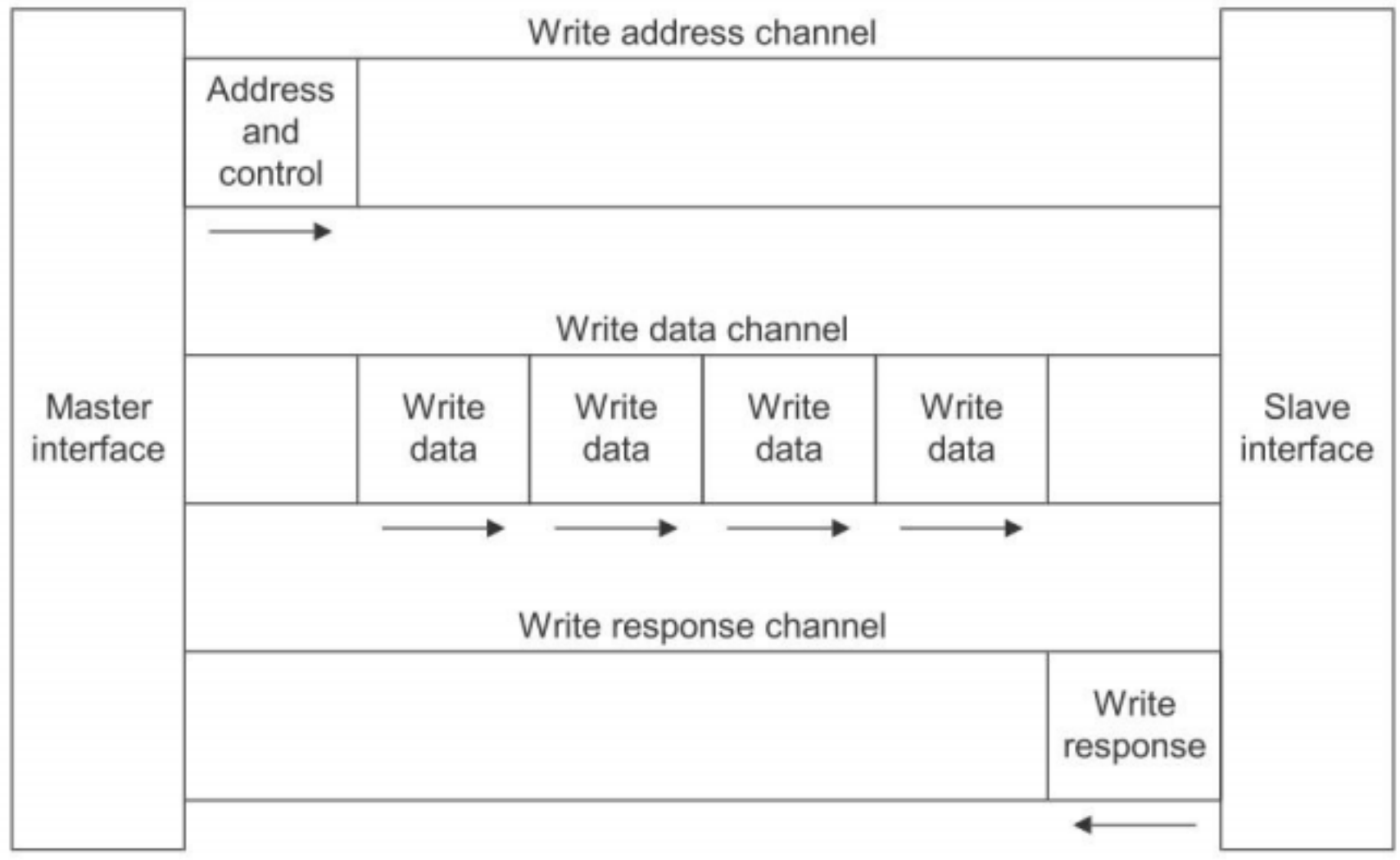


Figure 1-2 Channel architecture of writes

5、 这 5 条独立的通道都包含一个信息信号和一个双路的 VALD 、READY 握手机制。

6、 信息源通过 VALID 信号来指示通道中的数据和控制信息什么时候有效。目地源用 READY 信号来表示何时能够接收数据。 读数据和写数据通道都包括一个 LAST 信号，用来指明一个事物传输的最后一个数据。

7、 读和写事务都有他们自己的地址通道， 这地址通道携带着传输事务所必须的地址和信息。

8、 读数据通道传送着从设备到主机的读数据和读响应信息。 读响应信息指明读事务的完成状态。

9、 写数据通路传送着主机向设备的写数据。每八个数据都会有一个 byte lane，用来指明数据总线上面的哪些 byte 有效。写响应通道提供了设备响应写事务的一种方式。这完成信号每一次突发式读写会产生一个。

10、 主机和设备的接口和互联图如下：

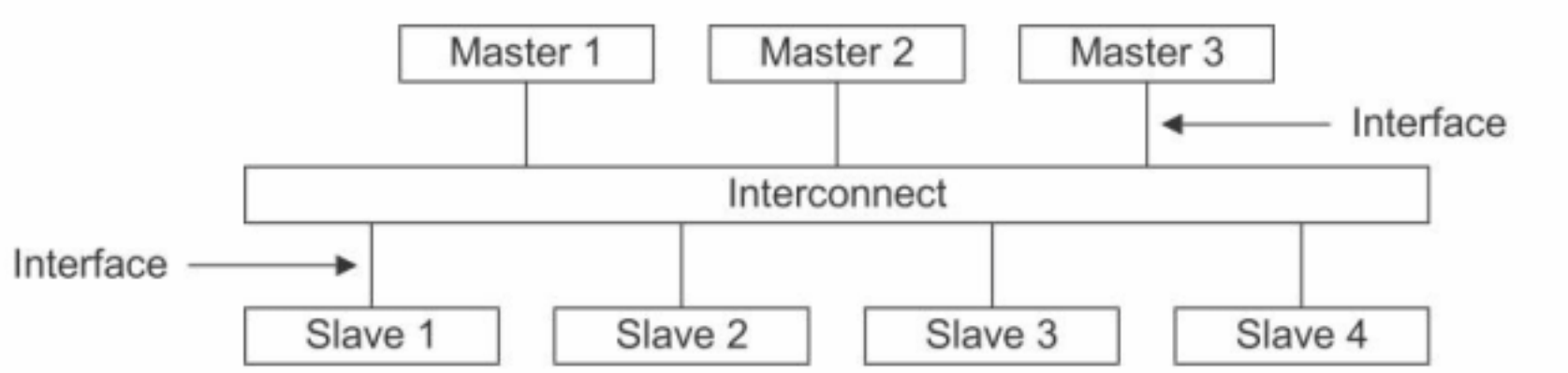


Figure 1-3 Interface and interconnect

11、 传输地址信息和数据都是在 VALID 和 READY 同时为高时有效。

Note

The master also drives a set of control signals showing the length and type of the burst, but these signals are omitted from the figure for clarity.

12、 突发式读的时序图如下：

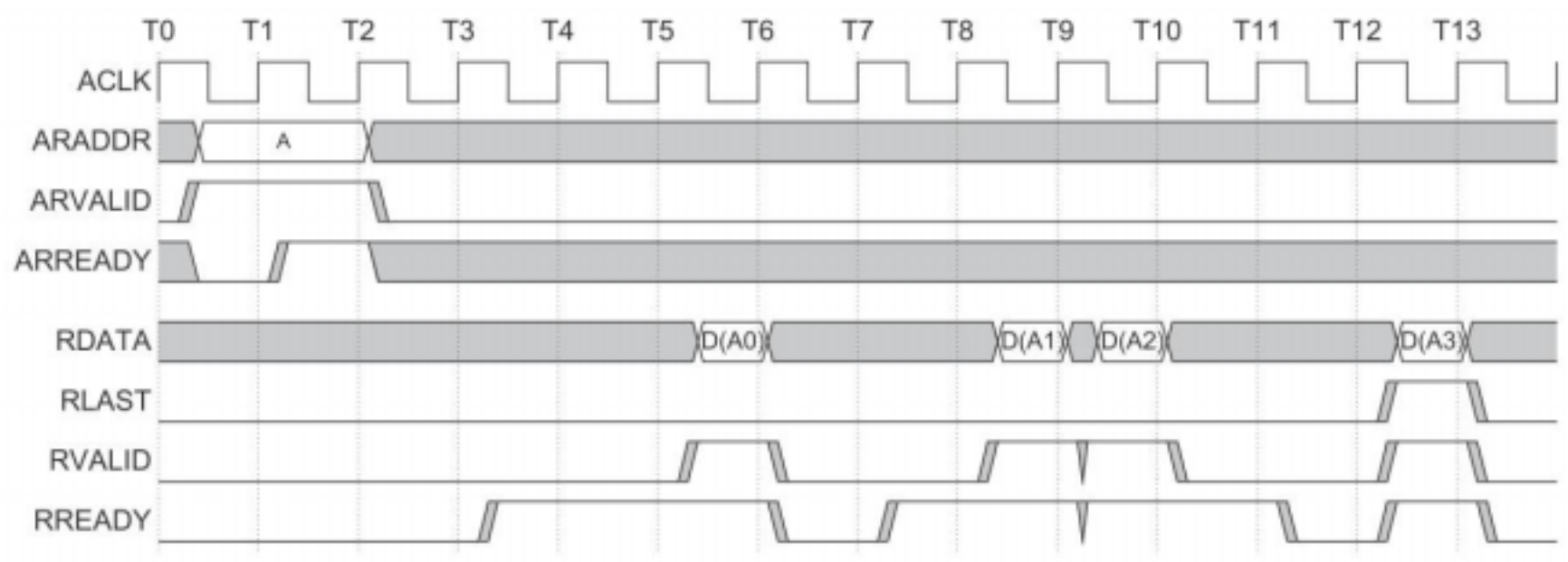


Figure 1-4 Read burst

当地址出现在地址总线后， 传输的数据将出现在读数据通道上。 设备保持 VALID 为低直到读数据有效。为了表明一次突发式读写的完成，设备用 RLAST 信号来表示最后一个被传输的数据。

13、 重叠突发式读时序图如下：

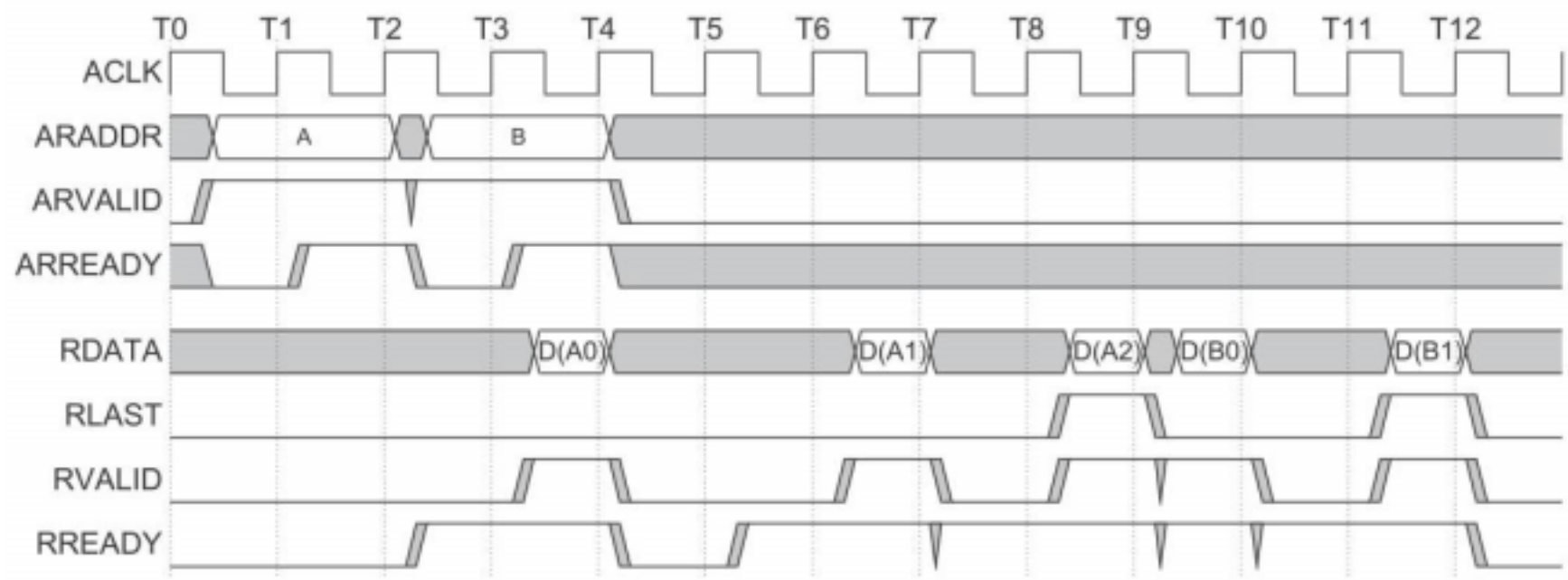


Figure 1-5 Overlapping read bursts

设备会在第一次突发式读完成后处理第二次突发式读数据。 也就意味着， 主机一开始传送了两个地址给设备。 设备在完全处理完第一个地址的数据之后才开始处理第二个地址的数据。

14、 突发式写时序图如下：

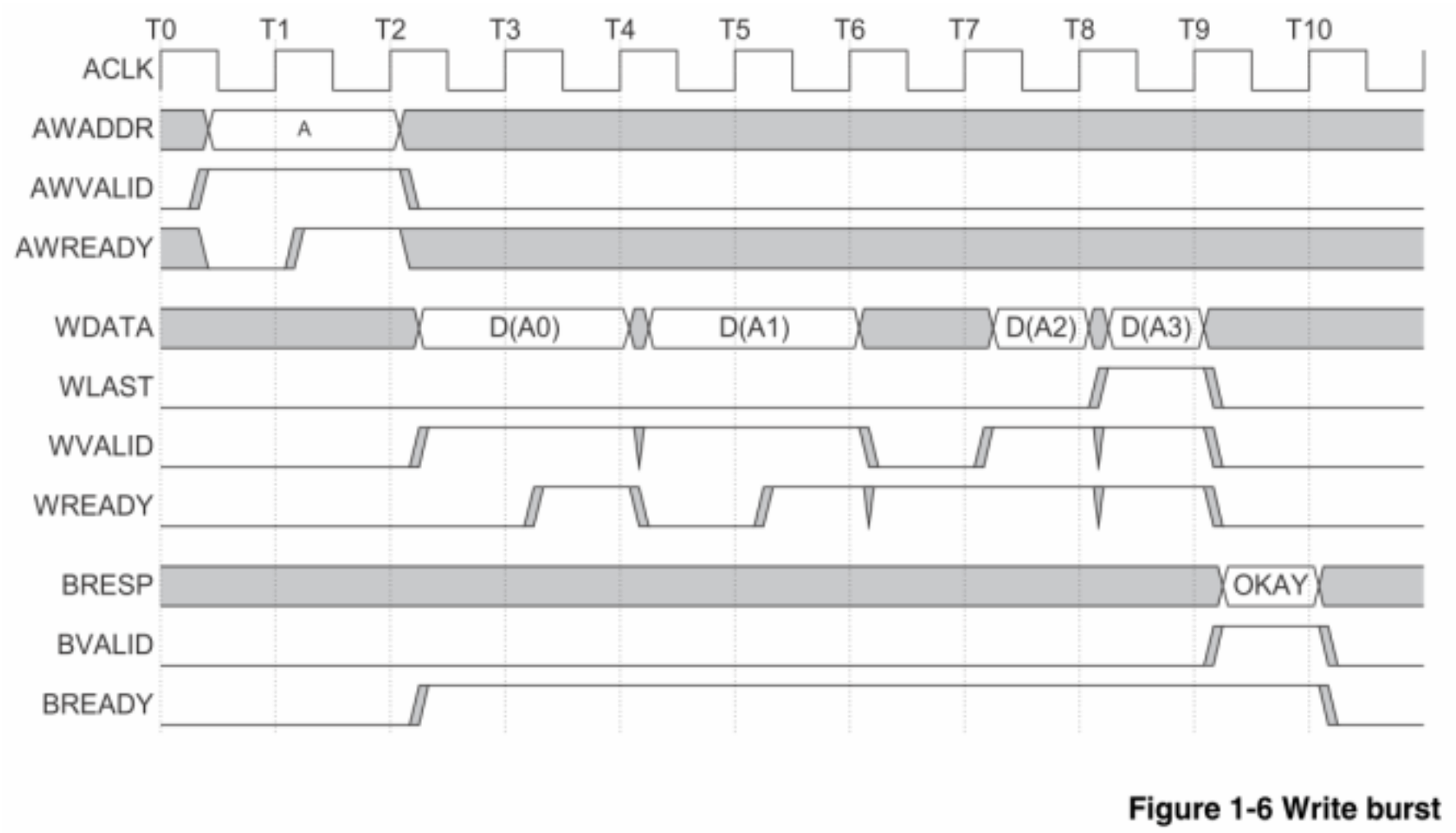


Figure 1-6 Write burst

这一过程的开始时， 主机发送地址和控制信息到写地址通道中， 然后主机发送每一个写数据到写数据通道中。当主机发送最后一个数据时， WLAST 信号就变为高。当设备接收完所有数据之后他将一个写响应发送回主机来表明写事务完成。

15、 AXI 协议支持乱序传输。他给每一个通过接口的事务一个 IDtag。协议要求相同 ID tag 的事务必须有序完成，而不同 ID tag 可以乱序完成。

第二章

本章主要介绍一些信号描述，　其中包括全局信号、 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。本章的所有表都是以　32 位的数据总线、 4 位的写数据闸门、 4 位的 ID 段。

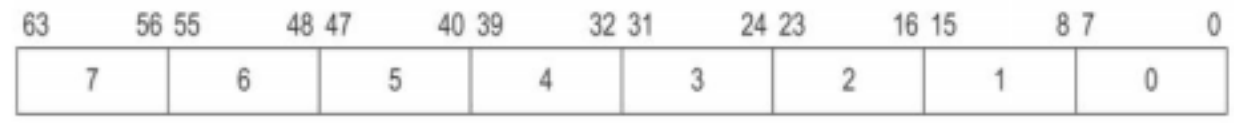
1、全局信号

信号	源	描述
ACLK	Clock source	全局时钟信号
ARESETn	Reset source	全局复位信号，低电平有效

2、写地址通道信号

信号	源	描述
AWID[3:0]	主机	写地址 ID，这个信号是写地址信号组的　ID tag。
AWADDR[31:0]	主机	写地址。
AWLEN[3:0]	主机	突发式写的长度。 此长度决定突发式写所传输的数据的个数。
AWSIZE[2:0]	主机	突发式写的大小。
AWBURST[1:0]	主机	突发式写的类型。
AWLOCK[1:0]	主机	锁类型。
AWCACHE[3:0]	主机	Cache类型。这信号指明事务的　bufferable、cacheable、write-through、 write-back、 allocate attributes信息。
AWPROT[2:0]	主机	保护类型。
AWVALID	主机	写地址有效。 1 = 地址和控制信息有效 0 = 地址和控制信息无效 这个信号会一直保持，直到　AWREADY 变为高。
AWREADY	设备	写地址准备好。这个信号用来指明设备已经准备好接受地址和控制信息了。 1 = 设备准备好 0 =　设备没准备好

3、写数据通道信号

信号	源	描述
WID[3:0]	主机	写 ID tag，WID 的值必须与 AWID 的值匹配
WDATA[31:0]	主机	写的数据。
WSTRB[3:0]	主机	写阀门。WSTRB[n] 标示的区间为 WDATA[(8*n)+7:(8*n)]  Figure 9-1 Byte lane mapping
WLAST	主机	写的最后一个数据。
WVALID	主机	写有效 1 = 写数据和阀门有效 0 = 写数据和阀门无效
WREADY	设备	写就绪。指明设备已经准备好接受数据了 1 = 设备就绪 0 = 设备未就绪

4、写响应通道信号

信号	源	描述
BID[3:0]	设备	响应 ID ， 这个数值必须与 AWID 的数值匹配。
BRESP[1:0]	设备	写响应。这个信号指明写事务的状态。可能有的响应： OKAY 、EXOKAY 、SLVERR、DECERR。
BVALID	设备	写响应有效。 1 = 写响应有效 0 = 写响应无效
BREADY	主机	接受响应就绪。该信号表示主机已经能够接受响应信息。 1 = 主机就绪 0 = 主机未就绪

5、读地址通道信号

信号	源	描述
ARID[3:0]	主机	读地址 ID。
ARADDR[31:0]	主机	读地址。
ARLEN[3:0]	主机	突发式读长度。
ARSIZE[2:0]	主机	突发式读大小。
ARBURST[1:0]	主机	突发式读类型。
ARLOCK[1:0]	主机	锁类型。
ARCACHE[3:0]	主机	Cache类型。
ARPROT[2:0]	主机	保护类型。
ARVALID	主机	读地址有效。信号一直保持，直到 ARREADY 为高。 1 = 地址和控制信息有效 0 = 地址和控制信息无效
ARREADY	设备	读地址就绪。指明设备已经准备好接受数据了。 1 = 设备就绪 0 = 设备未就绪

6、读数据通道信号

信号	源	描述
RID[3:0]	设备	读 ID tag。RID 的数值必须与 ARID 的数值匹配。
RDATA[31:0]	设备	读数据。
RRESP[1:0]	设备	读响应。这个信号指明读传输的状态： OKAY 、 EXOKAY 、SLVERR、DECERR。
RLAST	设备	读事务传送的最后一个数据。
RVALID	设备	读数据有效。 1 = 读数据有效。 0 = 读数据无效。

RREADY	主机	读数据就绪。 1 = 主机就绪 0 = 主机未就绪

7、低功耗接口信号

信号	源	描述
CSYSREQ	CLOCK controller	系统低功耗请求。此信号来自系统时钟控制器，使外围设备进入低功耗状态。
CSYSACK	外围设备	低功耗请求应答。
CACTIVE	外围设备	Clock active 1 = 外围设备时钟请求 0 = 外围设备时钟无请求

第三章

本章介绍主机 /设备之间的握手过程以及 READY 和 VALD 握手信号的关系以及默认值。

1、全部 5 个通道使用相同的 VALID/READY 握手机制传输数据及控制信息。传输源产生 VLAID 信号来指明何时数据或控制信息有效。而目地源产生 READY 信号来指明已经准备好接受数据或控制信息。传输发生在 VALID 和 READY 信号同时为高的时候。 VALID 和 READY 信号的出现有三种关系。

（1） VALID 先变高 READY 后变高。时序图如下：

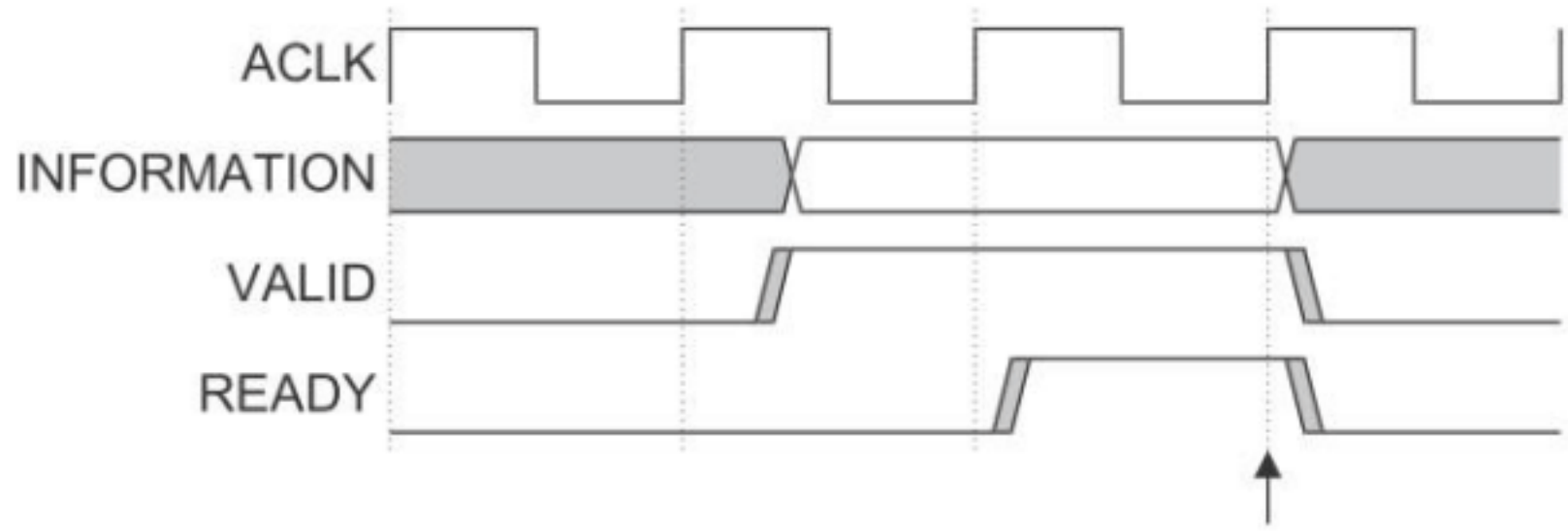
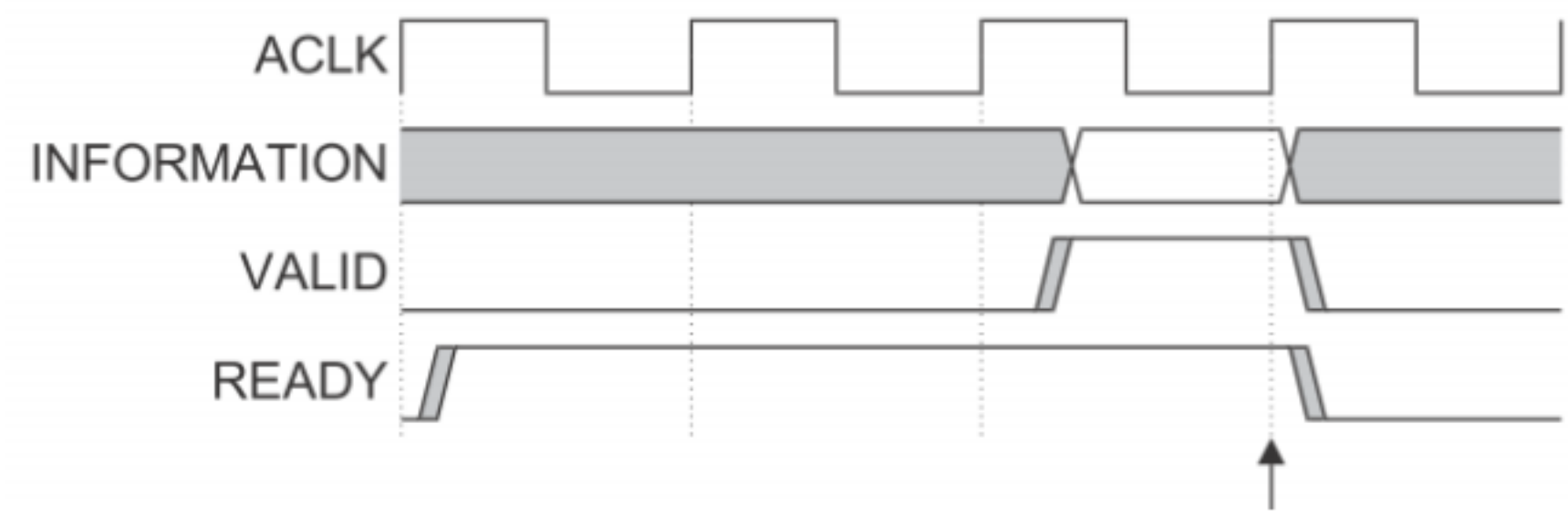


Figure 3-1 VALID before READY handshake

在箭头处信息传输发生。

（2） READY 先变高 VALID 后变高。时序图如下：



同样在箭头处信息传输发生。

（3） VALID 和 READY 信号同时变高。时序图如下：

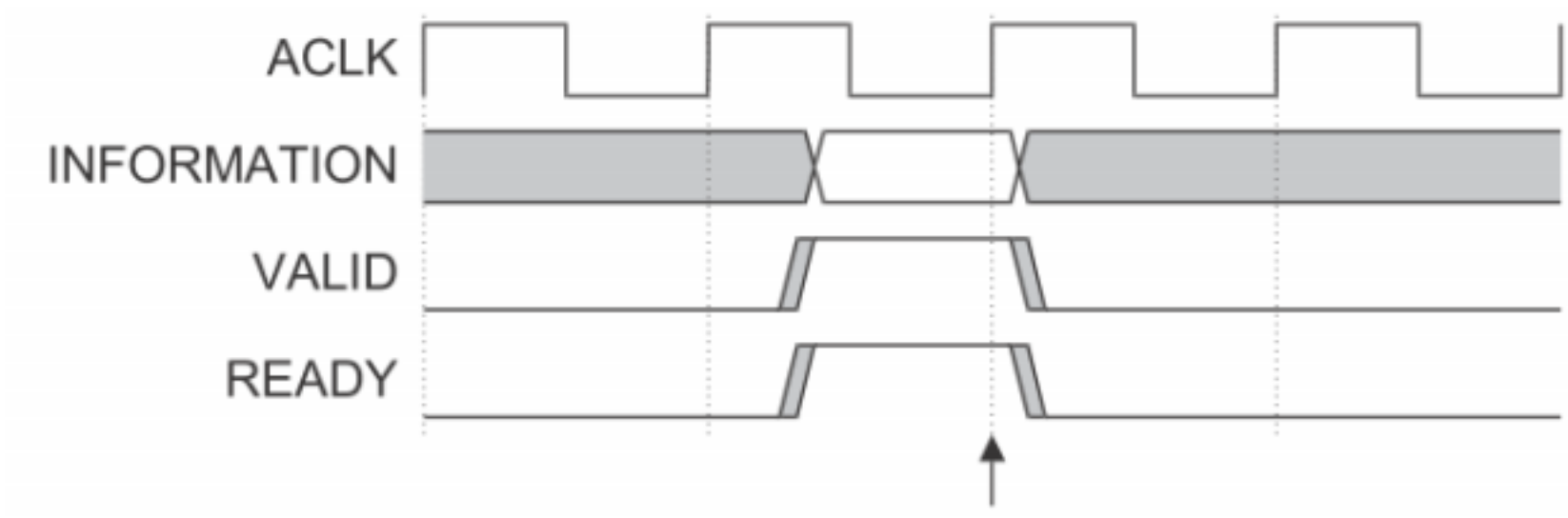


Figure 3-3 VALID with READY handshake

在这种情况下，信息传输立马发生，如图箭头处指明信息传输发生。

2、通道之间的关系

地址、读、写和写响应通道之间的关系是灵活的。

例如，写数据可以出现在接口上早于与其相关联的写地址。也有可能写数据与写地址在一个周期中出现。

两种关系必须被保持：

（1）读数据必须总是跟在与其数据相关联的地址之后。

（2）写响应必须总是跟在与其相关联的写事务的最后出现。

3、通道握手信号之间的依赖性

读事务握手依赖关系如图：

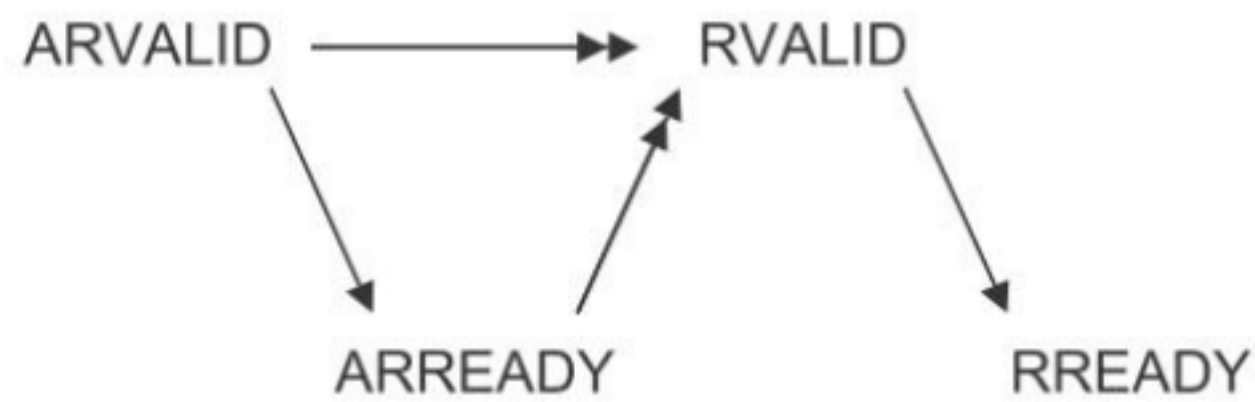


Figure 3-4 Read transaction handshake dependencies

（1）设备可以在 ARVALID 出现的时候在给出 ARREADY 信号，也可以先给出 ARREADY 信号，再等待 ARVALID 信号。

（2）但是设备必须等待 ARVALID 和 ARREADY 信号都有效才能给出 RVALID 信号，开始数据传输。

写事务握手依赖关系如图：

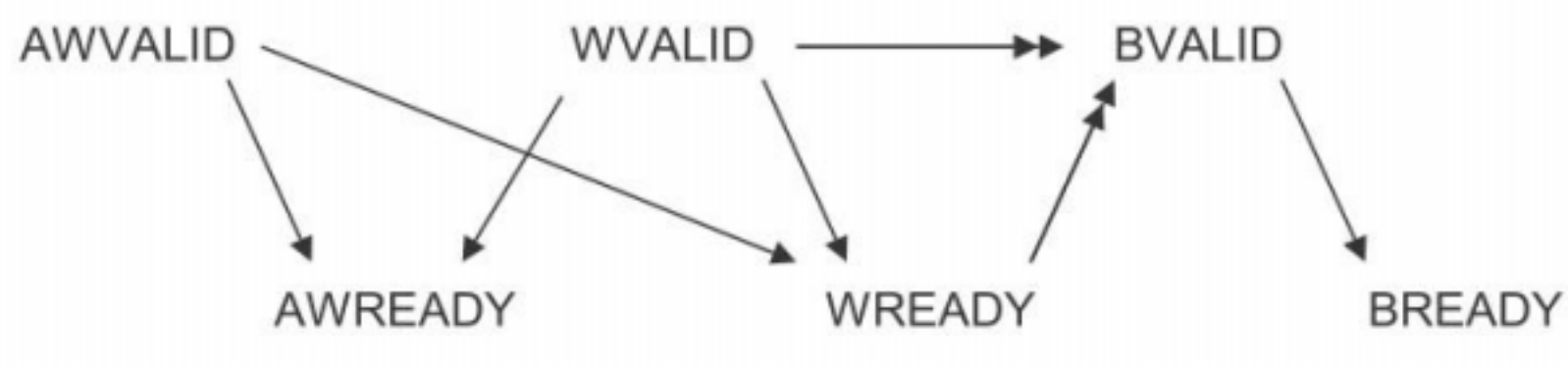


Figure 3-5 Write transaction handshake dependencies

（1）主机必须不能够等待设备先给出 AWREADY 或 WREADY 信号信号后再给出信号 AWVALID 或 WVLAID 。

（2）设备可以等待信号 AWVALID 或 WVALID 信号有效或者两个都有效之后再给出 AWREADY 信号。

（3）设备可以等待 AWVALID 或 WVALID 信号有效或者两个信号都有效之后再给出 WREADY 信号。

Note

It is important that during a write transaction, a master must not wait for **AWREADY** to be asserted before driving **WVALID**. This could cause a deadlock condition if the slave is conversely waiting for **WVALID** before asserting **AWREADY**.

第四章

本章主要介绍 AXI 突发式读写的类型和在一次突发式读写事务内如何计算地址和 byte lanes。

1、 突发式读写的地址必须以 4KB 对齐。

Bursts must not cross 4KB boundaries to prevent them from crossing boundaries between slaves and to limit the size of the address incrementer required within slaves.

2、 信号 AWLEN 或信号 ARLEN 指定每一次突发式读写所传输的数据的个数。具体信息如下图：

Table 4-1 Burst length encoding

ARLEN[3:0] AWLEN[3:0]	Number of data transfers
b0000	1
b0001	2
b0010	3
. . .	
b1101	14
b1110	15
b1111	16

3、 ARSIZE 信号或 AWSIZE 信号指定每一个时钟节拍所传输的数据的最大位数。

具体信息如下图：

Table 4-2 Burst size encoding

ARSIZE[2:0] AWSIZE[2:0]	Bytes in transfer
b000	1
b001	2
b010	4
b011	8
b100	16
b101	32
b110	64
b111	128

需要注意的是任何传输的 SIZE 都不能超过数据总线的宽度。

4、 AXI 协议定义了三种突发式读写的类型：固定式的突发读写、增值式突发读写、包装式突发读写。用信号 ARBURST 或 AWBURST 来选择突发式读写的类型。具体信息如下图：

Table 4-3 Burst type encoding

ARBURST[1:0] AWBURST[1:0]	Burst type	Description	Access
b00	FIXED	Fixed-address burst	FIFO-type
b01	INCR	Incrementing-address burst	Normal sequential memory
b10	WRAP	Incrementing-address burst that wraps to a lower address at the wrap boundary	Cache line
b11	Reserved	-	-

（1）固定式突发读写是指地址是固定的，每一次传输的地址都不变。这样的突发式读写是重复的对一个相同的位置进行存取。例如 FIFO。

（2）增值式突发读写是指每一次读写的地址都比上一次的地址增加一个固定的值。

（3）包装式突发读写跟增值式突发读写类似。包装式突发读写的地址是包数据

的低地址当到达一个包边界。

包装式突发读写有两个限制：

◇①起始地址必须以传输的 size 对齐。

◇②突发式读写的长度必须是 2、4、8 或者 16。

5、关于一些地址的计算公式。

Start_Address 主机发送的起始地址

Number_Bytes 每一次数据传输所能传输的数据 byte 的最大数量

Data_Bus_Bytes 数据总线上面 byte lanes的数量

Aligned_Address 对齐版本的起始地址

Burst_Length 一次突发式读写所传输的数据的个数

Address_N 每一次突发式读写所传输的地址数量，范围是 2-16

Wrap_Boundary 包装式突发读写的最低地址

Lower_Byte_Lane 传输的最低地址的 byte lane

Upper_Byte_Lane 传输的最高地址的 byte lane

INT(x) 对 x 进行向下取整

下面是计算公式：

Start_Address = ADDR

Number_Bytes = 2^{SIZE}

Burst_Length = LEN + 1

Aligned_Address = (INT(Start_Address / Number_Bytes)) x Number_Bytes

Address_1 = Start_Address

Address_N = Aligned_Address + (N － 1) x Number_Bytes

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

x (Number_Bytes x Burst_Length)

如果有 Address_N = Wrap_Boundary + (Number_Bytes x Burst_Length)，则后面的公式成立 Address_N = Wrap_Boundary。

第一次突发式读写：

Lower_Byte_Lane = Start_Address - (INT(Start_Address / Data_Bus_Bytes))

x Data_Bus_Bytes

Upper_Byte_Lane = Aligned_Address + (Number_Bytes - 1) -

(INT(Start_Address / Data_Bus_Bytes)) x Data_Bus_Bytes

除了第一次读写之后的读写：

Lower_Byte_Lane = Address_N － (INT(Address_N / Data_Bus_Bytes))

x Data_Bus_Bytes

Upper_Byte_Lane = Lower_Byte_Lane + Number_Bytes － 1

DATA[(8 x Upper_Byte_Lane) + 7 : (8 x Lower_Byte_Lane)]。

第五章

本章描述了 AXI 协议支持的系统级的 Cache和保护单元。

1、ARCACHE[3:0] 和 AWCACHE[3:0] 的编码如下图：

WA	RA	C	B	Transaction attributes
0	0	0	0	Noncacheable and nonbufferable
0	0	0	1	Bufferable only
0	0	1	0	Cacheable, but do not allocate
0	0	1	1	Cacheable and bufferable, but do not allocate
0	1	0	0	Reserved
0	1	0	1	Reserved
0	1	1	0	Cacheable write-through, allocate on reads only
0	1	1	1	Cacheable write-back, allocate on reads only
1	0	0	0	Reserved
1	0	0	1	Reserved
1	0	1	0	Cacheable write-through, allocate on writes only
1	0	1	1	Cacheable write-back, allocate on writes only
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Cacheable write-through, allocate on both reads and writes
1	1	1	1	Cacheable write-back, allocate on both reads and writes

在一些情况下， 信号 AWACAHE 可以用来确定哪个部件来提供写响应。 如果写事务被指定为 bufferable ， 那么他接受来自桥或者系统级的 cache提供的写响

应。如果事务被指定为 non-bufferable，那么写响应必须有最终目的源提供。

2、AWPROT 或者 ARPROT 信号提供三种级别的存取保护：

（1） 正常存取或者特权存取， ARPROT[0] 和 AWPROT[0]

（2） 安全性存取或者没有安全性存取， ARPROT[1] 和 AWPROT[1]

（3） 指令存取或者数据存取 ARPROT[2] 和 AWPROT[2]

信号 ARPROT[2:0] 和 信号 AWPROT[2:0]的编码如下图：

Table 5-2 Protection encoding

ARPROT[2:0] AWPROT[2:0]	Protection level
[0]	1 = privileged access 0 = normal access
[1]	1 = nonsecure access 0 = secure access
[2]	1 = instruction access 0 = data access

第六章

本章描述了 AXI 协议工具的独占式存取和锁存取机制。

1、当对自动存取时能之后，可以通过信号 ARLOCK[1:0] 或信号 AWLOCK[1:0] 来配置独占式存取和锁存取。编码如下图：

Table 6-1 Atomic access encoding

ARLOCK[1:0] AWLOCK[1:0]	Access type
b00	Normal access
b01	Exclusive access
b10	Locked access
b11	Reserved

我们通过信号 ARLOCK[1:0] 或 AWLOCK[1:0] 来选择独占式存取，用信号 RRESP[1:0]或 BRESP[1:0]来指明独占式存取的成功与否。

2、主机在请求独占式存取时，设备会返回两个响应分别是 EXOKAY 和 OKAY 。EXOKAY 是指设备支持独占式存取，而 OKAY 是指设备不支持独占式存取。 、

3、如果一个设备不支持独占式存取，可以忽略信号 ARLOCK[1:0] 和 AWLOCK[1:0] 。他必须提供 OKAY 响应对正常式存取和独占式存取。如果一个设备要支持独占式存取则必须有硬件监视器。

4、通过信号 ARLOCK[1:0] 或信号 AWLOCK[1:0] 对事务加锁，需要确定只允许主机存取设备区域直到一个未加锁的事务从同一个主机完成。 此处推荐锁存取只用来支持 legacy devices。

5、推荐遵循下面两天建议，但是不强制：

（1） 保持所有锁事务序列都在相同的 4KB 地址区域内。

（2） 限制用琐事务序列对两个事务加锁。

第七章

本章描述了 AXI 读写事务的四个设备响应。

1、AXI 协议对读事务和写事务都有响应。对于读事务，读响应与读数据一起发送给主机，而写事务将写响应通过写响应通道传送。 AXI 协议的响应类型有 OKAY 、EXOKAY 、SLVERR、DECERR。

2、通过信号 RRESP[1:0]和 BRESP[1:0]来编码响应信号，具体如下图：

Table 7-1 RRESP[1:0] and BRESP[1:0] encoding

RRESP[1:0] BRESP[1:0]	Response	Meaning
b00	OKAY	Normal access okay indicates if a normal access has been successful. Can also indicate an exclusive access failure.
b01	EXOKAY	Exclusive access okay indicates that either the read or write portion of an exclusive access has been successful.
b10	SLVERR	Slave error is used when the access has reached the slave successfully, but the slave wishes to return an error condition to the originating master.
b11	DECERR	Decode error is generated typically by an interconnect component to indicate that there is no slave at the transaction address.

协议规定请求的需要传输的数据数量必须被执行， 即使有错误报告。 在一次突发式读写的剩余数据不会被取消传输，即使有单个错误报告。

3、AXI 协议的四种响应类型：正常存取成功、独占式存取、设备错误、译码错误。AXI 协议要求， 在一个传输事务中的所有数据必须传输完成， 即使有错误状态发生。

第八章

本章描述 AXI 协议用事务 ID tags 来处理多地址和乱序传输。

1、下面介绍 5 中事务 IDs：

（1）AWID 这个 ID tag 是写地址群组信号。

（2）WID 这个是写 ID tag 在写事务中，与写数据在一起，主机传送一个 WID 去匹配与地址相一致的 AWID。

（3）BID 这个 ID tag 是写响应事务中。设备会传送 BID 去匹配与 AWID 和 WID 相一致的事务。

（4）ARID 这个 ID tag 是读地址群组信号。

（5）RID 这个 ID tag 是在读事务中。设备传送 RID 去匹配与 ARID 相一致的事务。

2、主机可以使用一个事务的 ARID 或者 AWID 段提供的附加信息排序主机的需要。事务序列规则如下：

（1） 从不同主机传输的事务没有先后顺序限制。他们可以以任意顺序完成。

（2） 从同一个主机传输的不同 ID 事务，也没有先后顺序限制。他们可以以任

意顺序完成。

（3） 相同数值的 AWID 写事务数据序列必须按照顺序依次写入主机发送的地址内。

（4） 相同数值的 ARID 读事务数据序列必须遵循下面的顺序：

◈① 当从相同设备读相同的 ARID 时，设备必须确保读数据按照相同的地址顺序接受。

◈② 当从不同的设备读相同的 ARID 时，接口处必须确保读数据按照主机发送的相同的地址顺序。

（5） 在相同的 AWID 和 ARID 的读事务和写事务之间没有先后顺序限制。 如果主机要求有顺序限制， 那么必须确保第一次事务完全完成后才开始执行第二个事务。

3、当一个主机接口与 interconnect 相连时， interconnect 会在信号 ARID 、AWID 、WID 段添加一位，每一个主机端口都是独一无二的。

这样做有两个影响：

（1）主机不需要去知道其他主机的 ID 数值，因为 interconnect是 ID 值是唯一的，当将主机 number 添加到段中。

（2）在设备接口处的 ID 段的宽度要比主机接口处的 ID 段宽。

对于读数据， interconnect 附加一位到 RID 段中，用来判断哪个主机端口读取数据。Interconnect 会移除 RID 段中的这一位在将 RID 的值送往正确的主机端口之前。

第九章

本章描述了 AXI 读写数据总线传输的不同大小和接口如何用字节不变 endian 去握手混合 endian 传输。

1、Narrow 传输，当主机产生的数据宽度小于数据总线宽度时，地址和控制信息决定哪一个 byte lanes为有效的数据。下面是两个应用 byte lanes的例子：

Example 1：

In Figure 9-2:

- the burst has five transfers
- the starting address is 0
- each transfer is eight bits
- the transfers are on a 32-bit bus.

Byte lane used

			DATA[7:0]	1st transfer
		DATA[15:8]		2nd transfer
	DATA[23:16]			3rd transfer
DATA[31:24]				4th transfer
			DATA[7:0]	5th transfer

Figure 9-2 Narrow transfer example with 8-bit transfers

Example 2：

In Figure 9-3:

- the burst has three transfers
- the starting address is 4
- each transfer is 32 bits
- the transfers are on a 64-bit bus.

Byte lane used

DATA[63:32]		1st transfer
	DATA[31:0]	2nd transfer
DATA[63:32]		3rd transfer

Figure 9-3 Narrow transfer example with 32-bit transfers

2、下面是一个数据不变性存取需求的数据结构的例子。他包含头信息，例如 source、destination identifiers 这些信息是采用 little-endian 格式，但是 payload 是 big-endian 字节流，具体情况如下图：

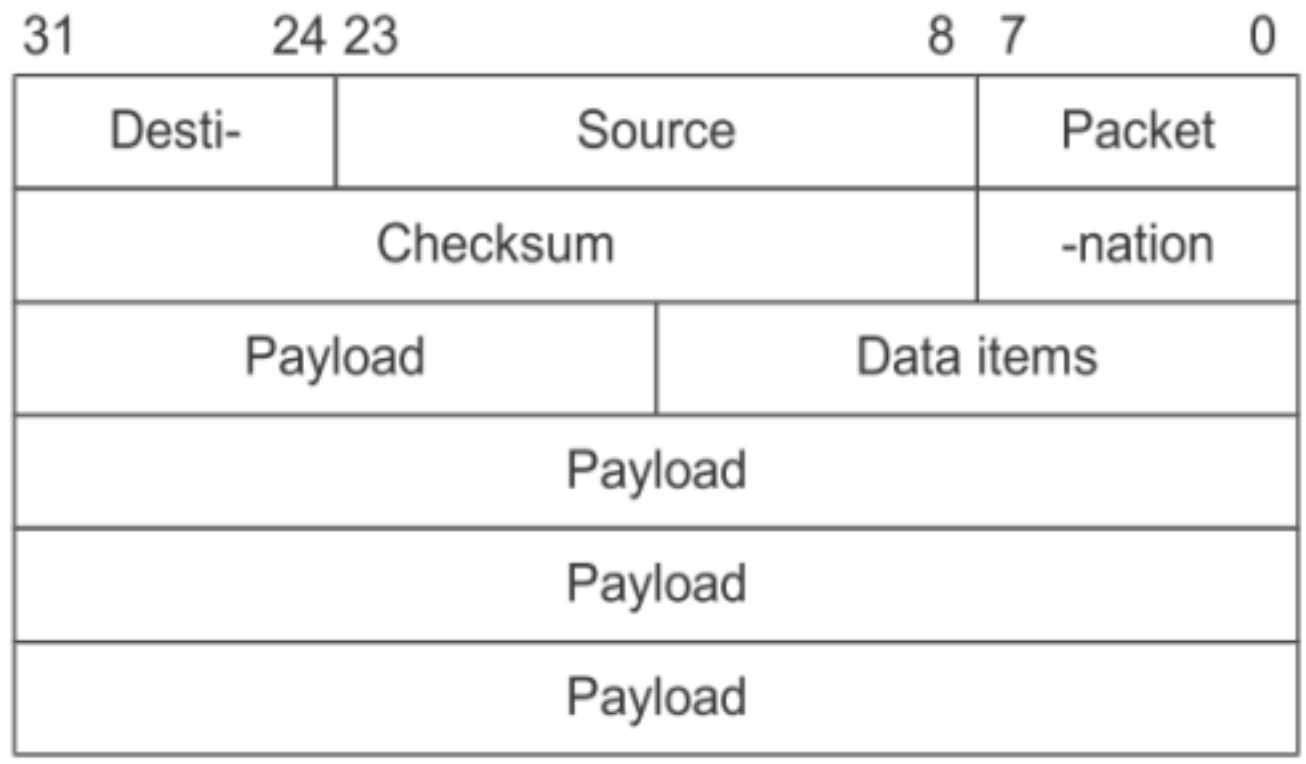


Figure 9-4 Example mixed-endian data structure

数据不变性确保在数据结构中 little-endian 存取头信息的部分不会破坏其他 big-endian 数据。

第十章

本章描述 AXI 协议不对齐握手传输。

1、AXI 协议允许主机使用低阶地址行去标示一个不对齐的起始地址在突发读写中。低阶地址行的信息必须包含 byte lane strobes信息。

2、下面是几个例子来表明数据以对齐或者不对齐的地址为起始地址, 分别在 32 位和 64 位数据总线上面传输的情况。其中暗色的框表示没有传输的数据。

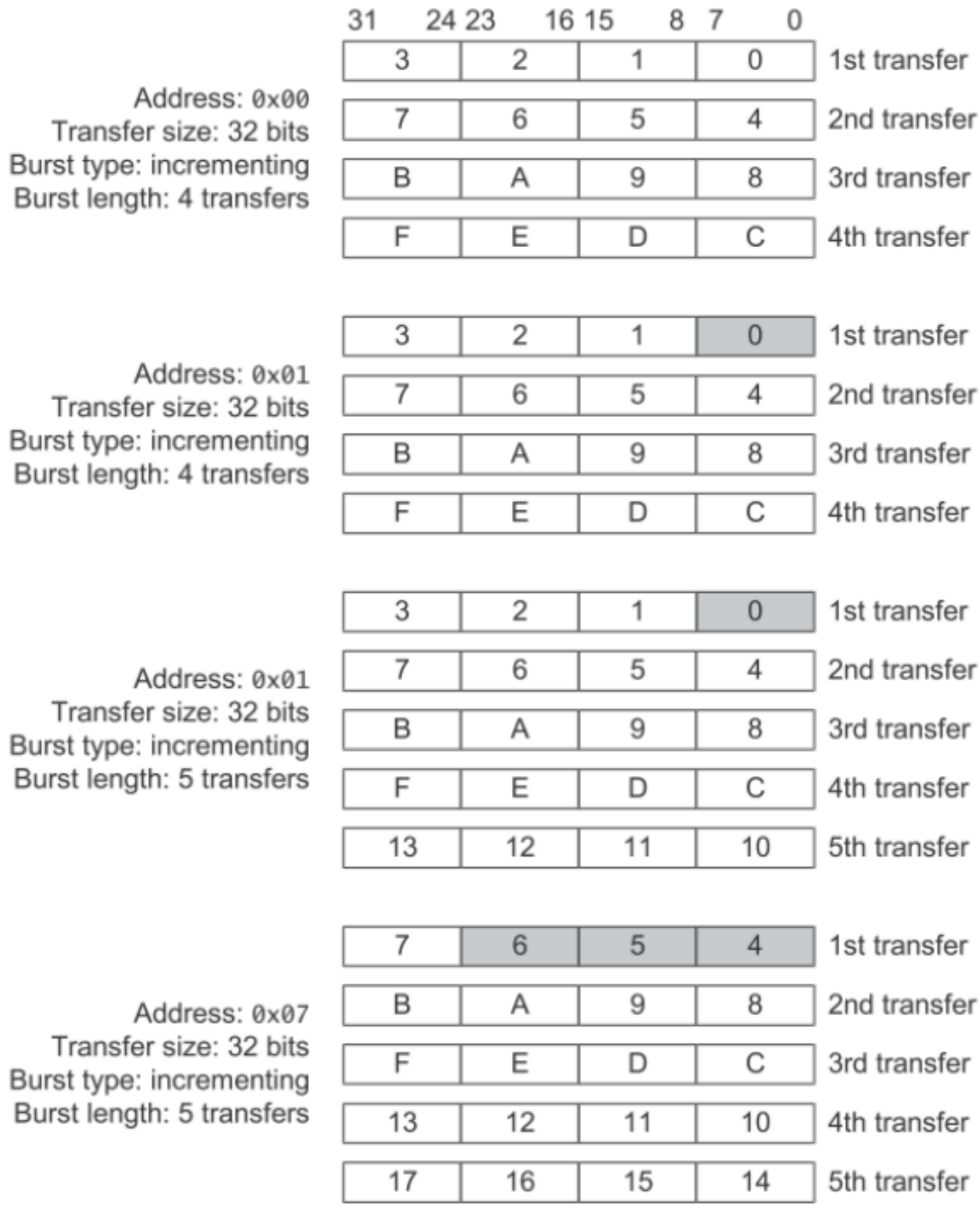


Figure 10-1 Aligned and unaligned word transfers on a 32-bit bus

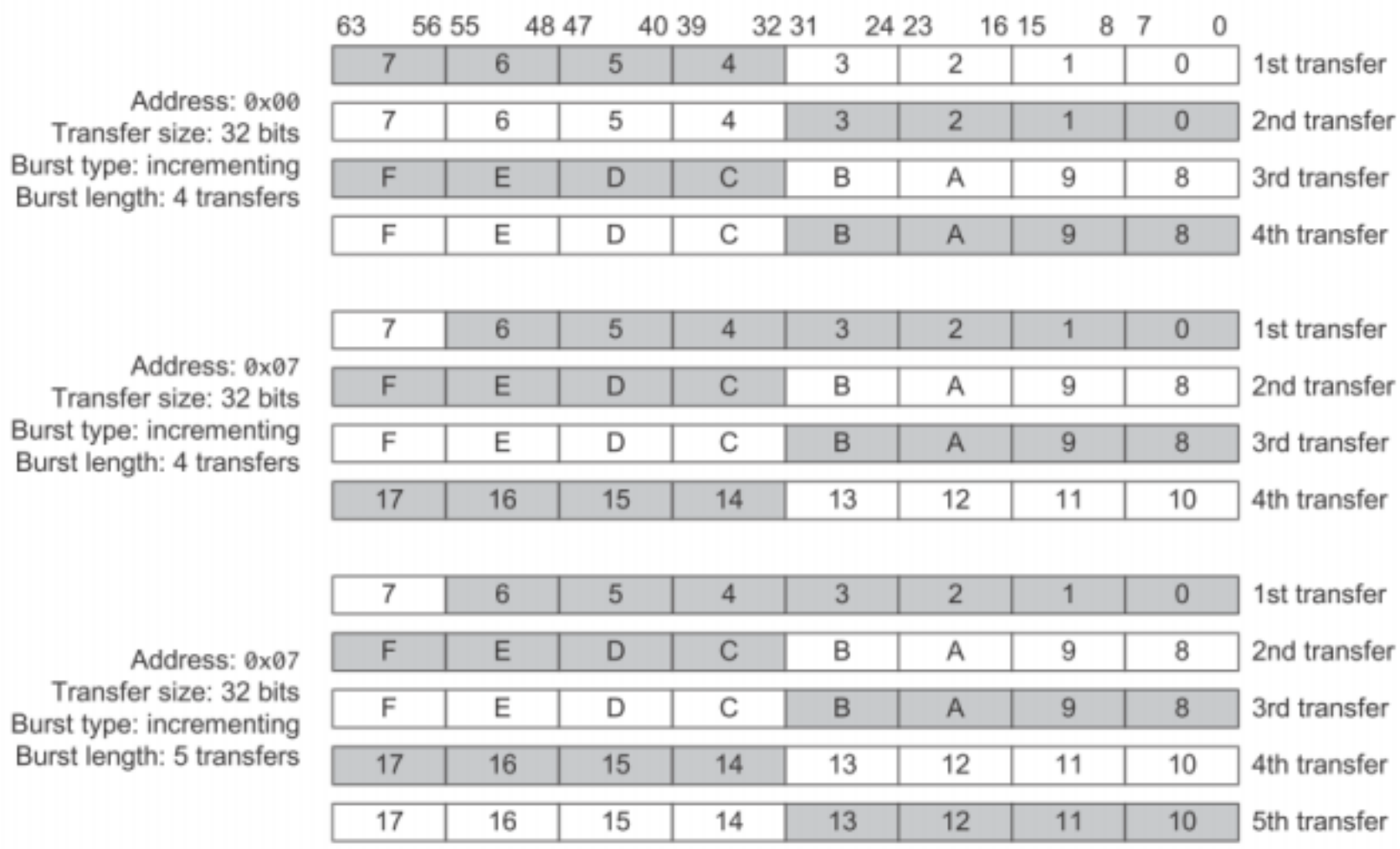


Figure 10-2 Aligned and unaligned word transfers on a 64-bit bus

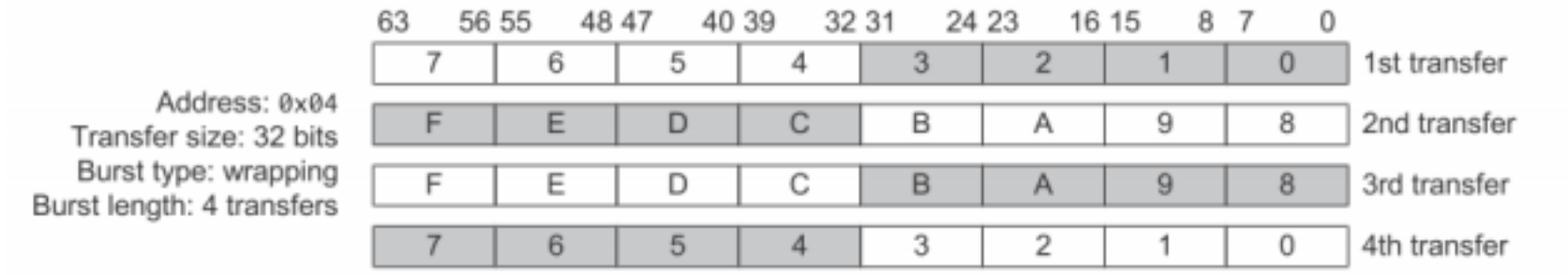


Figure 10-3 Aligned wrapping word transfers on a 64-bit bus

第十一章

本章主要描述 AXI 时钟和复位信号的时序。

1、在复位期间，以下接口必须遵循：

（1）主机接口必须将 ARVALID 、AWVALID 、WVALID 信号置低。

（2）设备接口必须将 RVALID 、BVALID 信号置低。

（3）所有其它信号可以为任意值。

2、主机接口必须开始将 ARVALID 、AWVLAID 或 WVALID 置高仅仅在 ARESETn 信号变高后的 ACK 的第一个上升沿。具体情况如下图：

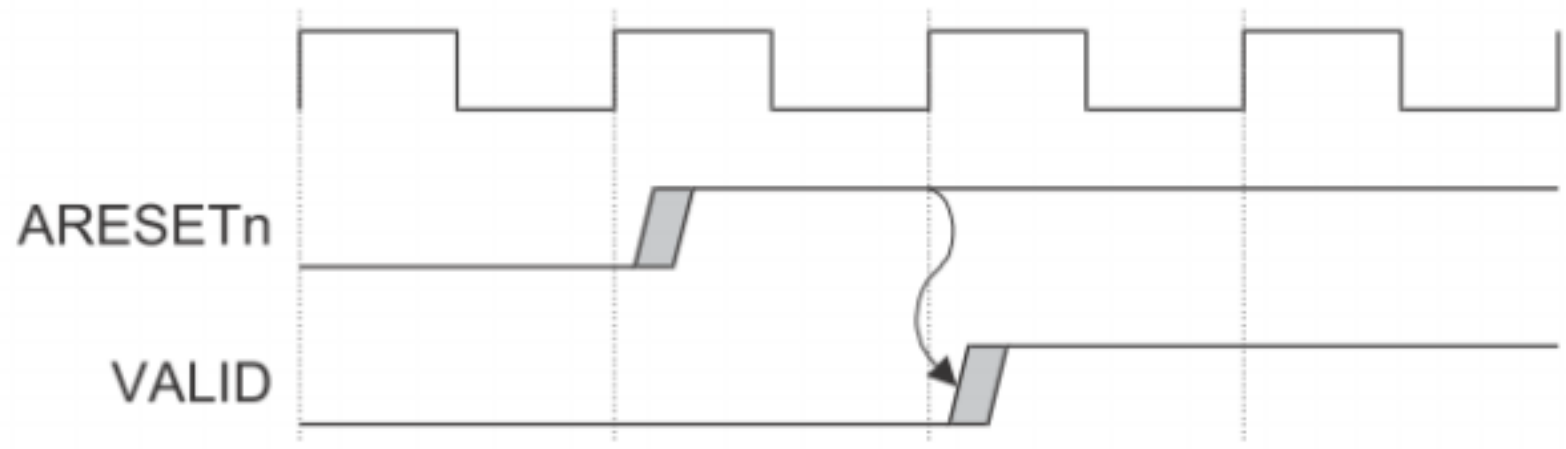


Figure 11-1 Exit from reset

第十二章

本章主要描述 AXI 协议在进入和离开低功耗状态期间的时钟控制接口。

1、低功耗时钟控制接口包括下面两个信号：

（1） 来自外围设备的信号，用于指明什么时候时钟使能能或者禁能。

（2） 两个握手信号用于系统时钟控制器请求退出或者进入低功耗状态。

2、时钟控制接口的一个主要信号时 CACTIVE，外围设备用这个信号来指明请求时钟使能。外围设备置 CACTIVE 有效去请求时钟，系统时钟控制器必须马上使能时钟。如果外围设备将 CACTIVE 置为无效，则系统时钟控制器将自己决定是否使能或者禁能外围设备时钟。

3、AXI 协议提供双线 request/acknowledge 握手来支持请求：

（1）CSYSREQ 当外围设备请求进入低功耗状态时，系统时钟控制器将 CSYSREQ 置低，平时 CSYSREQ 都是置高的。

（2）CSYSACK 外围设备用 CSYSACK 信号作为进入低功耗状态和离开低功耗状态的应答信号。

下面是 CSYSREQ 和 CSYSACK 信号之间的时序图：

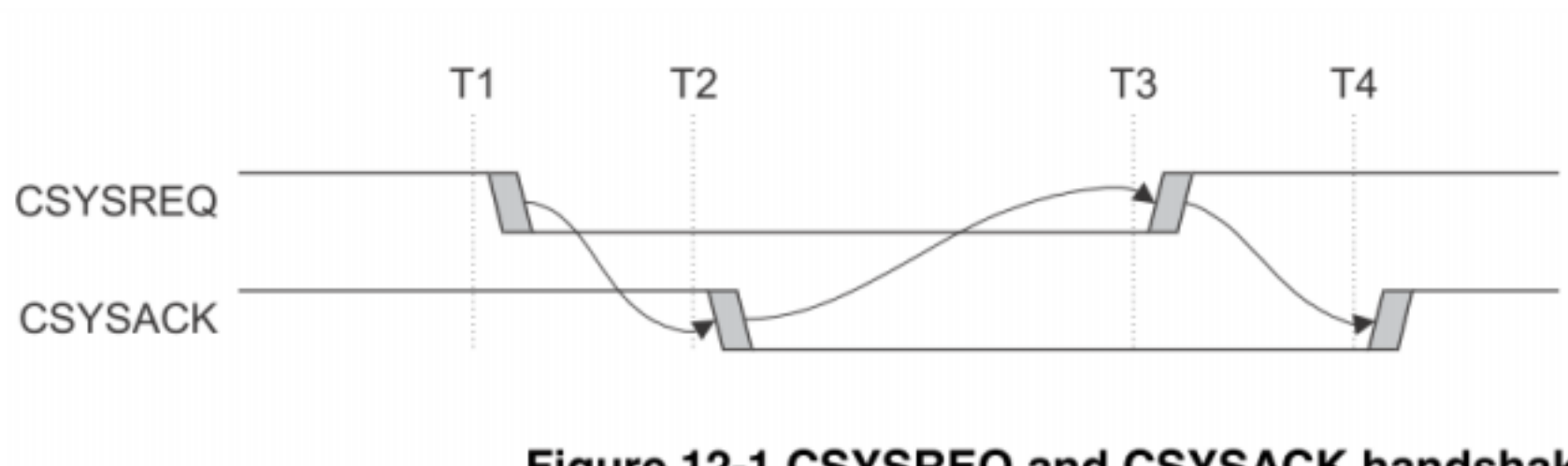


Figure 12-1 CSYSREQ and CSYSACK handshake

系统时钟控制器在 T1 时刻发出请求，外围设备在 T2 时刻给予应答，此时进入低功耗状态。在 T3 时刻，CSYSREQ 变高，请求离开低功耗状态，在 T4 时刻得到应答，此时离开低功耗状态进入正常模式。

4、外围设备可以选择接受请求也可以选择不接受请求。 主要通过信号 CACTIVE 来决定。

接受请求的情况：

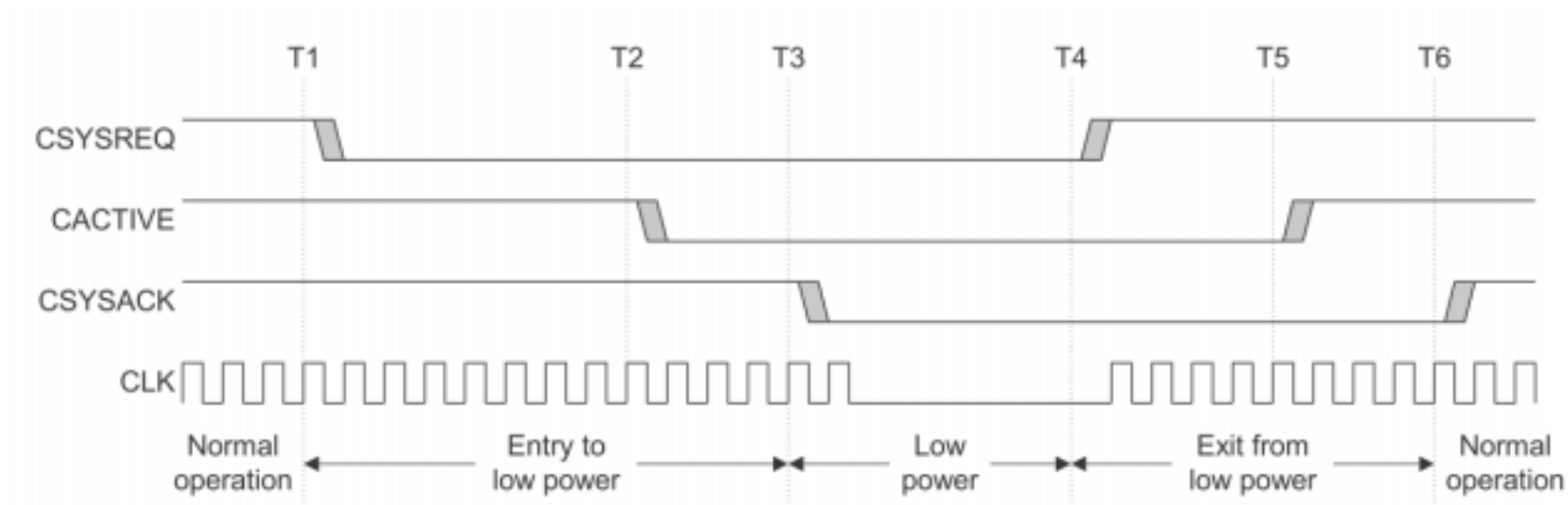


Figure 12-2 Acceptance of a low-power request

不接受请求的情况：

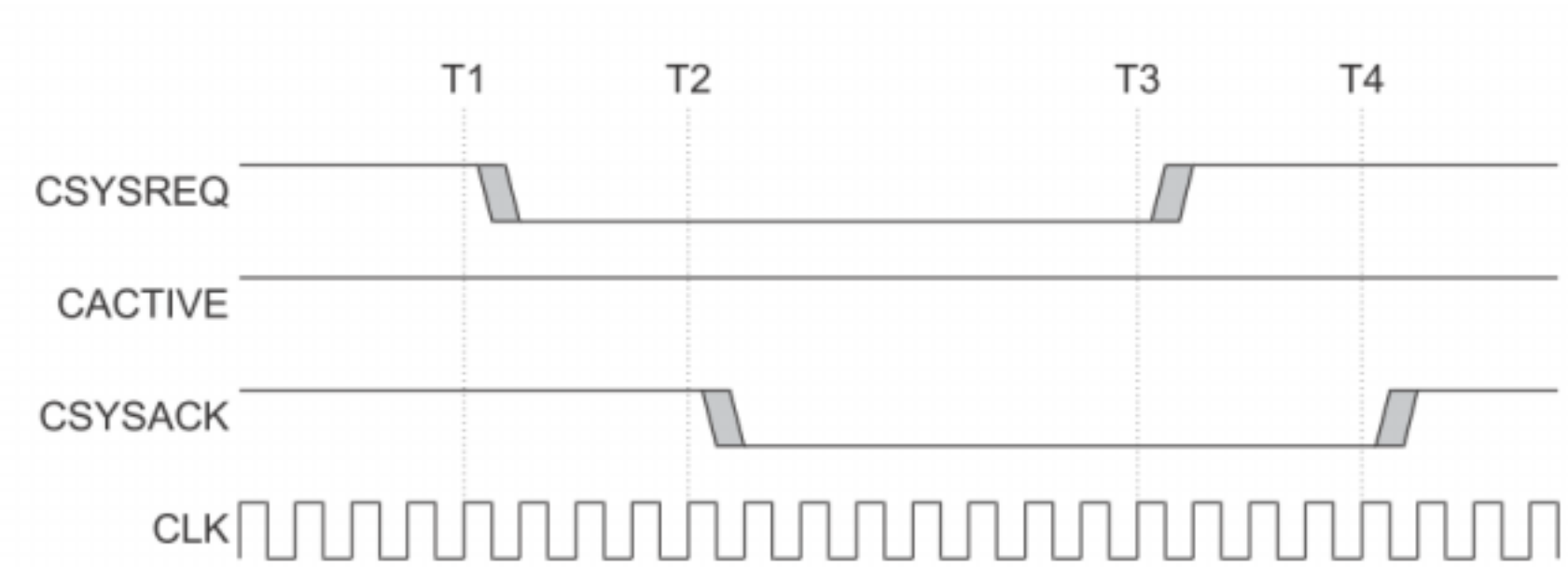


Figure 12-3 Denial of a low-power request

5、既可以通过系统也可以通过外围设备来退出低功耗状态。只要置信号 CACTIVE 和 CSYSREQ 这两个信号中的一个为高就可以退出低功耗模式。而系统可以通过置 CSYSREQ 为高来退出低功耗模式。

6、时钟控制框图如下：

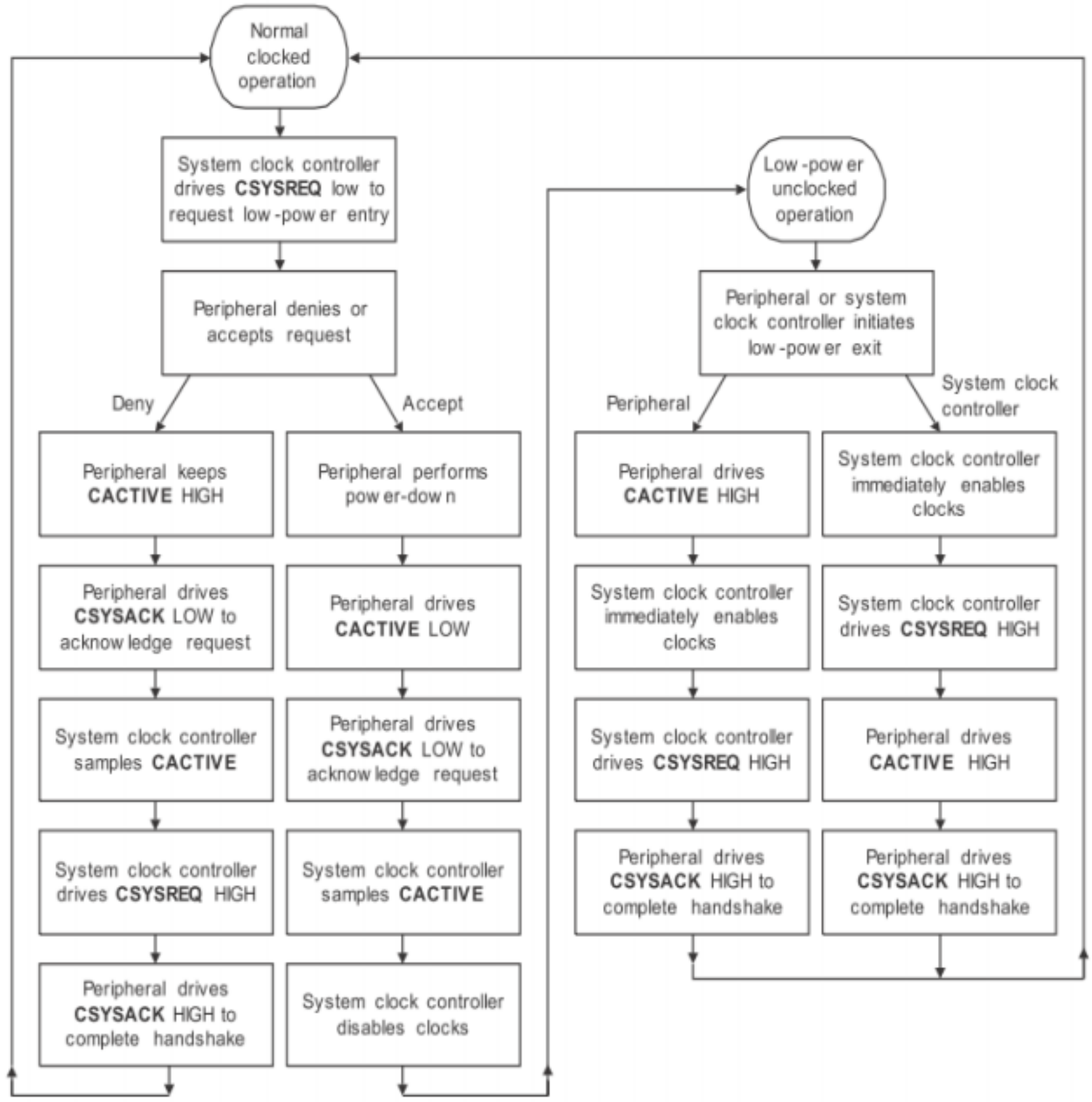


Figure 12-4 Low-power clock control sequence