AMBA Bus Architecture

AXI Specification

Contents

Introduction (1)

- AXI (Advanced eXtensible Interface)
  - ARM IHI 0022C, AMBA AXI Protocol v2.0, ARM Ltd.
    - ✓ https://developer.arm.com/documentation/ihi0022/c
  - AXI uses separate channels for address and data to improve throughput.
  - AXI supports burst-based and out-of-order transactions to enhance efficiency in high-speed systems.

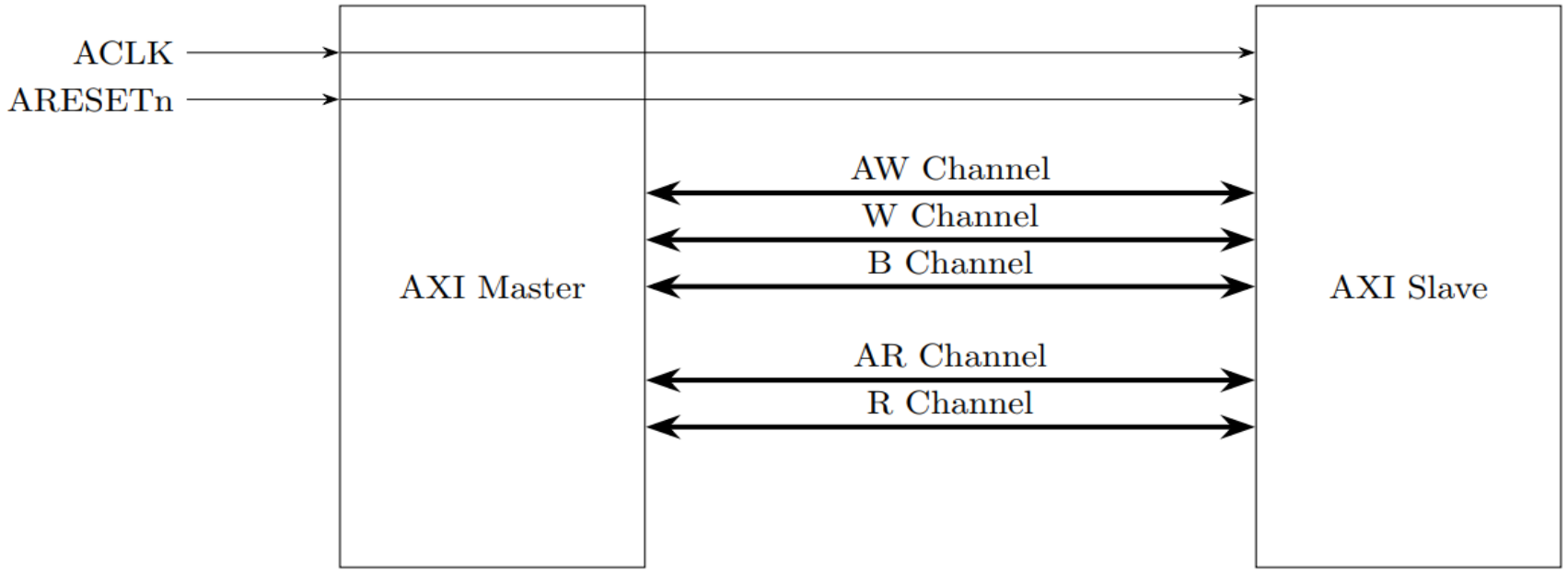
- AXI defines five independent channels.
  - Read Address (AR) / Read Data (R)
  - Write Address (AW) / Write Data (W) / Write Response (B)

Introduction (2)

- The AXI specification has evolved through several versions:
  - AXI3 was introduced in AMBA 3.
    - ✓ It supports burst length up to 16 and write interleaving.
  - AXI4 was defined in AMBA 4.
    - ✓ It increases burst length to 256 and simplifies write transactions.
  - AXI4-Lite is a simplified subset of AXI4.
    - ✓ It removes burst and complex ordering features for simple peripheral access.
    - ✓ It is ideal for register-level communication with low logic cost.
  - AXI4-Stream was introduced for high-speed streaming.
    - ✓ It removes the address channel and supports continuous data flow.
  - AXI5 introduces atomic transactions and barrier transactions.
  - ACE (AXI Coherency Extensions) enables coherent communication between CPU cores and shared memory.

Introduction (3)

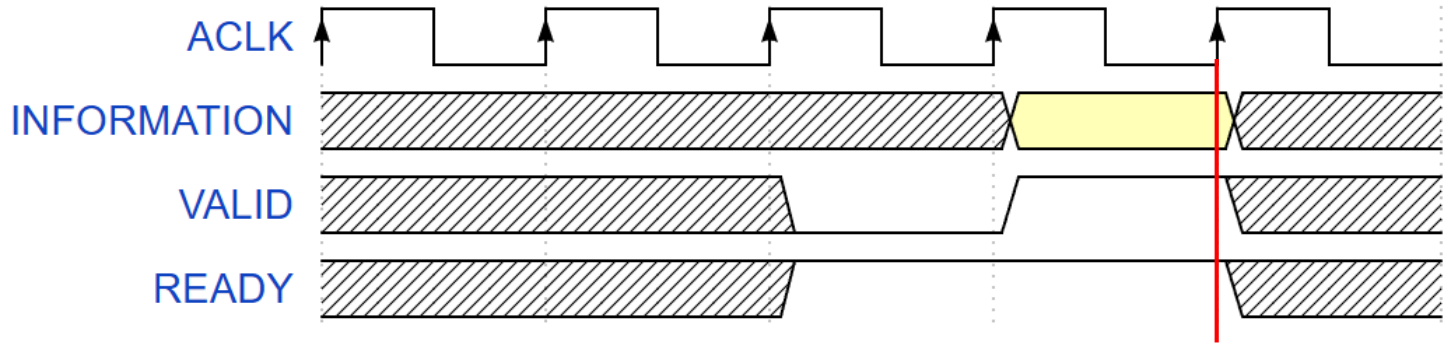
- There are several functional signal categories:
  - Global
  - Write address channel / Write data channel / Write response channel
  - Read address channel / Read data channel



< AXI Master / Slave Interface >

Channel Handshake (1)

- All five AXI channels use a common VALID/READY handshake for transferring data and control signals.
  - This bidirectional flow control allows both the master and slave to manage the data transfer rate.
  - The source asserts VALID when it has data or control information ready, and the destination asserts READY when it is prepared to receive it.
  - A transfer takes place only when both VALID and READY are high.



< AXI Handshake >

Channel Handshake (2)

- Write address channel
  - The master should assert AWVALID only when it presents valid address and control information.
  - It must keep AWVALID high until the slave accepts the transfer by asserting AWREADY.
  - By default, AWREADY can be either HIGH or LOW.
    - ✓ HIGH is recommended because it allows the slave to immediately accept valid addresses.
    - ✓ If AWREADY defaults to LOW, at least two clock cycles are needed—one for the master to assert AWVALID, and another for the slave to respond with AWREADY.

Channel Handshake (3)

- Write data channel
  - During a write burst, the master should assert WVALID only when it provides valid write data.
  - It must keep WVALID high until the slave signals readiness by asserting WREADY.
  - The master must assert WLAST when sending the final data beat of the burst.

- Write response channel
  - The slave should assert BVALID only when it provides a valid write response.
  - It must keep BVALID high until the master acknowledges it by asserting BREADY.

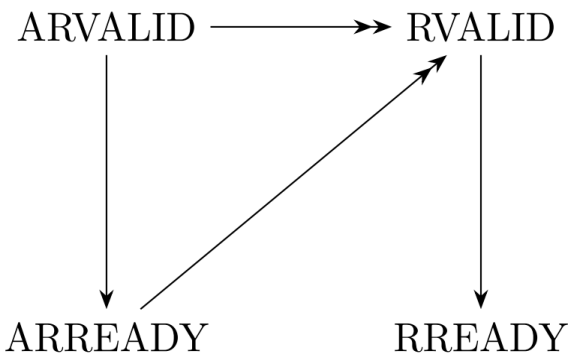
Channel Handshake (4)

- Read address channel
  - The master should assert ARVALID only when it presents valid address and control information.
  - It must keep ARVALID high until the slave accepts the transfer by asserting ARREADY.
  - By default, ARREADY can be either HIGH or LOW.

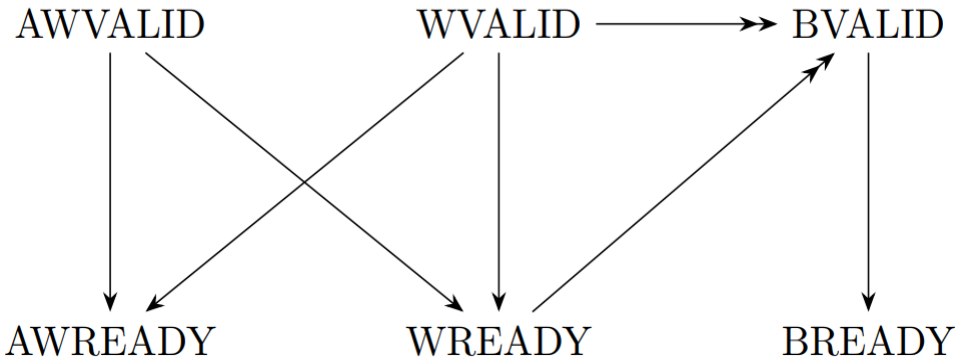
- Read data channel
  - The slave should assert RVALID only when it provides valid read data.
  - It must keep RVALID high until the master accepts the data by asserting RREADY.
  - The slave must assert RLAST when sending the final data beat of the read burst.

Channel Handshake (5)

- Relationships between the channels
  - The timing between the address, read, write, and write response channels is flexible.
    - ✓ For instance, write data may arrive before its associated write address if the address channel has more pipeline stages than the data channel.
  - Two ordering rules must always be followed:
    - ✓ Read data must be returned after the corresponding read address.
    - ✓ A write response must follow the final write data beat of its associated write transaction.



< Read transaction handshake dependencies >



< Write transaction handshake dependencies >

Contents

AXI-Lite Specification

- AXI4-Lite is intended for simple, low-bandwidth communication between a master and one or more slave devices.
  - Unlike the full AXI4 protocol, AXI4-Lite does not support burst transactions, quality-of-service signaling, or transaction IDs.
    - ✓ By removing complex features, AXI4-Lite reduces resource requirements and eases implementation in both hardware and verification environments.
    - ✓ This simplification makes it particularly well-suited for register-level access to peripherals such as timers, GPIO controllers, and UART interfaces.
  - Due to its simplicity, AXI4-Lite is commonly used in systems where configuration and control registers need to be accessed reliably, but high throughput is not required.

Signal Descriptions (1)

- Global signals

Signal	Meaning	Description
ACLK	Clock	All transfers on the AXI are synchronized to the rising edge of ACLK.
ARESETn	Reset	The AXI reset signal is active LOW.

- Write address channel signals (AW)

Signal	Meaning	Description
AWADDR[31:0]	Address	Specifies the address for the write transfer.
AWPROT[2:0]	Protection type	Indicates the privilege level, security, and access type (e.g., instruction or data).
AWVALID	Address valid	Master asserts this to indicate that the write address is valid.
AWREADY	Address ready	Slave asserts this to indicate that it is ready to accept the address.

Signal Descriptions (2)

- Write data channel signals (W)

Signal	Meaning	Description
WDATA[31:0]	Write data	Contains the actual data to be written to the slave.
WSTRB[3:0]	Write strobes	Indicates which byte lanes of WDATA are valid.
WVALID	Write valid	Master asserts this to signal that WDATA is valid.
WREADY	Write ready	Slave asserts this to signal it can accept WDATA.

- Write response channel signals (B)

Signal	Meaning	Description
BRESP[1:0]	Write response	Indicates the status of the write transaction (OKAY, SLVERR, DECERR).
BVALID	Response valid	Slave asserts this to indicate that BRESP is valid.
BREADY	Response ready	Master asserts this to signal it can accept the response.

Signal Descriptions (3)

- Read address channel signals (AR)

Signal	Meaning	Description
ARADDR[31:0]	Address	Specifies the address for the read transfer.
ARPROT[2:0]	Protection type	Indicates the privilege level, security, and access type (e.g., instruction or data).
ARVALID	Address valid	Master asserts this to indicate that the write address is valid.
ARREADY	Address ready	Slave asserts this to indicate that it is ready to accept the address.

- Read data channel signals (R)

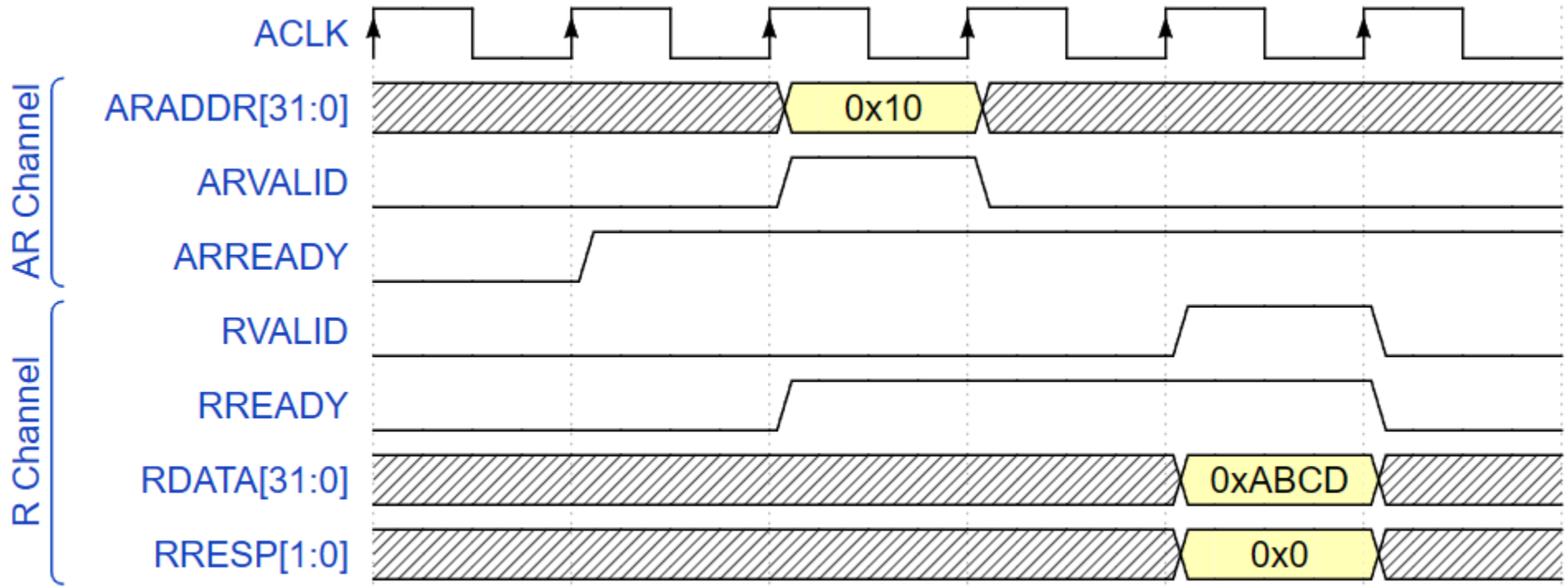
Signal	Meaning	Description
RDATA[31:0]	Read data	Contains the data returned from the slave to the master.
RRESP	Read response	Indicates the status of the read transfer (OKAY, SLVERR, DECERR).
RVALID	Read valid	Slave asserts this to signal it can accept RDATA.
RREADY	Read ready	Master asserts this to signal that RDATA is valid.

Read Transaction (1)

- A read transaction proceeds as follows:
  - AR Channel
    - ✓ 1. The master places a valid read address on the ARADDR bus and asserts ARVALID.
    - ✓ 2. The slave asserts ARREADY when it is ready to accept the address.
    - ✓ 3. When both ARVALID and ARREADY are high, the address is accepted.
  - R Channel
    - ✓ 4. The slave provides the read data on RDATA, sets RVALID high, and includes the response status in RRESP.
    - ✓ 5. The master asserts RREADY when it is ready to receive the data.
    - ✓ 6. The transaction completes when both RVALID and RREADY are high.

Read Transaction (2)

- A read transaction proceeds as follows:
  - RRESP[1:0] = 0x0 (OKAY) means a normal access has been successful.



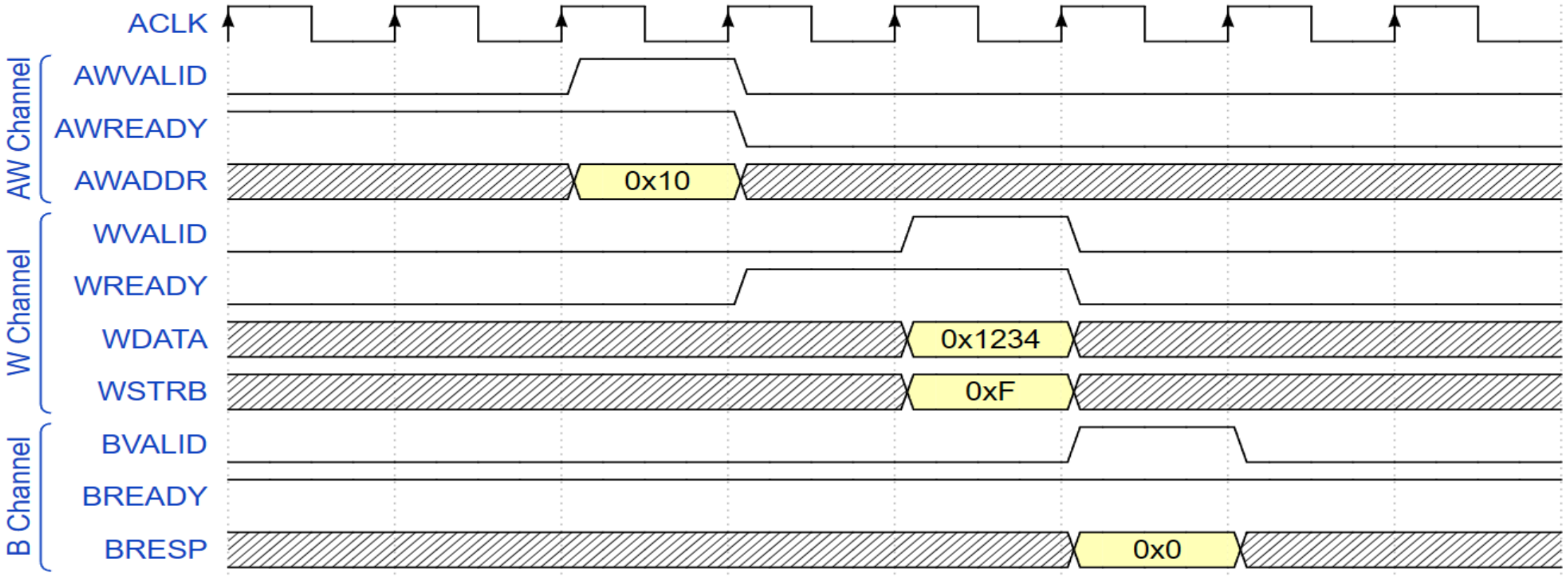
< Read Transaction >

Write Transaction (1)

- A write transaction proceeds as follows:
  - AW Channel
    - ✓ 1. The master places a valid write address on the AWADDR bus and asserts AWVALID.
    - ✓ 2. The slave asserts AWREADY when it is ready to accept the address.
    - ✓ 3. When both AWVALID and AWREADY are high, the address is accepted.
  - W Channel
    - ✓ 4. The master provides write data on WDATA, sets WSTRB, and asserts WVALID.
    - ✓ 5. The slave asserts WREADY when it is ready to receive the data.
    - ✓ 6. The data is accepted when both WVALID and WREADY are high.
  - B Channel
    - ✓ 7. The slave returns a response on BRESP and asserts BVALID.
    - ✓ 8. The master asserts BREADY to acknowledge the response.
    - ✓ 9. The transaction completes when both BVALID and BREADY are high.

Write Transaction (2)

- A write transaction proceeds as follows:
  - BRESP[1:0] = 0x0 (OKAY) means a normal access has been successful.



< Write Transaction >

Contents

Contents

AXI Specification

- AXI4 provides a more flexible and higher-performance communication interface compared to AXI4-Lite.
- The following are the key differences between AXI4-Lite and AXI4:
  - Burst Support
    - ✓ AXI4 supports burst transactions, allowing multiple data transfers with a single address phase
  - ID Signals
    - ✓ AXI4 includes transaction ID signals (such as AWID, WID, BID, ARID, and RID) to support out-of -order and interleaved transactions.
  - Data Width
    - ✓ AXI4 allows for wider data buses (up to 1024 bits).
  - Interleaving and Reordering
    - ✓ AXI4 enables multiple outstanding transactions that can be reordered or interleaved.

Signal Descriptions - Revisited (1)

- Write address channel signals (AW)

Signal	Meaning	Description
AWID[3:0]	Write ID tag	Identification tag for the write address group.
AWLEN[3:0]	Burst length	The number of data transfers is AWLEN + 1
AWSIZE[2:0]	Burst size	The size of each transfer in bytes is 2^AWSIZE
AWBURST[1:0]	Burst type	b00: Same address every time (FIXED). / b01: Address increases each time (INCR). b10: Address increases, but wraps around (WRAP).
AWLOCK[1:0]	Lock type	b00: Normal access / b01: Exclusive access / b10: Locked access
AWCACHE[3:0]	Cache type	AWCACHE[0] (Bufferable): Allows the transaction to be delayed. AWCACHE[1] (Cacheable): Can be merged. AWCACHE[2] (Read Allocate): Allocate in cache if read misses. AWCACHE[3] (Write Allocate): Allocate in cache if write misses

* Total data transferred (in bytes) = (AWLEN + 1) * (2^AWSIZE)

Signal Descriptions - Revisited (2)

- Write data channel signals (W)

Signal	Meaning	Description
WID[3:0]	Write ID tag	The WID must be equal to the AWID of the corresponding write transaction.
WLAST	Write last	Indicates the last transfer in a write burst.

- Write response channel signals (B)

Signal	Meaning	Description
BID[3:0]	Write ID tag	The BID must be equal to the AWID of the corresponding write transaction.

Signal Descriptions - Revisited (3)

- Read address channel signals (AR)

Signal	Meaning	Description
ARID[3:0]	Read ID tag	Identification tag for the read address group.
ARLEN[3:0]	Burst length	The number of data transfers is ARLEN + 1
ARSIZE[2:0]	Burst size	The size of each transfer in bytes is 2^ARSIZE
ARBURST[1:0]	Burst type	b00: Same address every time (FIXED). / b01: Address increases each time (INCR). b10: Address increases, but wraps around (WRAP).
ARLOCK[1:0]	Lock type	b00: Normal access / b01: Exclusive access / b10: Locked access
ARCACHE[3:0]	Cache type	ARCACHE[0] (Bufferable): Allows the transaction to be delayed. ARCACHE[1] (Cacheable): Can be merged. ARCACHE[2] (Read Allocate): Allocate in cache if read misses. ARCACHE[3] (Write Allocate): Allocate in cache if write misses

* Total data transferred (in bytes) = (ARLEN + 1) * (2^ARSIZE)

Signal Descriptions - Revisited (4)

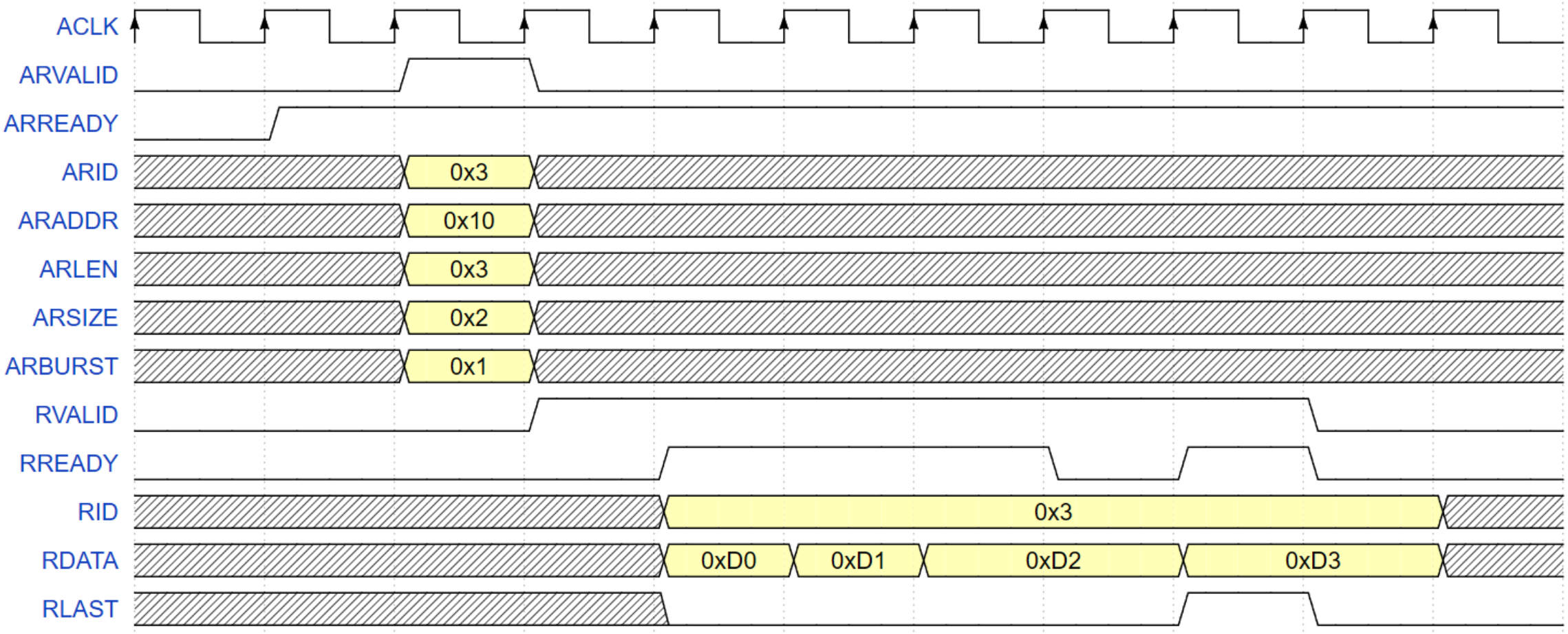
- Read data channel signals (R)

Signal	Meaning	Description
RID[3:0]	Read ID tag	The RID must be equal to the ARWID of the corresponding read transaction.
RLAST	Read last	Indicates the last transfer in a read burst.

- Relationship among IDs
  - By matching the IDs, the system can correctly associate each transaction with its corresp onding address, data, and response—even when multiple operations are in progress.
  - For write bursts, AWID, WID, and BID must be the same to ensure proper tracking of the a ddress, data, and response.
  - For read bursts, ARID and RID must match to ensure proper association between the add ress and the returned data.

Burst Transfer (1)

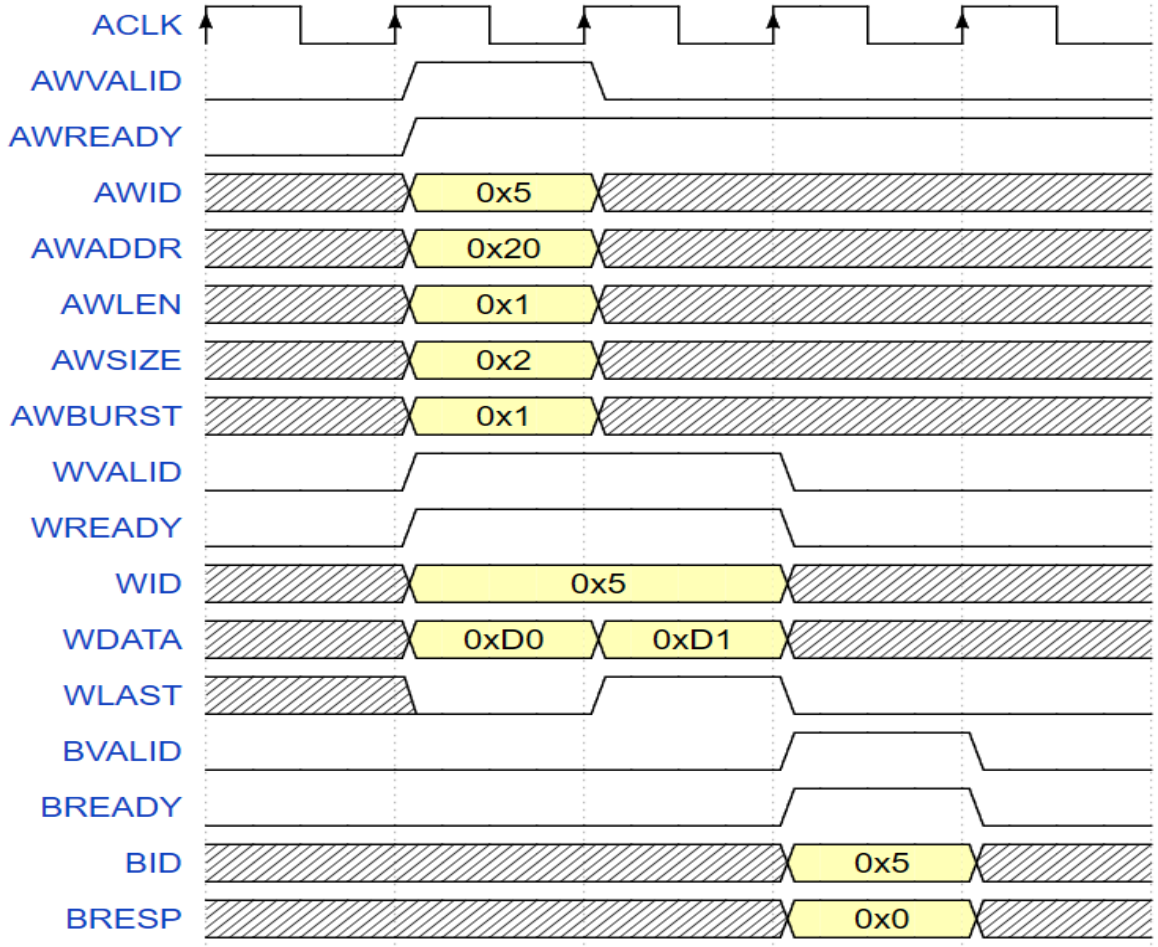
- A read transaction for a burst transfer proceeds as follows:



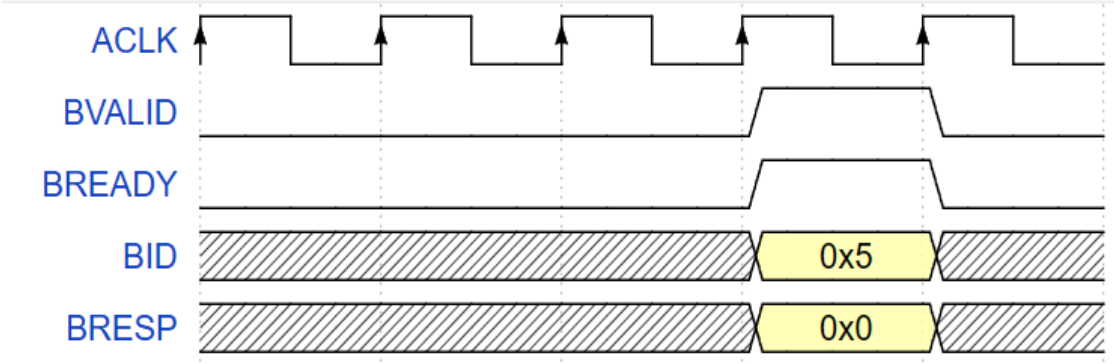
< Read Transaction - Burst Transfer (4 beats x 4 bytes) >

Burst Transfer (2)

- A write transaction for a burst transfer proceeds as follows:



< Write Transaction - Burst Transfer (2 beats x 4 bytes) >



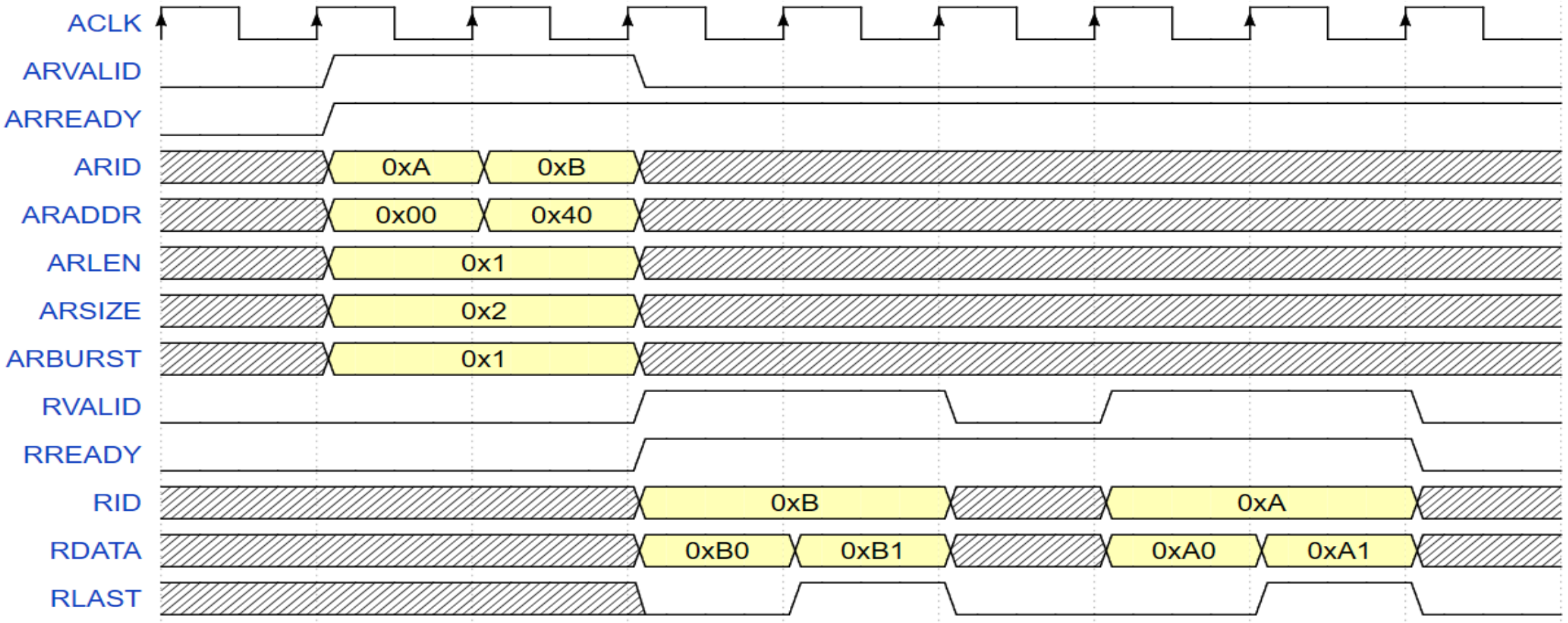
< Write Transaction - Burst Transfer (2 beats x 4 bytes) >

Reordering (1)

- The AXI protocol supports out-of-order completion and multiple outstanding transactions, which enhances system performance by allowing high-throughput communication.
- To support this, AXI uses ID signals to distinguish between transactions.
  - These IDs allow each port to behave as if it supports multiple independent transaction streams.
- Transactions with the same ID must be completed in order, but there are no ordering constraints between different IDs.
  - This allows masters to issue new transactions before previous ones are completed, and slaves to complete faster transactions earlier, depending on memory latency.

Reordering (2)

- A read transaction for reordering proceeds as follows:



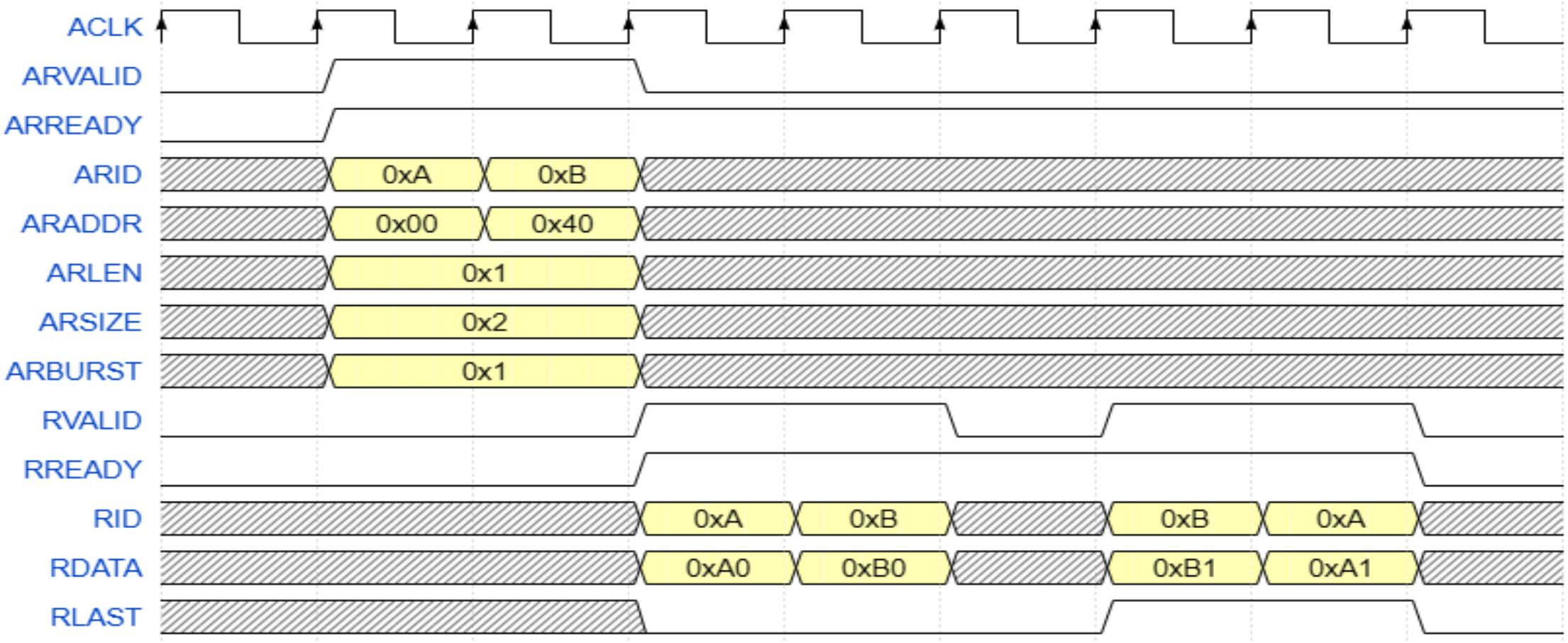
< Read Transaction - Reordering >

Interleaving (1)

- The AXI protocol allows interleaving of data beats from multiple transactions.
  - This improves bandwidth utilization by overlapping slow and fast memory accesses.
- Interleaving enables the system to switch between active transactions at the beat level.
  - Transactions with different IDs can be interleaved freely.
  - Each stream is still ordered within the same ID, but different IDs can deliver beats alternately.
- Ex) If two read bursts with ID=0xA and ID=0xB are in progress, the data may return in the order A0, B0, A1, B1.
- Most AXI-compliant slaves do not support interleaved write transfers from multiple transactions.

Interleaving (2)

- A read transaction for reordering proceeds as follows:



< Read Transaction - Reordering and Interleaving>

Contents

Exercise 1 (1)

- Write strobe
  - The WSTRB (write strobe) signals allow selective byte-level updates during a write transfer.
  - Each bit in WSTRB corresponds to a byte in the write data bus, and when set, it indicates that the corresponding byte should be written to memory.
  - WSTRB[n] corresponds to WDATA[(8 × n) + 7: (8 × n)].
  - This enables partial or sparse writes.

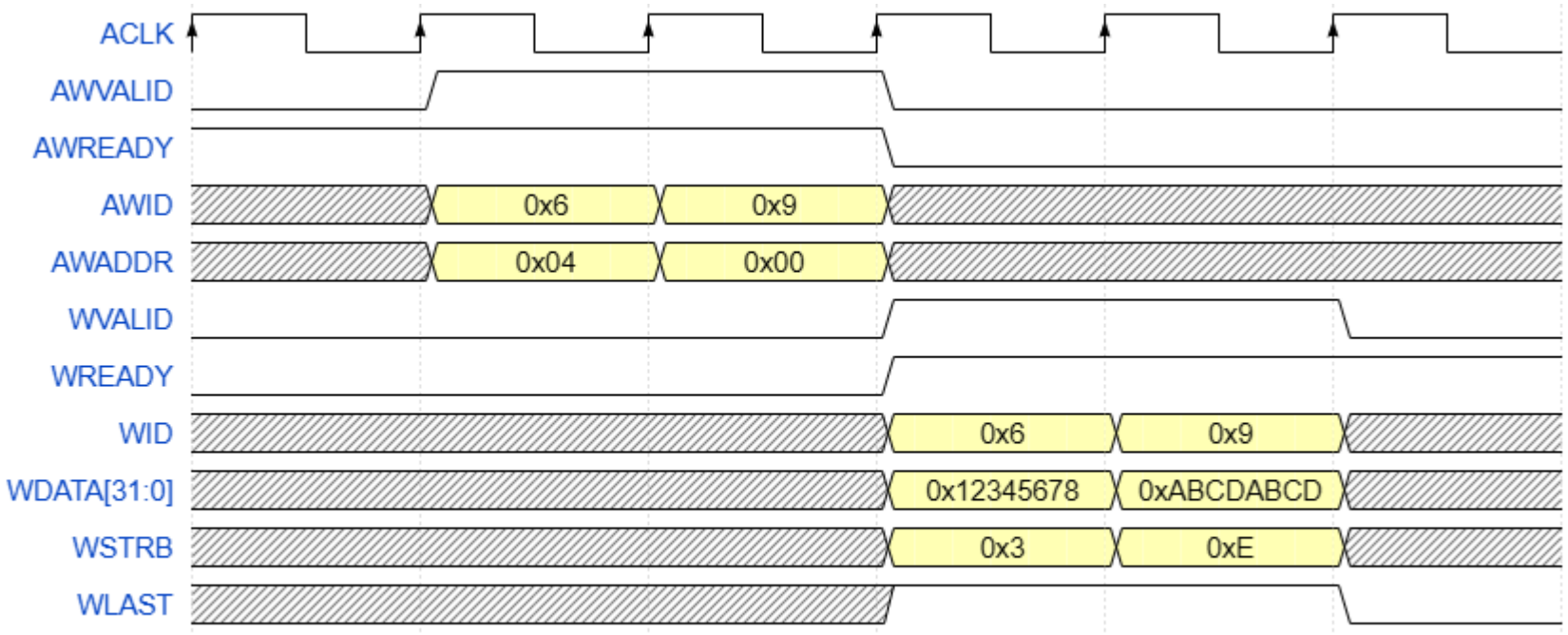
[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
7	6	5	4	3	2	1	0

  - Ex) If you write to bits [63:40] of the data bus, the WSTRB should be set to 0b11100000 (0xE0). Only the upper three bytes are valid and should be written to memory.

Exercise 1 (2)

- There is an AXI write transaction waveform below.
  - The write data bus is 32 bits wide, and memory is initially zeroed.
- What is the resulting memory content after write?

Address	Value
0x00	0x00000000
0x04	0x00000000

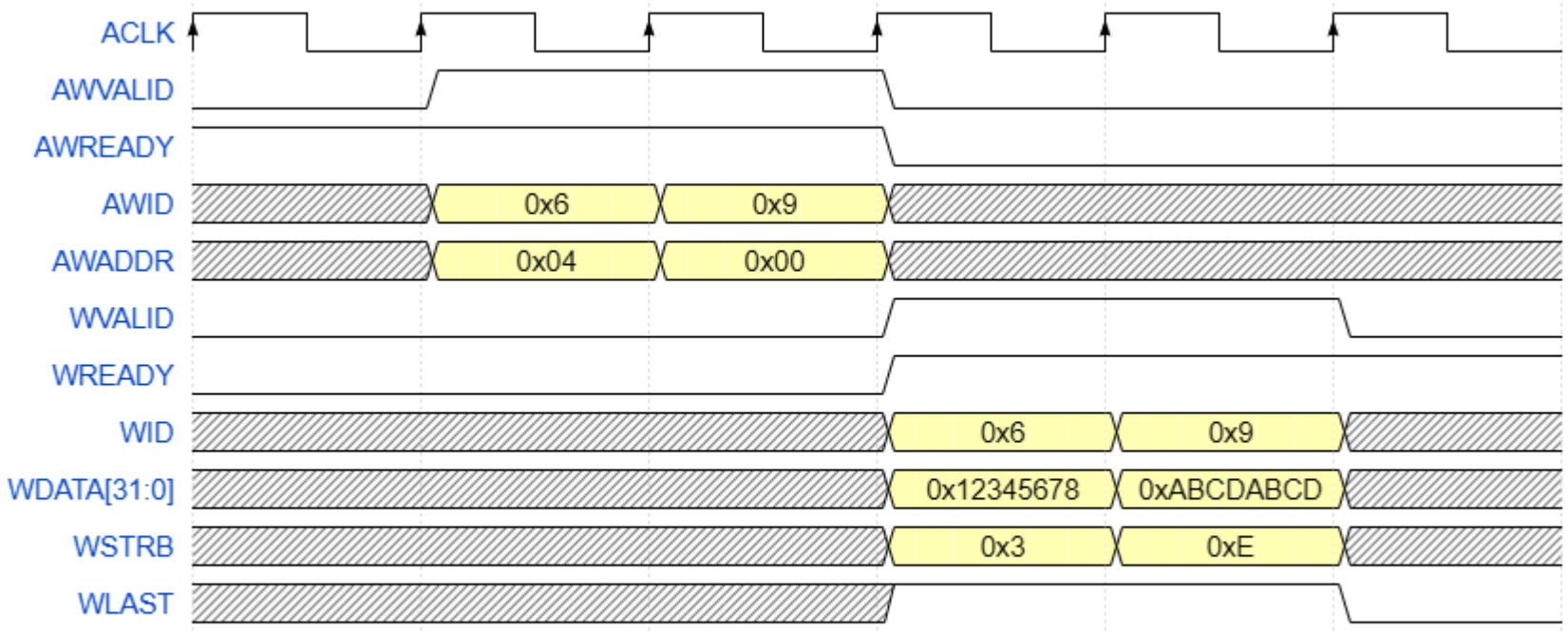


< Write Transaction - Write Strobe >

Exercise 1 (2)

- There is an AXI write transaction waveform below.
  - The write data bus is 32 bits wide, and memory is initially zeroed.
- What is the resulting memory content after write?

Address	Value
0x00	0xABCDAB00
0x04	0x00005678



< Write Transaction - Write Strobe >