Chapter A1 介绍

本章介绍了 AXI 协议的体系结构和本规范中使用的术语

A1.1 关于AXI协议

AMBA AXI 协议支持高性能、高频系统设计。

AXI 协议:

- 适用于高带宽和低延迟设计
- 无需使用复杂的桥即可提供高频操作
- 满足各种组件的接口要求
- 适用于具有高初始访问延迟的存储器控制器
- 提供了实施的灵活性互连架构
- 向后兼容现有的 AHB 和 APB 接口

AXI 协议的主要特点是:

- 独立的地址/控制和数据阶段
- 支持未对齐的数据传输，使用字节选通
- 使用仅发出起始地址的基于突发的事务
- 独立的读取和写入数据通道，可提供低成本的直接存储器访问 (DMA)
- 支持发布多个未完成的地址
- 支持乱序事务完成
- 允许轻松添加寄存器阶段以提供时序收敛

A1.2 AXI修订版本

A1.3 AXI结构

AXI 协议是基于突发的，并定义了以下独立的事务通道:

- 读地址
- 读数据
- 写地址
- 写数据
- 写回复

地址通道携带描述了待传输数据性质的控制信息。

数据在主机和从机之间传输，使用:

- 一个写入数据通道，用于将数据从主机传输到从机。在写事务中，从设备使用写响应通道向主设备发送传输完成信号。
- 一个读取数据通道，用于将数据从从设备传输到主设备。

AXI协议:

- 允许在实际数据传输之前发布地址信息
- 支持多个未完成的事务
- 支持事务的乱序完成

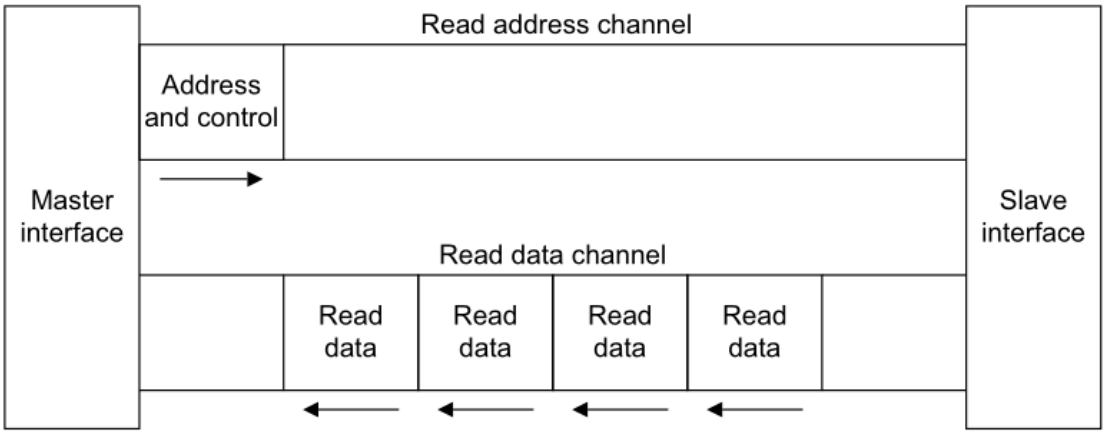


Figure A1-1 Channel architecture of reads

图 A1-1 显示了读取事务如何使用读取地址和读取数据通道。

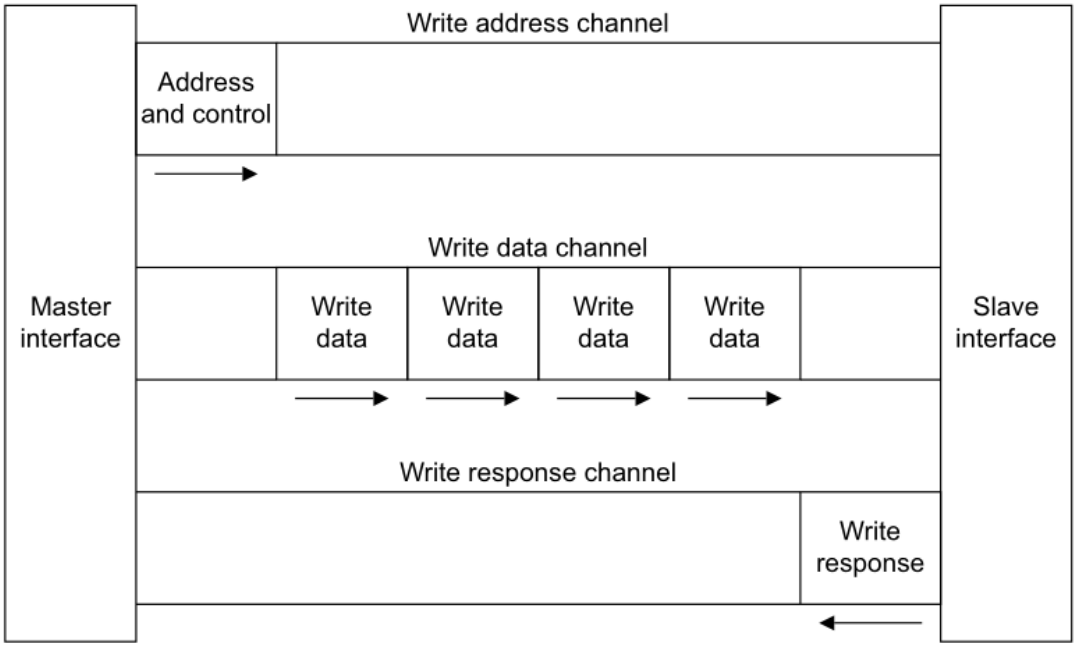


Figure A1-2 Channel architecture of writes

图 A1-2 显示了写事务如何使用写地址、写数据和写响应通道。

A1.3.1 通道定义

每个独立通道由一组信息信号和**VALID**和**READY**信号组成，提供双向握手机制。

数据来源方使用**VALID**信号来显示有效地址、数据或控制信息何时在通道上可用。数据目的地使用**READY**信号来显示它何时可以接受信息。

读数据通道和写数据通道还包括一个**LAST**信号，用于指示**一次传输事务中的最后一个数据**（transaction）。

读写地址通道

每个读写事务都有自己的地址通道。适当的地址通道携带传输事务所需的所有地址和控制信息。

读取数据通道

读数据通道承载从机到主机的读取**数据**和读取**响应**信息，包括：

- 数据总线，可以是 8、16、32、64、128、256、512 或 1024 位宽
- 指示读取事务完成状态的读取响应信号。

写数据通道

写入数据通道将写入数据从主机传送到从机，包括：

- 数据总线，可以是 8、16、32、64、128、256、512 或 1024 位宽
- 字节通道选通信号，用于每八个数据位，指示数据的哪些字节是有效的

写入数据通道信息始终被视为缓冲，**因此主机可以执行写入事务，而无需从机确认先前的写入事务。**

写响应通道

从机使用写响应通道来响应写事务。所有写事务都需要在写响应通道上发出完成信号。仅针对**完整的事务完成后发出信号**，而不是针对事务中的每个数据传输。

A1.3.2 接口和互连

一个典型的系统由许多通过某种形式的互连连接在一起的主设备和从设备组成，如图 A1-3 所示。

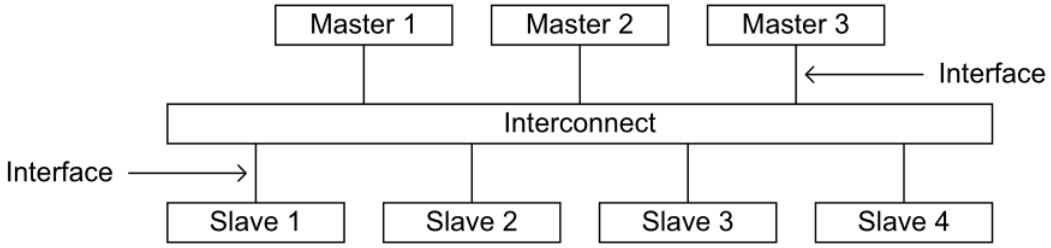


Figure A1-3 Interface and interconnect

AXI 协议为接口提供单一接口定义：

- 在主机和互连之间
- 在从机和互连之间
- 在主机和从机之间

经典的系统拓扑结构

大多数系统使用三种互连拓扑结构中的一种：

- 共享地址和数据总线
- 共享地址总线和多个数据总线
- 多层，有多个地址和数据总线

在大多数系统中，**地址通道的带宽要求明显低于数据通道的带宽要求。**这样的系统可以通过使用共享地址总线与多个数据总线来实现并行数据传输，从而在系统性能和互连复杂性之间取得良好的平衡。

A1.3.3 寄存器片（Register slices）

每个AXI通道只在一个方向上传输信息，而且该架构不要求通道之间有任何固定的关系。这意味着一个寄存器片可以插入任何通道中的几乎任何一点，代价是增加一个周期的延迟。

A1.4 术语（手册P323）

本节总结了本规范中使用的术语，并在术语表或其他地方进行了定义。

在适当的地方，本节中列出的术语链接到相应的词汇表定义

Chapter A2 信号描述

A2.1 全局信号

表A2-1显示了全局AXI信号。这些信号是由AXI3和AXI4协议使用的。

信号	来源	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

所有信号都在全局时钟的上升沿进行采样。

A2.2 写地址通道信号

表A2-2显示了AXI写地址通道的信号。除非描述中另有说明，否则一个信号是由AXI3和AXI4使用的。

信号	来源	描述
AWID	Master	写地址ID。这个信号是写地址组信号的识别标签。
AWADDR	Master	写地址。写地址给出了写突发传输中第一次传输的首地址。
AWLEN	Master	写突发长度。突发长度给出了一个突发中传输的确切数量。这个信息决定了与地址相关的数据传输的数量。 这在AXI3和AXI4之间不同。
AWSIZE	Master	写突发数据位宽。该信号表示突发中每个传输的大小。
AWBURST	Master	写突发类型。突发类型和大小信息，决定了如何计算突发内每个传输的地址。
AWLOCK	Master	锁定类型。提供有关传输的原子特性的额外信息。 这在AXI3和AXI4之间不同。
AWCACHE	Master	内存类型。这个信号表明交易是如何在系统中进行的。
AWPORT	Master	保护类型。这个信号表示交易的权限和安全级别，以及交易是数据访问还是指令访问。
AWQOS	Master	服务质量，QoS。为每个写交易发送的QoS标识符。 仅在AXI4中实现。
AWREGION	Master	区域标识符。允许从机上的一个物理接口用于多个逻辑接口。 仅在AXI4中实现。
AWUSER	Master	用户信号。可选的用户在写地址通道中定义的信号。 仅在AXI4中支持。
AWVALID	Master	写地址有效。该信号表明通道发出有效的写地址和控制信息的信号。
AWREADY	Slave	写地址准备就绪。这个信号表示从机已经准备好接受地址和相关的控制信号。

A2.3 写数据通道信号

表 A2-3 显示了 AXI 写数据通道信号。除非另有说明，否则 AXI3 和 AXI4 使用信号

信号	来源	描述
WID	Master	写ID标签。该信号是写数据传输的ID标签。 仅在AXI3中支持。
WDATA	Master	写数据
WSTRB	Master	写入选通位。该信号指示哪些字节通道持有有效数据。写数据总线的每8位有一个写选通位。
WLAST	Master	最后写入位。这个信号表示在一个写突发中的最后一次传输。
WUSER	Master	用户信号。可选的用户在写数据通道中定义的信号。 仅在AXI4中支持。
WVALID	Master	写入有效。该信号表明有效的写数据和写选通是可用的。
WREADY	Slave	写入准备。这个信号表示从机可以接受写数据。

A2.4 写回复通道信号

表A2-4显示了AXI写响应通道的信号。除非描述中另有说明，否则一个信号是由AXI3和AXI4使用的

信号	来源	描述
BID	Slave	响应ID标签。这个信号是写响应的ID标签。
BRESP	Slave	写入响应。该信号表示写传输事务的状态。
BUSER	Slave	用户信号。可选的用户在写响应通道中定义的信号。 仅在AXI4中支持。
BVALID	Slave	写入响应有效。该信号表明通道发出了有效的写入响应信号。
BREADY	Master	响应就绪。该信号表明主控端可以接受写响应。

A2.5 读地址通道信号

表A2-5显示了AXI读地址通道的信号。除非描述中另有说明，否则一个信号是由AXI3和AXI4使用的。

信号	来源	描述
ARID	Master	读地址ID。该信号是读取地址组信号的识别标签。
ARADDR	Master	读地址。读地址给出了读突发事务中第一次传输的地址。
ARLEN	Master	突发长度。这个信号表示一个突发的确切传输数量。 这在AXI3和AXI4之间发生变化。
ARSIZE	Master	数据位宽。该信号表示突发中每个传输的数据大小。
ARBURST	Master	突发类型。突发类型和大小信息决定了如何计算突发内每个传输的地址。
ARLOCK	Master	锁定类型。这个信号提供了关于传输的原子特性的额外信息。 这在AXI3和AXI4之间有所变化。
ARCACHE	Master	内存类型。这个信号表明传输事务是如何在系统中进行的。
ARPORT	Master	保护类型。这个信号表示交易的权限和安全级别，以及交易是数据访问还是指令访问。
ARQOS	Master	服务质量，QoS。为每个读事务发送的QoS标识符。 仅在AXI4中实施
ARREGION	Master	区域标识符。允许从机上的一个物理接口用于多个逻辑接口。 仅在AXI4中实现。
ARUSER	Master	用户信号。可选的用户定义的信号，在读地址通道。 仅在AXI4中支持。
ARVALID	Master	读取地址有效。该信号表明通道发出有效的读取地址和控制信息的信号。
ARREADY	Slave	读取地址就绪。这个信号表示从机已经准备好接受地址和相关的控制信号。

A2.6 读数据通道信号

表A2-6显示了AXI读数据通道的信号。除非描述中另有说明，否则一个信号是由AXI3和AXI4使用的。

信号	来源	描述
RID	Slave	读ID标签。这个信号是由从机产生的读数据组信号的识别标签。
RDATA	Slave	读数据
RRESP	Slave	读响应。该信号表示读传输的状态。
RLAST	Slave	读取最后位。该信号表示在一个读突发中的最后一次传输。
RUSER	Slave	用户信号。可选的用户定义的信号，在读数据通道。 仅在AXI4中支持。
RVALID	Slave	读取有效。该信号表明通道发出了所需的读取数据的信号。
RREADY	Slave	读取就绪。该信号表明主控端可以接受读数据和响应信息。

A2.7 低功耗接口信号

表A2-7显示了可选的低功耗接口的信号。这些信号是由AXI3和AXI4协议使用的。

信号	来源	描述
CSYSREQ	Clock controller	系统退出低功耗状态请求。该信号是系统时钟控制器对外设退出低功耗状态的请求。
CSYSACK	Peripheral device	退出低功耗状态的确认。该信号是外设对系统退出低功率状态请求的确认。
CACTIVE	Peripheral device	时钟有效。该信号表明外设需要其时钟信号。

Chapter A3 单一接口需求

本章描述了单个主站和从站之间的基本AXI协议交易要求。

A3.1 时钟和复位

本节介绍了实现AXI全局时钟和复位信号**ACLK**和**ARESETn**的要求。

A3.1.1 时钟

每个AXI组件使用一个单一的时钟信号，ACLK。所有的输入信号都在ACLK的上升沿进行采样。

所有输出信号的变化必须发生在ACLK的上升沿之后。

在主机和从机接口上，输入和输出信号之间必须没有组合路径。

A3.1.2 复位

AXI协议使用单一的低电平复位信号，即**ARESETn**。复位信号可以以异步方式断言，但释放必须与**ACLK**的上升沿同步。

复位期间，以下接口要求：

- 主接口必须驱动**ARVALID**、**AWVALID**和**WVALID**为低电平

- 从属接口必须驱动**RVALID**和**BVALID**为低电平
- 所有其他信号可以被驱动到任何值。

复位后，主站被允许开始驱动**ARVALID**、**AWVALID**或**WVALID**的最早时间是在ARESETn为高电平后的ACLK的第一个上升沿。

图A3-1显示了复位后**ARVALID**、**AWVALID**或**WVALID**可以被驱动为高电平的最早点。

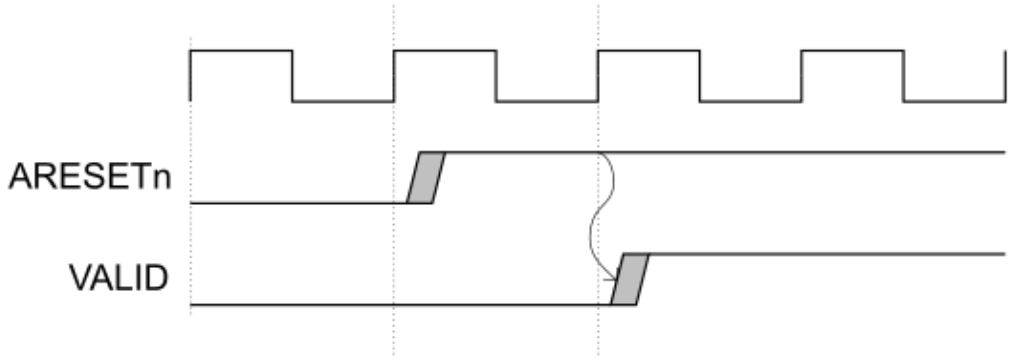


Figure A3-1 Exit from reset

A3.2 基础读写传输事务

本节定义了AXI协议传输事务的基本机制：

A3.2.1 握手过程

所有五个传输通道都使用相同的**VALID/READY**握手过程来传输地址、数据和控制信息。

这种双向流量控制机制意味着主机和从机都可以控制信息在主机和从机之间移动的速度。

源端产生**VALID**信号以指示地址、数据或控制信息何时可用。目的地产生READY信号，表示它可以接受该信息。

只有当**VALID**和**READY**信号都为高电平时才会发生传输。

在主机和从机接口上，输入和输出信号之间必须没有组合路径。

在图A3-2中，源端在T1之后提出地址、数据或控制信息并断言**VALID**信号。

目的地在T2之后断言**READY**信号，源端必须保持其信息的稳定性，直到传输在T3发生，此时这一断言被识别。

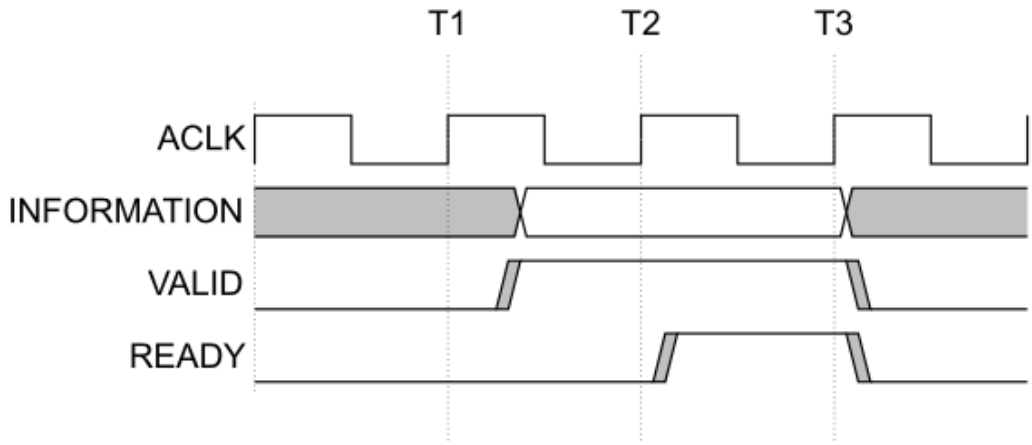


Figure A3-2 VALID before READY handshake

A source is not permitted to wait until **READY** is asserted before asserting **VALID**.

翻译：信号源不允许等到**READY**被断言后才断言**VALID**

一旦**VALID**被断言，它必须保持断言，直到握手发生（发生在上升的时钟边缘，此时**VALID**和**READY**都被断言。）

在图A3-3中，在地址、数据或控制信息有效之前，目的地在T1之后断言**READY**，表明它可以接受该信息。

源端提出信息，并在T2之后断言**VALID**，当该断言被识别时，传输在T3发生。在这种情况下，传输发生在一个周期内。

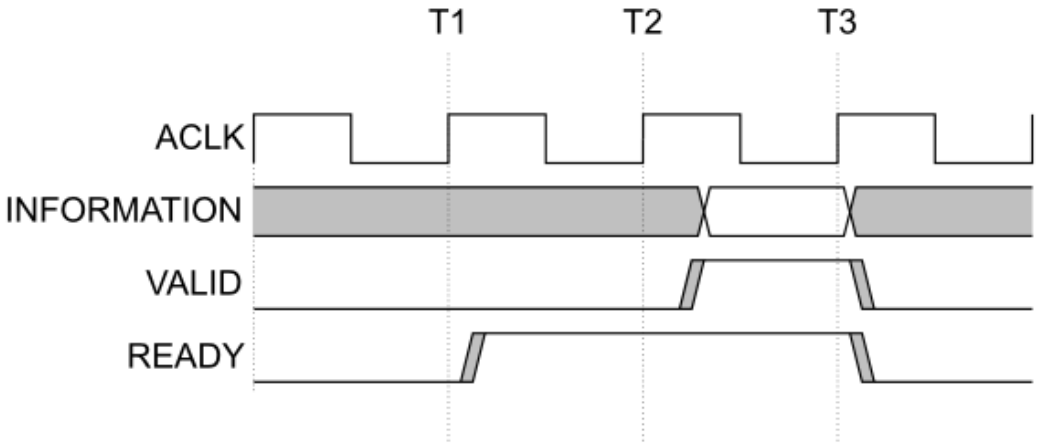


Figure A3-3 READY before VALID handshake

目的地允许等待**VALID**被断言后再断言相应的**READY**。

如果**READY**被断言，则允许在**VALID**被断言之前取消**READY**的断言。

在图A3-4中，在T1之后，源端和目的端都恰好表明它们可以传输地址、数据或控制信息。

在这种情况下，当**VALID**和**READY**的断言可以被识别时，传输发生在上升的时钟边缘。这意味着传输发生在T2。

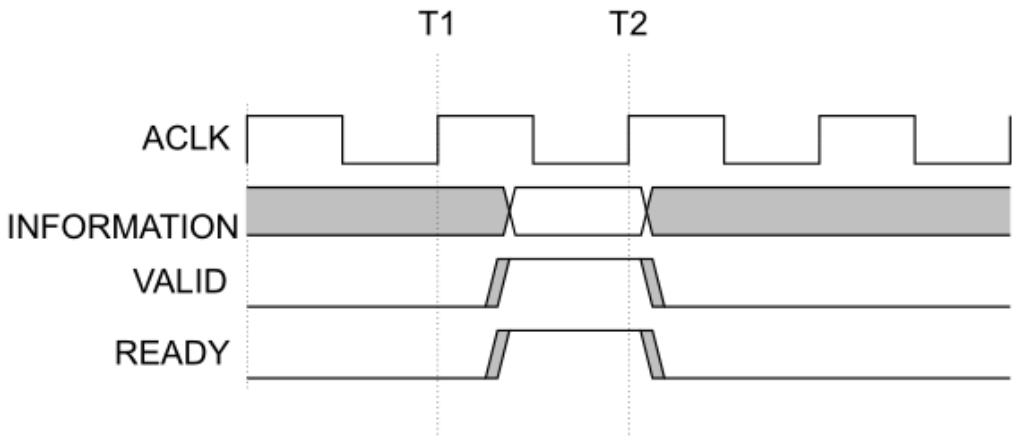


Figure A3-4 VALID with READY handshake

A3.2.2 通道信号要求

通道握手信号

每个通道都有自己的**VALID/READY**握手信号对。表A3-1显示了每个通道的信号。

Transaction channel	Handshake pair
Write address channel	**AWVALID, AWREADY**
Write data channel	**WVALID, WREADY**
Write response channel	**BVALID, BREADY**
Read address channel	**ARVALID, ARREADY**
Read data channel	**RVALID, RREADY**

写地址通道

主机只有在驱动有效的地址和控制信息时才能断言**AWVALID**信号。

当断言时，**AWVALID**必须保持断言，直到从机断言**AWREADY**后的一个上升时钟沿才能释放。

AWREADY的默认状态可以是高或低。本规范推荐默认状态为**高**。当**AWREADY**为高电平时，从属设备肯定能够接受呈现给它的任何有效地址。

Note:

本规范不推荐默认的**AWREADY**状态为LOW，因为它迫使传输至少需要两个周期，一个周期断言AWVALID，另一个周期断言AWREADY。

写数据通道

在一个写突发期间，主机只有在驱动有效的写数据时才能断言**WVALID**信号。

当断言时，**WVALID**必须保持断言，直到从机断言**WREADY**后的一个上升时钟沿才能释放。

WREADY的默认状态可以是高电平，但只有当从机在一个周期内总能接受写数据时才可以。

主机必须在驱动突发事件中的最后一次写传输（transfer）时发出**WLAST**信号。

写回复通道

从机只有在驱动有效的写回复信号时，才能断言**BVALID**信号。

当断言时，**BVALID**必须保持断言状态，直到主站断言**BREADY**后的一个上升时钟沿才能释放。

BREADY的默认状态可以是高电平，但前提是主机在一个周期内总是能接受一个写响应。

读地址通道

主站只有在驱动有效的地址和控制信息时才能断言**ARVALID**信号。

当断言时，**ARVALID**必须保持断言，直到从机断言**ARREADY**后的一个上升时钟沿才能释放。

ARREADY的默认状态可以是高或低。本规范推荐默认状态为**高**。如果**ARREADY**为高电平，那么从机肯定能够接受任何呈现给它的有效地址。

Note：

本规范不推荐默认的**ARREADY**值为LOW，因为它迫使传输至少需要两个周期，一个周期断言**ARVALID**，另一个周期断言**ARREADY**。

读数据通道

从机只有在驱动有效的读数据时才可以断言**RVALID**信号。

当断言时，**RVALID**必须保持断言，直到主机断言**RREADY**后的一个上升时钟沿才能释放。

即使从机只有一个读取数据的来源，它也必须只在响应数据请求时断言**RVALID**信号。

主机接口使用**RREADY**信号来表示它接受数据。**RREADY**的默认状态可以是高电平，但只有当主站能够立即接受读数据时，每当它开始一次读传输事务（transaction）。

从机必须在驱动突发事件中的最后一次读传输（transfer）时断言**RLAST**信号。

A3.3 通道间关系

AXI协议要求保持以下关系：

- 写回复信号必须总是在它作为一部分的写入事务中的最后一次写入传输之后
- 读取的数据必须始终跟在与数据有关的地址后面
- 通道握手必须符合依赖性

在其他方面，该协议没有定义通道之间的任何关系。

这意味着，例如，写数据可以在传输事务的写地址数据之前出现在一个接口上。（如果写地址通道比写数据通道包含更多的寄存器阶段，就会发生这种情况。同样地，写数据可能出现在与写地址相同的周期内。）

Note：

当互连需要确定目标地址空间或从机空间时，它必须重新调整地址和写数据。这是为确保写数据只对它要去的从属空间发出有效信号。

A3.3.1 通道握手信号之间的依赖性

为了防止出现死锁情况，必须遵守握手信号之间存在的依赖规则。

正如在通道信号要求中总结的一样，在任何传输事务中：

- 发送信息的AXI接口的**VALID**信号不能依赖于接收该信息的AXI接口的**READY**信号
- 一个正在接收信息的AXI接口可以等待，直到它检测到一个**VALID**信号，然后再断言其相应的**READY**信号。

Note：

虽然等待**VALID**被断言后再断言**READY**是可以接受的，但在检测到相应的**VALID**前断言**READY**也是可以接受的，这可以使设计更有效。

此外，不同通道上的握手信号之间存在着依赖性，AXI4定义了一个额外的写回复依赖性。下面的小节定义了这些依赖关系：

- 读传输事务依赖
- 写传输事务依赖
- AXI4的写回复依赖

在依赖关系图中：

- 单头箭头指的是可以在箭头开头的信号**之前或之后断言**的信号。
- 双头箭头指向的信号必须在箭头开始处的**信号断言后才会被断言。**

读传输事务依赖

图A3-5显示了读传输事务握手信号的依赖性，并显示在一个读传输事务中：

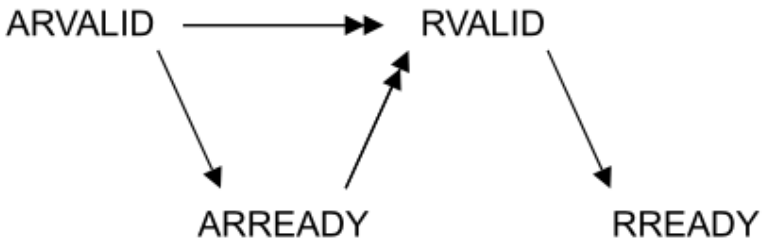
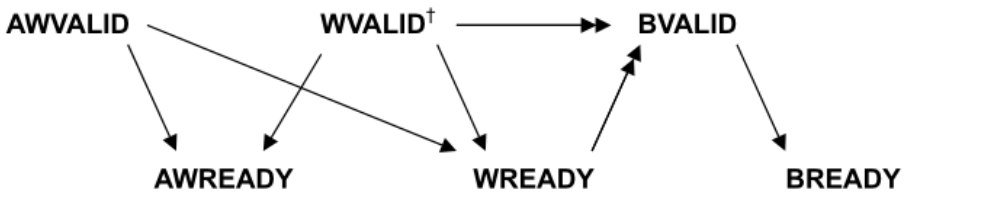


Figure A3-5 Read transaction handshake dependencies

- 主机在断言**ARVALID**之前，不得等待从机断言**ARREADY**
- 从机可以等待**ARVALID**被断言后再断言**ARREADY**
- 从机可以在**ARVALID**被断言之前断言**ARREADY**
- 从机必须等待**ARVALID**和**ARREADY**都被断言，然后再断言**RVALID**，以表明有效数据可用
- 从机在断言**RVALID**之前，不得等待主机断言**RREADY**
- 主机可以等待**RVALID**被断言后再断言**RREADY**
- 主站可以在**RVALID**被断言之前断言**RREADY**

写传输事务依赖

图A3-6显示了写传输事务握手信号的依赖性，并表明在一个写传输事务中：



† Dependencies on the assertion of **WVALID** also require the assertion of **WLAST**

Figure A3-6 Write transaction handshake dependencies

- 主机不能等待从站断言**AWREADY**或**WREADY**后再断言**AWVALID**或**WVALID**

- 从机可以等待**AWVALID**或**WVALID**，或两者都等待，然后再断言**AWREADY**
- 从机可以在**AWVALID**或**WVALID**（或两者）被断言之前断言**AWREADY**
- 从机可以等待**AWVALID**或**WVALID**，或两者，然后再断言**WREADY**
- 从机可以在**AWVALID**或**WVALID**（或两者）被断言之前断言**WREADY**
- 从机必须等待**WVALID**和**WREADY**都被断言，然后再断言**BVALID**
- 从机也必须等待**WLAST**被断言，然后再断言**BVALID**，因为写响应，**BRESP**，必须在写事务的最后一次数据传输之后才被发出信号
- 从机在断言**BVALID**之前，不得等待主机断言**BREADY**
- 主机可以在断言**BREADY**之前等待**BVALID**
- 主机可以在**BVALID**被断言之前断言**BREADY**

Caution：

必须遵守依赖规则以防止出现死锁情况。

例如，主机在驱动**WVALID**之前，不得等待**AWREADY**被断言。如果从机在断言**AWREADY**之前等待**WVALID**，就会发生死锁情况。

AXI4写回复依赖

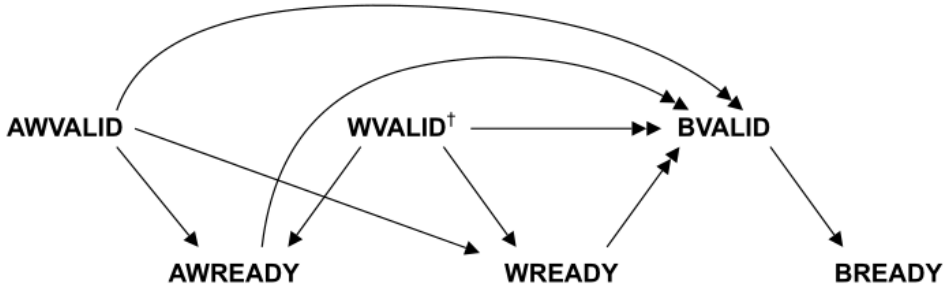
AXI4定义了一个额外的AXI4从机写响应依赖性规则

Note：

这种额外的依赖关系反映了AXI3中的预期用途，因为不希望任何组件在地址被接受之前接受所有写入数据并提供写入响应。

通过发布写响应信号，从设备负责针对所有后续事务对写入事务进行风险检查

图A3-7总结了AXI4从属写入响应握手依赖关系。这些依赖关系是：



† Dependencies on the assertion of **WVALID** also require the assertion of **WLAST**

Figure A3-7 Slave write response handshake dependencies

- 主机不能等待从机断言**AWREADY**或**WREADY**后再断言**AWVALID**或**WVALID**
- 从机可以等待**AWVALID**或**WVALID**，或两者，然后再断言**AWREADY**
- 从机可以在**AWVALID**或**WVALID**（或两者）被断言之前断言**AWREADY**
- 从机可以等待**AWVALID**或**WVALID**，或两者，然后再断言**WREADY**
- 从机可以在**AWVALID**或**WVALID**（或两者）被断言之前断言**WREADY**
- 从机必须等待**AWVALID**、**AWREADY**、**WVALID**和**WREADY**被断言，然后再断言**BVALID**

- 从机也必须等待**WLAST**被断言，然后再断言**BVALID**，因为写响应，**BRESP**必须在写事务的最后一次数据传输之后才被发出信号
- 从机在断言**BVALID**之前，不得等待主机断言**BREADY**
- 主机可以在断言**BREADY**之前等待**BVALID**
- 主机可以在**BVALID**被断言之前断言**BREADY**

A3.3.2 注意事项

AXI4写响应依赖性中描述的附加依赖性意味着AXI3的从机在接受地址之前接受所有写数据并提供写响应不适用于AXI4。

将AXI3从机转换为AXI4需要添加包装器，以确保在从属设备接受适当地址之前不会提供返回的写响应。

Note：

本规范强烈建议任何新的AXI3从机实现都包含此附加依赖项

任何AXI3主机都符合AXI4写响应要求。

A3.4 传输事务结构

本节介绍传输事务的结构。以下部分定义了地址、数据和响应的结构

A3.4.1 地址结构

AXI协议是基于突发的。

主机开始每个突发的时候，都会向从机发送控制信息和事务中第一个字节的地址。随着突发的进行，从机必须计算出突发中后续传输的地址。

一个突发事件不能跨越**4KB**的地址边界

Note：

这可以防止脉冲串跨越两个从机之间的边界。它还限制了一个从机必须支持的地址增量的数量。

突发事件长度

突发事件的长度由以下方面指定：

- **ARLEN[7:0]**，读传输
- **AWLEN[7:0]**，写传输

在本规范中，**AxLEN**表示**ARLEN**或**AWLEN**

AXI3支持所有突发类型的1到16次传输的突发长度

AXI4将对INCR突发类型的突发长度支持扩展到1到256次传输。在AXI4中对所有其他突发类型的支持仍然是1到16次传输。

AXI3的突发长度定义为Burst_Length = **AxLEN[3:0]** + 1

AXI4的突发长度定义为Burst_Length = **AxLEN[7:0]** + 1,以适应AXI4中INCR突发类型的扩展突发长度.

AXI有以下规则来管理突发的使用:

- 对于回转突发，突发长度必须是2、4、8或16
- 一次突发不能跨越4KB的地址边界
- 不支持提前终止突发事件

没有组件可以提前终止突发。然而，为了减少写入脉冲中的数据传输数量，主控设备可以通过禁止所有写入选通来禁用进一步的写入。在这种情况下，主机必须完成突发中的剩余传输。在读突发中，主机可以丢弃读数据，但必须完成突发中的所有传输

Note：

在访问读敏感设备（如FIFO）时，丢弃不需要的读取数据可能会导致数据丢失。当访问这样的设备时，主机必须使用与所需数据传输大小完全匹配的突发长度。

在AXI4中，INCR突发类型和长度大于16的事务可以转换为多个较小的突发，即使事务属性指示事务不可修改。

在这种情况下，生成的突发必须保留与原始事务相同的事务特征，唯一的例外是：

- 突发长度缩短
- 生成的突发的地址被适当地调整

Note：

为了与AXI3兼容，需要将较长的突发分解为多个较短的突发的能力，并且可能还需要这种能力来减少较长的突发对服务质量保证的影响。

突发事件位宽

每次数据传输或节拍中要传输的最大字节数由：

- **ARSIZE[2:0]**，读传输
- **AWSIZE[2:0]**，写传输

在本规范中，**AxSIZE**表示**ARSIZE**或**AWSIZE**

Table A3-2 Burst size encoding

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

表A3-2显示了**AxSIZE**的编码。

如果AXI总线比突发位宽更宽，AXI接口必须从传输地址中确定每次传输要使用数据总线的哪些字节通道

任何传输的大小都不能超过传输事务中任何一个代理的数据总线宽度。

突发事件类型

AXI协议定义了三种突发类型：

FIXED（固定突发）

在一个固定的突发事件中：

- 对于突发传输中的每个传输，地址都是相同的
- 有效的字节通道对于突发传输中的所有节拍是恒定的。然而，在这些字节通道中，实际有**WSTRB**断言的字节在突发的每个节拍中可能不同。

这种突发类型用于对同一位置的重复访问，如加载或清空FIFO时。

INCR（增量突发）

在增量突发中，突发中每次传输的地址是前一次传输的地址的增量。

递增值取决于传输的大小。例如，在一个大小为4个字节的突发事件中，每个传输的地址是前一个地址加4。

这种突发类型用于访问正常的顺序存储器。

WARP（回转突发）

回转突发与增量突发类似，只是如果达到地址上限，地址会环绕到一个较低的地址。

以下限制适用于回转突发事件：

- 开始地址必须与每个传输的位宽对齐
- 突发传输的长度必须是2、4、8或16个传输

回转突发的行为是：

- 突发传输使用的最低地址与要传输的数据的总大小对齐，即与（（突发中每个传输的位宽）×（突发中的传输数量））对齐。这个地址被定义为回转边界。
- 每次传输后，地址的递增方式与INCR突发的方式相同。然而，如果这个递增的地址是（（回转边界）+（要传输的数据总大小）），那么地址就会绕到回转边界。
- 突发传输中的第一个传输可以使用高于回转边界的地址，但要遵守适用于回转突发的限制。这意味着，对于任何第一个地址高于回转边界的WRAP突发，其地址都会被回转。

这种突发类型用于高速缓存行访问。

在本规范中，**AxBURST**表示**ARBURST**或**AWBURST**。

Table A3-3 Burst type encoding

AxBURST[1:0]	Burst type
0b00	FIXED
0b01	INCR
0b10	WRAP
0b11	Reserved

突发事件地址

本节提供了确定脉冲串内传输的地址和字节通道的方法。这些公式使用了以下变量：

Start_Address 主机发出的起始地址

Number_Bytes 每次数据传输中的最大字节数

Data_Bus_Bytes 数据总线中的字节通道的数量

Aligned_Address 起始地址的对齐版本

Burst_Length 突发长度

Address_N 一个突发事件中传输N的地址。N是1，表示一个突发事件中的第一个传输。

Wrap_Boundary 在一个回转突发中的最低地址

Lower_Byte_Lane 传输的最低寻址字节的字节通道

Upper_Byte_Lane 传输的最高寻址字节的字节通道

INT(x) x的四舍五入的整数值

这些方程式决定了一个突发事件中的传输地址:

- **Start_Address** = **AxADDR**
- **Number_Bytes** = 2 ^ **AxSIZE**
- **Burst_Length** = **AxLEN** + 1
- **Aligned_Address** = (**INT**(**Start_Address** / **Number_Bytes**)) × **Number_Bytes**

这些方程式决定了第一次传输的传输地址:

- **Address_1** = **Start_Address**

对于一个INCR突发，以及一个地址没有包裹的WRAP突发，这个方程式决定了突发中第一次传输后的任何传输的地址:

- **Address_N** = **Aligned_Address** + (N – 1) × **Number_Bytes**

对于一个WRAP突发，**Wrap_Boundary**变量定义了回转的边界:

- **Wrap_Boundary** = (**INT**(**Start_Address** / (**Number_Bytes** × **Burst_Length**)))× (**Number_Bytes** × **Burst_Length**)

对于一个WRAP突发，如果Address_N = Wrap_Boundary + (Number_Bytes × Burst_Length)，那么:

- 使用此公式进行当前传输:

  — **Address_N** = **Wrap_Boundary**

- 使用此公式进行任何后续的传输:

— **Address_N** = **Start_Address** + ((N – 1) × **Number_Bytes**) – (**Number_Bytes** × **Burst_Length**)

这些公式决定了在一个突发事件中的第一次传输要使用哪个字节通道:

- **Lower_Byte_Lane** = **Start_Address** – (**INT**(**Start_Address** / **Data_Bus_Bytes**)) × **Data_Bus_Bytes**
- **Upper_Byte_Lane** = **Aligned_Address** + (**Number_Bytes** – 1) – (**INT**(**Start_Address** / **Data_Bus_Bytes**)) × **Data_Bus_Bytes**.

这些公式决定了在一个突发的第一次传输之后，所有的传输要使用哪个字节通道:

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

数据的传输是在:

- DATA((8 × **Upper_Byte_Lane**) + 7 : (8 × **Lower_Byte_Lane**)).

A3.4.3 数据的读写结构

本节介绍了AXI读和写数据总线上不同大小的传输，以及接口如何执行混合大小端和非对齐传输。

写选通

WSTRB[n:0]信号为高电平时，指定数据总线上包含有效信息的字节通道。写数据总线的每八位有一个写选通，因此**WSTRB[n]**对应**WDATA[(8n)+7: (8n)]**

主设备必须确保仅针对包含有效数据的字节通道，写入选通为HIGH。

当**WVALID**为低电平时，写选通可以采取任何数值，尽管本规范建议它们要么被驱动为低电平，要么保持在之前的数值。

窄传输

当一个主机产生一个比其数据总线更窄的传输时，地址和控制信息决定了传输使用哪个字节通道

- 在递增或回转突发中，在突发的每个节拍上使用不同的字节通道
- 在一个固定的突发事件中，每个节拍都使用相同的字节通道

图A3-8和图A3-9给出了两个使用字节通道的例子。阴影部分表示没有传输的字节。

在图A3-8中

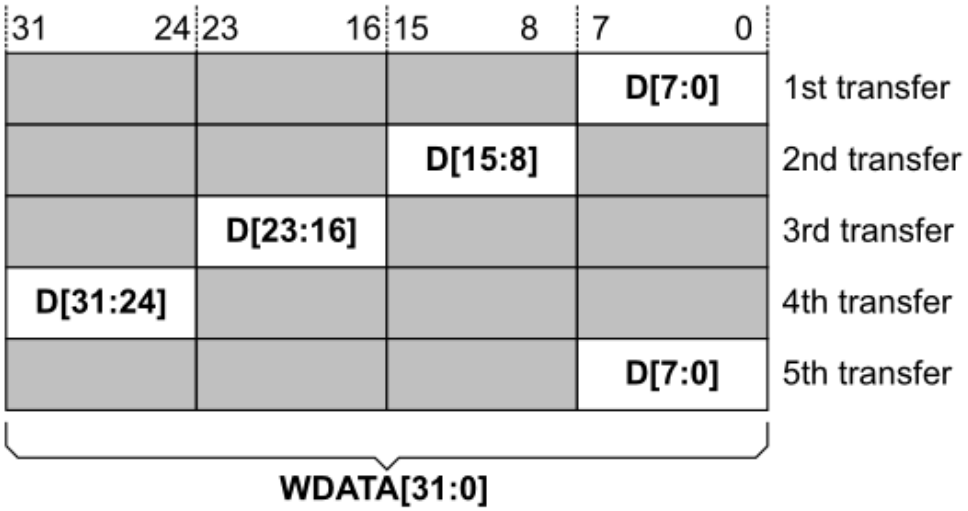


Figure A3-8 Narrow transfer example with 8-bit transfers

- 该突发有五个传输
- 起始地址是0
- 每个传输是8位
- 传输在32位总线上
- 突发类型是INCR

在图A3-9中

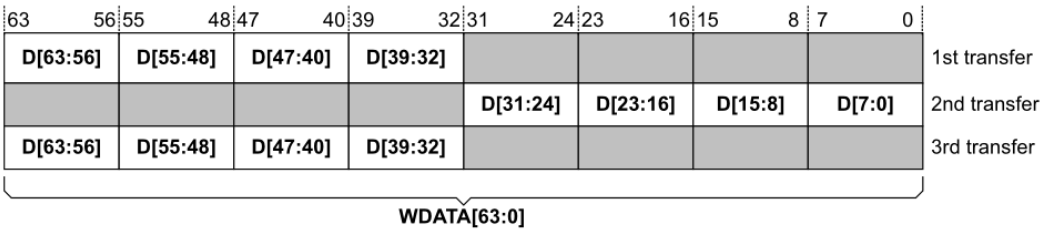


Figure A3-9 Narrow transfer example with 32-bit transfers

- 该突发有三次传输
- 起始地址是4
- 每次传输是32位
- 传输在64位总线上

字节不变性

为了访问单个内存空间中的混合字符顺序数据结构，AXI协议使用字节不变的字符顺序方案

字节不变字符顺序意味着，对于数据结构中的任何多字节元素：

- 无论数据的字节顺序如何，该元素都使用相同的连续字节内存
- 字符顺序确定这些字节在内存中的顺序，这意味着它确定内存中的第一个字节是元素的最高有效字节(MSB)还是最低有效字节(LSB)
- 传输到给定地址的任何字节都会将相同数据总线线上的8位数据传递到相同的地址位置，而不考虑该字节所属的任何数据元素的字节顺序

只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的相应字节通道上。

支持多种传输宽度的组件可能需要一个更复杂的接口来转换一个不是天然的字节不变的接口。

大多数小端组件可以直接连接到字节不变接口。只支持大端传输的组件需要字节不变操作的转换函数。

图A3-10和图A3-11的例子显示了一个32位的数字0x0A0B0C0D，存储在一个寄存器和一个存储器中。

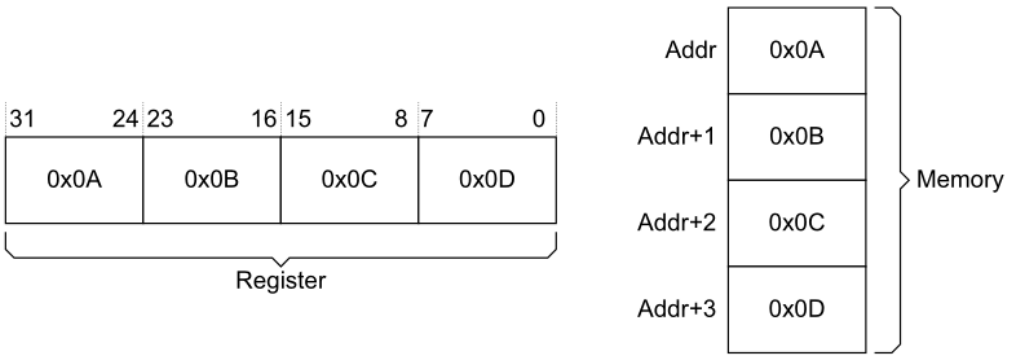


Figure A3-10 Example big-endian byte invariant data structure

图A3-10显示了大端字节不变数据结构的一个例子。在这个结构中：

- 数据的最高有效字节(MSB)为0x0A，存储在寄存器中的MSB位置
- 数据的MSB存储在具有最低地址的存储位置
- 其他数据字节按重要性降序排列

图A3-11显示了小端字节不变数据结构的一个例子。在这个结构中

- 数据的最低有效字节(LSB)为0x0D，存储在寄存器中的LSB位置
- 数据的LSB存储在具有最低地址的存储位置
- 其他数据字节按重要性递增的顺序放置

图A3-10和图A3-11中的例子表明，字节不变性确保大编码和小编码结构可以在一个内存空间中共存而不会损坏。

图A3-12显示了一个需要字节不变访问的数据结构示例。在本例中，报头字段使用小端排序，而有效负载使用大端排序

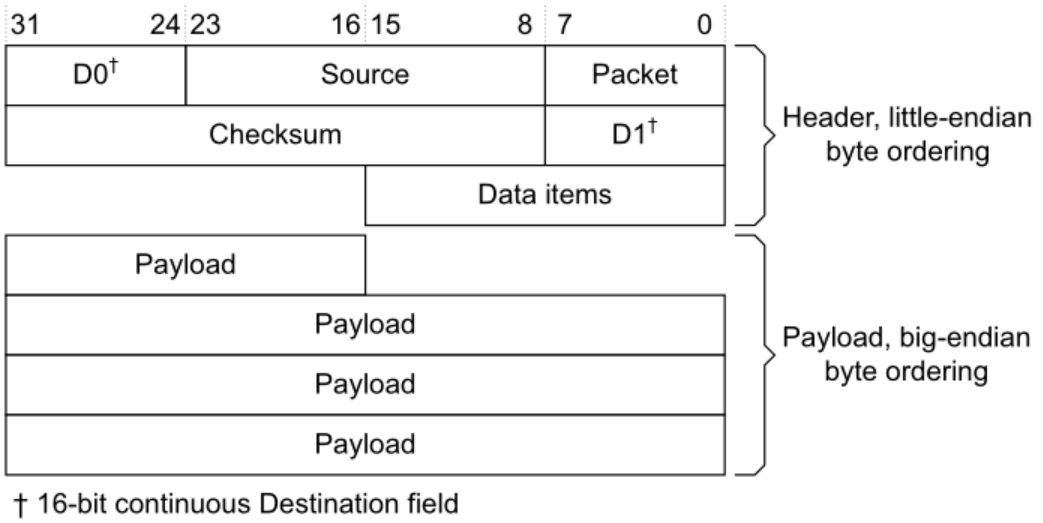


Figure A3-12 Example mixed-endian data structure

例如，在这种结构中，Data items是一个两字节的小端元素，这意味着它的最低地址是它的LSB

字节不变性的使用确保了对有效负载的大端访问不会破坏小端元素。

非对齐传输

AXI支持非对齐的传输。对于由超过一个字节的数据传输组成的任何突发，访问的第一个字节可能与自然地址边界不对齐。

例如，从字节地址0x1002开始的32位数据分组不与自然的32位地址边界对齐。

主机可以：

- 使用低位地址线发出未对齐的起始地址的信号
- 提供对齐的地址，并使用字节通道选通信号来标记未对齐的起始地址

Note：

低位地址线上的信息必须与字节选通通道上的信息一致。

从属设备不需要基于来自主设备的任何对齐信息采取特殊操作。

图A3-13显示了32位总线上有对齐和无对齐的32位传输的增量突发的例子。

图中的每一行代表一次传输，阴影单元表示没有传输的字节。

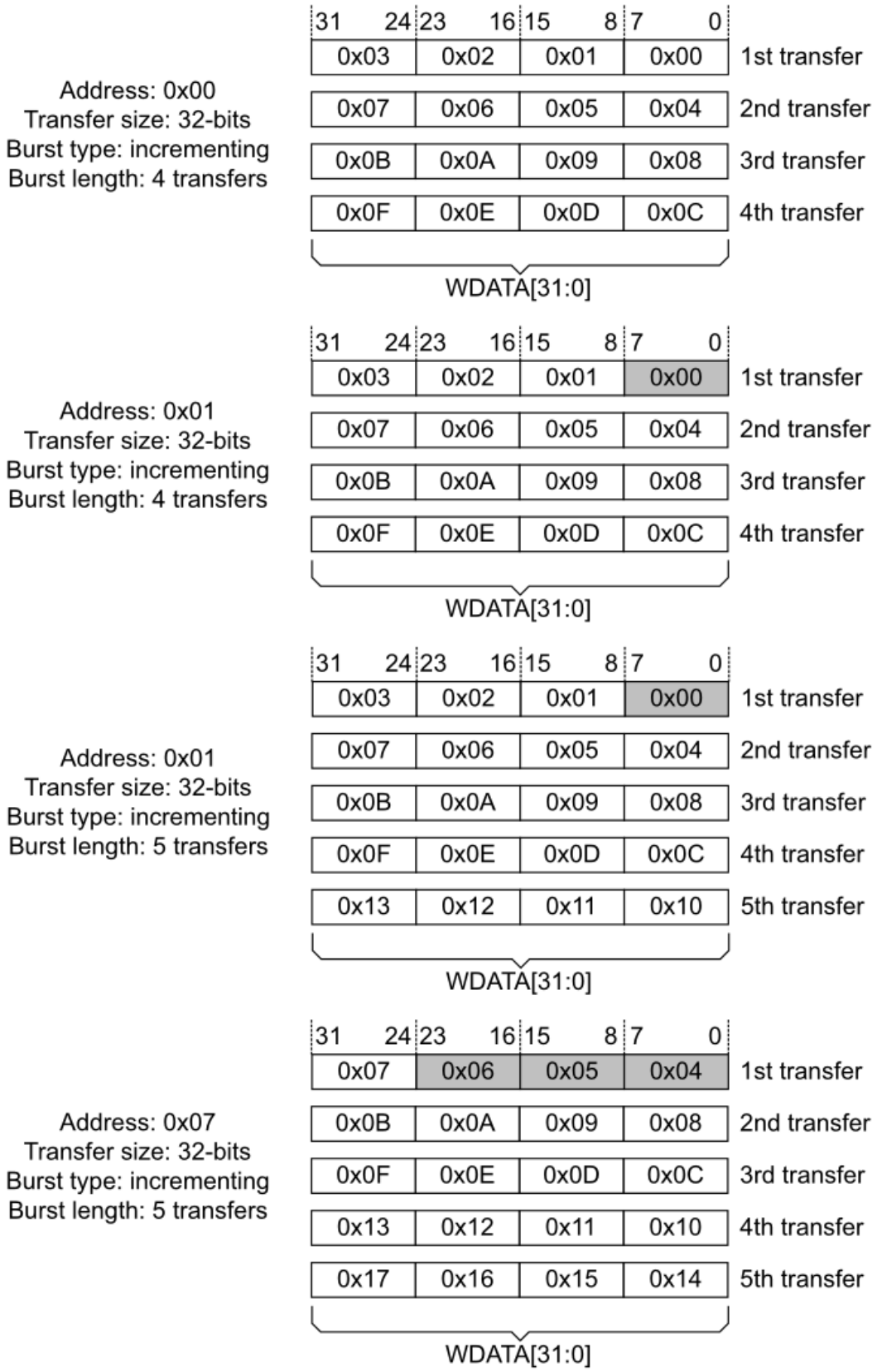


Figure A3-13 Aligned and unaligned transfers on a 32-bit bus

图A3-14显示了64位总线上具有对齐和未对齐的32位传输的递增突发的示例。

图中的每一行表示传输，阴影单元格表示未传输的字节

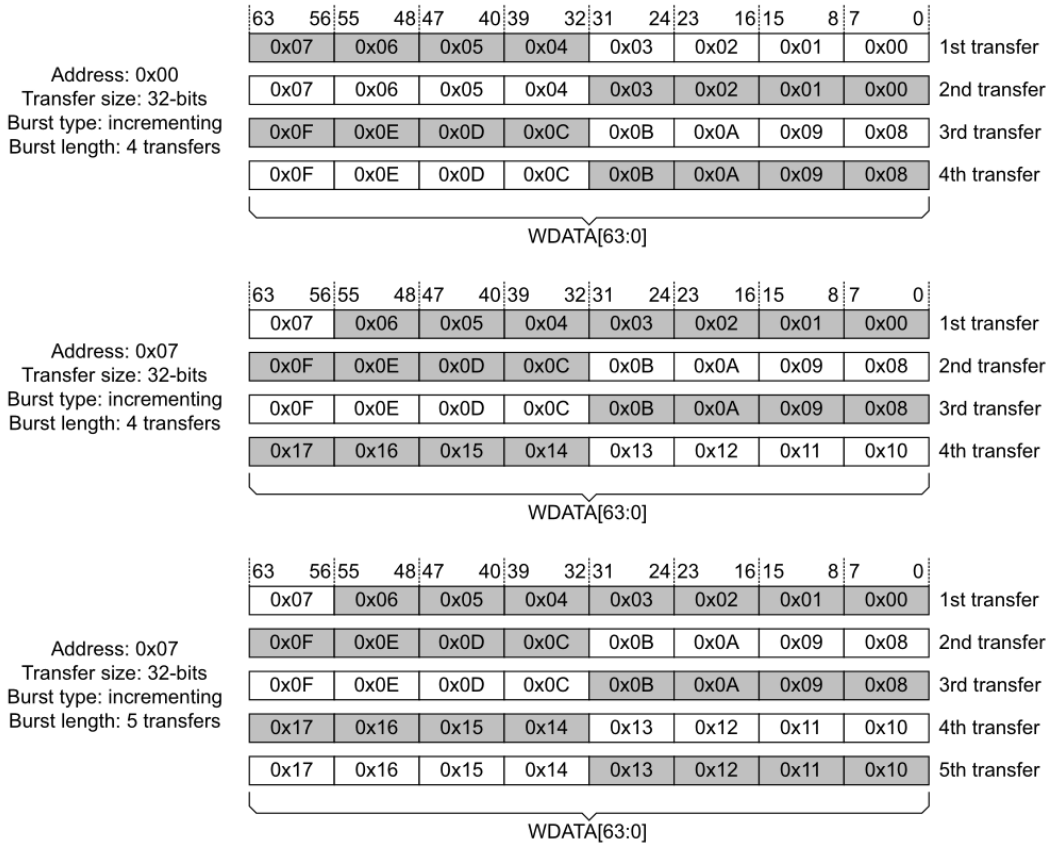


Figure A3-14 Aligned and unaligned transfers on a 64-bit bus

图A3-15显示了一个回转突发的例子，在一个64位总线上有对齐的32位传输。

图中每一行代表一次传输，阴影部分表示未传输的字节。

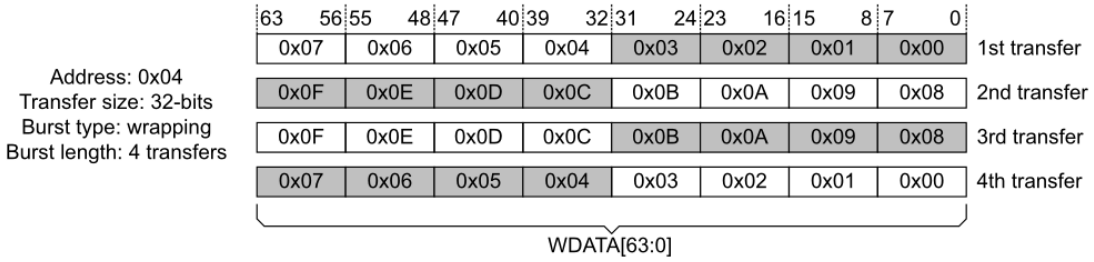


Figure A3-15 Aligned wrapping transfers on a 64-bit bus

A3.4.4 读写回复结构

AXI协议为读写事务提供响应信号：

- 对于读事务，从机的响应信息在读数据通道上发出信号
- 对于写交易，响应信息在写响应通道上发出信号

响应的信号是由：

- **RRESP[1:0]**，对于读传输
- **BRESP[1:0]**，对于写传输

回复的内容：

OKAY 正常访问成功。表示正常访问已成功。也可以表示独占访问失败

EXOKAY 独占访问正常。表示独占访问的读或写部分都已成功

SLVERR 从机错误。当访问已成功到达从机，但从机因为各种原因希望向发起的主机返回一个错误条件时使用

DECERR 解码错误。通常由一个互连（interconnect）组件产生，以表明在交易地址没有从属设备

表A3-4显示了**RRESP**和**BRESP**信号的编码情况

Table A3-4 RRESP and BRESP encoding

RRESP[1:0] BRESP[1:0]	Response
0b00	OKAY
0b01	EXOKAY
0b10	SLVERR
0b11	DECERR

对于一个写事务，对整个突发事件（burst）发出一个响应信号，而不是对突发事件中的每个数据传输（transfer）发出信号。

对于一个读事务，从属设备可以为一个突发中的不同传输发出不同的响应信号。
例如，在一个16次读传输的突发中，从属设备可能对15次传输返回一个**OKAY**响应，对其中一次传输返回一个**SLVERR**响应

该协议规定，即使报告了错误，也必须执行所需数量的数据传输。
例如，如果要求从属机构进行8次传输的读取，但从属机构有一个错误条件，从属机构必须执行8次数据传输，每次都有一个错误响应。如果从属设备给出一个错误响应，则突发的剩余部分不会被取消。

OKAY 正常访问成功

OKAY响应表示以下任何一种情况：

- 正常访问的成功
- 独占访问的失败
- 对不支持独占访问的从机的独占访问

OKAY是对大多数事务的回应

EXOKAY 独占访问成功

一个**EXOKAY**响应表示独占访问的成功

SLVERR 从机错误

SLVERR响应表示一个不成功的传输事务

为了简化系统监控和调试，本规范建议ERROR响应仅用于错误条件，而不用于通知正常的预期事件。以下是从设备错误情况的示例：

- FIFO或缓冲区超载或欠载条件
- 尝试不支持的传输大小
- 尝试对只读位置进行写访问
- 从机中的超时条件
- 尝试对已禁用或已断电的功能访问

DECERR 解码错误

DECERR响应表示互连组件不能成功解码一个从属访问

如果互连不能成功解码从属访问，它必须返回 **DECERR** 响应。
本规范建议互连网将访问路由到一个默认的从属机构，并且默认的从属机构返回 **DECERR** 响应

AXI协议要求一个事务的所有数据传输都要完成，即使发生错误情况。
任何给出DECERR响应的组件都必须满足这一要求