ARM® AMBA® AXI

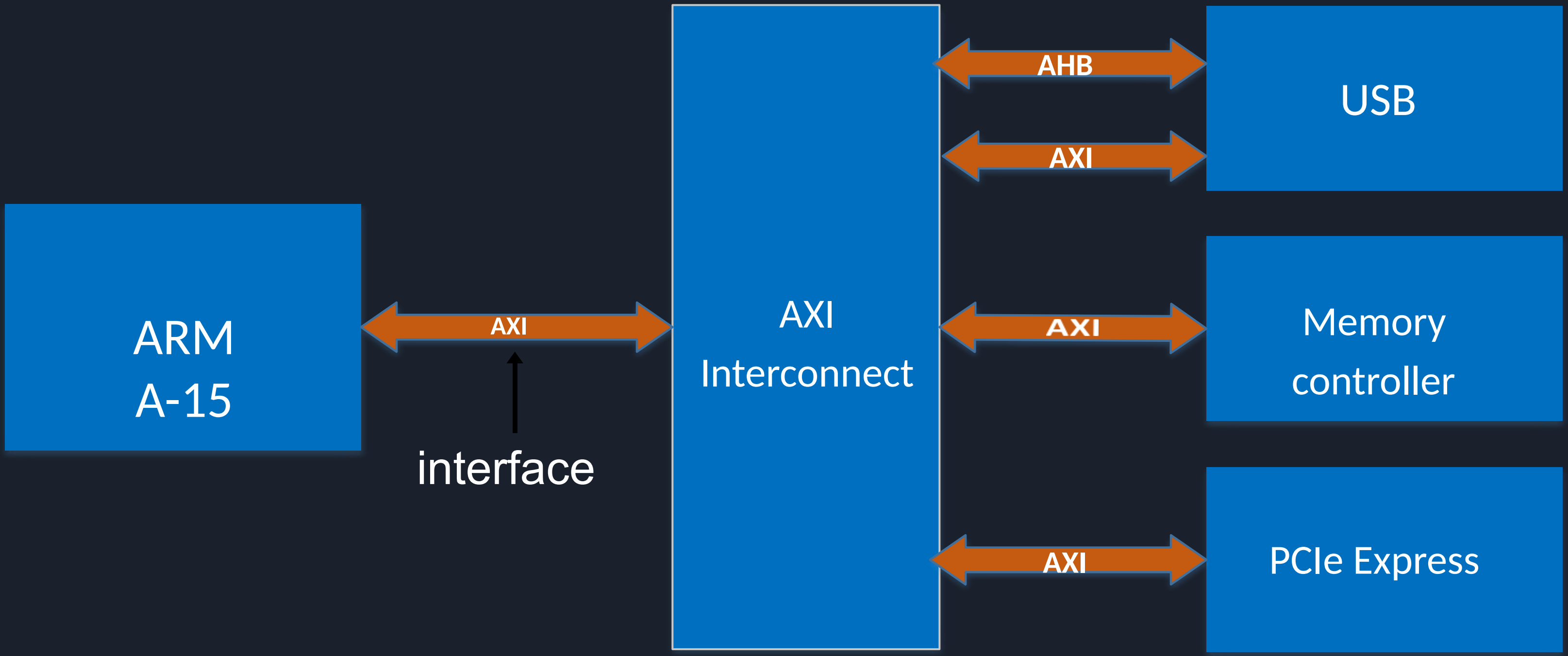
Advanced e**X**tensible **I**nterface

Topics

- Overview of AXI
- AXI architecture and features
- Channel definitions and signals
- Handshaking process
- Basic AXI transactions
- Types of Access (Atomic Accesses)
- Ordering model (Transaction ID tags)
- AXI4 & AXI4 Lite

Overview of AXI

System-On-Chip (SoC)



AXI Overview

Protocol basics

- On-chip communication protocols
- Connects multiple masters and slaves with AXI interconnect.
- **Supports Features:**
  - Burst write & Read transactions
  - Out of order transactions,
  - Parallel Write & Read transactions
  - Over lapped transactions;
  - Align & Unaligned transfers;
  - protected and locked transfers
- **Phased transactions**
- **Write transaction**
  - WR addr phase
  - WR data phase
  - WR response phase
- **Read transaction**
  - RD addr phase
  - RD data & response phase

- Signals classified in channels AW, W, B, AR, R.

Handshaking

- Each channel has valid & ready

  signal.
- AWVALID = 1 @ posedge clk meaning master driving valid information on WR address channel signals.
- AWREADY = 1 (slave driven) meaning slave ready to accept master given addr & ctrl info.
- At clock posedge, when both AWVALID and AWREADY ‘1’ indicate completion of transfer.

Topics

- Overview of AXI
- **AXI Architecture and Features**
- Channel definitions and Signals
- Handshaking process
- Basic AXI transactions
- Types of   Access (Atomic Access)
- Ordering Model (Transaction ID tags)
- AXI4 & AXI4 Lite

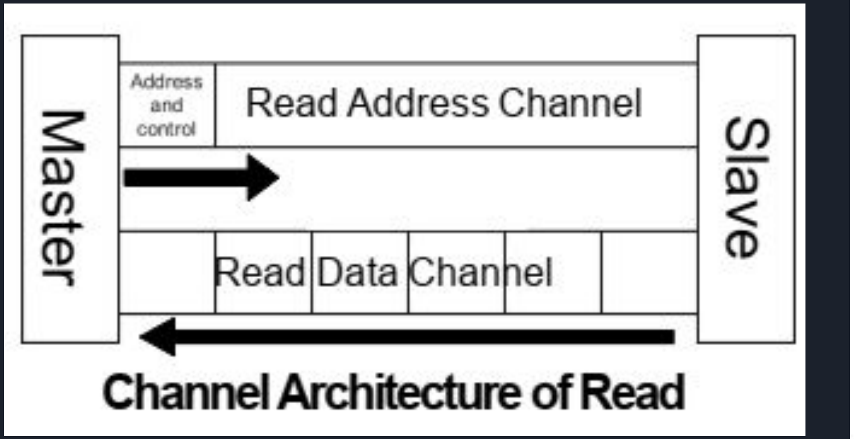
Architecture

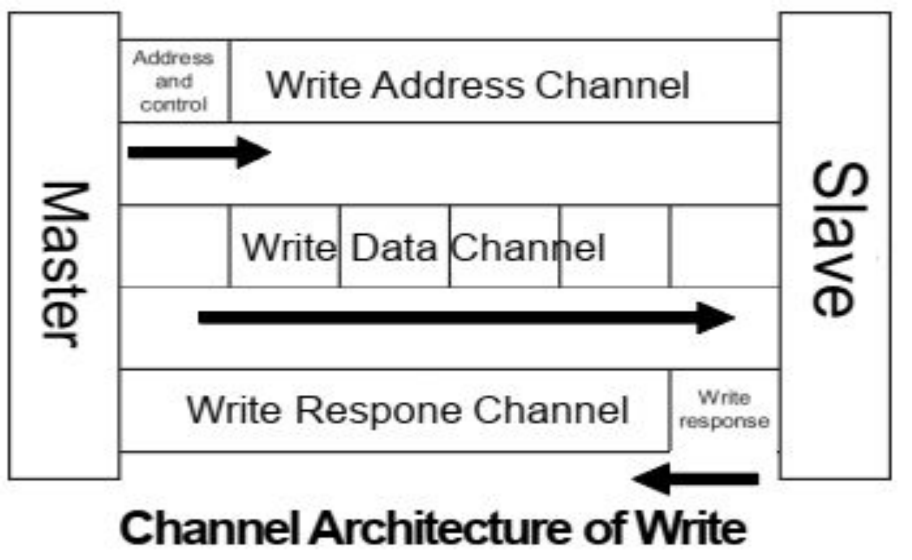
Terminology

- **AXI : A**dvanced e**X**tensible **I**nterface (Interface divided into 5 channels)
- **Protocol :** Set of rules.
- **Channel :** Physical transmission medium (wires) to carry signals.
- **Signal :** It’s an electric current or electromagnetic field that carries information from one place (source) to another (destination).
- **Master :** Initiates transactions.
- **Slave :** receives and responds to transactions.
- **Interface :** A point where two blocks meet and interact.
- **Interconnect :** a device that allows to connect multiple masters and multiple slaves.

Architecture

- AXI is *burst-based* protocol, where *transactions with only start address issued*.
- Address/Control issued ahead of actual data transfer.
- Interface divided into 5 channels.

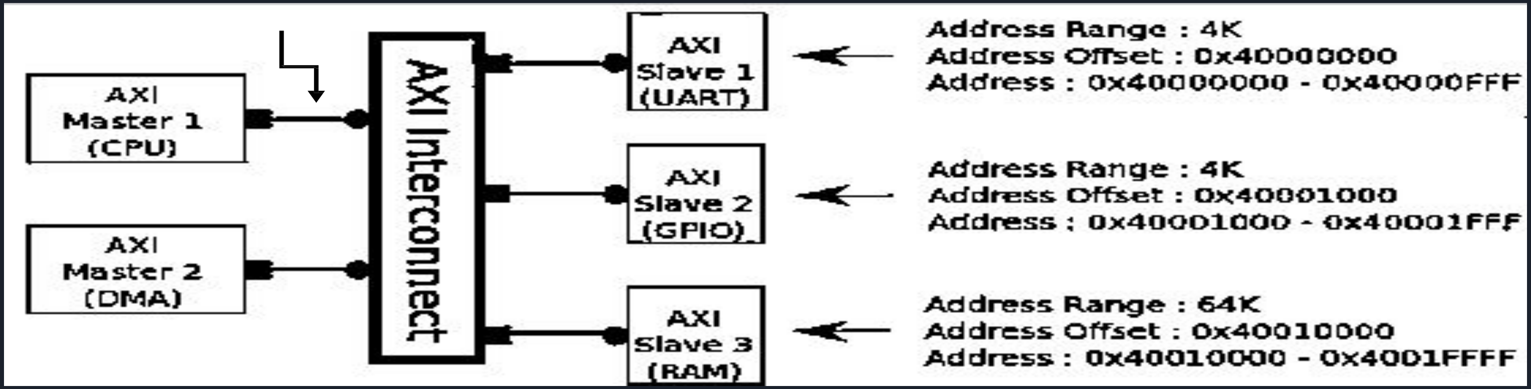




Interface and Interconnect

- Each master should be able to initiate transaction to each of slaves with use of interconnect.
- Interconnect connects **multiple masters to multiple slaves**.
- Interconnect is slave device for masters & master device for Slaves

AXI Interface


AXI Master 1 (CPU)
AXI Master 2 (DMA)
AXI Interconnect
AXI Slave 1 (UART)
AXI Slave 2 (GPIO)
AXI Slave 3 (RAM)
Address Range : 4K
Address Offset : 0x40000000
Address : 0x40000000 - 0x40000FFF
Address Range : 4K
Address Offset : 0x40001000
Address : 0x40001000 - 0x40001FFF
Address Range : 64K
Address Offset : 0x40010000
Address : 0x40010000 - 0x4001FFFF

Source: M.S. Sadri, Zynq Training

Comparing AMBA AHB to AXI Bus-System Modeling

AMBA AHB

- single-channel, shared bus.
- A 128 bit bus running at 400 MHz.
- The AHB bus speed was assumed to be double the AXI Bus, and two times the width.

AMBA AXI

- Multi-channel, read/write optimized bus.
- A 64 bit bus running at 200 MHz
- The primary throughput channels- R/W data channels, while the address, response channels are to improve pipelining of multiple requests.

The simulation study between AMBA AHB and AMBA AXI-Results

Bus Type	***Latency***	***Throughput***	***Power***
AHB Bus	Excellent	Good	Excellent
AXI Bus	Good	Excellent	Good

Features

Terminology

- **Transaction :** Multiple *transfers* of data or an entire *burst* of transfers.
- **Transfer :** A single exchange of information, with one valid and ready handshake.
- **Beat :** An individual data transfer within an AXI burst.
- **Burst :** A transaction where multiple data items are transferred based upon a single address.
- **Aligned :** Each data transfer is aligned to the size of the transfer. E.g.: 32-bit transfer aligned to 4-byte boundaries (0, 4, 8, C).
- **Unaligned :** A data access performed to an address which is not aligned to the size of the access. E.g.: A 32-bit data packet that starts at a byte address of 0x1002 is not aligned to the natural 32-bit address boundary.
- **Multiple outstanding addresses :** Master able to provide transaction addresses without waiting for earlier transactions to complete.
- **Out of order transaction :** Transactions to faster memory regions can complete without waiting for last transaction to slower memory regions
- **Data Interleaving :** Masters producing write data sequence to same slave but data not arriving each clock, interleave to avoid idle cycles on bus.

Features

- **Burst Write** and **Read** transactions.
- **Separate address/control** and **data phases**.
- **Separate Read** and **Write data channels**.
- **unaligned data transfers** using byte strobes.
    - Ex: Accessing 32-bit data starting at address 0x2
- Issues **multiple outstanding addresses**
    - ID signals
- **Out of order transaction** and **Data Interleaving**
    - ID signals

Topics

- Overview of AXI
- AXI Architecture and Features
- Handshaking Process
- Basic AXI Transactions
- Types of Access (Atomic Access)
- Ordering Model (Transaction ID tags)
- AXI4 & AXI4 Lite

Channel Definitions

Channel Definition

- Each channel is **independent** and use **two-way flow control**

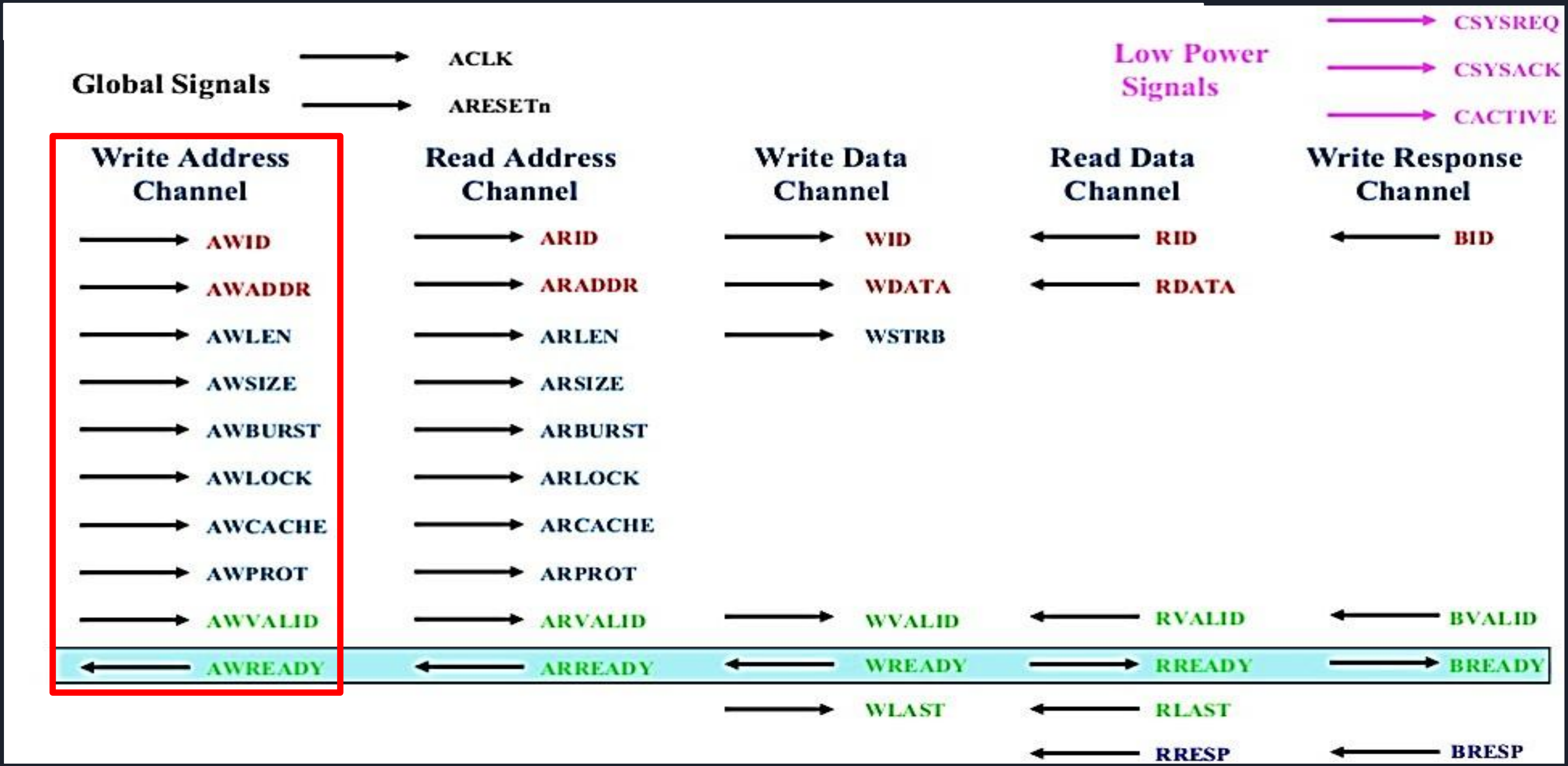
- **VALID**
  - Asserted when valid data or control information available on channel.

- **READY**
  - Asserts when receiver can accept the data.

- **LAST**
  - Asserts while the final data completes

AXI Signals

AXI Signals



Global Signals

→ ACLK

→ ARESETn

Low Power Signals

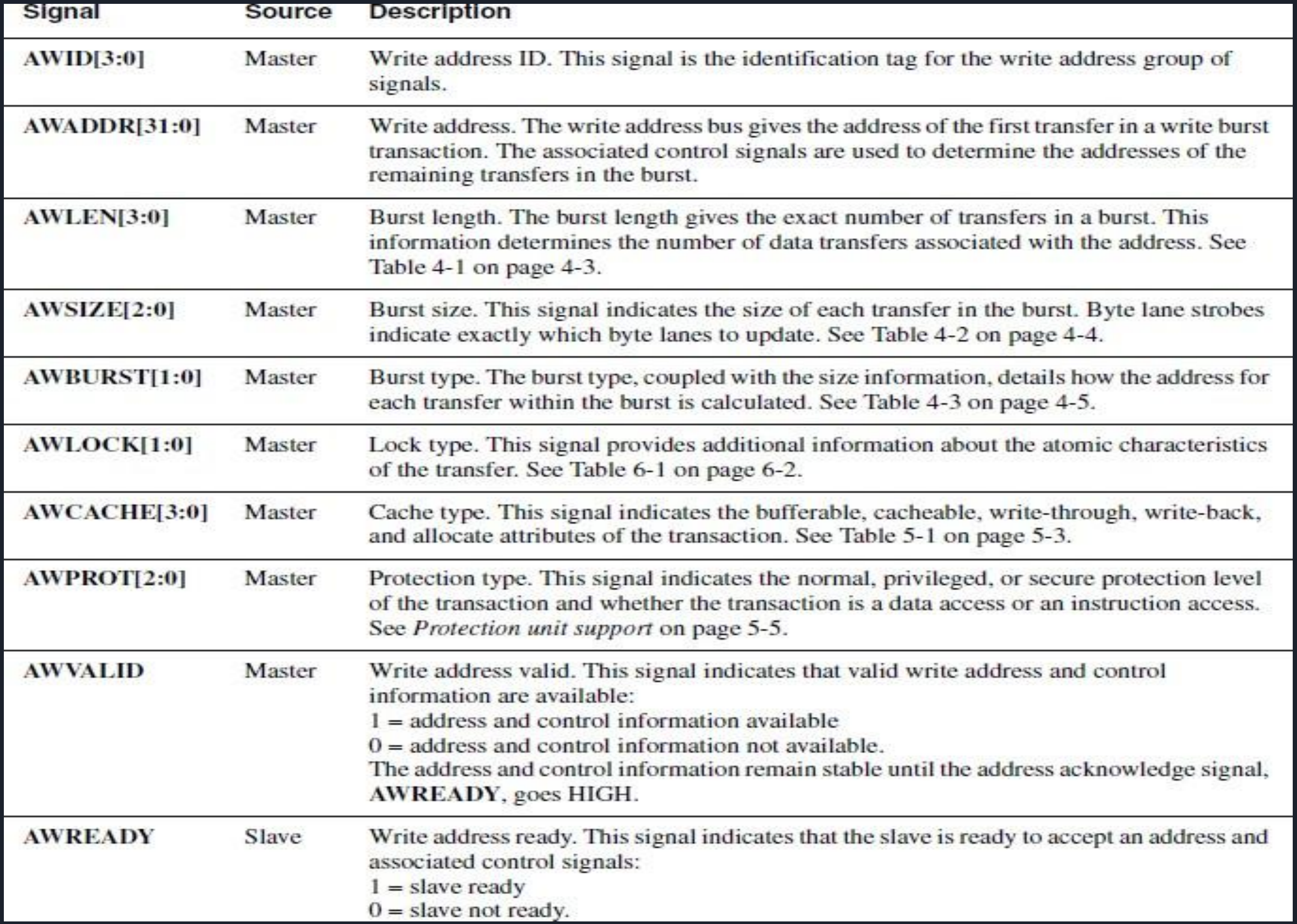
→ CSYSREQ

→ CSYSACK

→ CACTIVE

Write Address Channel	Read Address Channel	Write Data Channel	Read Data Channel	Write Response Channel
→ AWID	→ ARID	→ WID	← RID	← BID
→ AWADDR	→ ARADDR	→ WDATA	← RDATA	
→ AWLEN	→ ARLEN	→ WSTRB		
→ AWSIZE	→ ARSIZE			
→ AWBURST	→ ARBURST			
→ AWLOCK	→ ARLOCK			
→ AWCACHE	→ ARCACHE			
→ AWPROT	→ ARPROT			
→ AWVALID	→ ARVALID	→ WVALID	← RVALID	← BVALID
← AWREADY	← ARREADY	← WREADY	→ RREADY	→ BREADY
		→ WLAST	← RLAST	
			← RRESP	← BRESP

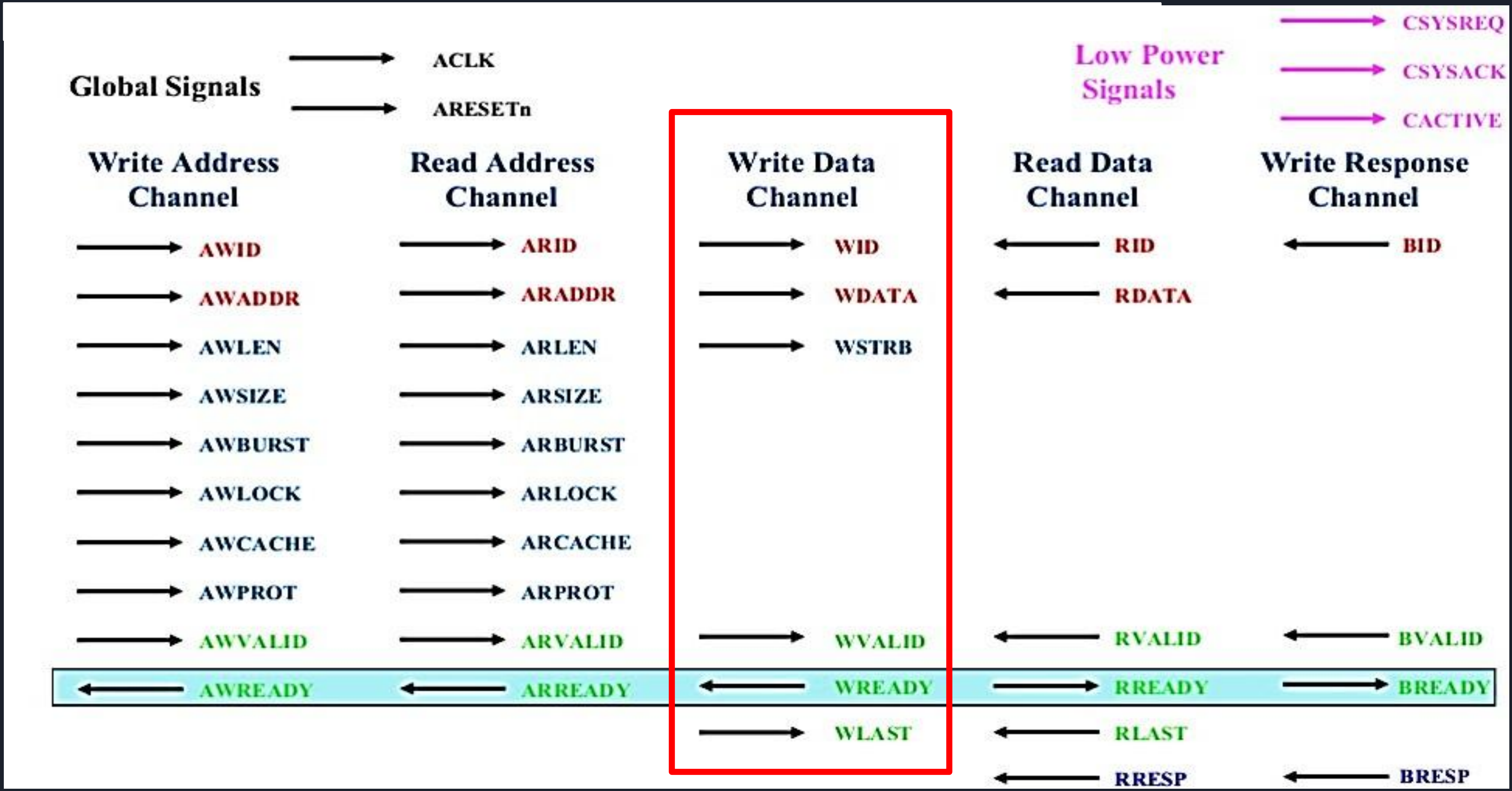
Write Address Channel Signals



Signal	Source	Description
AWID[3:0]	Master	Write address ID. This signal is the identification tag for the write address group of signals.
AWADDR[31:0]	Master	Write address. The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst.
AWLEN[3:0]	Master	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. See Table 4-1 on page 4-3.
AWSIZE[2:0]	Master	Burst size. This signal indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update. See Table 4-2 on page 4-4.
AWBURST[1:0]	Master	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. See Table 4-3 on page 4-5.
AWLOCK[1:0]	Master	Lock type. This signal provides additional information about the atomic characteristics of the transfer. See Table 6-1 on page 6-2.
AWCACHE[3:0]	Master	Cache type. This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction. See Table 5-1 on page 5-3.
AWPROT[2:0]	Master	Protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. See *Protection unit support* on page 5-5.
AWVALID	Master	Write address valid. This signal indicates that valid write address and control information are available: 1 = address and control information available 0 = address and control information not available. The address and control information remain stable until the address acknowledge signal, **AWREADY**, goes HIGH.
AWREADY	Slave	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready 0 = slave not ready.

Source: ARM AMBA AXI Protocol v1.0: Specification

AXI Signals



Global Signals

ACLK

ARESETn

Low Power Signals

CSYSREQ

CSYSACK

CACTIVE

Write Address Channel

AWID
AWADDR
AWLEN
AWSIZE
AWBURST
AWLOCK
AWCACHE
AWPROT
AWVALID
AWREADY

Read Address Channel

ARID
ARADDR
ARLEN
ARSIZE
ARBURST
ARLOCK
ARCACHE
ARPROT
ARVALID
ARREADY

Write Data Channel

WID
WDATA
WSTRB
WVALID
WREADY
WLAST

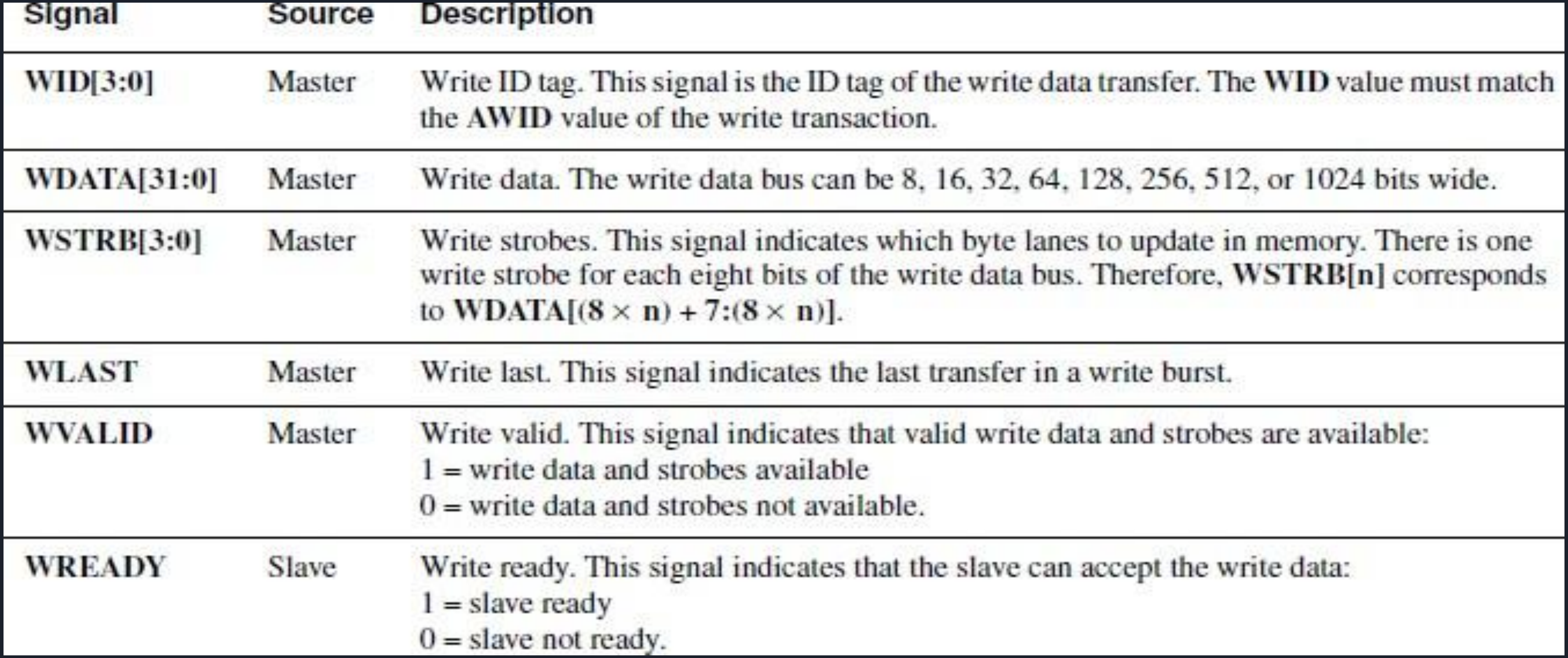
Read Data Channel

RID
RDATA
RVALID
RREADY
RLAST
RRESP

Write Response Channel

BID
BVALID
BREADY
BRESP

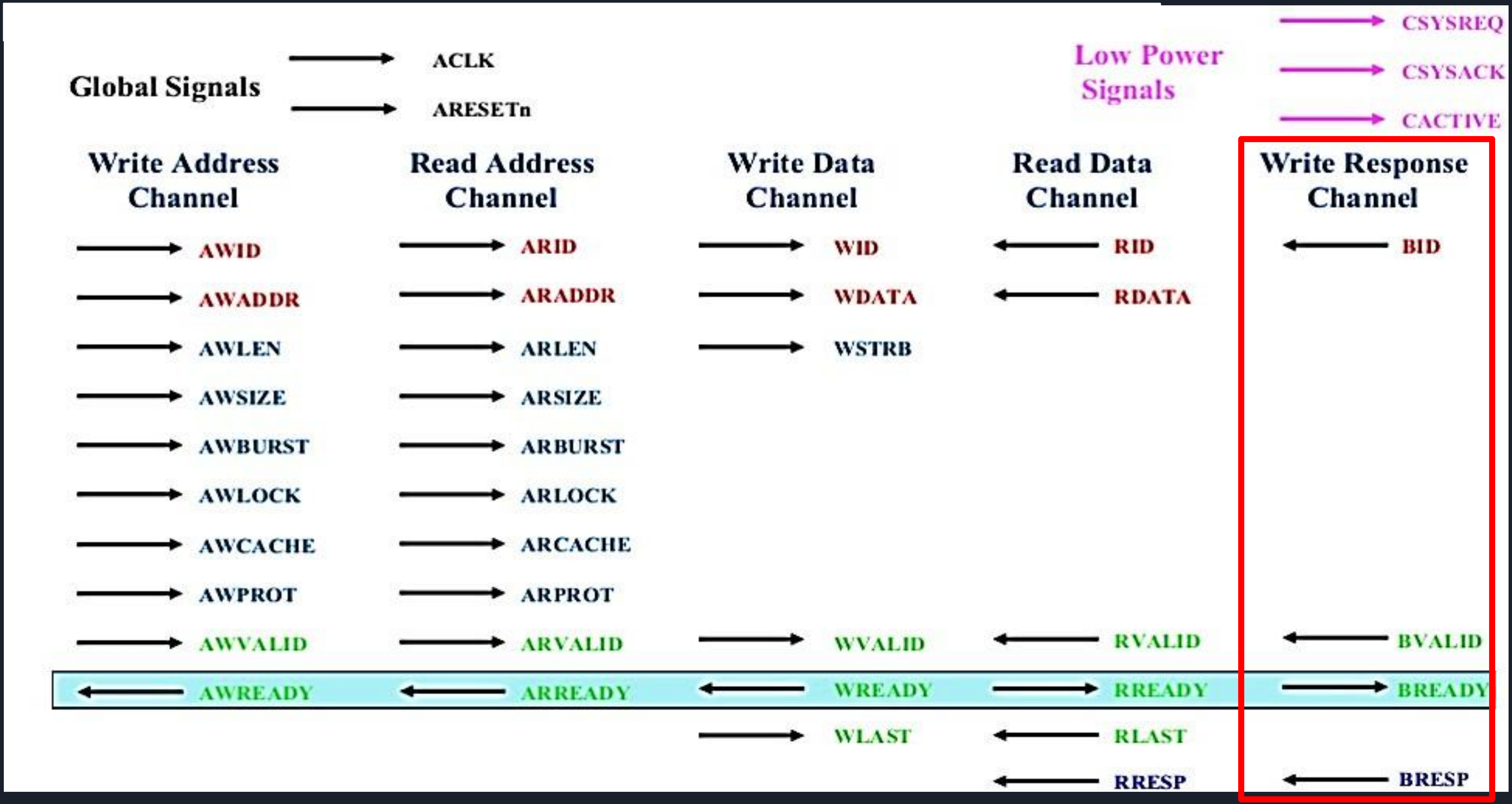
Write Data Channel Signals



Signal	Source	Description
WID[3:0]	Master	Write ID tag. This signal is the ID tag of the write data transfer. The **WID** value must match the **AWID** value of the write transaction.
WDATA[31:0]	Master	Write data. The write data bus can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide.
WSTRB[3:0]	Master	Write strobes. This signal indicates which byte lanes to update in memory. There is one write strobe for each eight bits of the write data bus. Therefore, **WSTRB[n]** corresponds to **WDATA[(8 × n) + 7:(8 × n)]**.
WLAST	Master	Write last. This signal indicates the last transfer in a write burst.
WVALID	Master	Write valid. This signal indicates that valid write data and strobes are available: 1 = write data and strobes available 0 = write data and strobes not available.
WREADY	Slave	Write ready. This signal indicates that the slave can accept the write data: 1 = slave ready 0 = slave not ready.

Source: ARM AMBA AXI Protocol v1.0: Specification

AXI Signals



Global Signals

→ ACLK

→ ARESETn

Low Power Signals

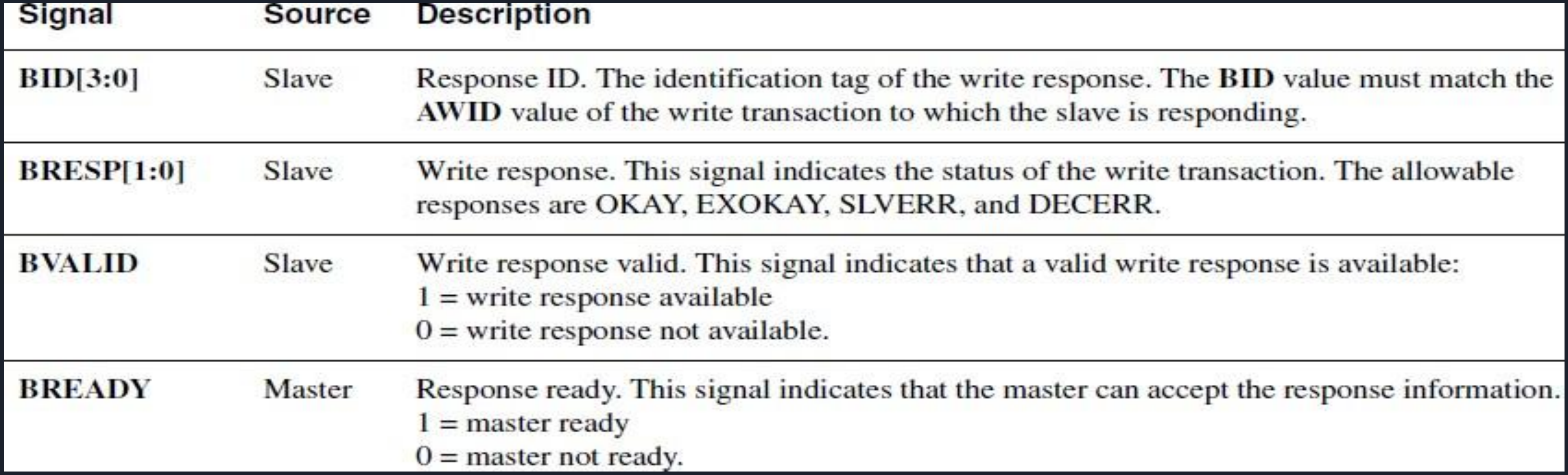
→ CSYSREQ

→ CSYSACK

→ CACTIVE

Write Address Channel	Read Address Channel	Write Data Channel	Read Data Channel	Write Response Channel
→ AWID	→ ARID	→ WID	← RID	← BID
→ AWADDR	→ ARADDR	→ WDATA	← RDATA	
→ AWLEN	→ ARLEN	→ WSTRB		
→ AWSIZE	→ ARSIZE			
→ AWBURST	→ ARBURST			
→ AWLOCK	→ ARLOCK			
→ AWCACHE	→ ARCACHE			
→ AWPROT	→ ARPROT			
→ AWVALID	→ ARVALID	→ WVALID	← RVALID	← BVALID
← AWREADY	← ARREADY	← WREADY	→ RREADY	→ BREADY
		→ WLAST	← RLAST	
			← RRESP	← BRESP

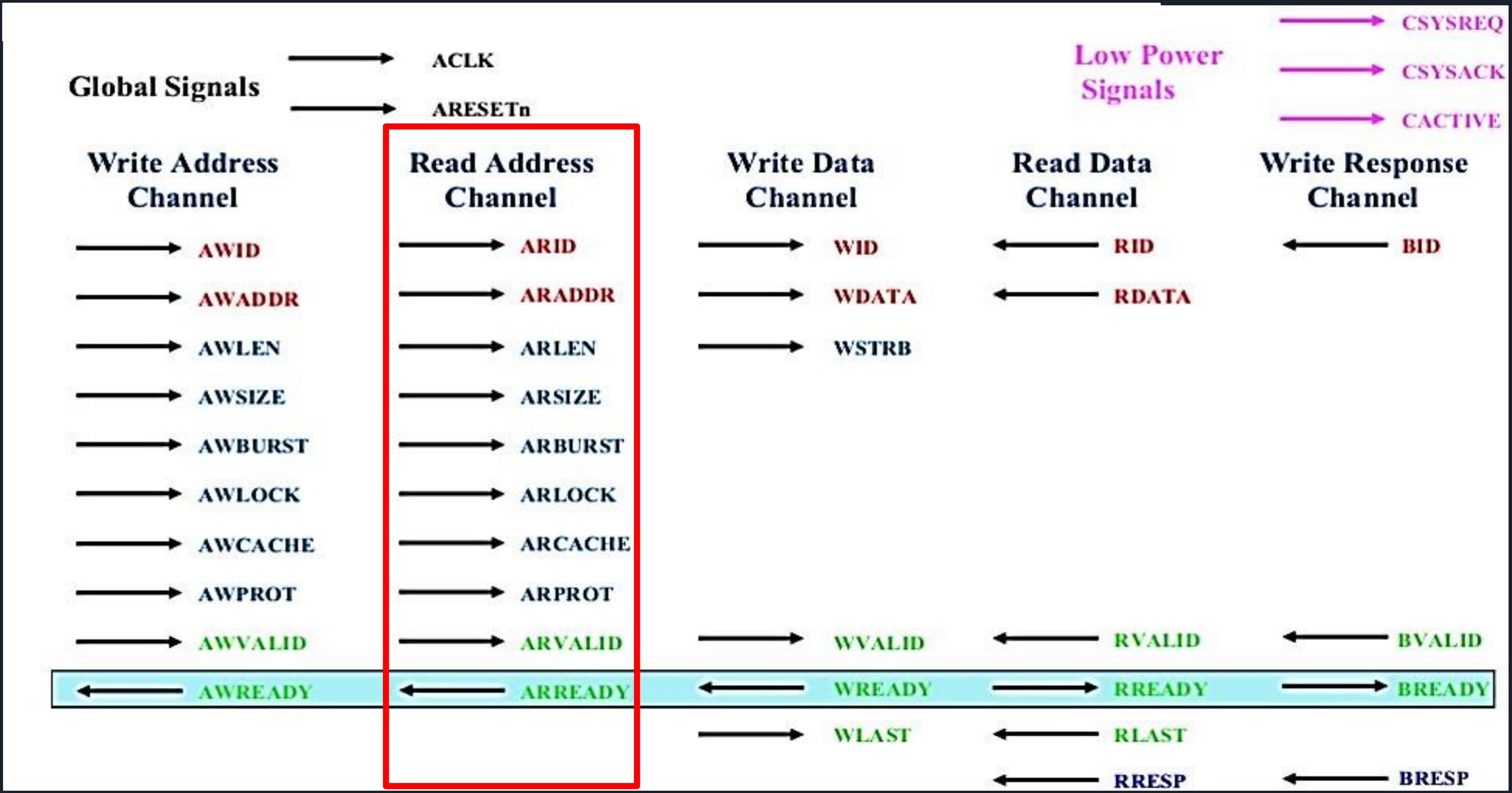
Write Resp Channel Signals



Signal	Source	Description
BID[3:0]	Slave	Response ID. The identification tag of the write response. The **BID** value must match the **AWID** value of the write transaction to which the slave is responding.
BRESP[1:0]	Slave	Write response. This signal indicates the status of the write transaction. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
BVALID	Slave	Write response valid. This signal indicates that a valid write response is available: 1 = write response available 0 = write response not available.
BREADY	Master	Response ready. This signal indicates that the master can accept the response information. 1 = master ready 0 = master not ready.

Source: ARM AMBA AXI Protocol v1.0: Specification

AXI Signals



Global Signals

→ ACLK

→ ARESETn

Low Power Signals

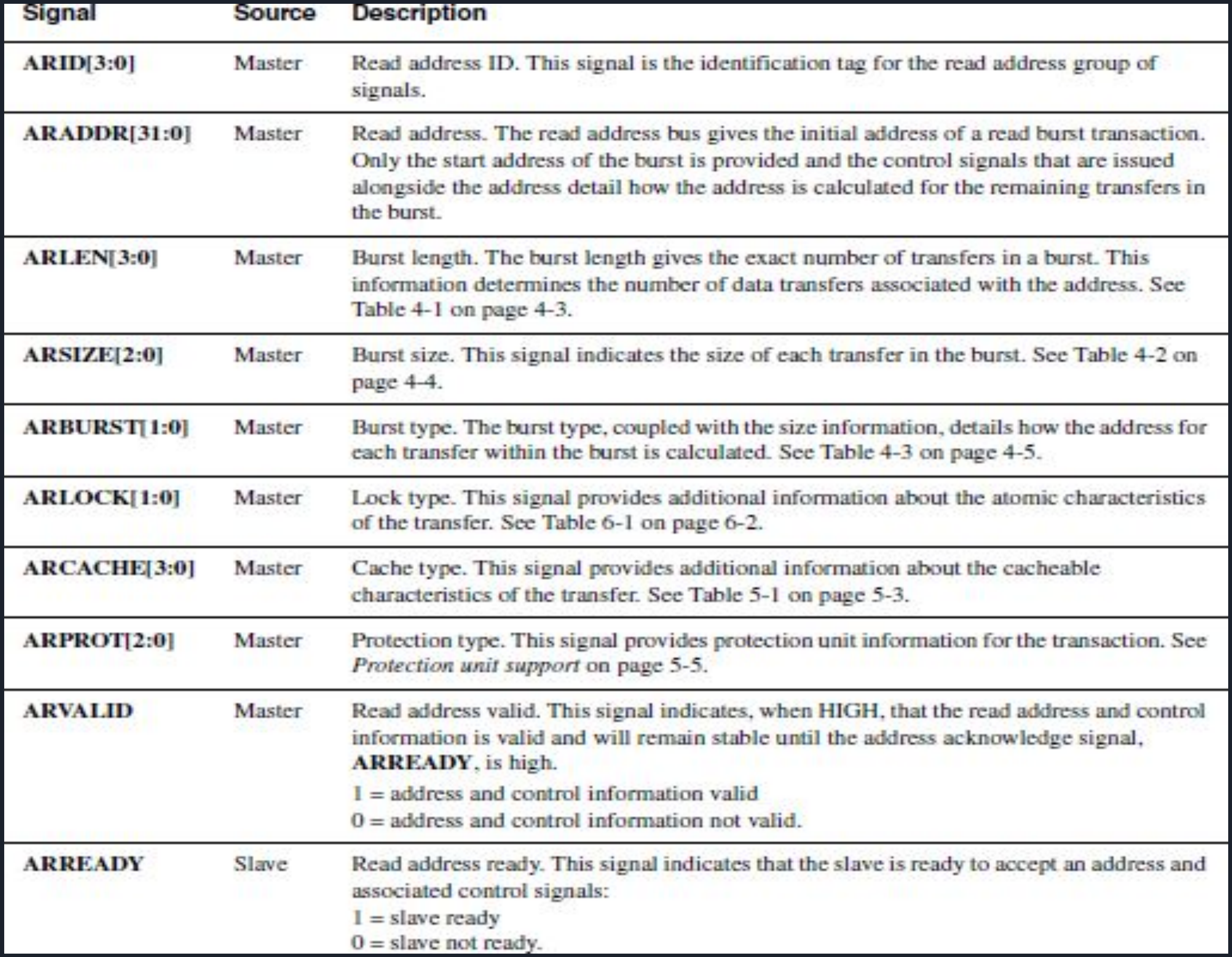
→ CSYSREQ

→ CSYSACK

→ CACTIVE

Write Address Channel	Read Address Channel	Write Data Channel	Read Data Channel	Write Response Channel
→ AWID	→ ARID	→ WID	← RID	← BID
→ AWADDR	→ ARADDR	→ WDATA	← RDATA	
→ AWLEN	→ ARLEN	→ WSTRB		
→ AWSIZE	→ ARSIZE			
→ AWBURST	→ ARBURST			
→ AWLOCK	→ ARLOCK			
→ AWCACHE	→ ARCACHE			
→ AWPROT	→ ARPROT			
→ AWVALID	→ ARVALID	→ WVALID	← RVALID	← BVALID
← AWREADY	← ARREADY	← WREADY	→ RREADY	→ BREADY
		→ WLAST	← RLAST	
			← RRESP	← BRESP

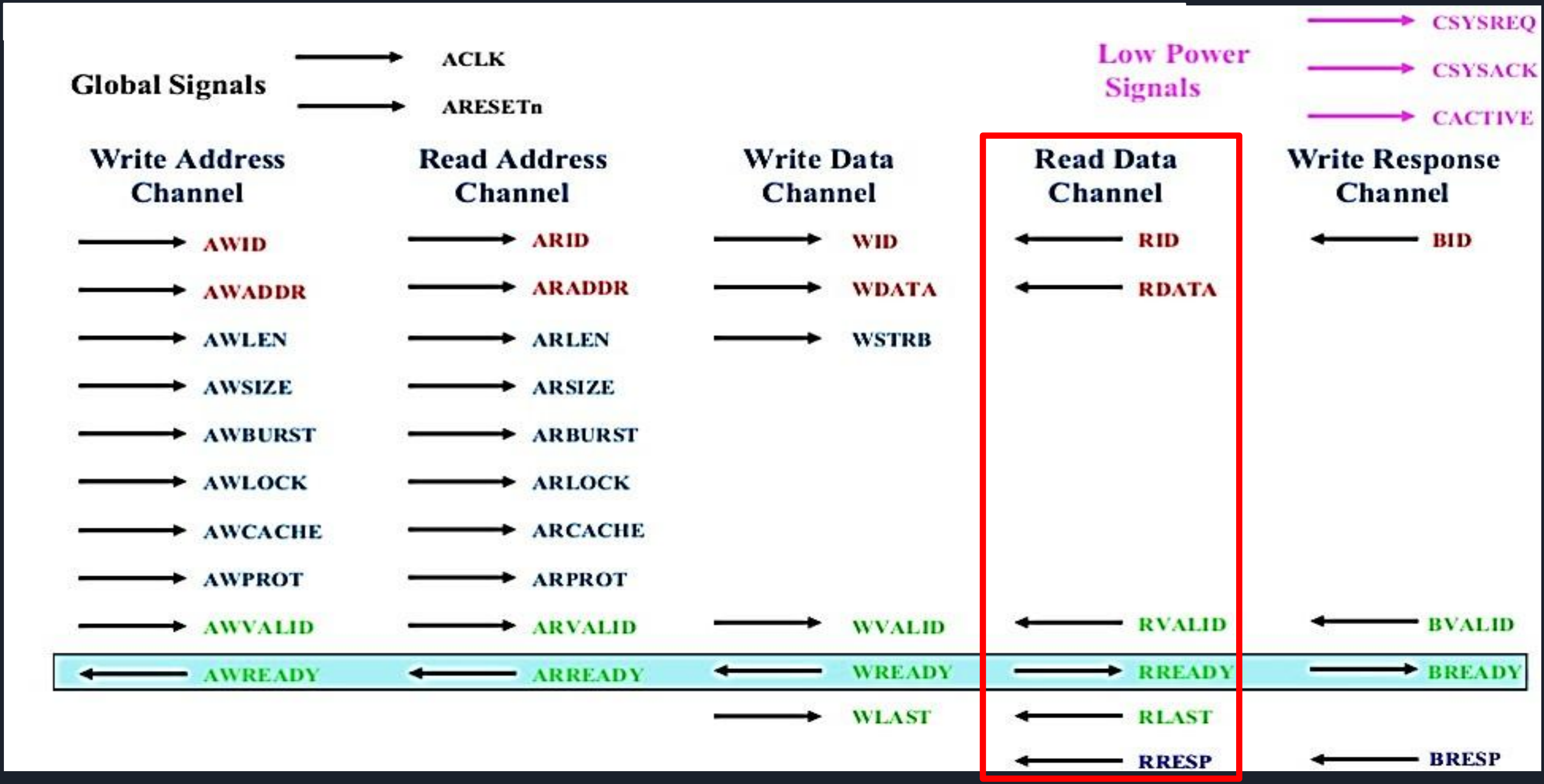
Read Address Channel Signals



Signal	Source	Description
ARID[3:0]	Master	Read address ID. This signal is the identification tag for the read address group of signals.
ARADDR[31:0]	Master	Read address. The read address bus gives the initial address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.
ARLEN[3:0]	Master	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. See Table 4-1 on page 4-3.
ARSIZE[2:0]	Master	Burst size. This signal indicates the size of each transfer in the burst. See Table 4-2 on page 4-4.
ARBURST[1:0]	Master	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. See Table 4-3 on page 4-5.
ARLOCK[1:0]	Master	Lock type. This signal provides additional information about the atomic characteristics of the transfer. See Table 6-1 on page 6-2.
ARCACHE[3:0]	Master	Cache type. This signal provides additional information about the cacheable characteristics of the transfer. See Table 5-1 on page 5-3.
ARPROT[2:0]	Master	Protection type. This signal provides protection unit information for the transaction. See *Protection unit support* on page 5-5.
ARVALID	Master	Read address valid. This signal indicates, when HIGH, that the read address and control information is valid and will remain stable until the address acknowledge signal, **ARREADY**, is high. 1 = address and control information valid 0 = address and control information not valid.
ARREADY	Slave	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready 0 = slave not ready.

Source: ARM AMBA AXI Protocol v1.0: Specification

AXI Signals



Global Signals

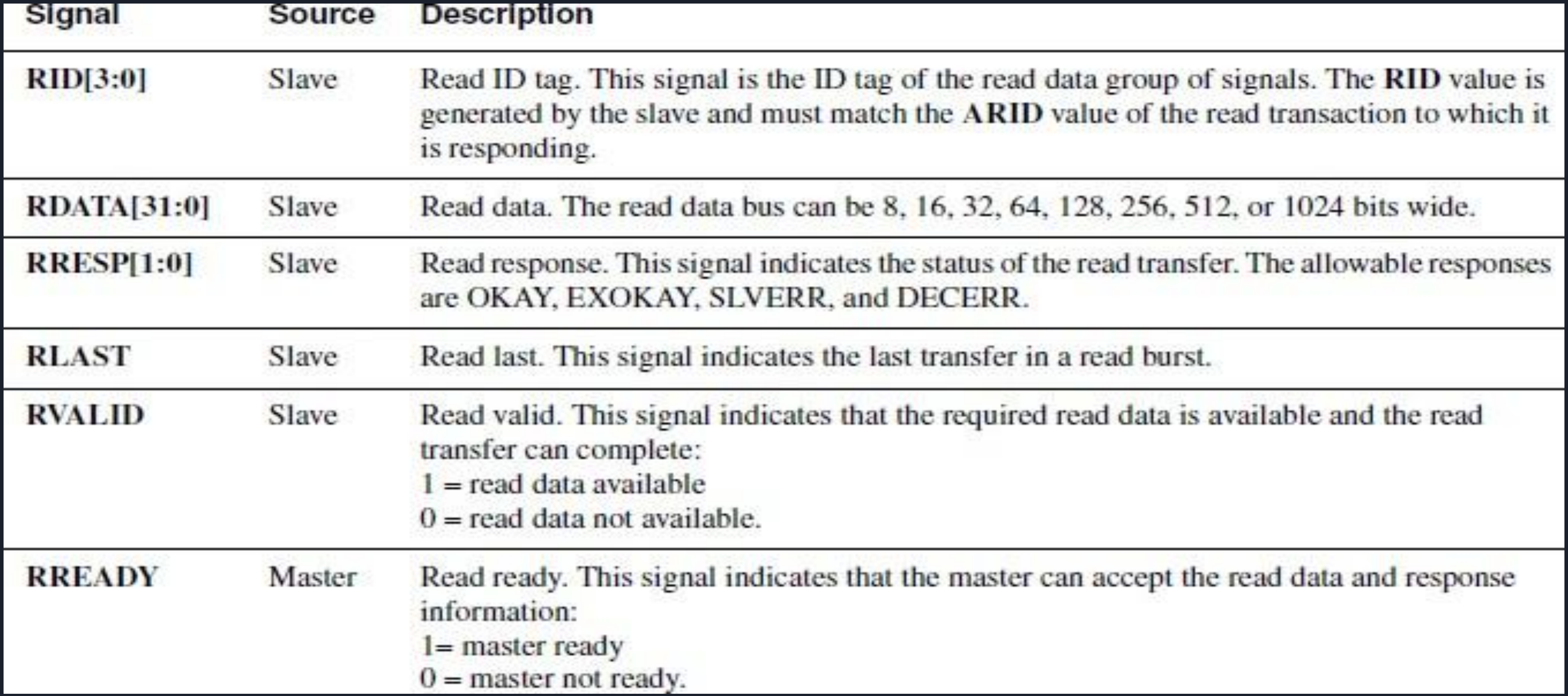
→ ACLK
→ ARESETn

Low Power Signals

→ CSYSREQ
→ CSYSACK
→ CACTIVE

Write Address Channel	Read Address Channel	Write Data Channel	Read Data Channel	Write Response Channel
→ AWID	→ ARID	→ WID	← RID	← BID
→ AWADDR	→ ARADDR	→ WDATA	← RDATA	
→ AWLEN	→ ARLEN	→ WSTRB		
→ AWSIZE	→ ARSIZE			
→ AWBURST	→ ARBURST			
→ AWLOCK	→ ARLOCK			
→ AWCACHE	→ ARCACHE			
→ AWPROT	→ ARPROT			
→ AWVALID	→ ARVALID	→ WVALID	← RVALID	← BVALID
← AWREADY	← ARREADY	← WREADY	→ RREADY	→ BREADY
		→ WLAST	← RLAST	
			← RRESP	← BRESP

Read Data + Resp Channel Signals



Signal	Source	Description
RID[3:0]	Slave	Read ID tag. This signal is the ID tag of the read data group of signals. The **RID** value is generated by the slave and must match the **ARID** value of the read transaction to which it is responding.
RDATA[31:0]	Slave	Read data. The read data bus can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide.
RRESP[1:0]	Slave	Read response. This signal indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
RLAST	Slave	Read last. This signal indicates the last transfer in a read burst.
RVALID	Slave	Read valid. This signal indicates that the required read data is available and the read transfer can complete: 1 = read data available 0 = read data not available.
RREADY	Master	Read ready. This signal indicates that the master can accept the read data and response information: 1= master ready 0 = master not ready.

Source: ARM AMBA AXI Protocol v1.0: Specification

Topics

- Overview of AXI
- AXI Architecture and Features
- Channel Definitions and Signals
- **Handshaking Process**
- Basic AXI Transactions
- Types of Access (Atomic Access)
- Ordering Model (Transaction ID tags)
- AXI4 & AXI4 Lite

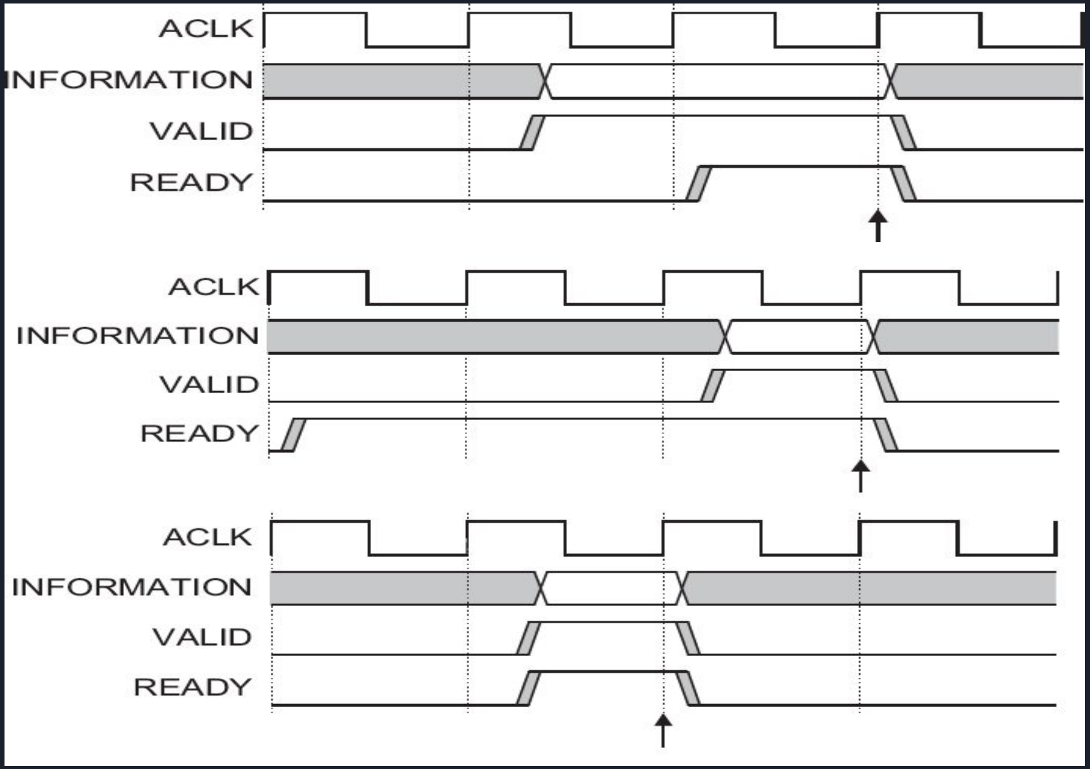
Handshaking Process

Handshake Process

- Information moves only when
  - Source is **Valid**
  - Destination is **Ready**

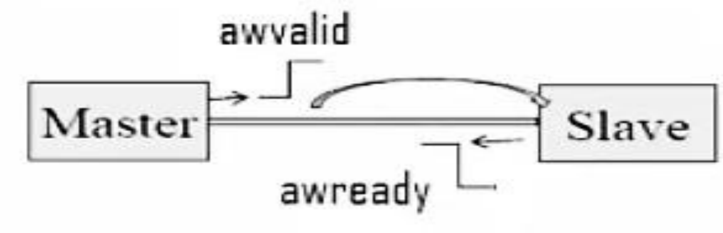
- On each channel the master or slave can limit the flow

- Very flexible



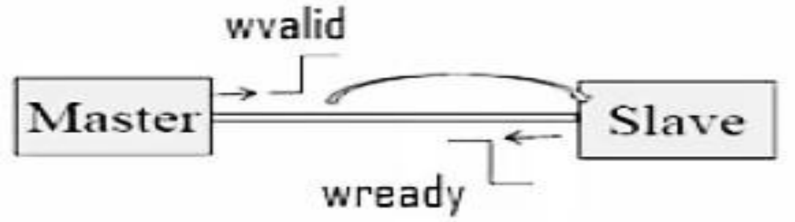
Source: ARM AMBA AXI Protocol v1.0: Specification

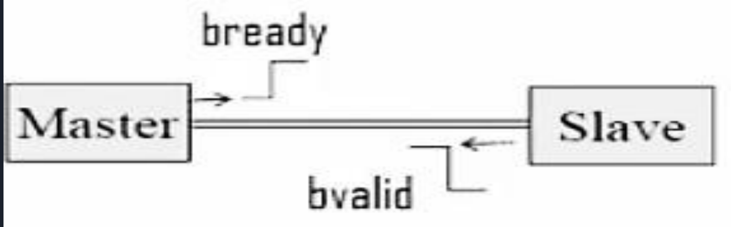
Write channel handshaking



- Master provides **awvalid** to show information valid on write addr channel.
- Slave gives **awready** indicating address information accepted.

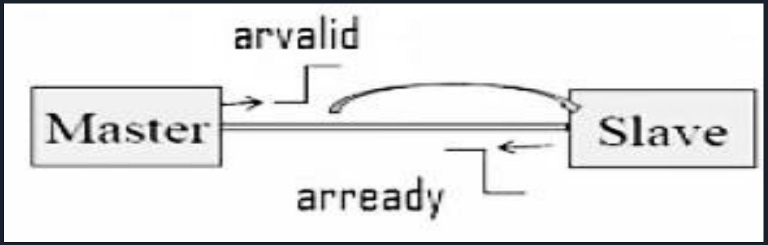
- Master provides **wvalid** to say information is valid in write data channel.
- Slave gives **wready** indicating write data information accepted.





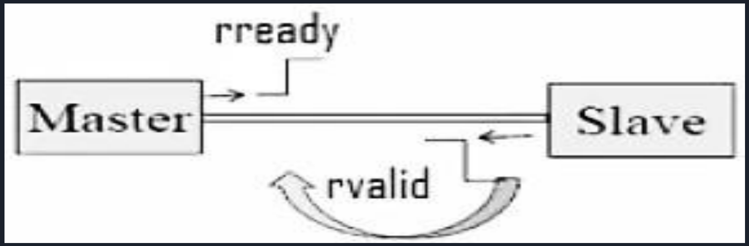
- BVALID from slave shows valid information on write response channel
- Master gives BREADY indicating write response information accepted.

Read channel handshaking



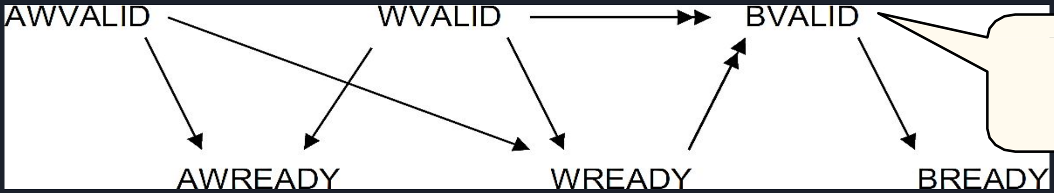
- Master provides **arvalid** to show information valid on read addr channel.
- Slave gives **arready** indicating read address information accepted.

- Slave gives **rvalid** indicating valid information on read data & response channel.
- Master provides **rready** saying read data & response information accepted.



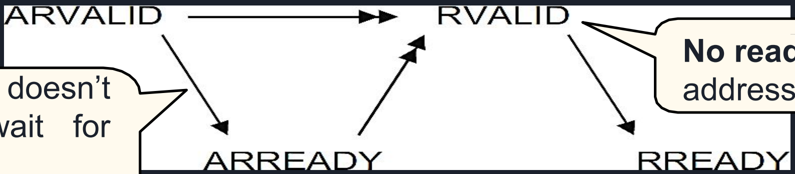
Dependencies between channel handshake signals

Write dependency



Slave mustn't give read **BRESP** unless write addr & data phase completes.

Read dependency



Slave doesn't need to wait for ARVALID assertion

No read data by slave unless read address phase completes.

- Write data can appear before Write address
- Write data appear in the same cycle as the address
- Read data always come after Read address
- Write response always come after write data

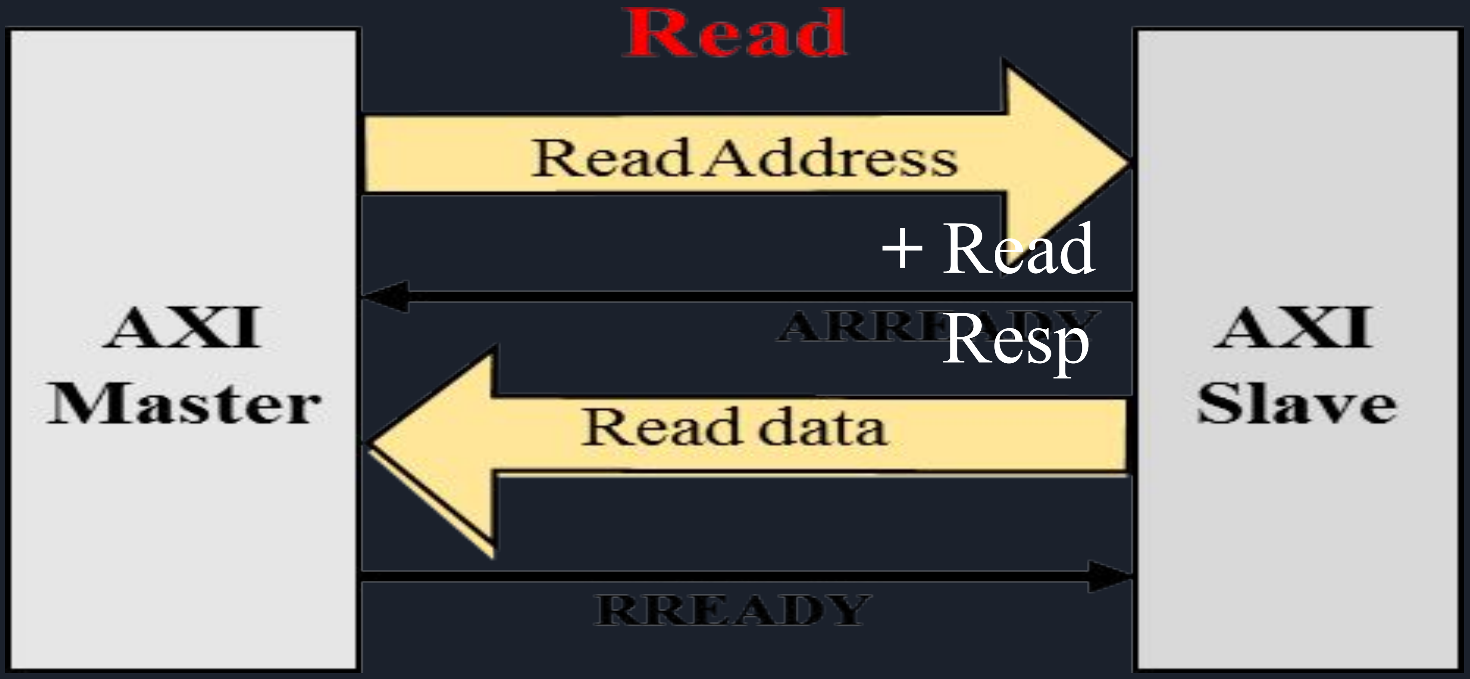
Source: ARM AMBA AXI Protocol v1.0: Specification

Topics

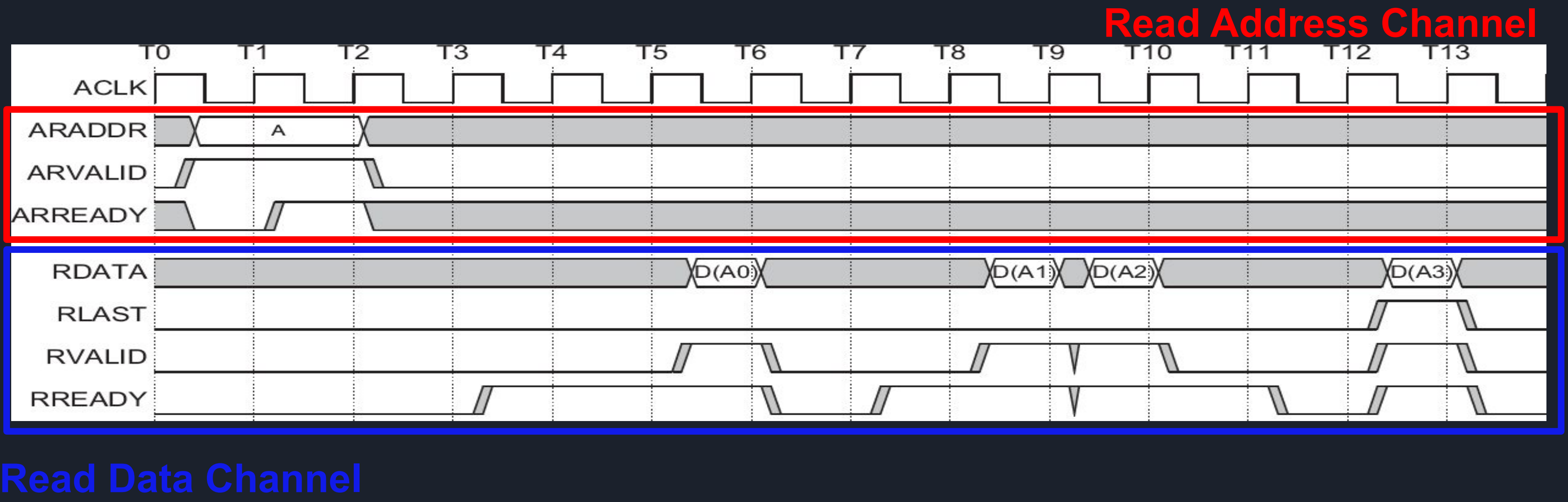
- Overview of AXI
- AXI Architecture and Features
- Channel Definitions and Signals
- Handshaking Process
- **Basic AXI Transactions**
- Types of Access (Atomic Accesses)
- Ordering Model (Transaction ID tags)
- AXI4 & AXI4 Lite

AXI Transactions

Read Transaction



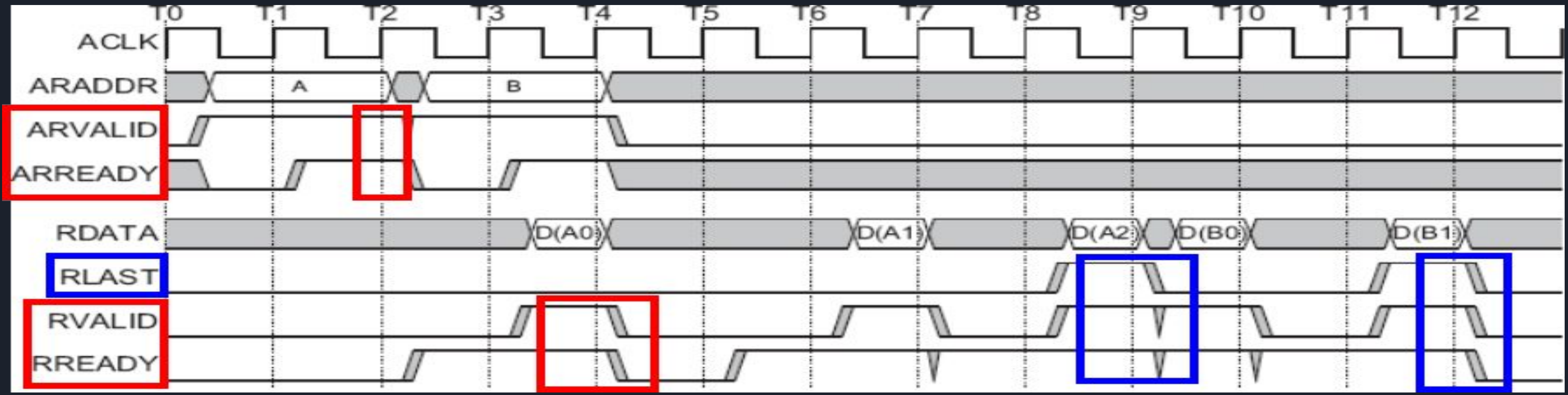
Read Bursts



Source: AMBA® AXI Protocol v 1.0 Specification, ARM, 2003.

- The slave keeps the **RVALID** signal LOW until the read data is available.
- Slave asserts **RLAST** for final data transfer in a burst indicating last data item has been transferred.

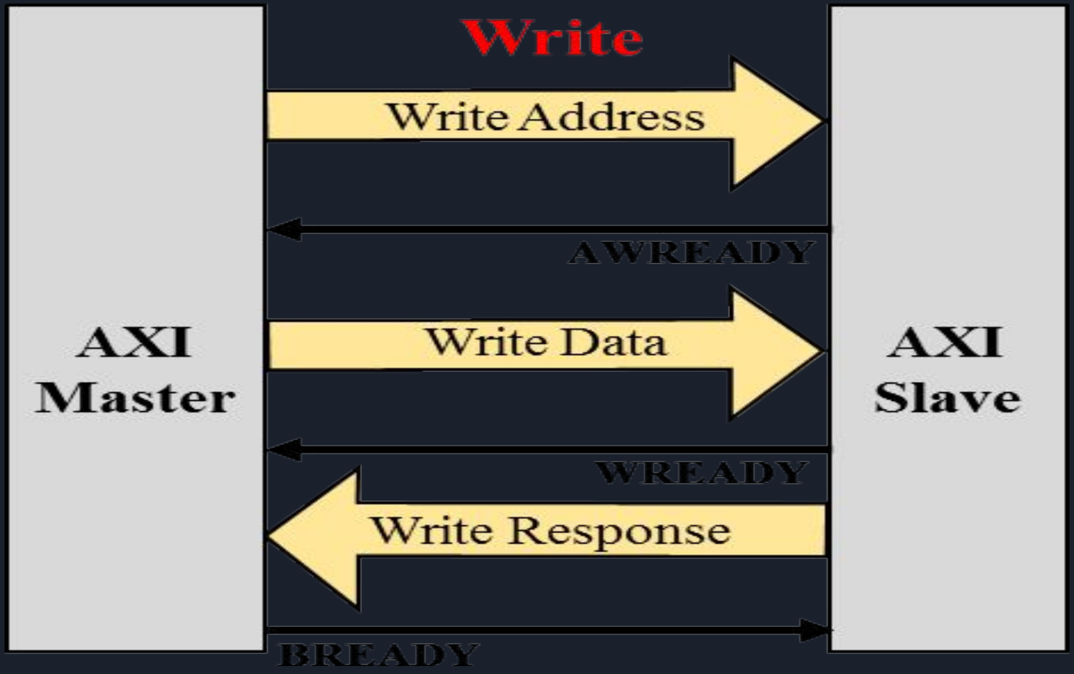
Overlapping Read Bursts



Source: AMBA AXI Protocol v 1.0 Specification, ARM, 2003.

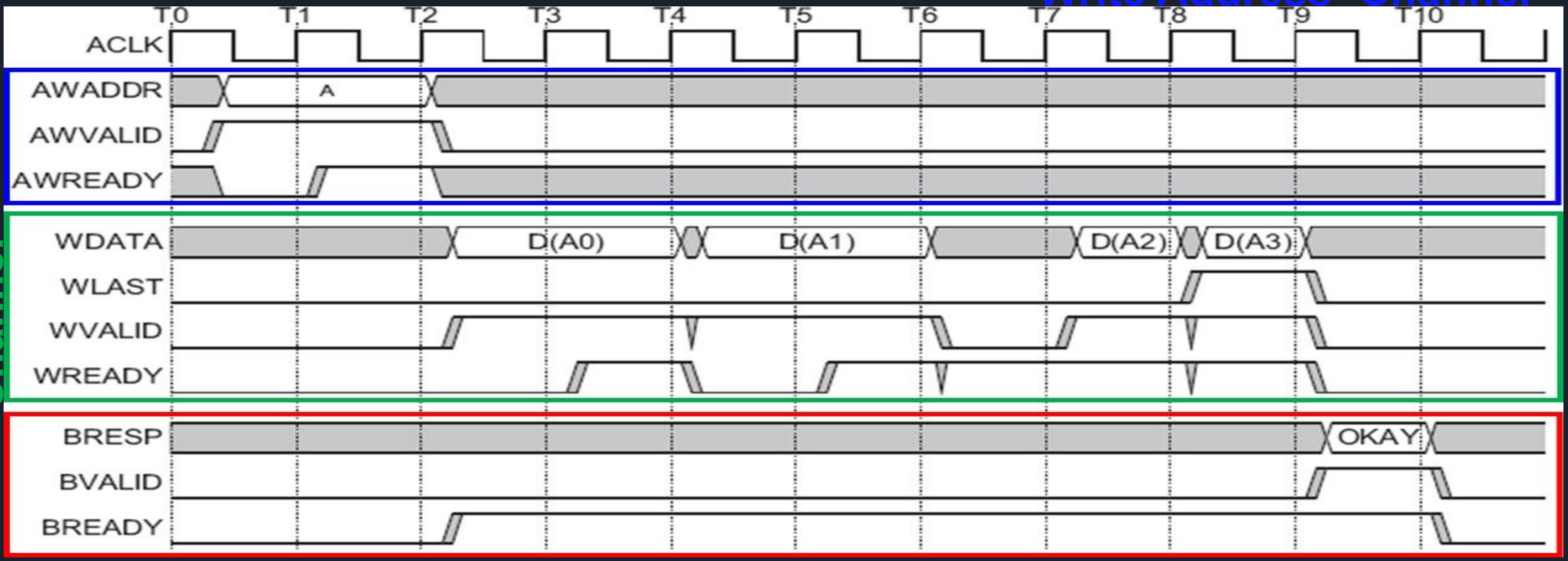
- A master can drive another burst address (B) after first addr accepted by slave.
- First addr allows slave to process data for the second burst in parallel with the completion of the first burst (RLAST, D(A2))

Write Transactions



Write Burst

Write Address `Channel



Write Data Channel

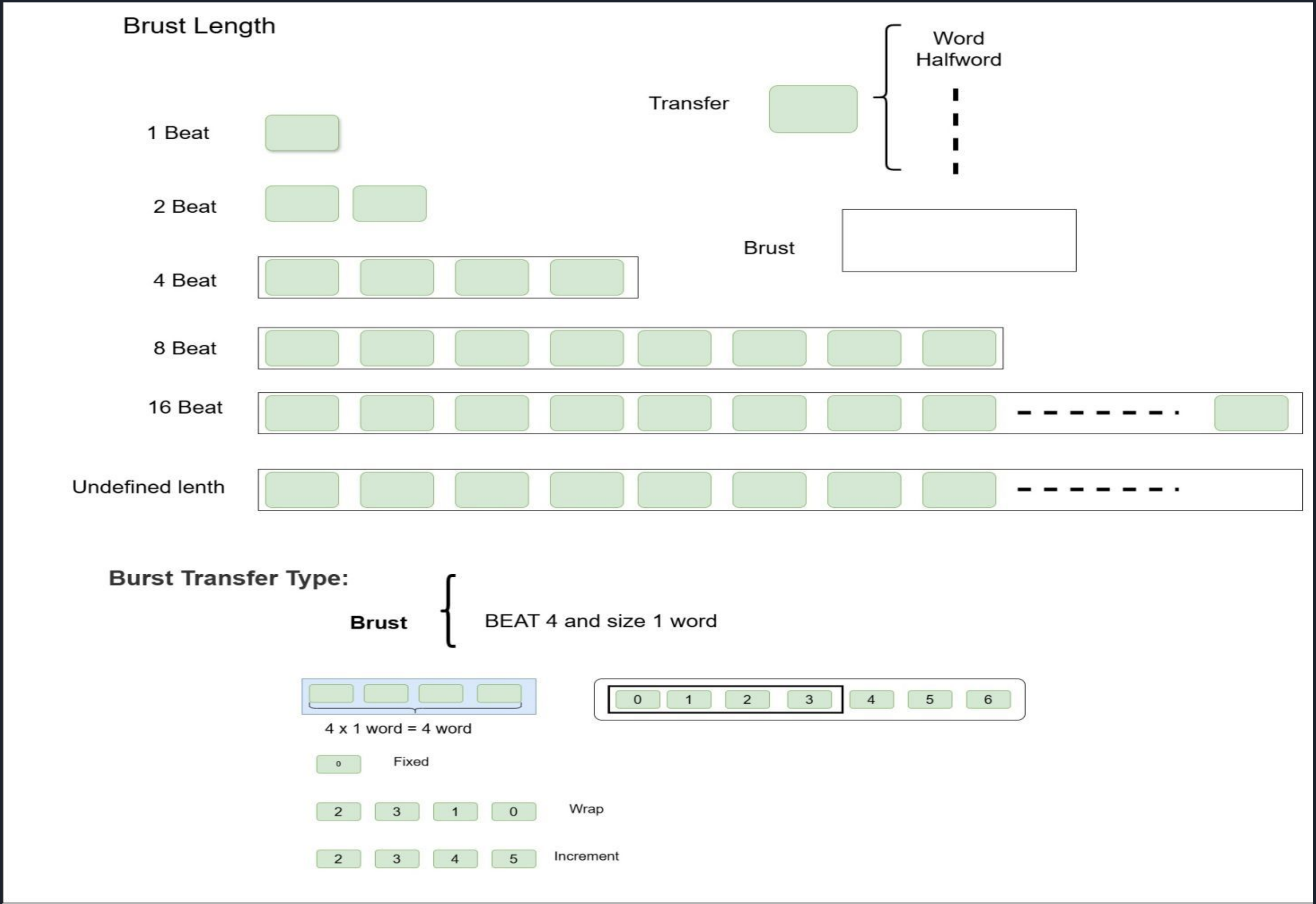
Write Response Channel

Source: AMBA® AXI Protocol v 1.0 Specification, ARM, 2003.

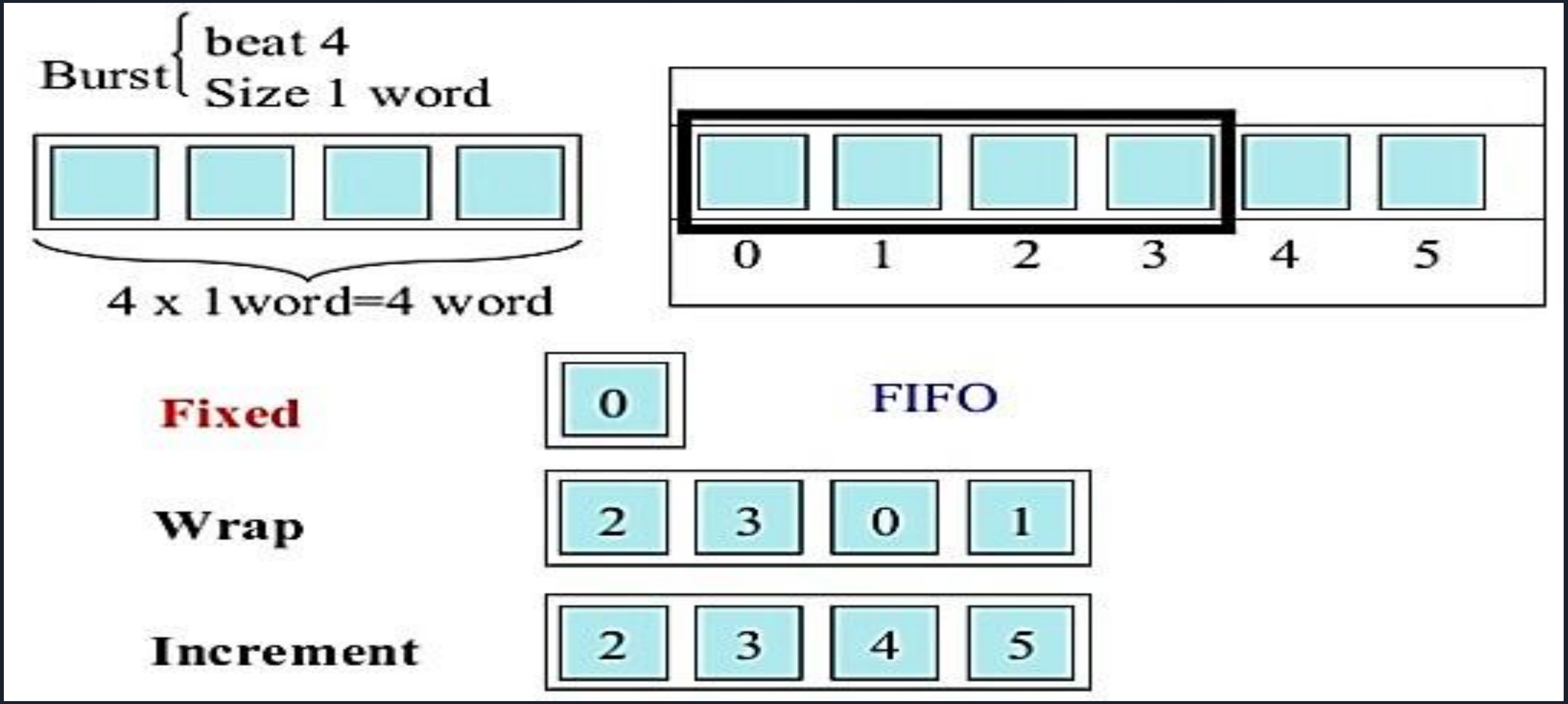
Source: AMBA AXI Protocol v 1.0 Specification, ARM, 2003.

Addressing Options

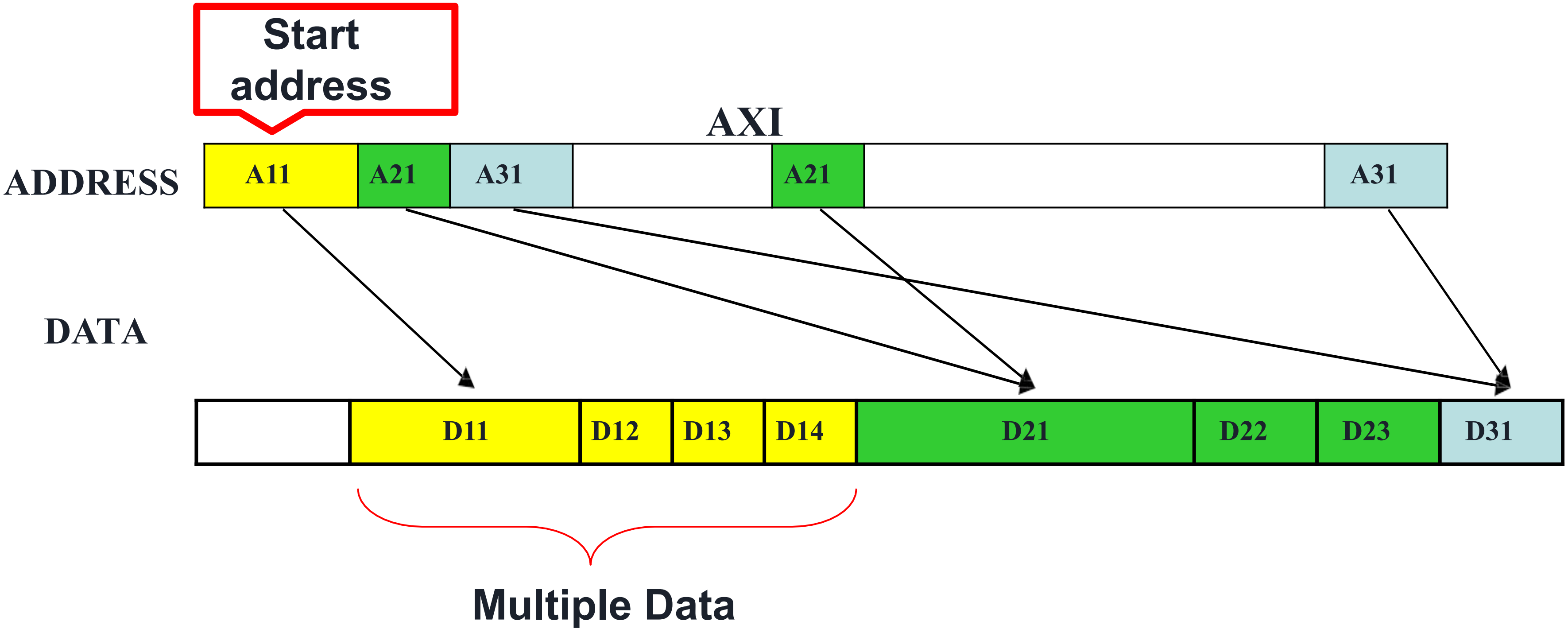
Burst Operation[5]



Burst Transfer Type

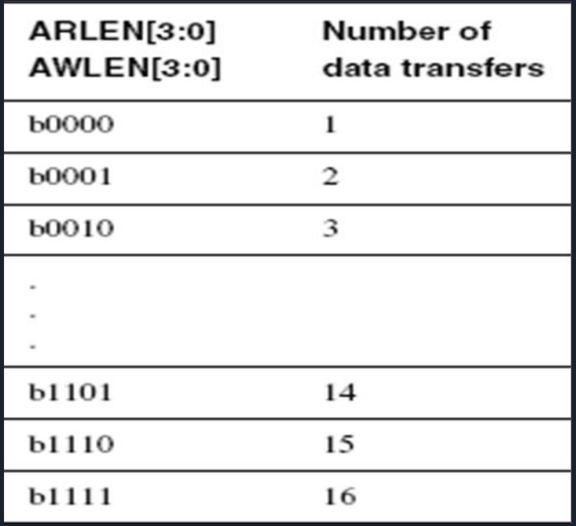


AXI Burst Transfer



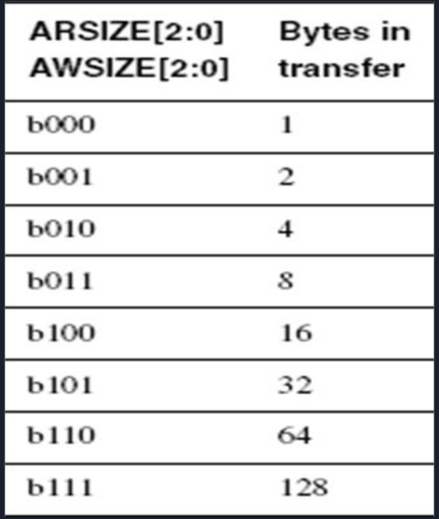
Burst Control signals

Burst length



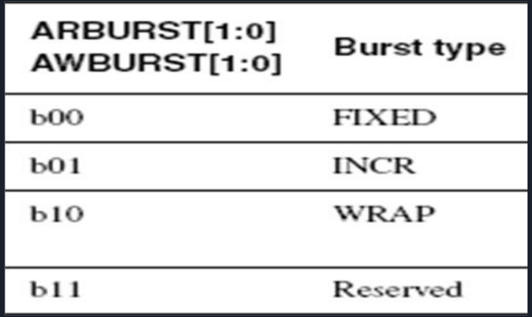
ARLEN[3:0] AWLEN[3:0]	Number of data transfers
b0000	1
b0001	2
b0010	3
. . .	
b1101	14
b1110	15
b1111	16

Burst Size



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

Burst Type

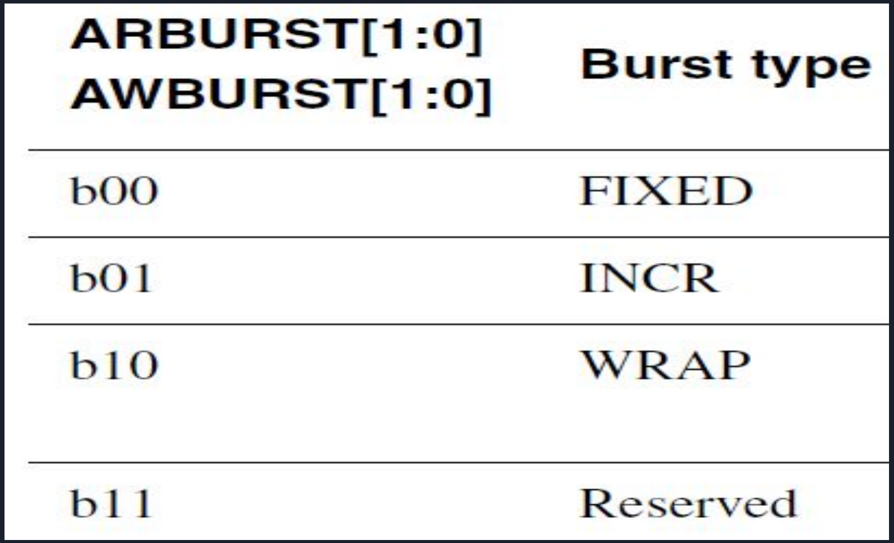


ARBURST[1:0] AWBURST[1:0]	Burst type
b00	FIXED
b01	INCR
b10	WRAP
b11	Reserved

Source: AMBA AXI Protocol v 1.0 Specification, ARM, 2003.

- Every transaction must have the number of transfers.
- No component can terminate a burst early to reduce the no. of data transfers.

Burst Type



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

Source: AMBA AXI Protocol v 1.0 Specification, ARM, 2003.

Address calculations

Burst address

- **Addresses of transfers within a burst:**

$$Start\_Address \;=\; ADDR$$
$$Number\_Bytes = \; 2^{AWSIZE}$$
$$Burst\_Length \;=\; AWLEN + 1$$
$$Aligned\_Address = \left(INT\left(\frac{Start\_Address}{Number\_Bytes}\right)\right) \times NumberBytes$$

- **Address of first transfers in a burst:**

$$Address\_1 = Start\_Address$$

- **Address of any transfer after first transfer in a burst:**

$$Address\_N \;=\; Aligned\_address + (N - 1) \times Number\_Bytes$$

Burst address

- **Wrap_Boundary variable extended to get wrapping boundary:**

$$Wrap\_Boundary = \left(INT\left(\frac{Start\_Address}{Number\_Bytes \times Burst\_Length}\right)\right) \times (Number\_Bytes \times Burst\_Length)$$

- **Address of first transfers in a burst:**

$$Address\_N = Wrap\_Boundary + (Number\_Bytes \times Burst\_Length)$$

- **Address of any transfer after first transfer in a burst:**

$$Address\_N = Wrap\_Boundary$$

INCR

Find the address of each transfer in a burst

- AWBURST = INCR
- AWSIZE = 2
- AWLEN = 3
- AWADDR = 23C0
- Bus_Size = 32 bits

Solution Start_Address =>

23C0

Number_Bytes => 2 ^{AWSIZE} => 2 ^{2} **= 4**

Burst_Length => AWLEN + 1 = 3 + 1 **= 4**

Aligned_Address = INT(23C0/4) x 4 = 8F0 x 4 = **23C0**

- **Address_1 = Start_Address** = **23C0**
- **Address_N = Aligned_Address + (N-1) x Number_Bytes**
- Address_2 = 23C0 + (2 - 1) x 4 = 23C4
- Address_3 = 23C0 + (3 - 1) x 4 = 23C8
- Address_4 = 23C0 + (4 - 1) x 4 = 23CC
- **Address : 23C0 , 23C4 , 23C8, 23CC**

Fixed

Burst address (Fixed)

- AWBURST = FIXED
- AWSIZE = 1
- AWLEN = 4
- AWADDR = 44B0
- Bus Size = 32 bits

Start_Address = 44B0 **Number_Bytes** =

$2^{AWSIZE} = 2^1 = 2$ **Burst_Length** = AWLEN + 1

= 4 + 1 =5

Aligned_Address = INT(Start_Address/Number_Bytes) x Number_Bytes

= INT(44B0/2) x 2 = 2258 x 2 = 44B0

Burst Address (Fixed…)

- Since, it’s a Fixed Burst , so the Address will be same for each transfer i.e.

- **Address_1** = 44B0
- **Address_2** = 44B0
- **Address_3** = 44B0
- **Address_4** = 44B0
- **Address_5** = 44B0

Wrap

Burst Address (WRAP)

- AWBURST = WRAP
- AWSIZE = 1
- AWLEN = 2
- AWADDR = A242
- Bus Size = 32-bits

Start_Address = A242 **Number_Bytes** =

$2^{\mathbf{AWSIZE}}$ = 2^1 = 2 **Burst_Length** = AWLEN +

1 = 2 + 1 =3

Aligned_Address = INT(A242/2) x 2 = 5121 x 2 = A242

Burst Address (WRAP…)

- Wrap_Boundary = (INT(A242/(2 X 3))) X (2 X 3) = A242

Address_N=Wrap_Boundary+(Num_Bytes x Burst_Len)

- **Address_1** = A242
- **Address_2** = A242 + (2 - 1) x 2 = A244
- **Address_3** = A242 + (3 - 1) x 2 = A246
- **Address_4** = A242 + (4 - 1) x 2 = A248
- **Address : A242 , A244 , A246**

Additional Control Information

Cache Support

- **Support for system level caches and other performance enhancing components**
  - Bufferable (B) bit, ARCACHE[0] and AWCACHE[0]
  - Cacheable (C) bit, ARCACHE[1] and AWCACHE[1]
  - Read Allocate (RA) bit, ARCACHE[2] and AWCACHE[2]
  - Write Allocate (WA) bit, ARCACHE[3] and AWCACHE[3]

Cache Support

- **Cacheable Bit**
  - Write : a number of different writes can be **merged** together.
  - Read : a location can be **pre-fetched** or can be **fetched just once** for multiple read transactions
- **Read Allocate Bit**
  - If the transfer is a read and it misses in the cache then it should be allocated
- **Write Allocate Bit**
  - If the transfer is a **write** and it **misses** in the cache then it should be allocated

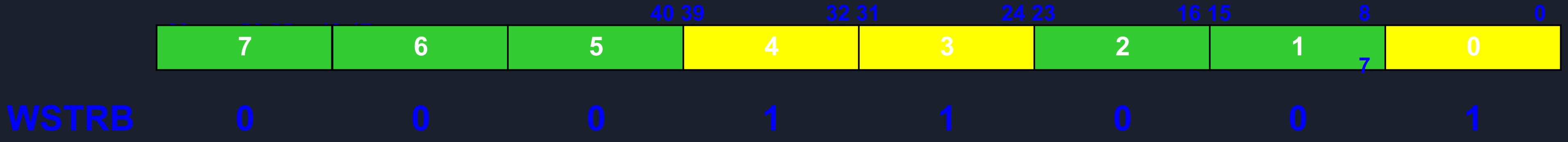
Cacheable Write – Sparse Strobes

Cacheable Bit

Write : a number of different writes can be merged together

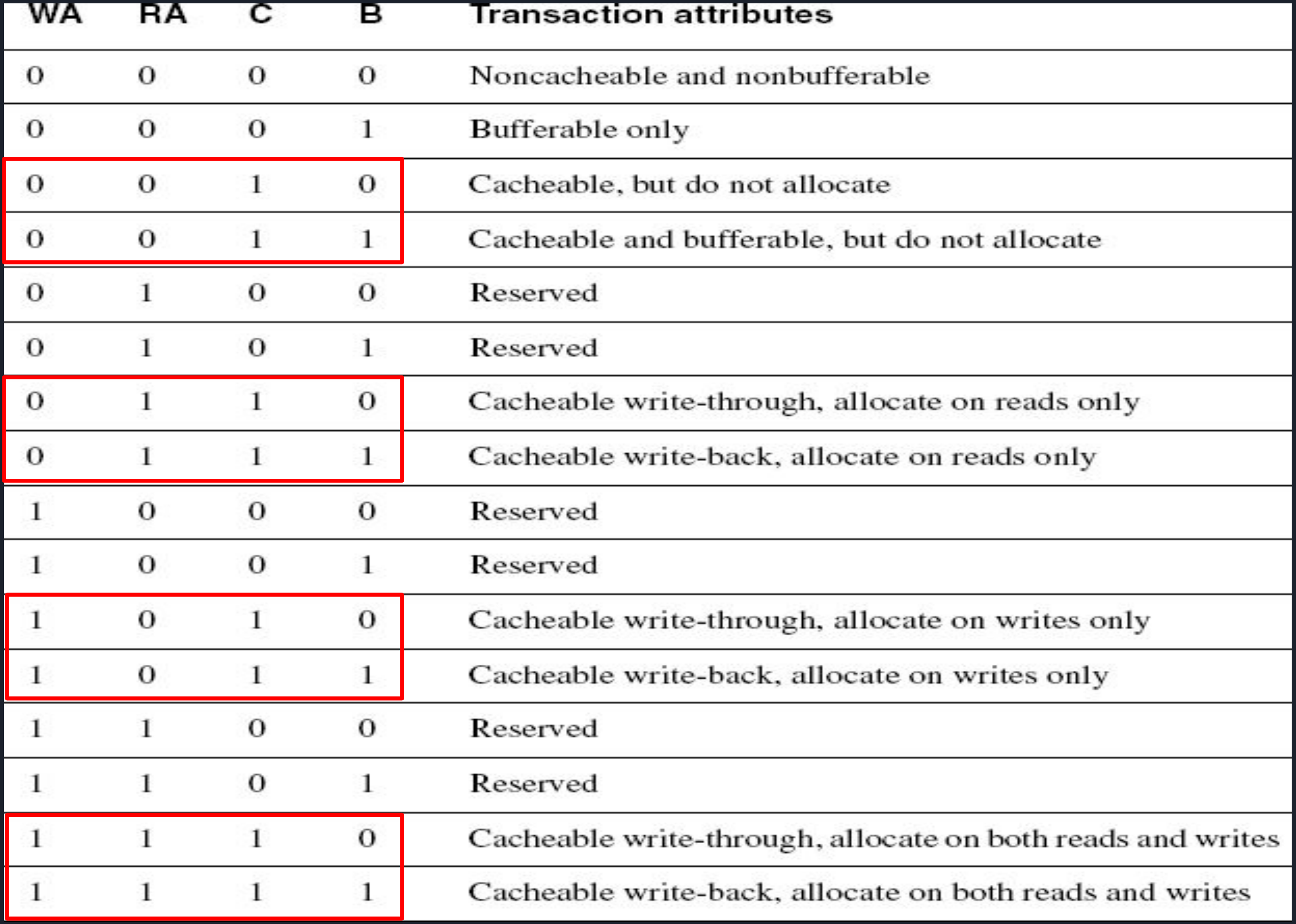
ARM11-MPCore processor and L220 Level 2 cache controller use sparse strobes. This is due to the fact that they use a merging write buffer. If the application writes several bytes at address 0x0, 0x3, 0x4, the write buffer will be drained using a 64- bit transfer, and strobes will be 0b00011001 and the AXI slave must only update the bytes that are enabled.

64-bit write data bus



				40 39	32 31	24 23	16 15	8 7	0
	7	6	5	4	3	2	1		0
WSTRB	0	0	0	1	1	0	0		1

Cache Encoding



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

Protection Unit Support

- To support complex system design, for the interconnect and other devices in the system to provide protection against illegal transactions

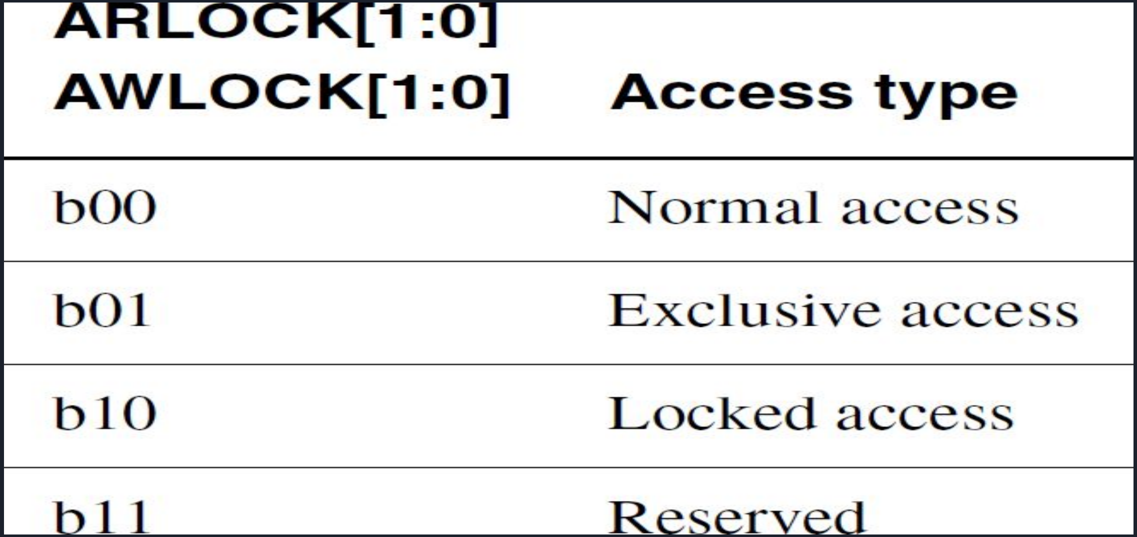
- **Normal or privileged, ARPROT[0] andAWPROT[0]**
  - **Low High**
- **Secure or non-secure, ARPROT[1] andAWPROT[1]**
  - **Low High**
- **Data or instruction, ARPROT[2] andAWPROT[2]**
  - **Low High**

Topics

- Overview of AXI
- AXI Architecture and Features
- Channel Definitions and Signals
- Handshaking Process
- Basic AXI Transactions
- **Types of Access (Atomic Access)**
- Ordering Model (Transaction ID tags)
- AXI4 & AXI4 Lite

Atomic Access

Atomic Accesses

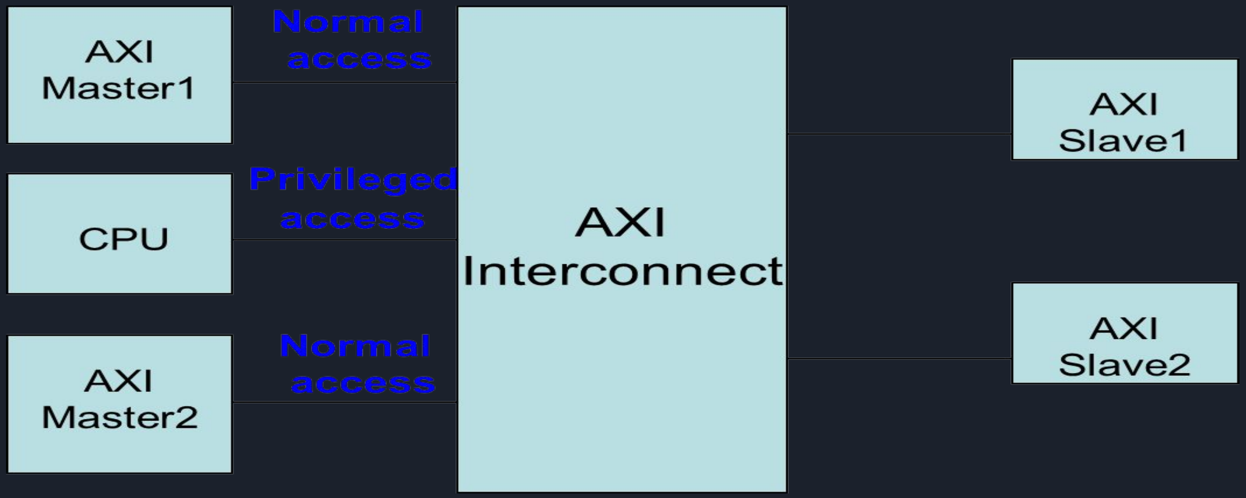


ARLOCK[1:0] AWLOCK[1:0]	Access type
b00	Normal access
b01	Exclusive access
b10	Locked access
b11	Reserved

Source: ARM AMBA AXI Protocol v1.0: Specification

Normal / Privileged (`b00)

- Used by some **masters** to indicate processing mode.
- A privileged processing mode has a **greater level of access** within a system.



Exclusive Access (`b01)

Master1 Exclusive Read

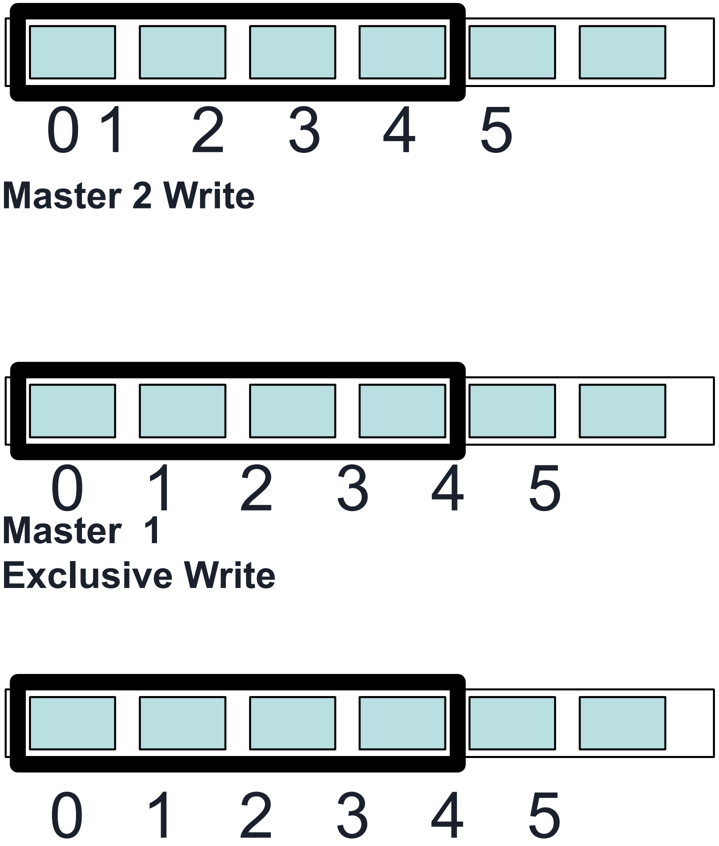
Cycle 1

Master 2 Write

Cycle 2

Master 1
Exclusive Write

Cycle 3



Semaphore type without locking the bus access

Exclusive Read success
Response : EXOKAY

Exclusive Read fail
Response : OKAY

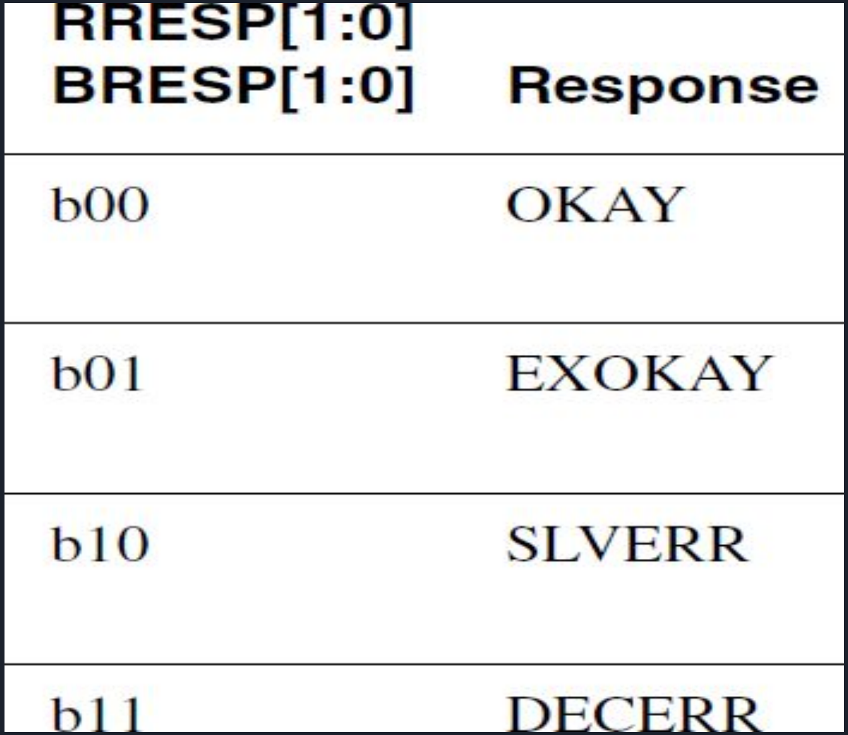
Exclusive Write success
Response : EXOKAY

Exclusive Write fail
Response : OKAY

Locked Access

- Interconnect makes sure only one specific master is allowed access to the slave region until an unlocked transfer from the same master completes.
- Purpose of use for longer and undisturbed transfers

Response Signals



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

Accessed reached slave but slave gives error to master

No slave present at transaction address interconnect responsible for decode error

Topics

- Overview of AXI
- AXI Architecture and Features
- Channel Definitions and Signals
- Handshaking Process
- Basic AXI Transactions
- Types of Access (Atomic Access)
- **Ordering Model (Transaction ID tags)**
- AXI4 & AXI4 Lite

Ordering Model

Ordering Model

- AXI protocol enables **out-of-order transaction** completion and issuing of **multiple outstanding addresses**.
- These features help maximize data throughput and system efficiency.
- All transactions with same ID must be ordered, but there is no restriction on the ordering of transactions with different IDs.

Ordering Model [1]

- **Multiple Outstanding Address**
  - Master is able to provide transaction addresses without waiting for earlier transactions to complete.
- **Out of Order Transaction**
- **Write data Interleaving**
  - If two masters generate write data sequence to the same slave, but write data doesn't arrive every clock cycle.
  - Instead of waiting for a data sequence to complete before other sequence starts, AXI system can interleave write data sequences avoiding idle cycles on the bus.

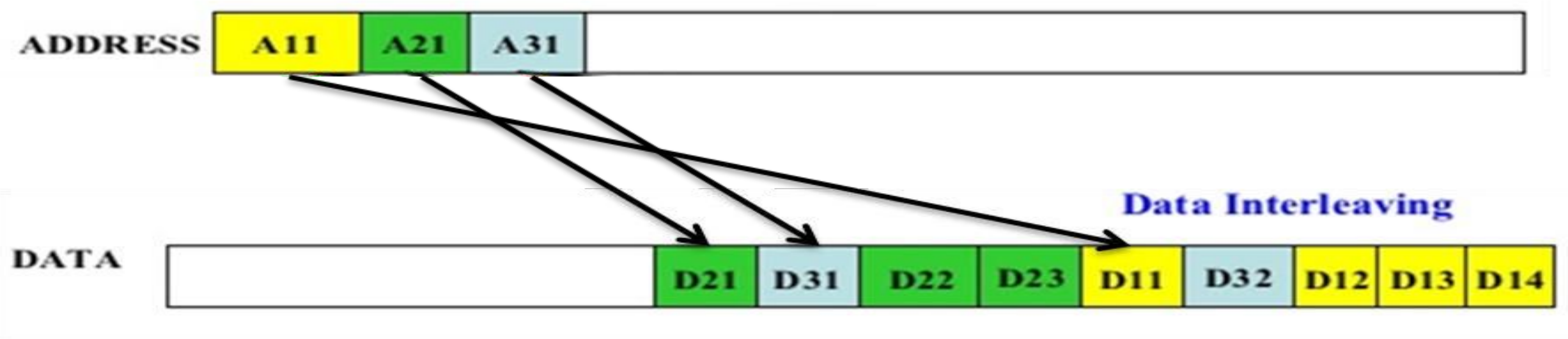
[1] AMBA AXI Protocol v 1.0 Specification, ARM, 2003.

Read Ordering

- At master interface, read data from read transactions with same **ARID** value must arrive in same order in which master issued addresses.

- Data from read transactions with different **ARID** values can return in any order and it is also acceptable to interleave the read data of transactions with different **ARID** fields.

Read Data Interleaving[5]



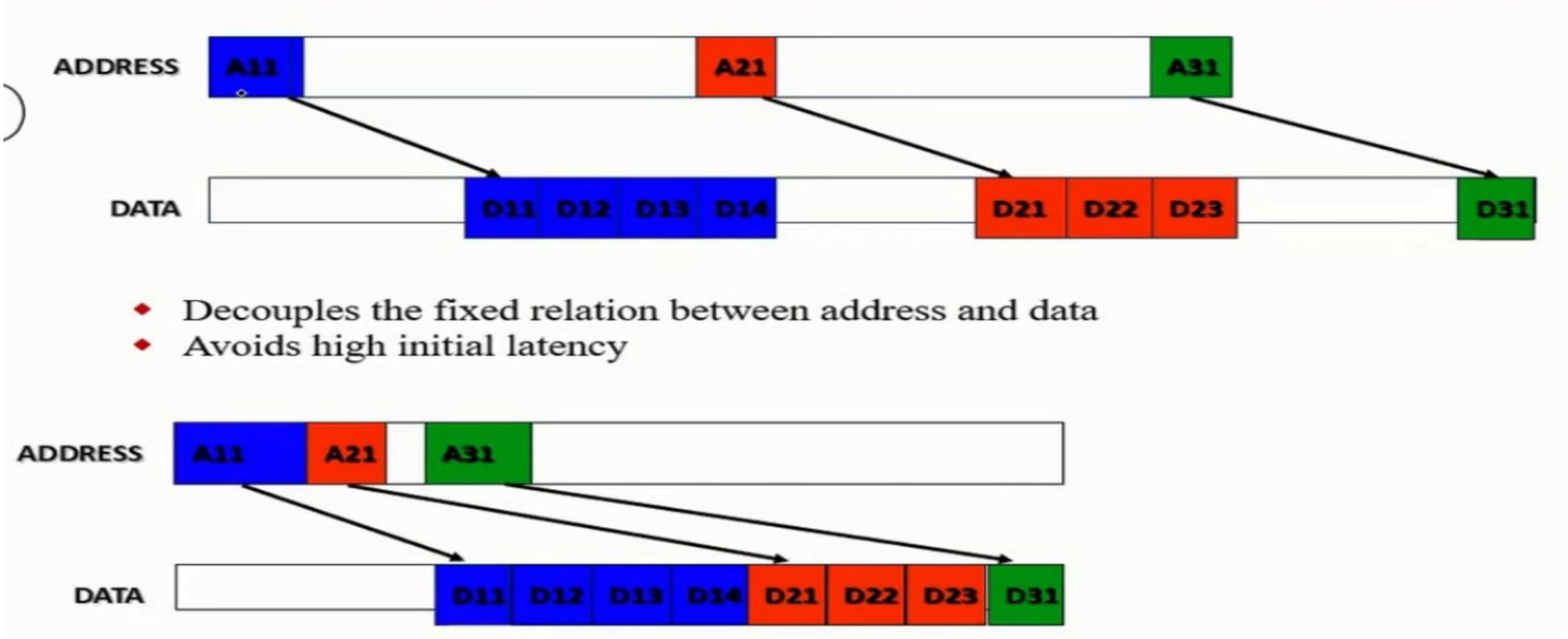
Read data Interleaving

Normal Write Ordering

- If a slave does not support write data interleaving the master must issue the data of write transactions in the same order in which it issues the transaction addresses.

- Write data does not support out-of-order transactions but support data interleaving.

Multiple Outstanding Address



- Decouples the fixed relation between address and data
- Avoids high initial latency

Figure: Multiple Outstanding Address

Write Data Interleaving[5]

- Write data interleaving enables a slave interface to accept interleaved write data with different **AWID** values.

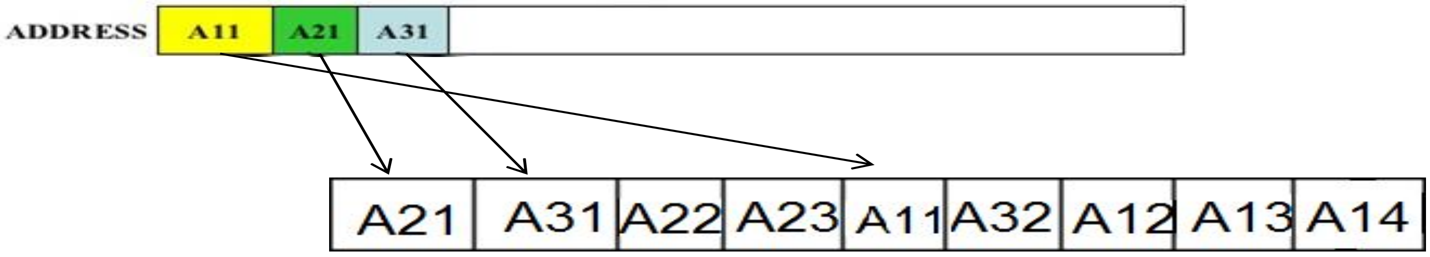
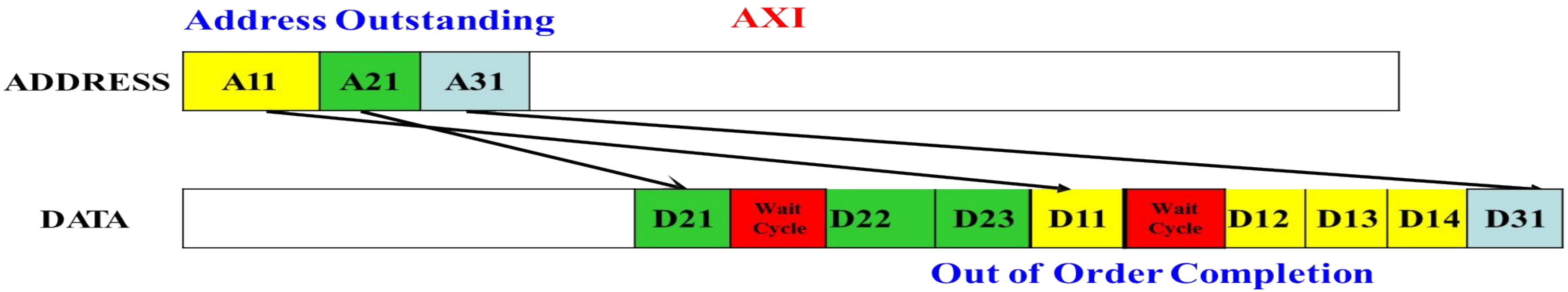


Figure: Write data Interleaving

Out of Order Transaction

- Transactions to faster memory regions can complete without waiting for previous transaction to slower memory regions.
- Fast slaves may return data ahead of slow slaves

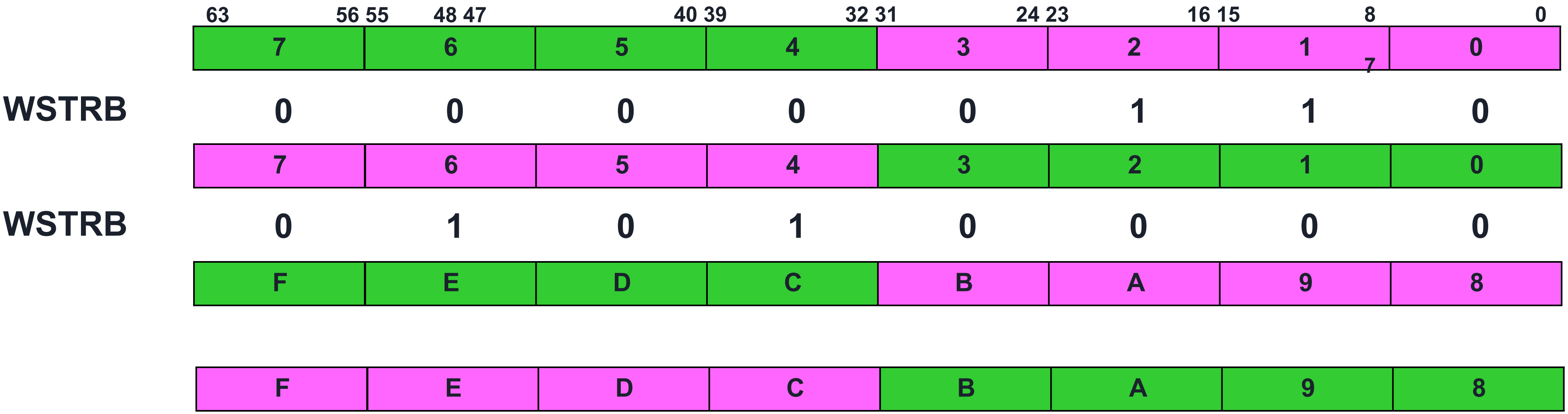


Aligned Transfer

Address: 0x00
Transfer size: 32 bits
Burst type: incrementing
Burst length: 4 transfers

64-bit write data bus

For wrapping burst type, all transfers are aligned transfers



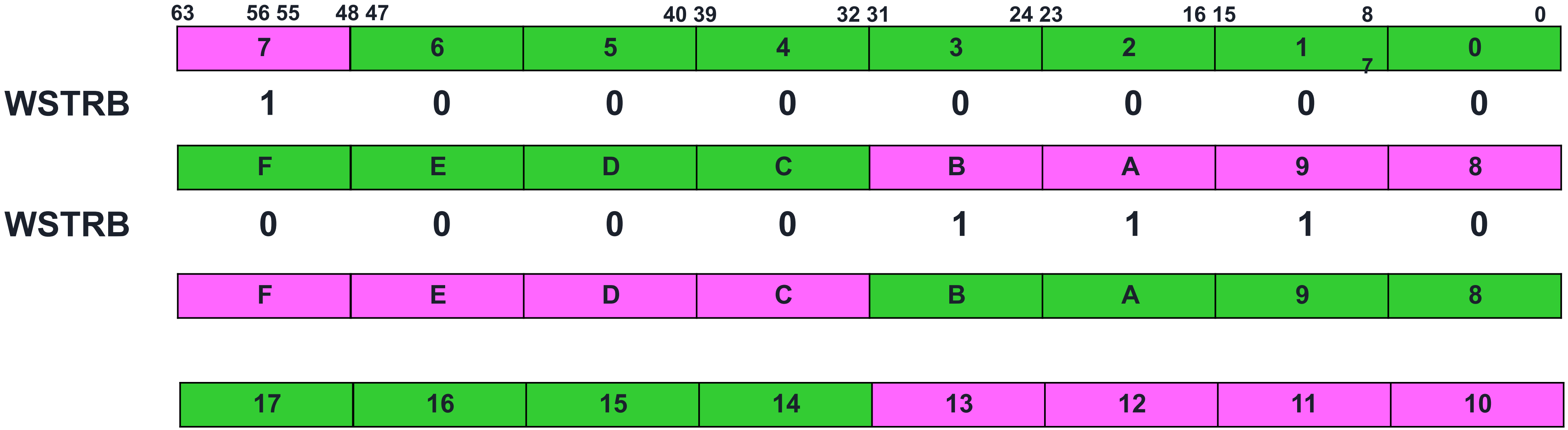
	63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
	7	6	5	4	3	2	1	0
WSTRB	0	0	0	0	0	1	1	0
	7	6	5	4	3	2	1	0
WSTRB	0	1	0	1	0	0	0	0
	F	E	D	C	B	A	9	8
	F	E	D	C	B	A	9	8

Unaligned Transfer

Address: 0x07
Transfer size: 32 bits
Burst type: incrementing
Burst length: 4 transfers

For incrementing burst type, fist transfer can be unaligned transfers, but the rest transfers are aligned transfers

64-bit write data bus

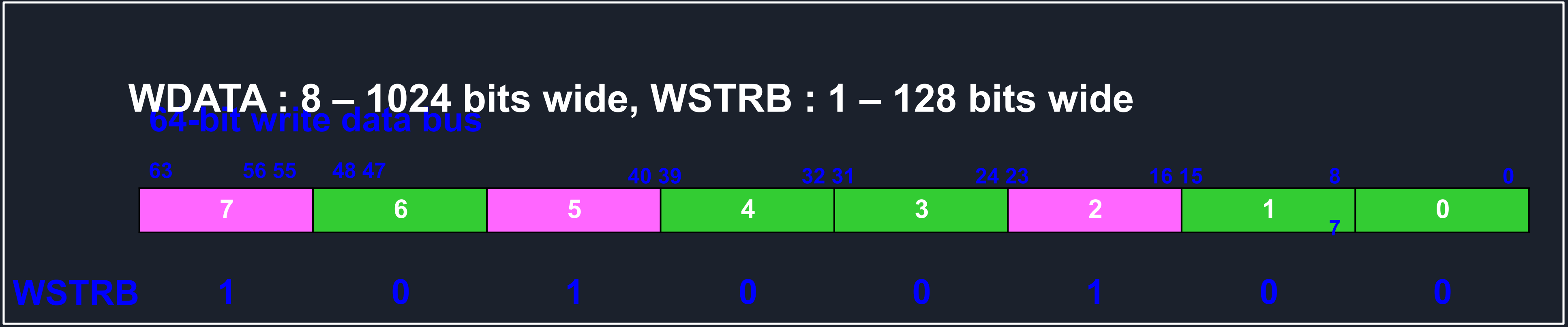


	63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
	7	6	5	4	3	2	1	0
WSTRB	1	0	0	0	0	0	0	0
	F	E	D	C	B	A	9	8
WSTRB	0	0	0	0	1	1	1	0
	F	E	D	C	B	A	9	8
	17	16	15	14	13	12	11	10

Write Strobes

- WSTRB Write strobes, signal indicates which byte lanes to update in memory.

- One strobe for each eight bits of the write data bus.



WDATA : 8 – 1024 bits wide, WSTRB : 1 – 128 bits wide

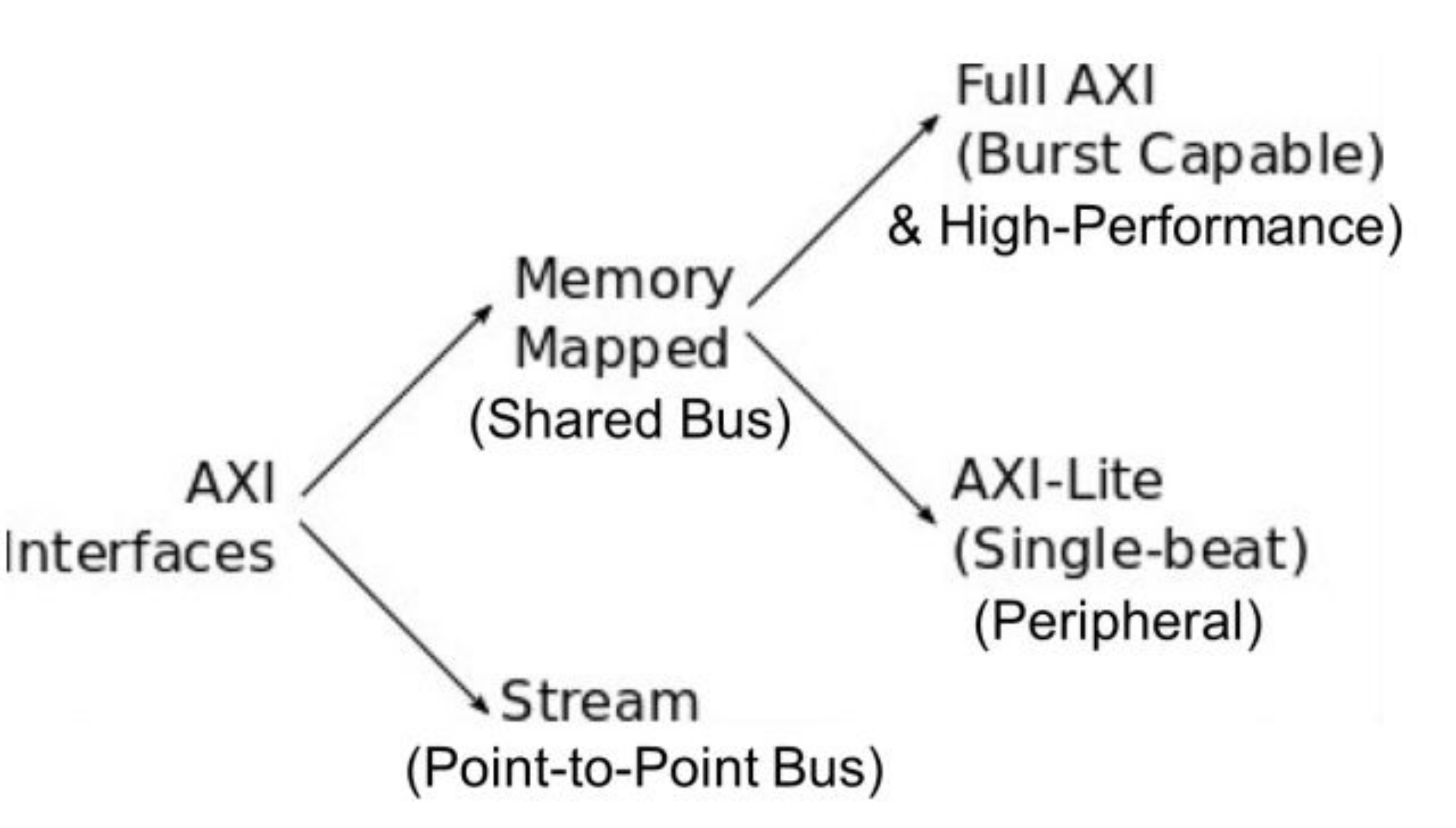
	7	6	5	4	3	2	1	0
WSTRB	1	0	1	0	0	1	0	0

Topics

- Overview of AXI
- AXI Architecture and Features
- Channel Definitions and Signals
- Handshaking Process
- Basic AXI Transactions
- Types of   Access (atomic accesses)
- Ordering Model (AXI ID transaction)
- **AXI4 & AXI4 Lite**

AXI4

AXI Interfaces



AXI Interfaces

Memory Mapped (Shared Bus)

Full AXI (Burst Capable) & High-Performance)

AXI-Lite (Single-beat) (Peripheral)

Stream (Point-to-Point Bus)

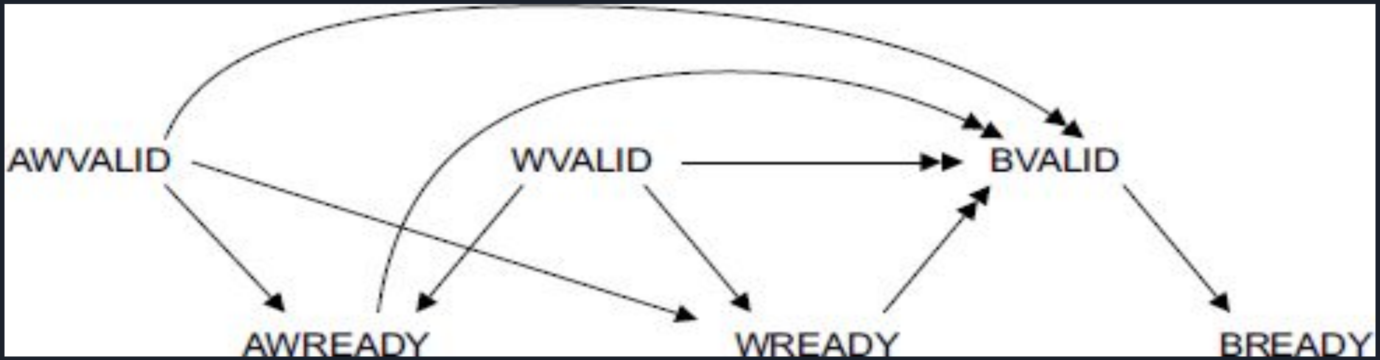
Source: M.S. Sadri, Zynq Training

AXI4 Features

- Support for cache coherency and enforced ordering.
- Support for burst lengths up to 256 beats.
- Support for Quality of Service signaling.
- Support for multiple region interfaces.
- AXI4 does not support locked transactions.

Write Response Dependencies

- The AXI4 protocol requires that the write response for all transactions must not be given until the clock cycle after address acceptance



Write Response Dependencies

AXI4-Lite

AXI4-Lite

Key functionality of AXI4-Lite operation.

- all **transactions are of burst length of 1**
- all **data accesses use full width** of the data bus
- Supports a data bus width of 32-bit or 64-bit.
- all accesses are Non-modifiable, Non-bufferable
- Exclusive accesses are not supported.
- **EXOKAY response is not supported on the read data and write response channels**
- **Length, Burst, Size related signals are not supported**.
- Bus Width can be 32/64 bit and supports strobe.

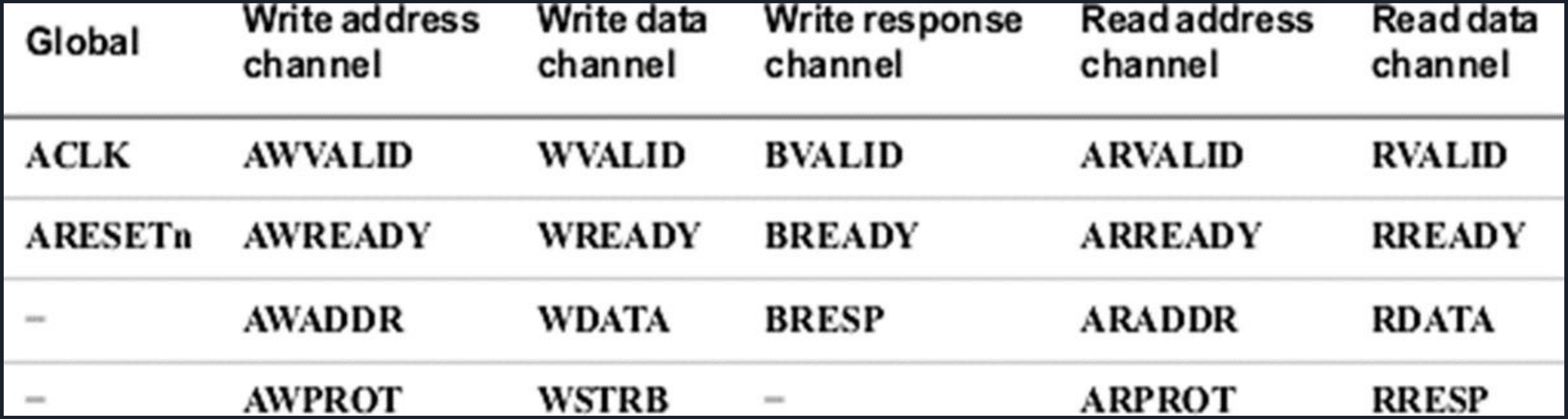
AXI4-Lite

- **No support data interleaving**, the burst length is defined as 1.

- **Does not support AXI IDs.** Meaning, all transactions must be in order, and all accesses use **a single fixed ID value.**

- **Supports multiple outstanding transactions**, but a slave can restrict this by the appropriate use of the handshake signals.

AXI4-Lite Signals



Global	Write address channel	Write data channel	Write response channel	Read address channel	Read data channel
ACLK	AWVALID	WVALID	BVALID	ARVALID	RVALID
ARESETn	AWREADY	WREADY	BREADY	ARREADY	RREADY
–	AWADDR	WDATA	BRESP	ARADDR	RDATA
–	AWPROT	WSTRB	–	ARPROT	RRESP

AXI4-Lite applications

- Simple, low-performance peripherals
    - General Purpose Input Output (GPIO).
        - It’s a port used to connect any peripheral based on requirement where same port is used as input and output.
    - Universal Asynchronous Receiver Transmitter (UART Lite)

References

[1] AMBA AXI Protocol v 1.0 Specification, ARM, 2003.

[2] AMBA Specification, Rev 2.0, ARM, 1999.

[3] AMBA 4 AXI4-Stream Protocol Version 1.0

[4] http://infocenter.arm.com/help/topic/com.arm.doc.set.amba

[5] https://www.slideshare.net/AzadMishra1/axi-protocol-55779579

Thank you