Introduction to AXI Protocol

For High-Performance Memory Controllers

What is AXI?

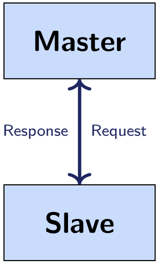
AMBA Advanced eXtensible Interface

Design Goals:

- High bandwidth, low latency
- High-frequency operation
- Flexible (wide range of components)
- **Optimized for memory controllers** with high initial access latency

Key Innovation:

Burst-based transactions with out-of-order completion



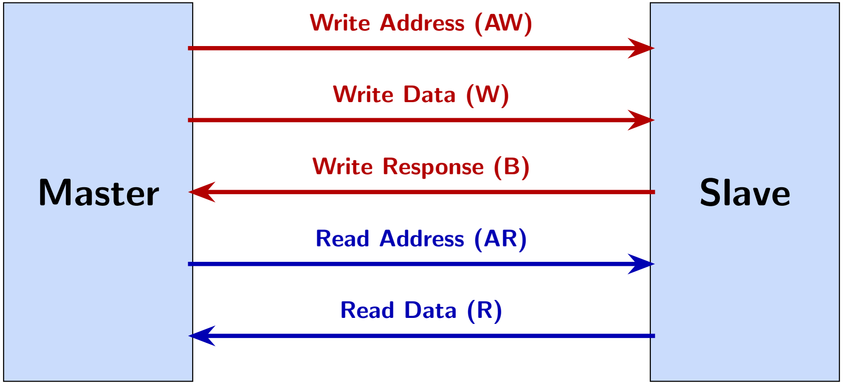
Perfect for DDR Memory!

AXI Protocol Versions

Feature	**AXI3**	**AXI4**	**AXI4-Lite**
Burst Length	1-16 beats	**1-256 beats**	1 beat only
Write Interleaving	Yes (WID)	**No (removed)**	N/A
QoS Signaling	No	**Yes**	No
Region Signals	No	Yes	No
Use Case	Legacy	**Memory Ctrl**	Simple Periph

AXI4 recommended for new memory controller designs

Five Independent Channels



Key Benefit: Read and Write operate **simultaneously** (full duplex)

Write Transaction Channels

Write Address (AW)

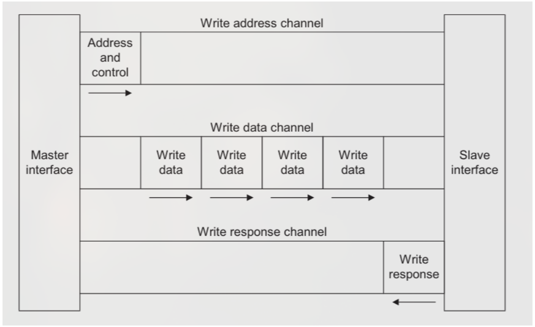
- AWID, AWADDR
- AWLEN, AWSIZE
- AWBURST, AWQOS

Write Data (W)

- WDATA, WSTRB
- WLAST

Write Response (B)

- BID, BRESP

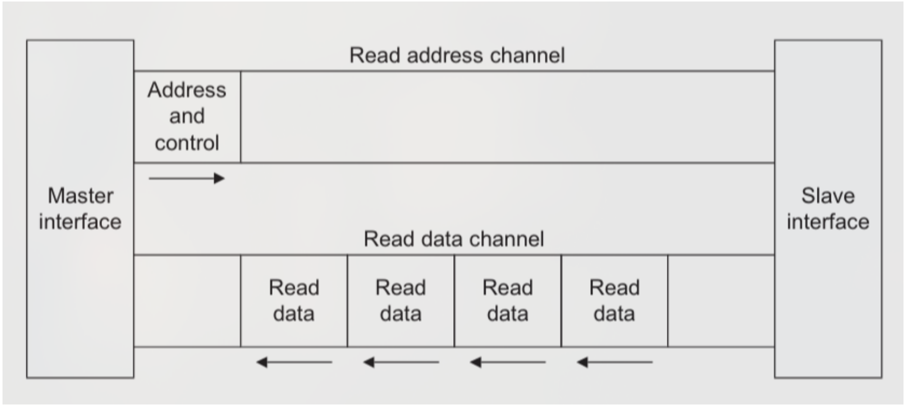


Read Transaction Channels

1. **Read Address (AR)**
   - ARID - Transaction ID
   - ARADDR - Start address
   - ARLEN - Burst length
   - ARSIZE - Bytes per beat
   - ARBURST - Type (FIXED/INCR/WRAP)
   - ARQOS - Priority (AXI4)

2. **Read Data (R)**
   - RID - Matches ARID
   - RDATA - Actual data
   - RRESP - Status
   - RLAST - Final beat indicator

Read Channel Diagram



VALID/READY Handshake Protocol

Every channel uses:

- **VALID** - Source has data
- **READY** - Destination can accept
- **Transfer** - Both HIGH

Critical Rules:

- VALID **must NOT** wait for READY
- READY **can** wait for VALID
- Prevents deadlock!

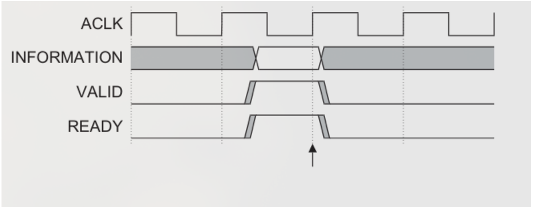


Figure 3-3 VALID with READY handshake

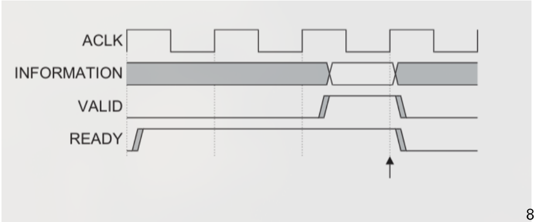


Figure 3-2 READY before VALID handshake

Burst Transaction Parameters

Parameter	Encoding	Meaning
AWLEN/ARLEN	0-255 (AXI4) 0-15 (AXI3)	Beats $=$ LEN $+ 1$
AWSIZE/ARSIZE	3 bits 000 $\rightarrow$ 1 byte 011 $\rightarrow$ 8 bytes 111 $\rightarrow$ 128 bytes	Bytes/beat $= 2^{SIZE}$

Total Burst Size $= (\text{AWLEN} + 1) \times 2^{AWSIZE}$

AXI4 Max: 256 beats $\times$ 128 bytes $=$ **32 KB** per burst!

Burst Types

FIXED (00)

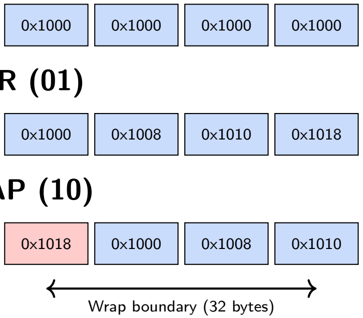
Address constant

INCR (01)

Increments by SIZE

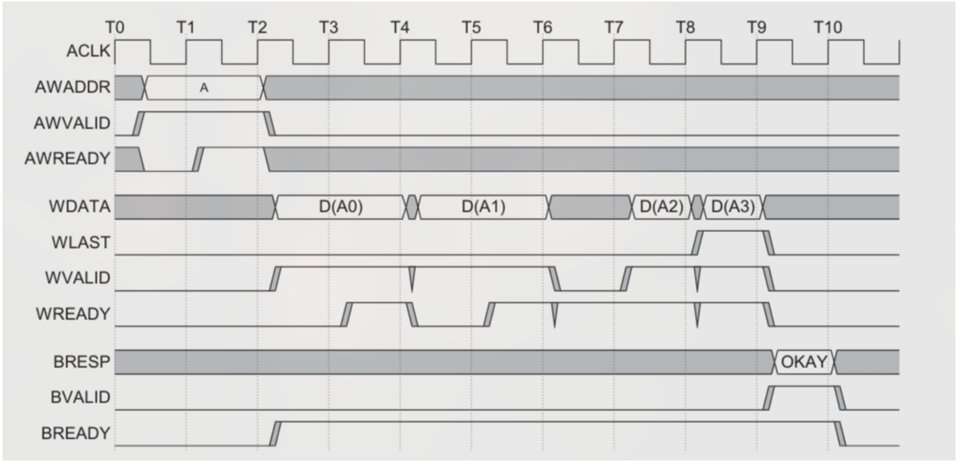
WRAP (10)

Wraps at boundary

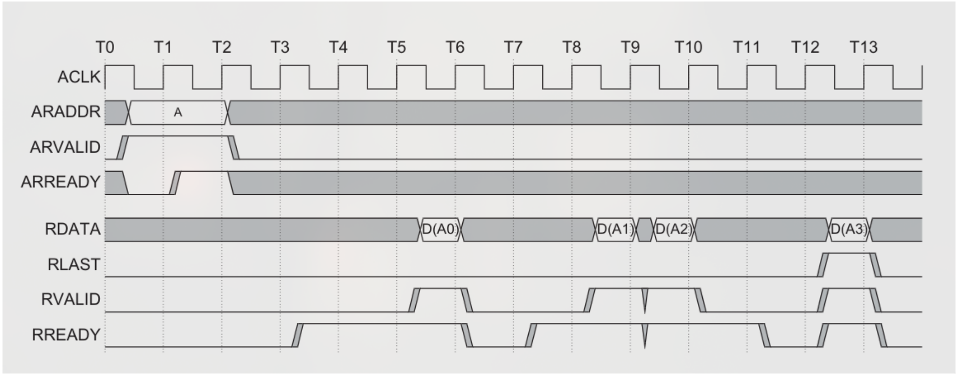

0x1000 | 0x1000 | 0x1000 | 0x1000
0x1000 | 0x1008 | 0x1010 | 0x1018
0x1018 | 0x1000 | 0x1008 | 0x1010
Wrap boundary (32 bytes)

Most common for DDR: INCR **For caches:** WRAP

Write Burst Example

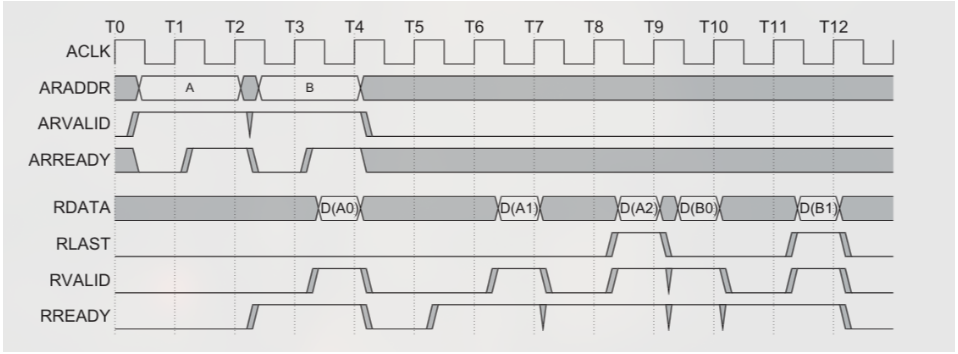


Read Burst Example



Slave returns multiple data beats for single address request

Overlapping Bursts (Pipelining)



Multiple transactions in flight → Hides memory latency!

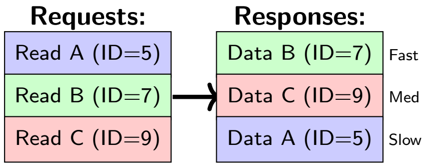
Transaction IDs & Out-of-Order Completion

Why IDs?

- Memory has high latency
- Issue multiple requests
- Complete when ready
- Use ID to route response

Ordering Rules:

- **Same ID** → In-order
- **Different ID** → Any order



Out-of-order = Performance!

AXI4 Enhancements Over AXI3

Performance:

- **256 beat bursts** (was 16)
- Longer sequential access
- Better DDR utilization

Quality of Service:

- ARQOS/AWQOS signals
- 4-bit priority (0-15)
- Real-time traffic prioritization

Simplification:

- **Removed WID** signal
- No write data interleaving
- Simpler slave implementation
- Less hardware complexity

Additional:

- Region identifiers
- User-defined signals
- Enhanced cache attributes

AXI to DDR Command Translation

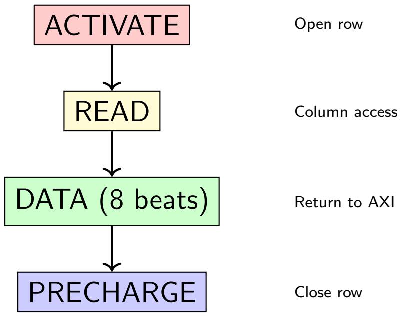
AXI Read Request:

- ARADDR = 0x9000_1000
- ARLEN = 7 (8 beats)
- ARSIZE = 3 (8 bytes)

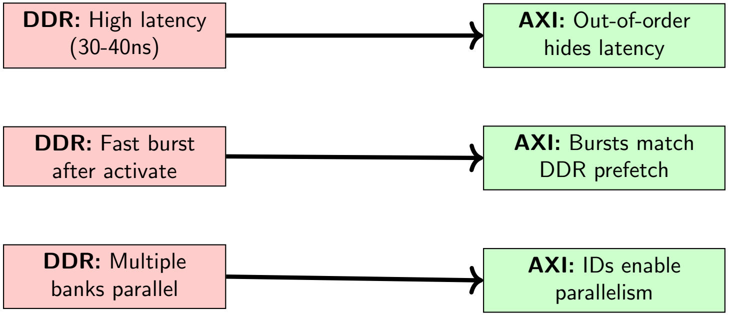
Memory Controller Decodes:

- Bank = 2
- Row = 0x900
- Column = 0x10

DDR Commands Issued:



Why AXI is Perfect for DDR Memory Controllers


DDR: High latency (30-40ns)
AXI: Out-of-order hides latency
DDR: Fast burst after activate
AXI: Bursts match DDR prefetch
DDR: Multiple banks parallel
AXI: IDs enable parallelism

Perfect match between protocol and memory!

Memory Controller Optimizations Using AXI

Transaction Reordering:

- Row buffer hits first
- Bank parallelism
- Min read/write switch

Write Buffering:

- Collect small writes
- Combine to DDR bursts
- Early response (BRESP)

QoS Scheduling:

- Priority-based (ARQOS/AWQOS)
- Real-time = high priority
- DMA = low priority

Burst Alignment:

- Match DDR BL8
- Align to cache lines
- Respect 4KB boundary

Key Takeaways

5 Independent Channels
Full duplex read/write

Burst-Based Protocol
One address, multiple data

Out-of-Order Capable
Transaction IDs enable flexibility

Perfect for DDR
Hides latency, maximizes bandwidth

Critical Rules:
VALID must NOT wait for READY • Bursts cannot cross 4KB • Same ID = ordered