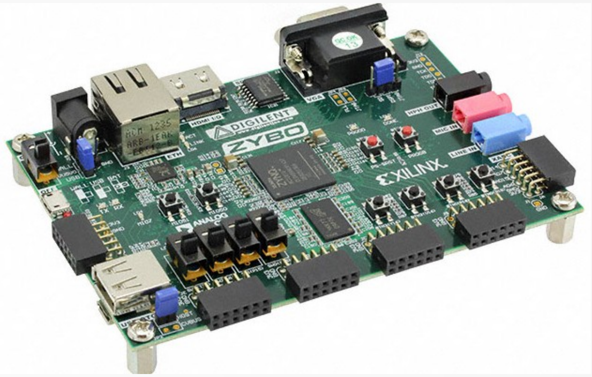
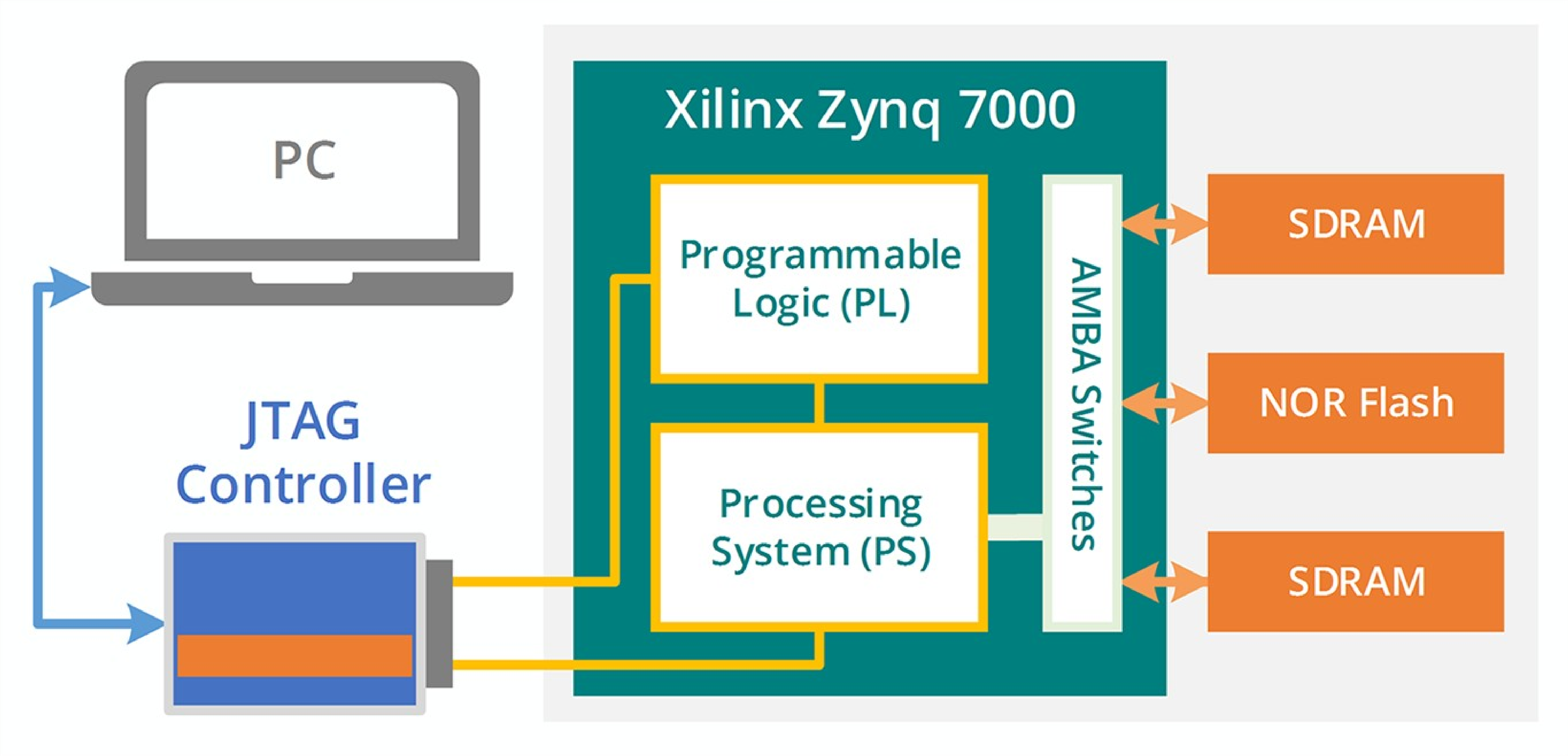


Course 02: PS-PL interface

CHERIF Bilel

5A-SIEC

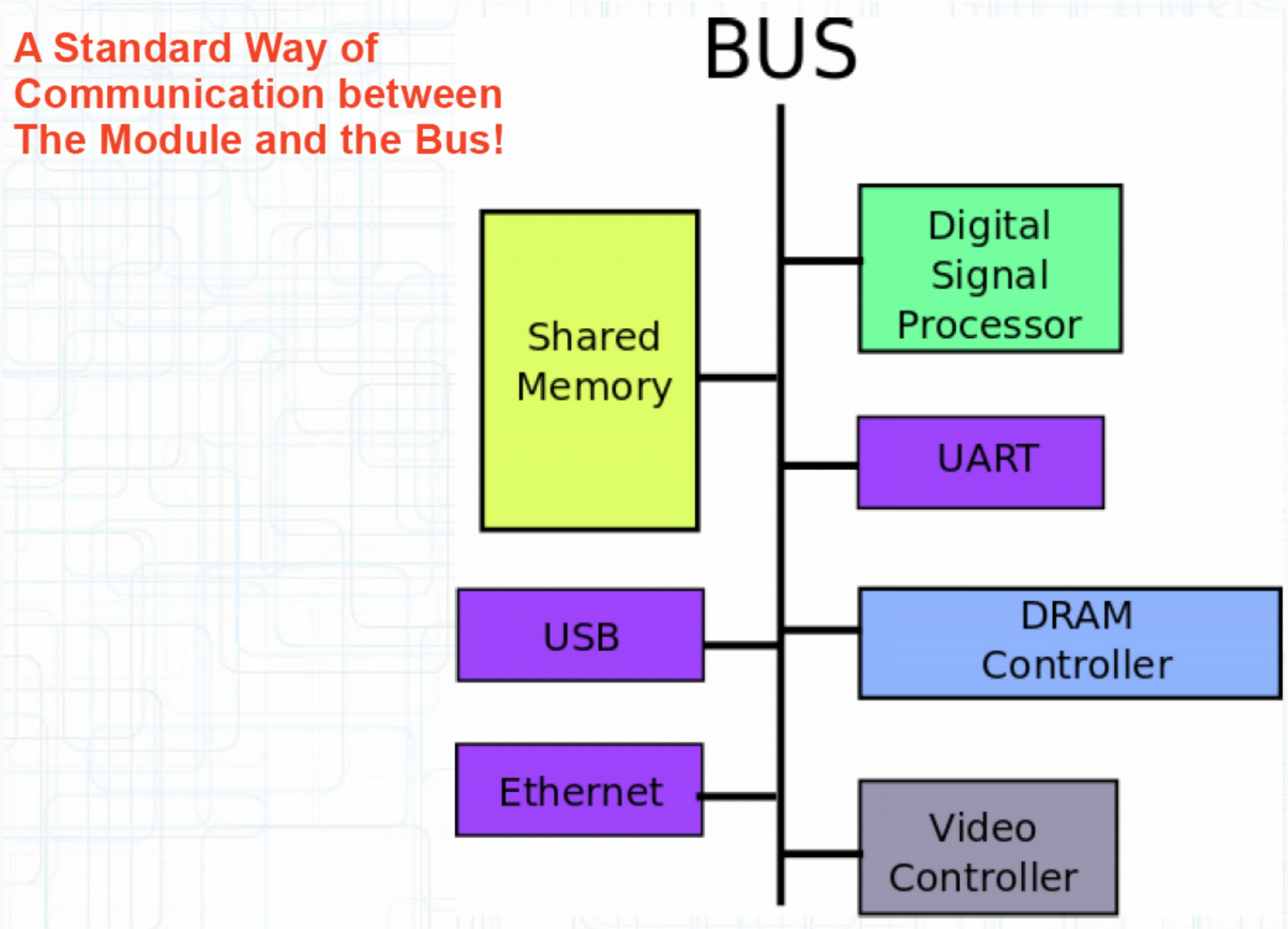
Zynq


PC
JTAG
Controller
Xilinx Zynq 7000
Programmable
Logic (PL)
Processing
System (PS)
AMBA Switches
SDRAM
NOR Flash
SDRAM

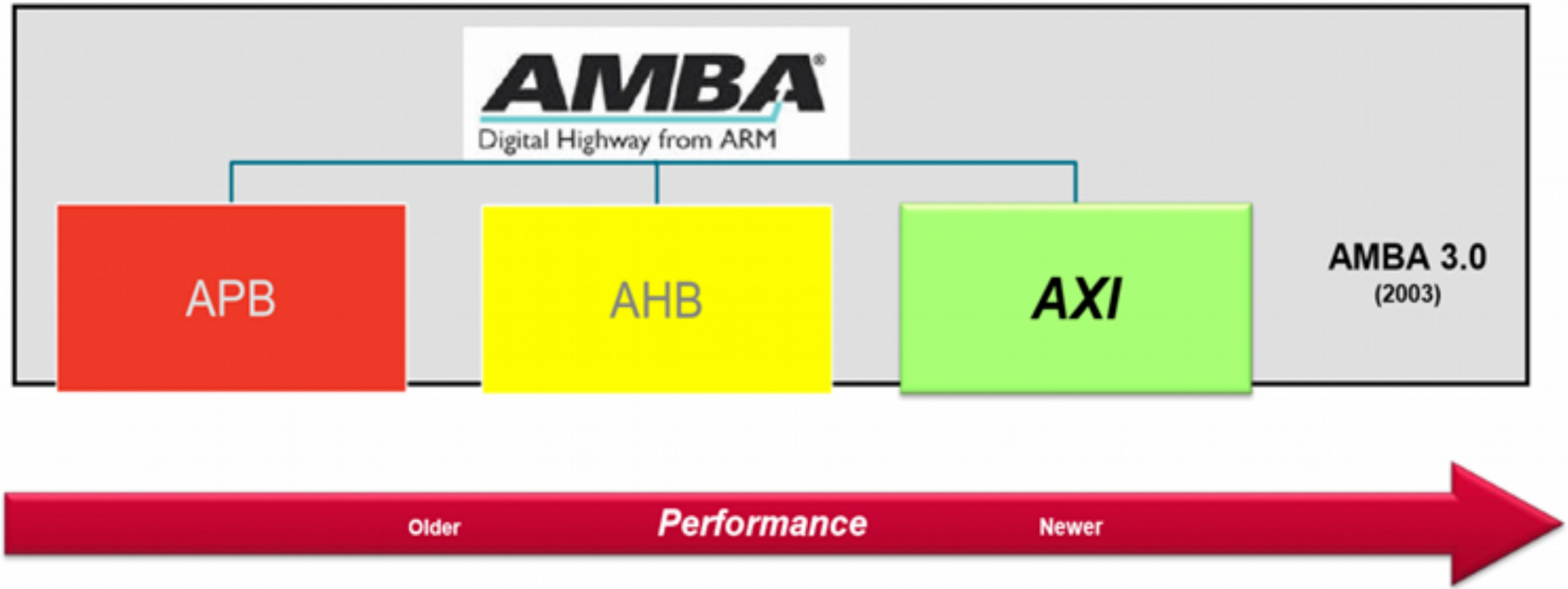
SOC Bus

- A standard
  - All units talk based on that standard
  - All units can talk easily to each other
- Maintenance
  - Design is easily maintained/updated, debugged
- Re-use
  - Units can be easily re-used in other desigs

Soc Bus


A Standard Way of
Communication between
The Module and the Bus!
BUS
Shared
Memory
Digital
Signal
Processor
UART
USB
DRAM
Controller
Ethernet
Video
Controller

AMBA

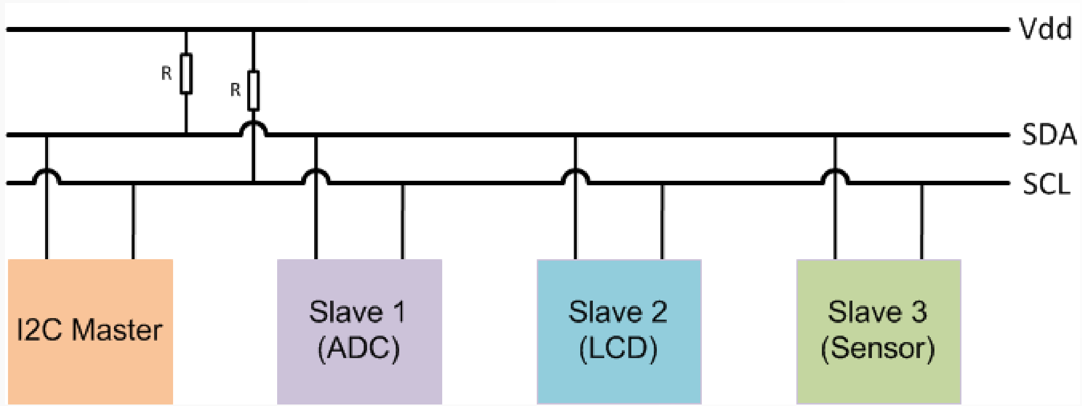


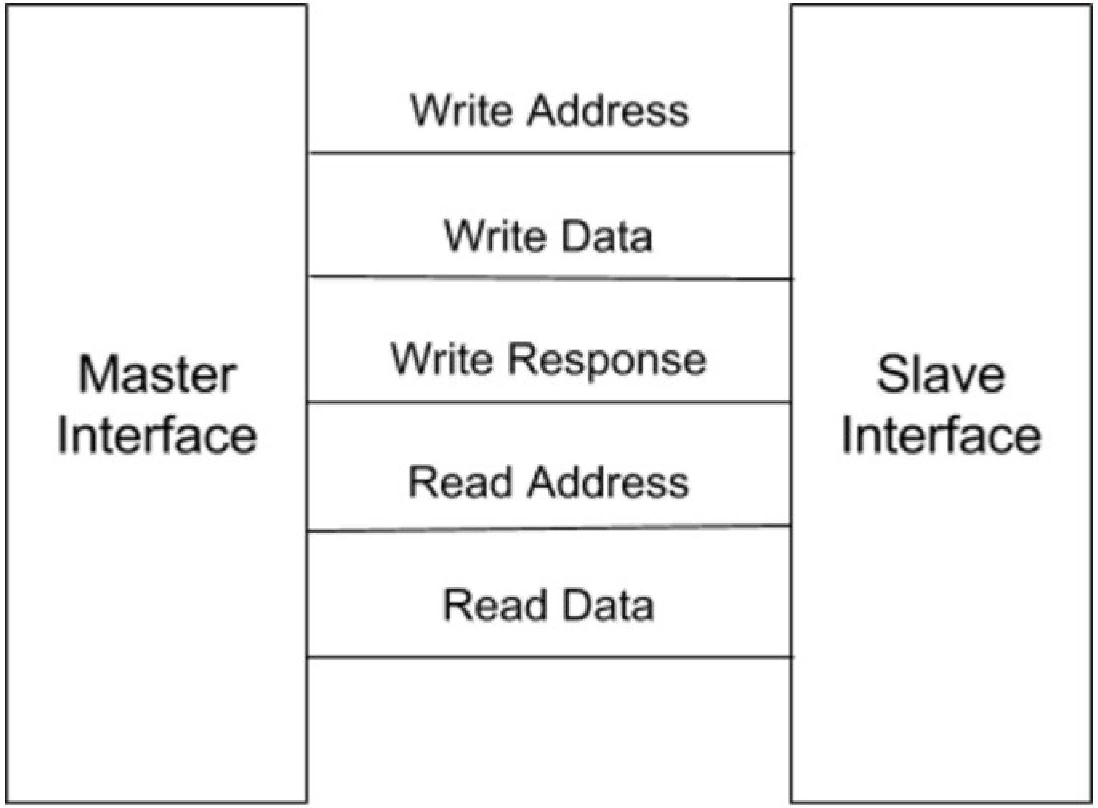
AMBA: Advanced Microcontroller Bus Architecture
AXI: Advanced Extensible Interface

AXI vs AHB

AHB : Advanced High-performance Bus	AXI : Advanced eXtensible Interface
Shared bus	Interface
single channel Bus	multi- channel Bus
each of the bus masters will connect to a single-channel shared bus	connect to a Read data channel, Read address channel, Write data channel, Write address channel and Write response channel
Low power	Uses around 50 % more power

Bus vs Interface



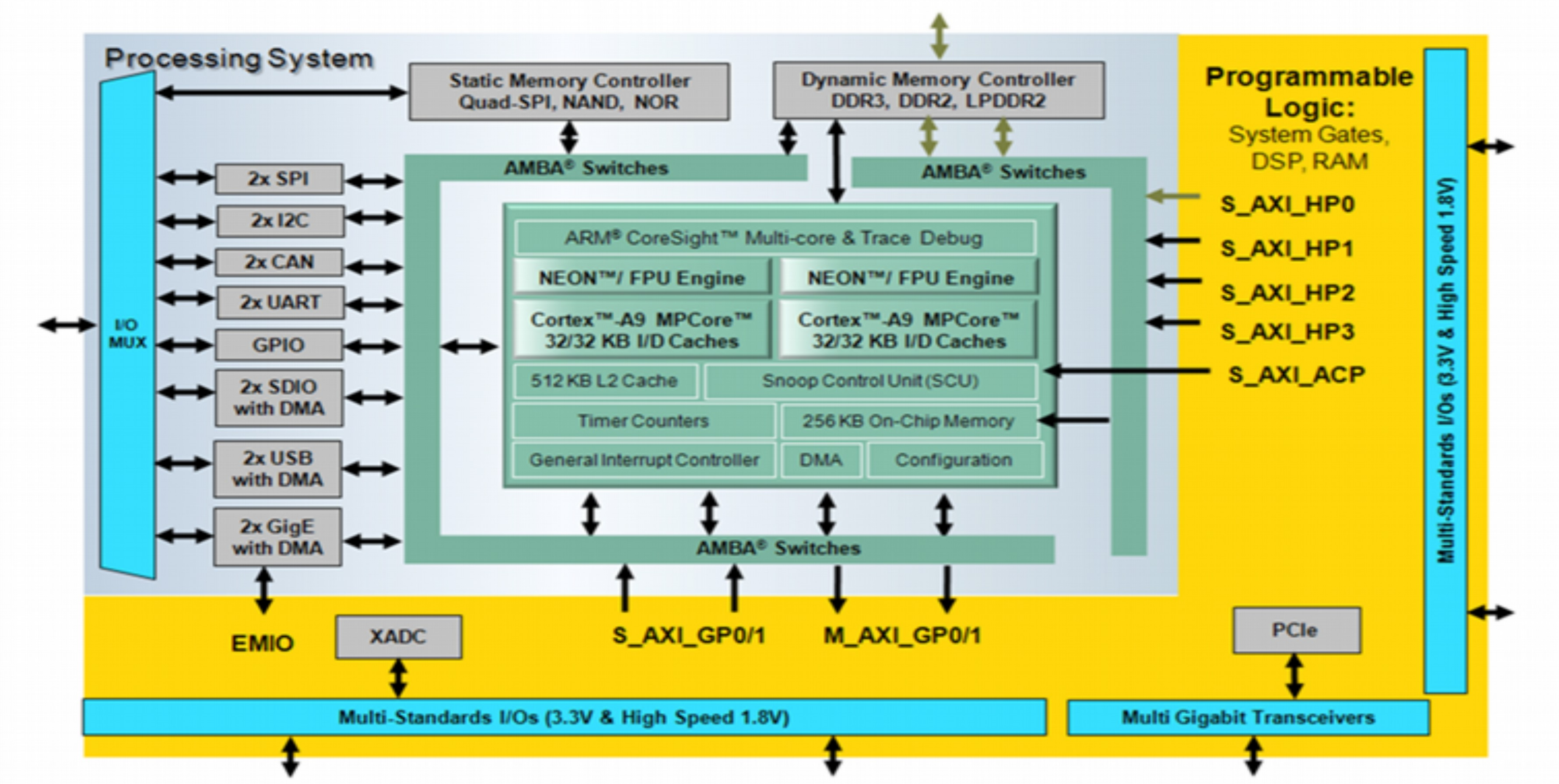


Channel connections between master and slave interfaces

Terminology

- Transaction :
  - Transfer of data from one point in the hardware to another point
- Master : Initiates the transaction
- Slave : Responds to the initiated transaction

PS-PL interfaces



PS-PL interfaces

- Two 32-bit Master AXI ports (PS master)
- Two 32-bit Slave AXI ports (PL Master)
- Four 32/64-bit Slave High Performance Ports (PL Master)
- One 64-bit Slave Accelerator Coherency Port (ACP) (PL Master)
- Four clocks from the PS to the PL
- PS to PL Interrupts
- PL to PS Interrupts

AXI interface

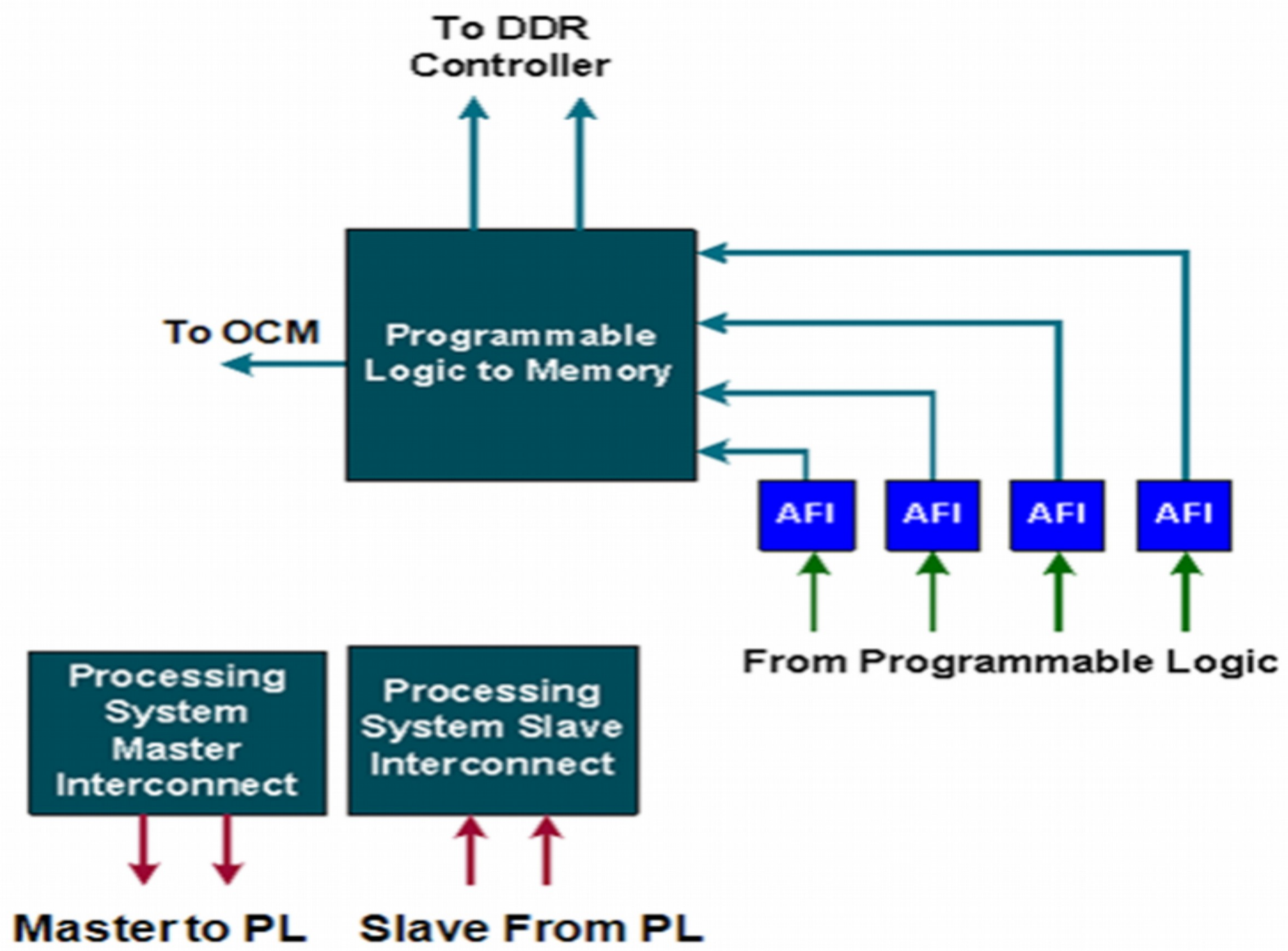
The Zynq SoC supports three different AXI transfer types that you can use to interface the PS to the PL side of the device:

- AXI4 Burst transfers
- AXI4-Lite for simple control interfaces
- AXI4-Streaming for unidirectional data transfers

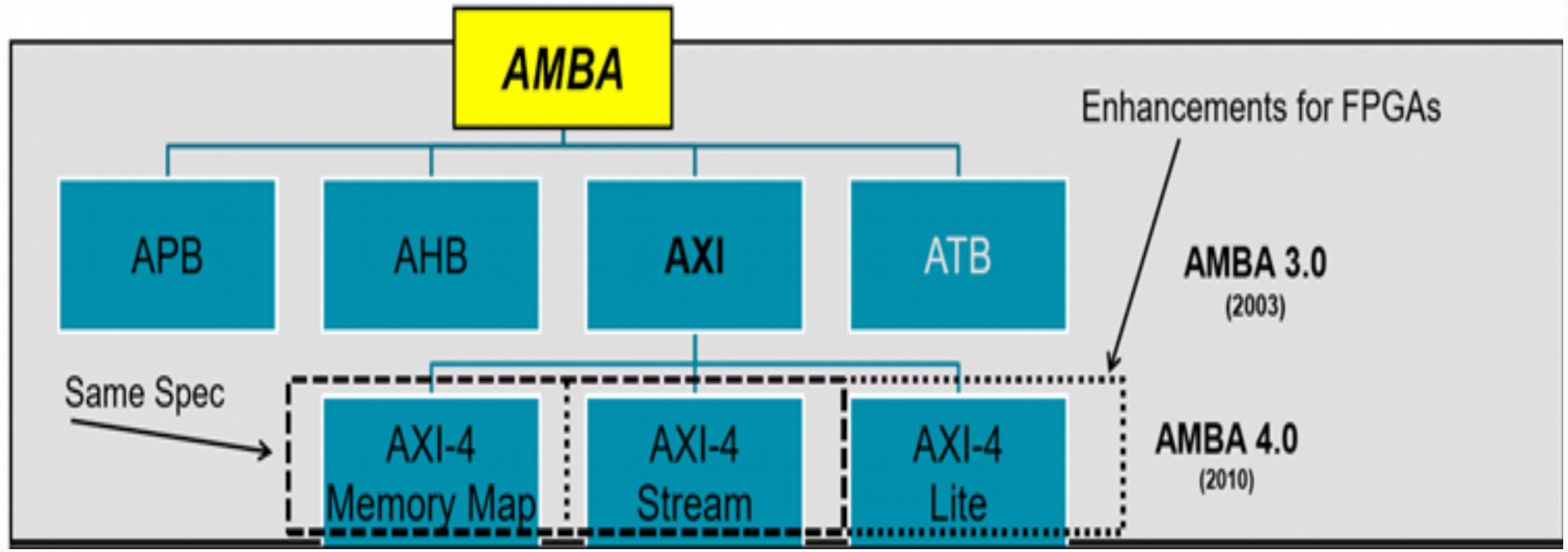
The theoretical bandwidths of each of the interfaces are defined in the table below:

Interface	Width	IF Clock	Read BW	Write BW	Combined	No Ports	Total BW
AXI GPIO	32	150 MHz	600 MBps	600 MBps	1200 MBps	2	2400 MBps
AXI HP	64	150 MHz	1200 MBps	1200 MBps	2400 MBps	4	9600 MBps
AXI ACP	64	150 MHz	1200 MBps	1200 MBps	2400 MBps	1	2400 MBps

AXI interface

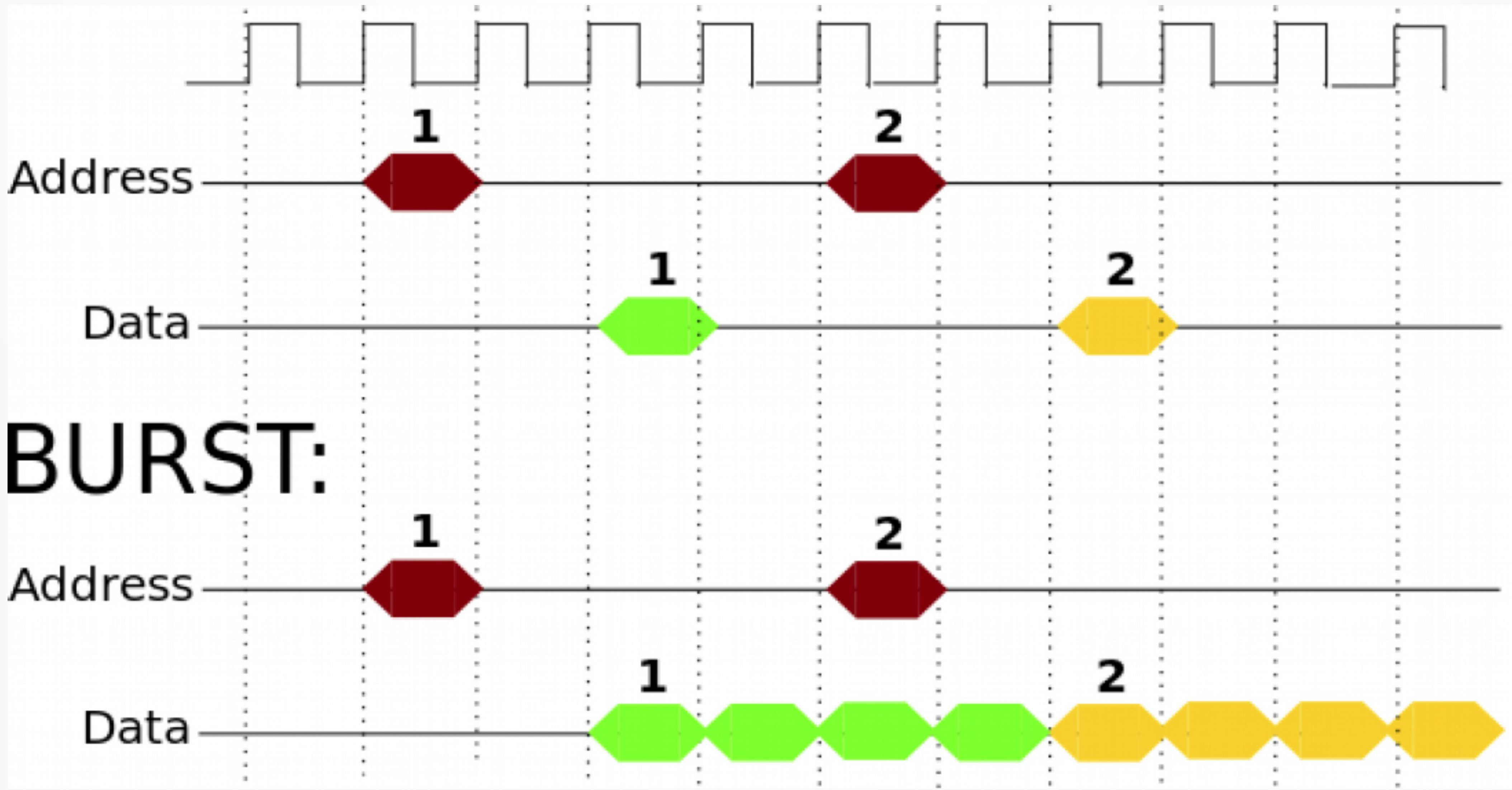


AXI Types



Interface	Features	Similar to
Memory Map / Full (AXI4)	Traditional Address/Data Burst (single address, multiple data)	PLBv46, PCI
Streaming (AXI4-Stream)	Data-Only, Burst	Local Link / DSP Interfaces / FIFO / FSL
Lite (AXI4-Lite)	Traditional Address/Data—No Burst (single address, single data)	PLBv46-single OPB

Burst



AXI Channels

The AXI interface has separate and independent read and write channels that can be used simultaneously.

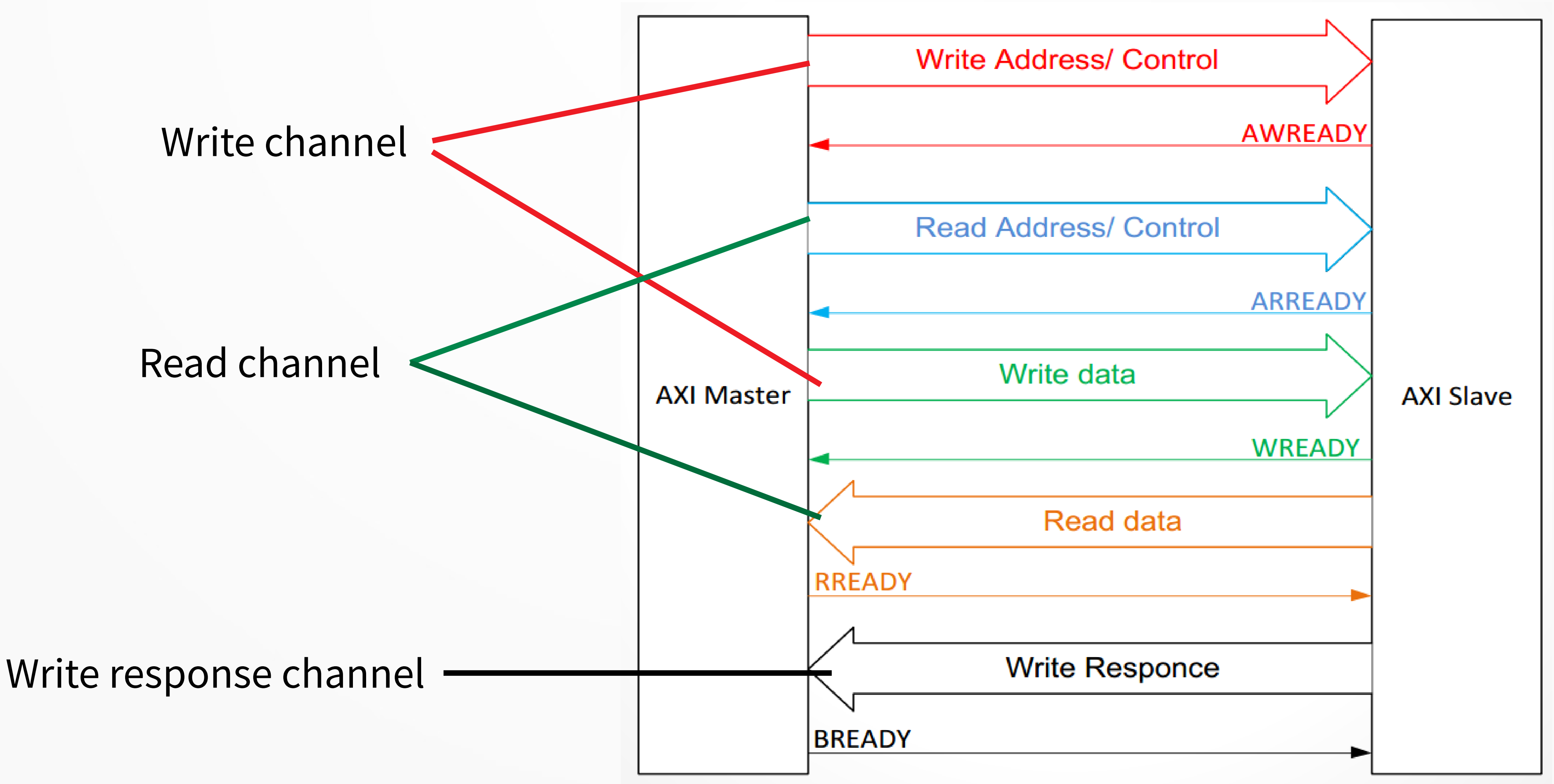
Each channel has its own address and data buses.

Both channels are non-posted (there is always a response).

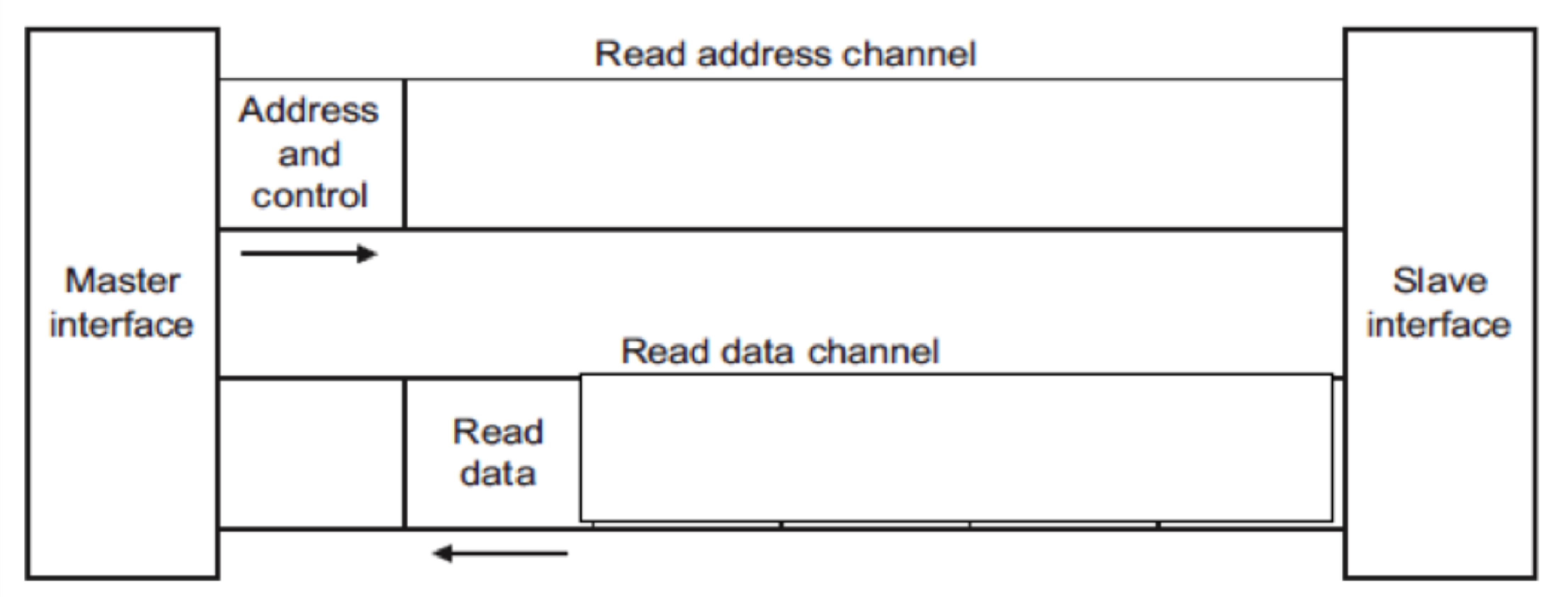
- In the read case the response is simply the read data coming back.

- For a write, a separate response bus acknowledges data delivery

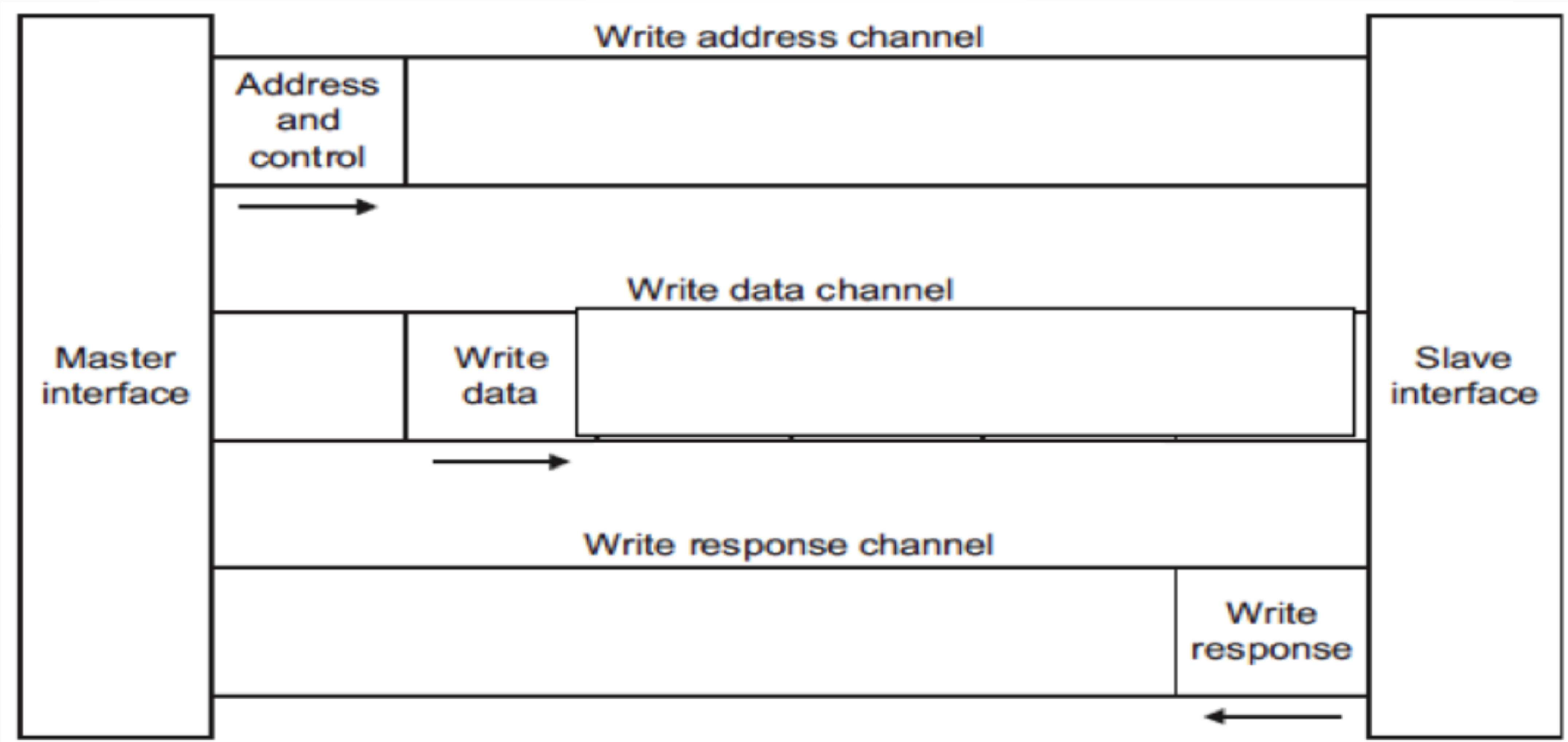
AXI channels



Read Channels



Write Channels



AXI Lite and AXI Full

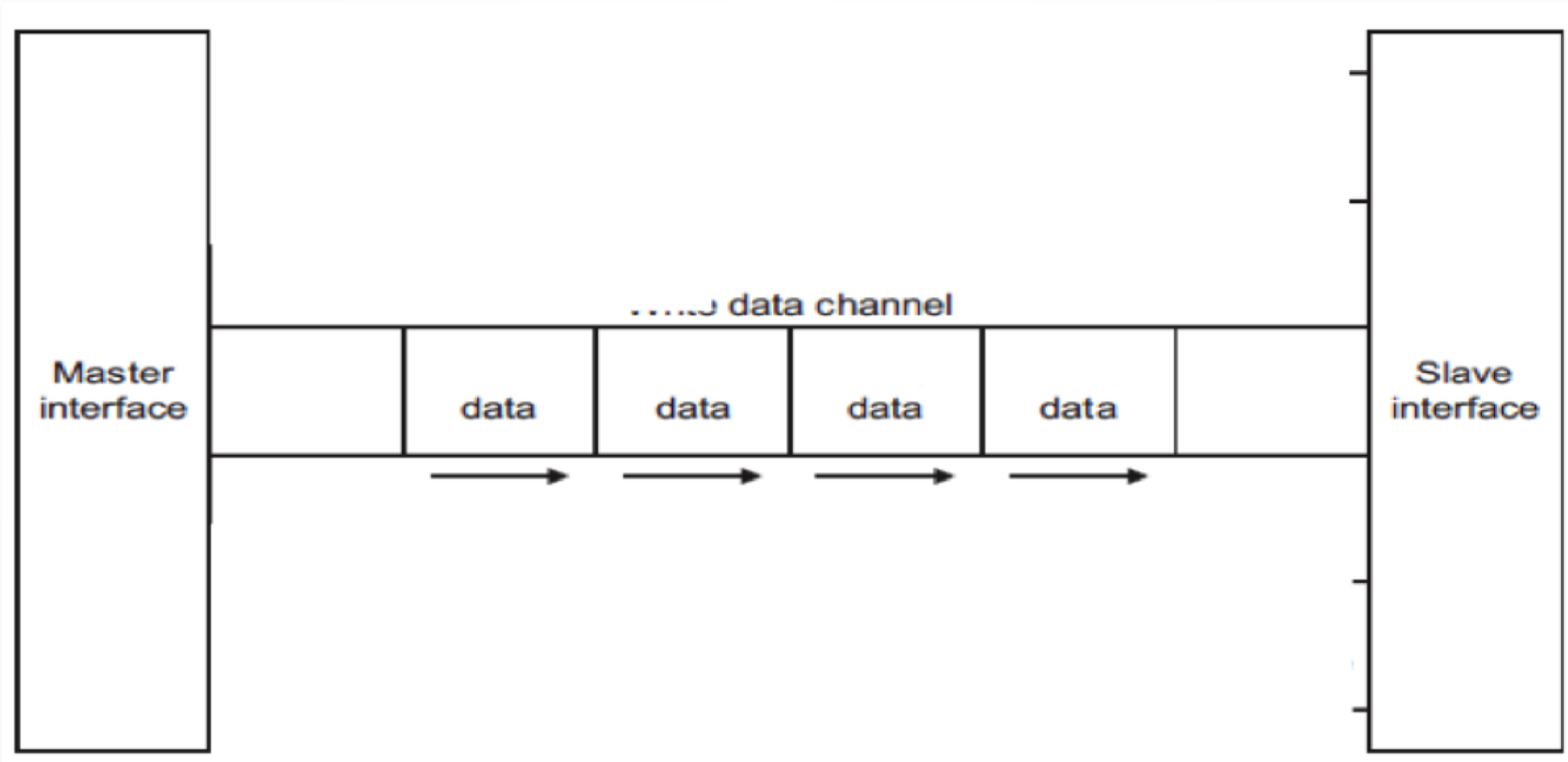
AXI LITE

- No burst
- Data width 32 or 64 only – Xilinx IP only supports 32-bits
- Very small footprint
- Bridging to AXI4 handled automatically by AXI_Interconnect (if needed)

AXI FULL

- Sometimes called "Full AXI" or "AXI Memory Mapped" – Not ARM-sanctioned names
- Single address multiple data – Burst up to 256 data beats
- Data Width parameterizable – 1024 bits

AXI Stream


data channel
Master interface
data
data
data
data
Slave interface

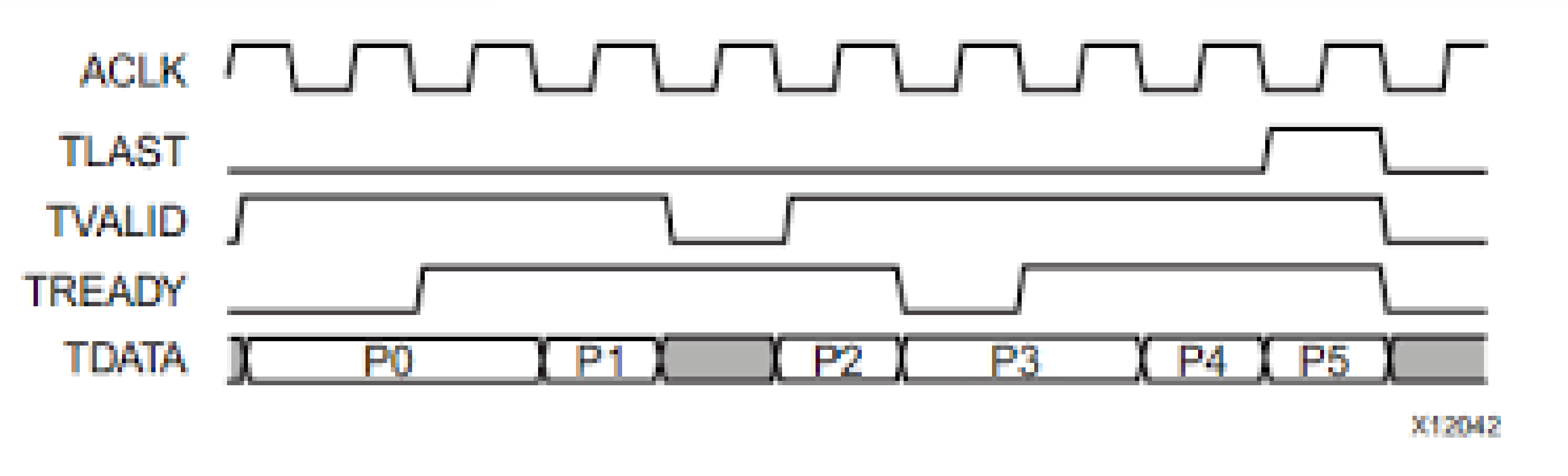
AXI Stream

No address channel, no read and write, always just master to slave

– Effectively an AXI4 “write data” channel Unlimited burst length

– Protocol allows merging, packing, width conversion

– Supports sparse, continuous, aligned, unaligned streams



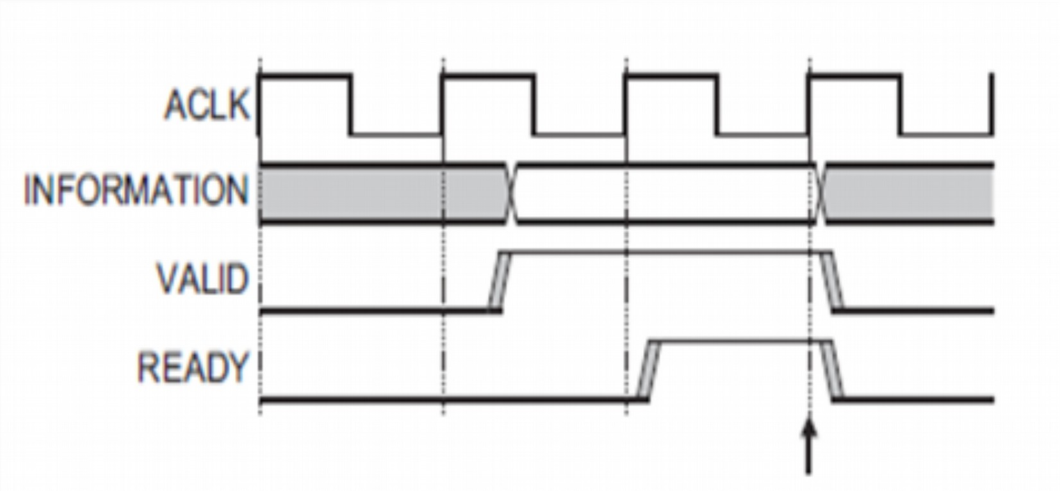
AXI Lite signals

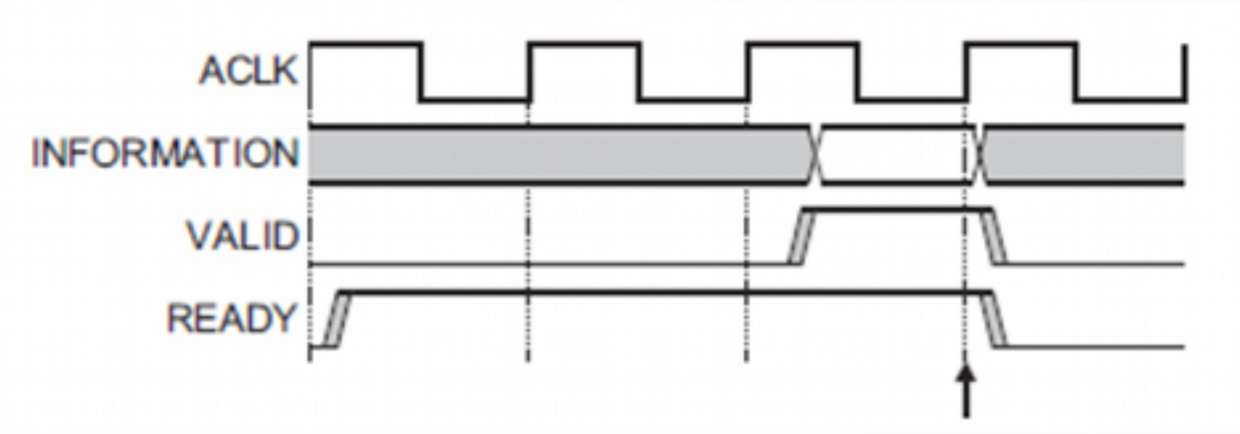
Global	Write address channel	Write data channel	Write response channel	Read address channel	Read data channel
ACLK	AWVALID	WVALID	BVALID	ARVALID	RVALID
ARESETn	AWREADY	WREADY	BREADY	ARREADY	RREADY
–	AWADDR	WDATA	BRESP	ARADDR	RDATA
–	AWPROT	WSTRB	–	ARPROT	RRESP

AXI Channel handshaking

Handshaking

- AXI uses a valid/ready handshake acknowledge
- Each channel has its own valid/ready
  - Address (read/write)
  - Data (read/write)
  - Response (write only)
- Flexible signaling functionality
  - Inserting wait states
  - Always ready
  - Same cycle acknowledge

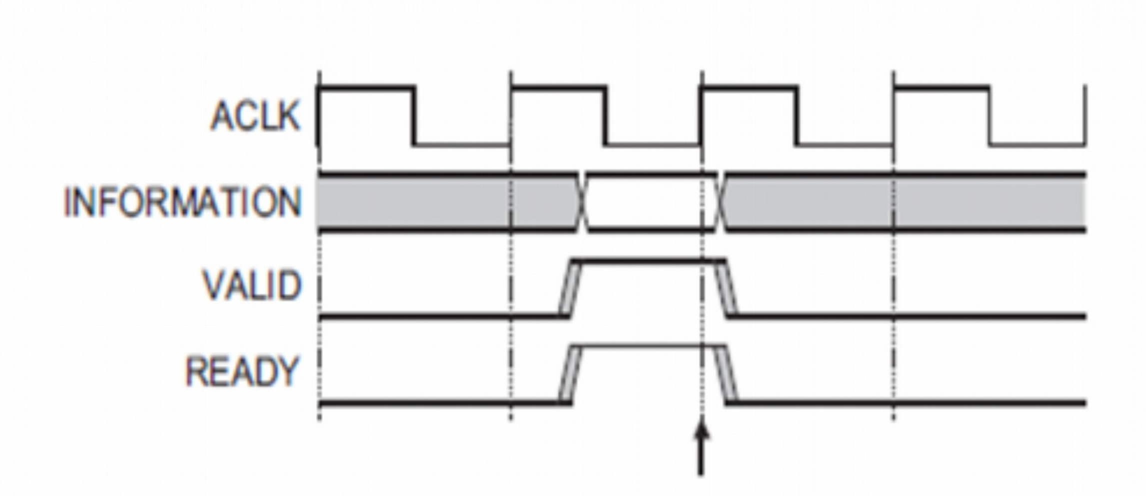




Note:

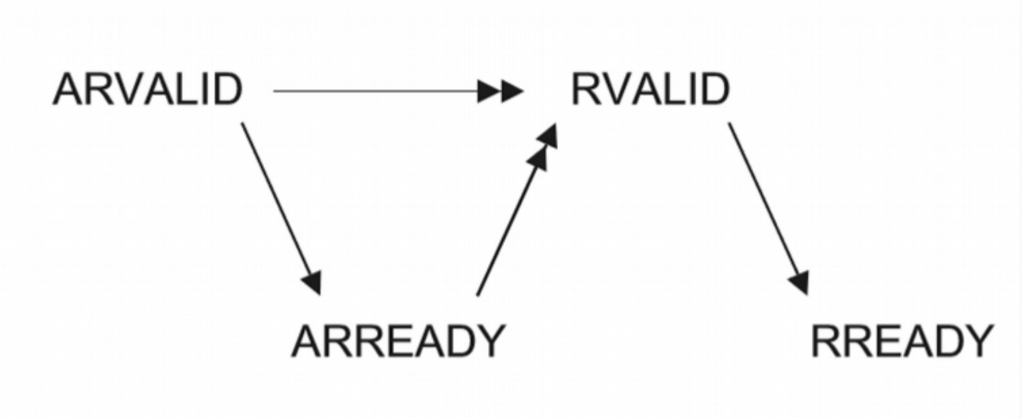
It is up to the master to assert the valid signal and the slave to assert the ready signals for all channels except the read data channel where the slave asserts valid to indicate that it is returning data.

The agent that asserts ready determines the flexibility as seen in the three waveform options.



In any transaction:

- the VALID signal of one AXI component must not be dependent on the READY signal of the other component in the transaction
- the READY signal can wait for assertion of the VALID signal.

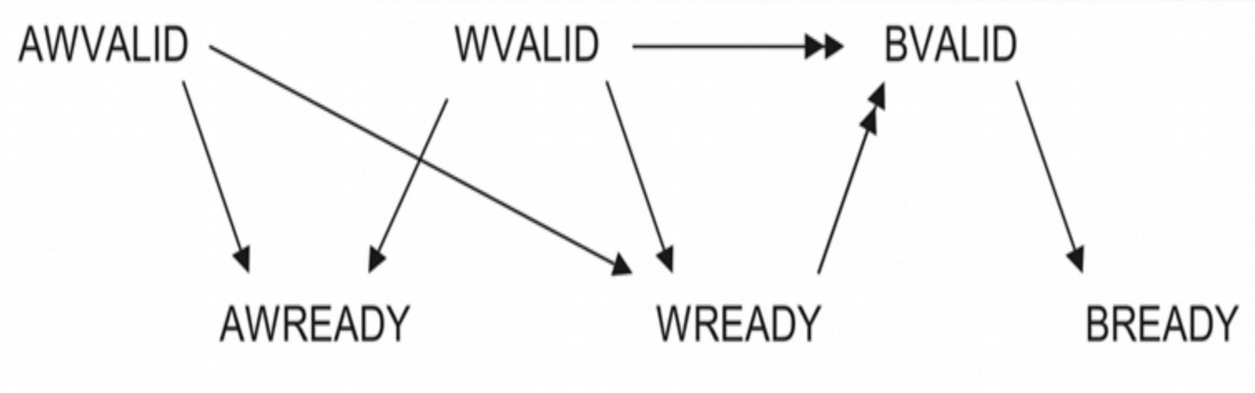


Read Transaction :

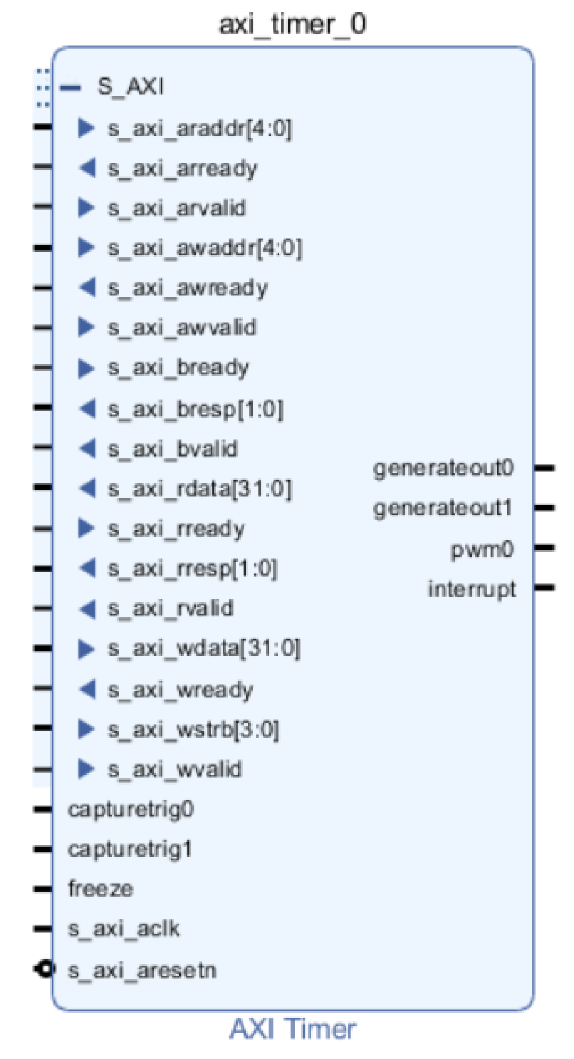
- the slave can wait for ARVALID to be asserted before it asserts ARREADY
- the slave must wait for both ARVALID and ARREADY to be asserted before it starts to return read data by asserting RVALID.

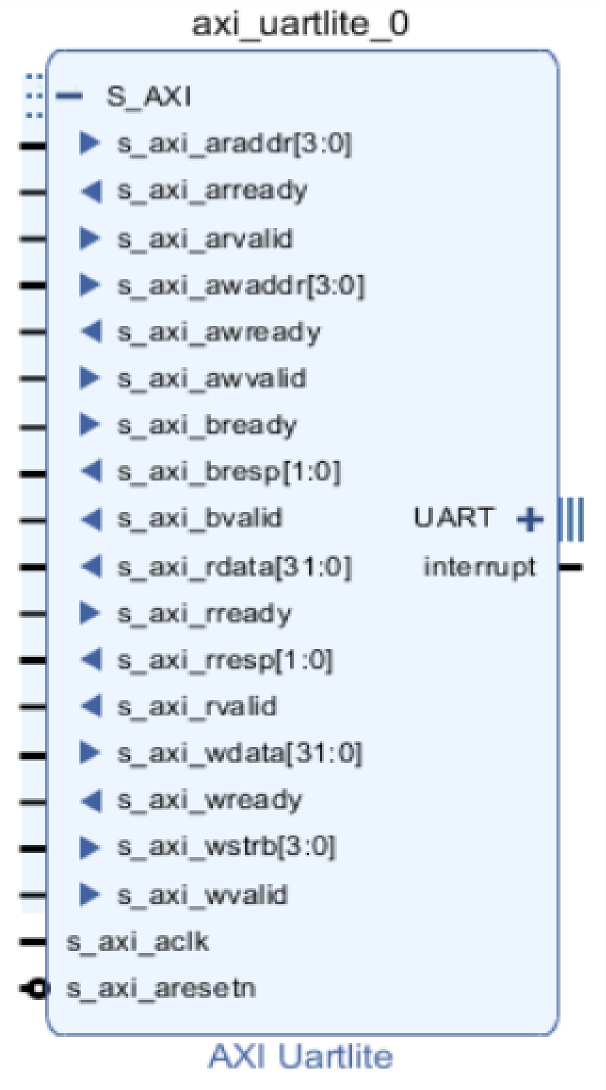
Write transaction

- the master must not wait for the slave to assert AWREADY or WREADY before asserting AWVALID or WVALID
- the slave can wait for AWVALID or WVALID, or both, before asserting AWREADY
- the slave can wait for AWVALID or WVALID, or both, before asserting WREADY
- the slave must wait for both WVALID and WREADY to be asserted before asserting BVALID.



AXI Lite IPs example





Role of Write Strobe WSTRB

OLD reg value

0xDE	0x05	0xBE	0x00

Write Data

0X02	0xAD	0x15	0xEF

WSTRB

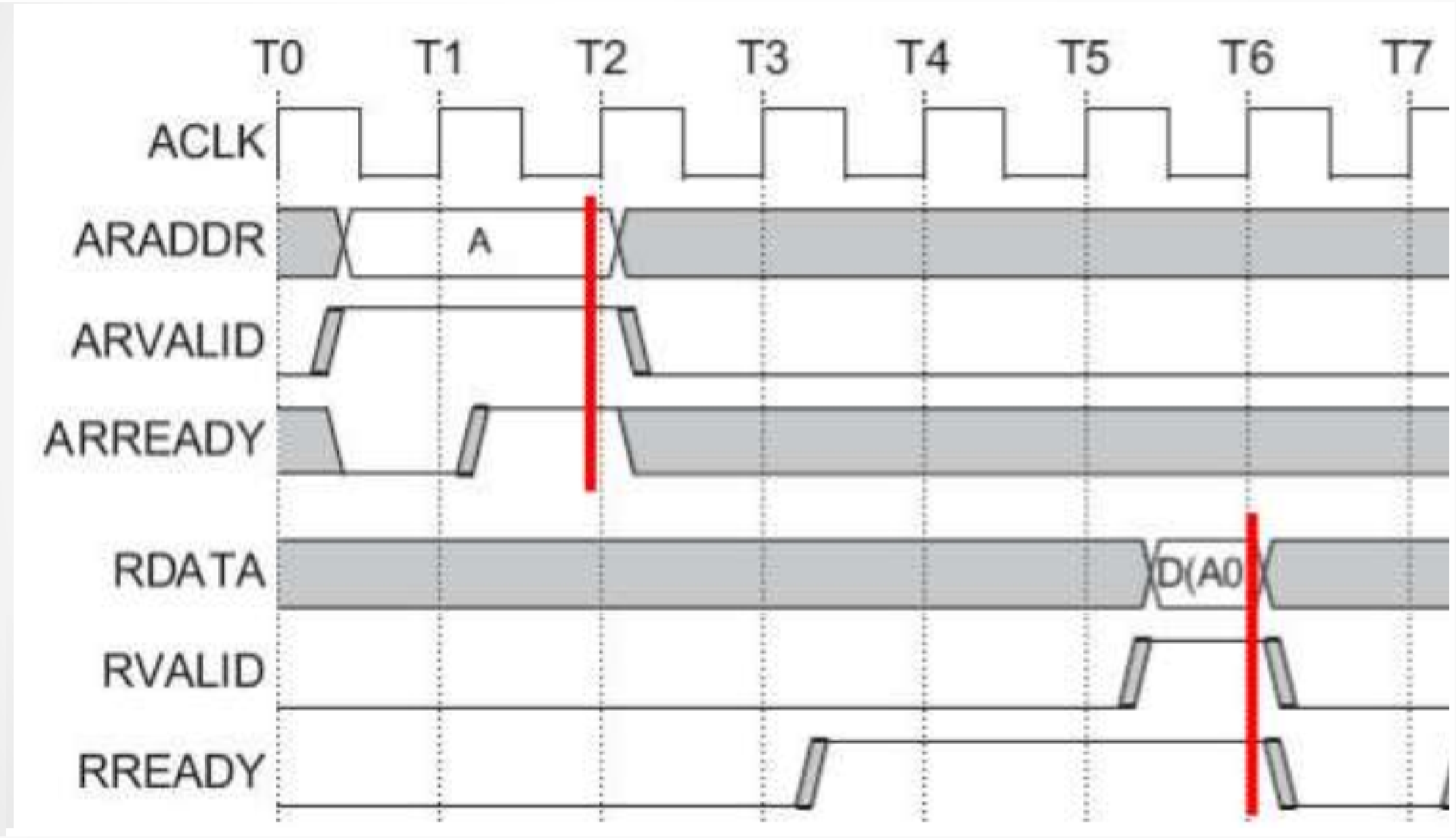
0	1	0	1

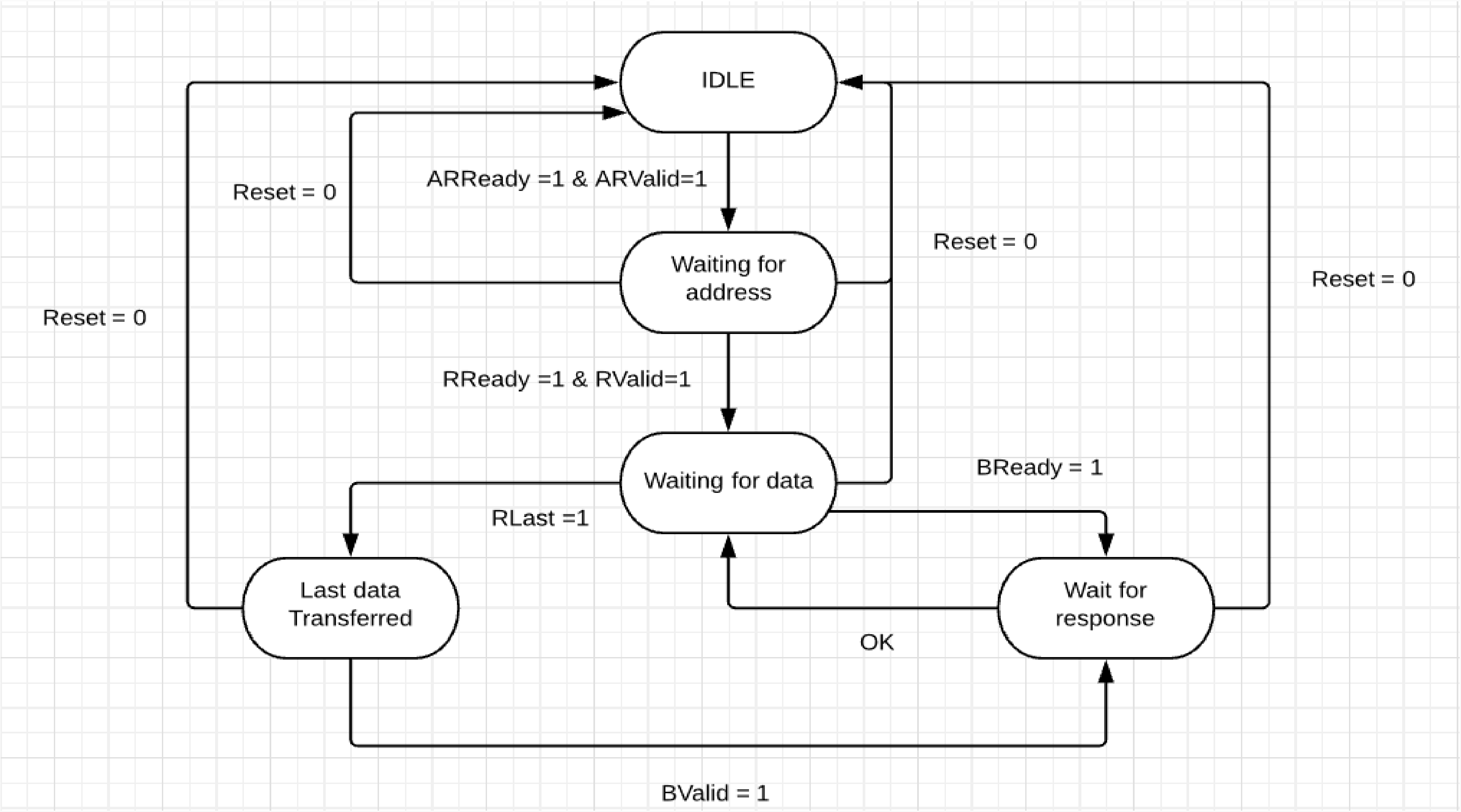
Resulting reg value

0xDE	0xAD	0xBE	0xEF

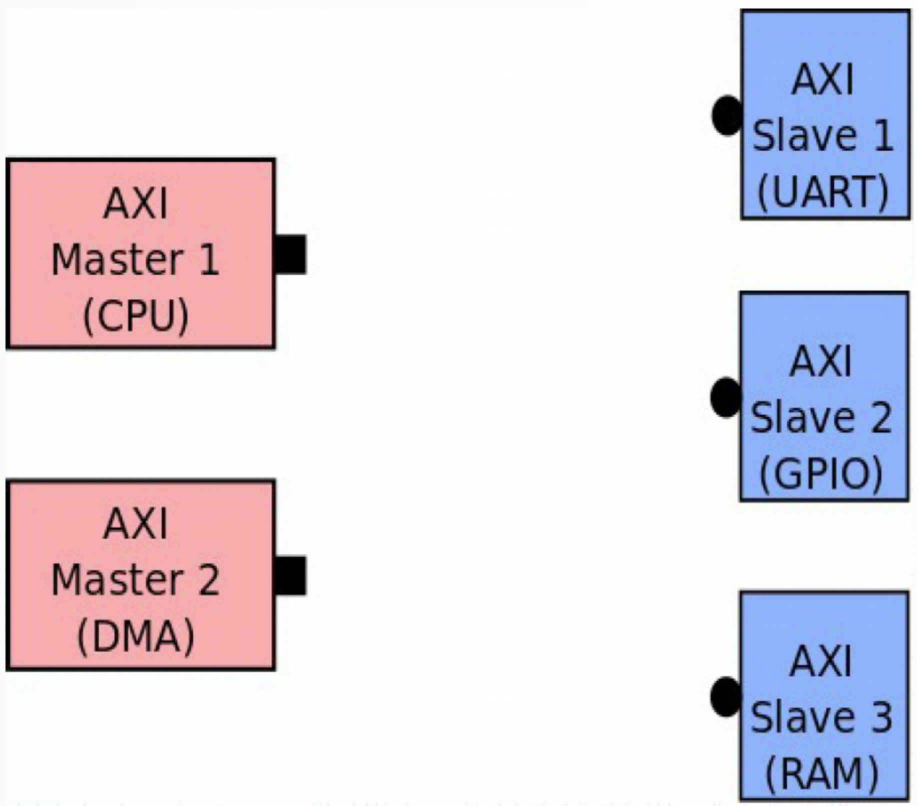
AXI Lite read operation

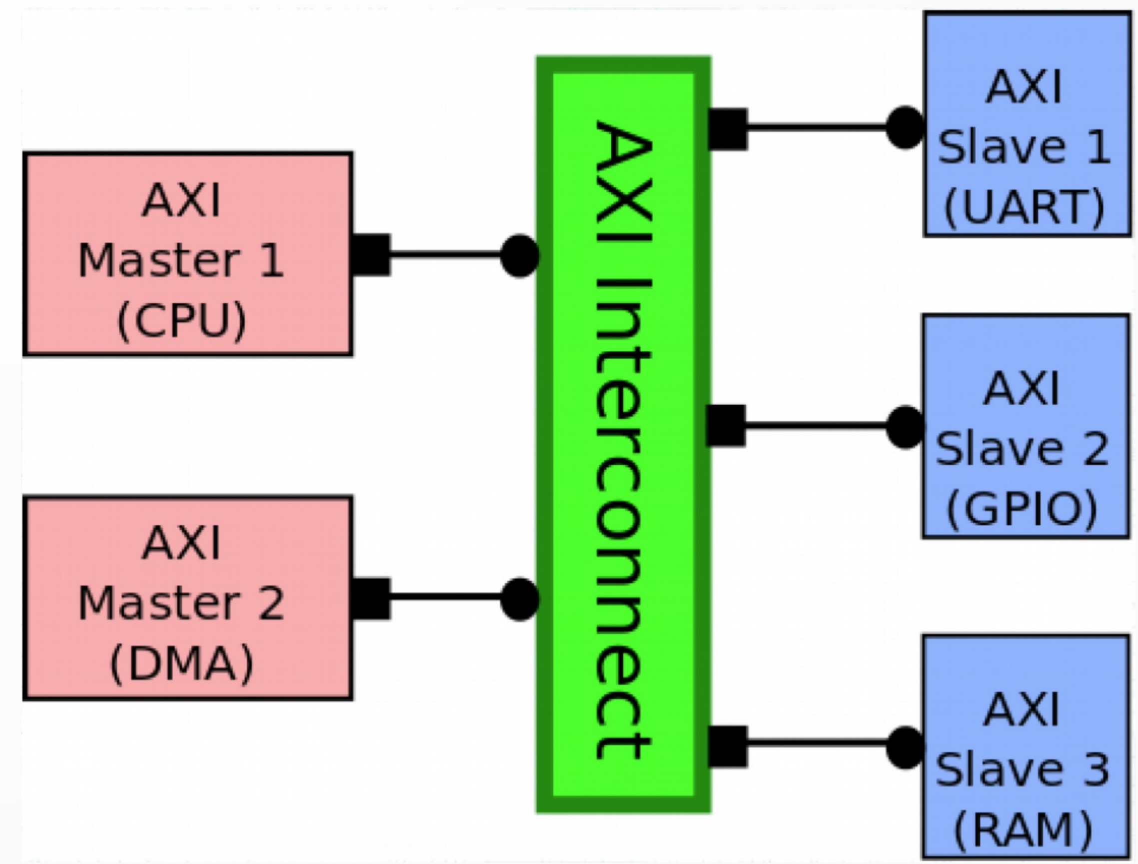
- ADDR handshake
- Read adresse
- DATA handshake
- Read Data




IDLE
ARReady =1 & ARValid=1
Reset = 0
Waiting for address
Reset = 0
Reset = 0
Reset = 0
RReady =1 & RValid=1
Waiting for data
BReady = 1
RLast =1
Last data Transferred
Wait for response
OK
BValid = 1

Connecting Masters and Slaves





Interconnect vs. Interface

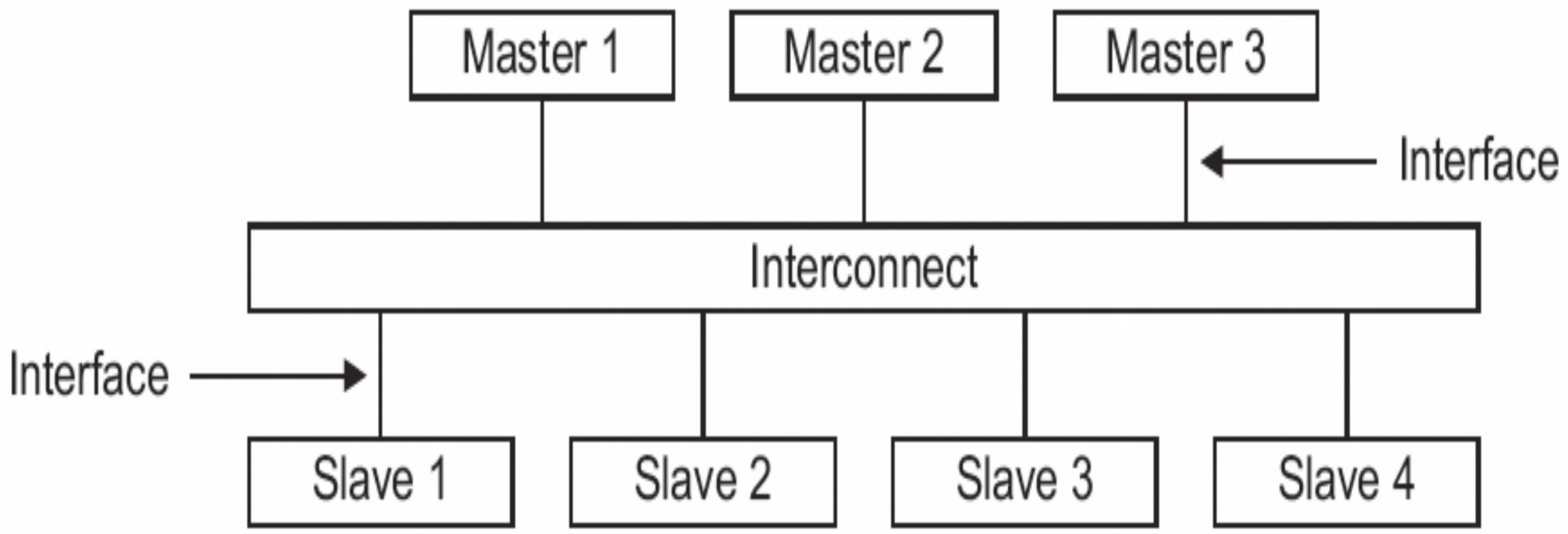
Interface

A point-to-point connection for passing data, addresses,
and hand-shaking signals between master and slave clients within the system

Interconnect

A switch which manages and directs traffic between attached AXI interfaces

Interconnect vs. Interface



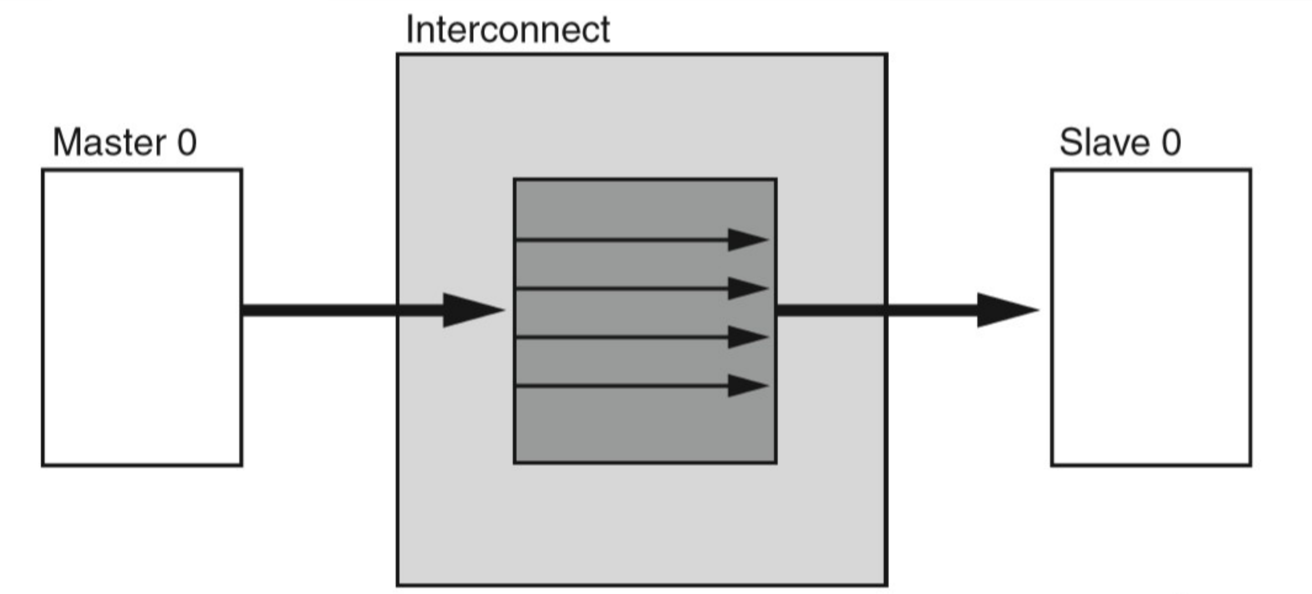
Use Models

The AXI Interconnect core connects one or more AXI memory-mapped master devices to one or more memory-mapped slave devices.

- Pass Through
- Conversion Only
- N-to-1 Interconnect
- 1-to-N Interconnect
- N-to-M Interconnect (Crossbar Mode)

Pass Through

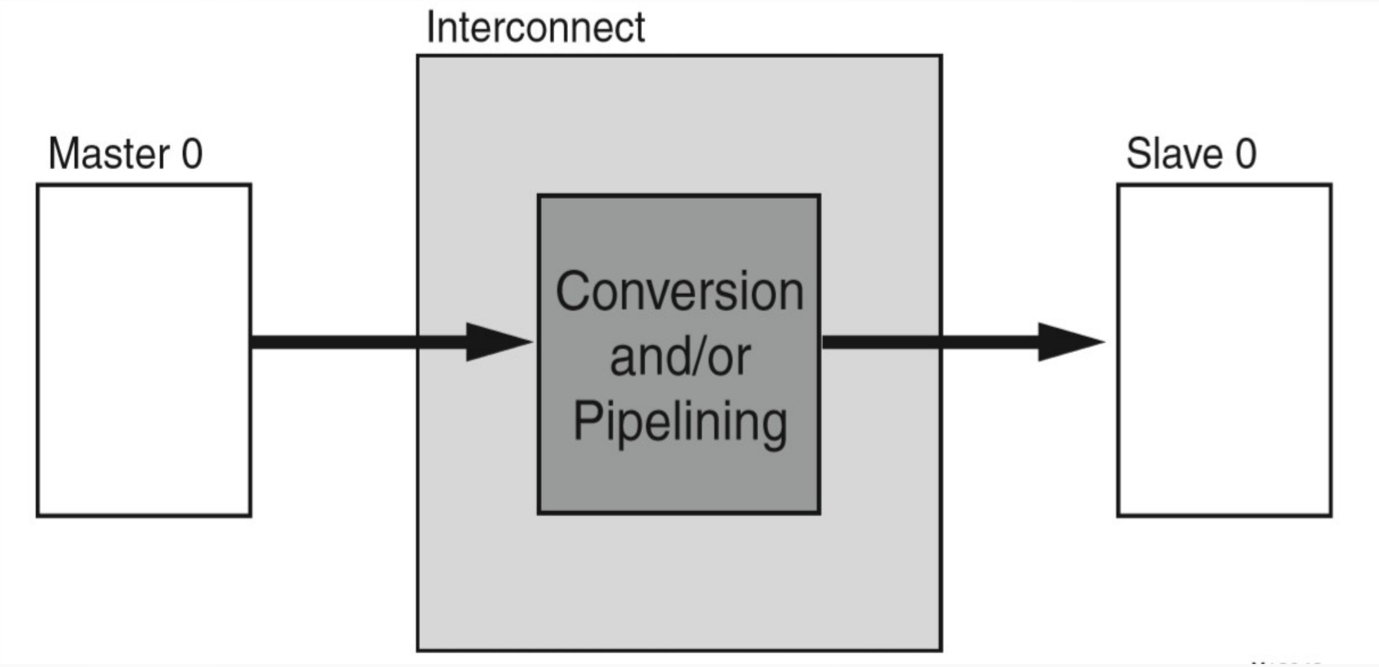
- When there is only one master device and only one slave device connected to the AXI Interconnect core, and the AXI Interconnect core is not performing any optional conversion functions or pipelining, all pathways between the slave and master interfaces degenerate into direct wire connections with no latency and consuming no logic resources.



Conversion Only

The AXI Interconnect core can perform various conversion and pipelining functions when connecting one master device to one slave device. These conversion and pipelining functions are:

- Data width conversion
- Clock rate conversion
- AXI4-Lite slave adaptation



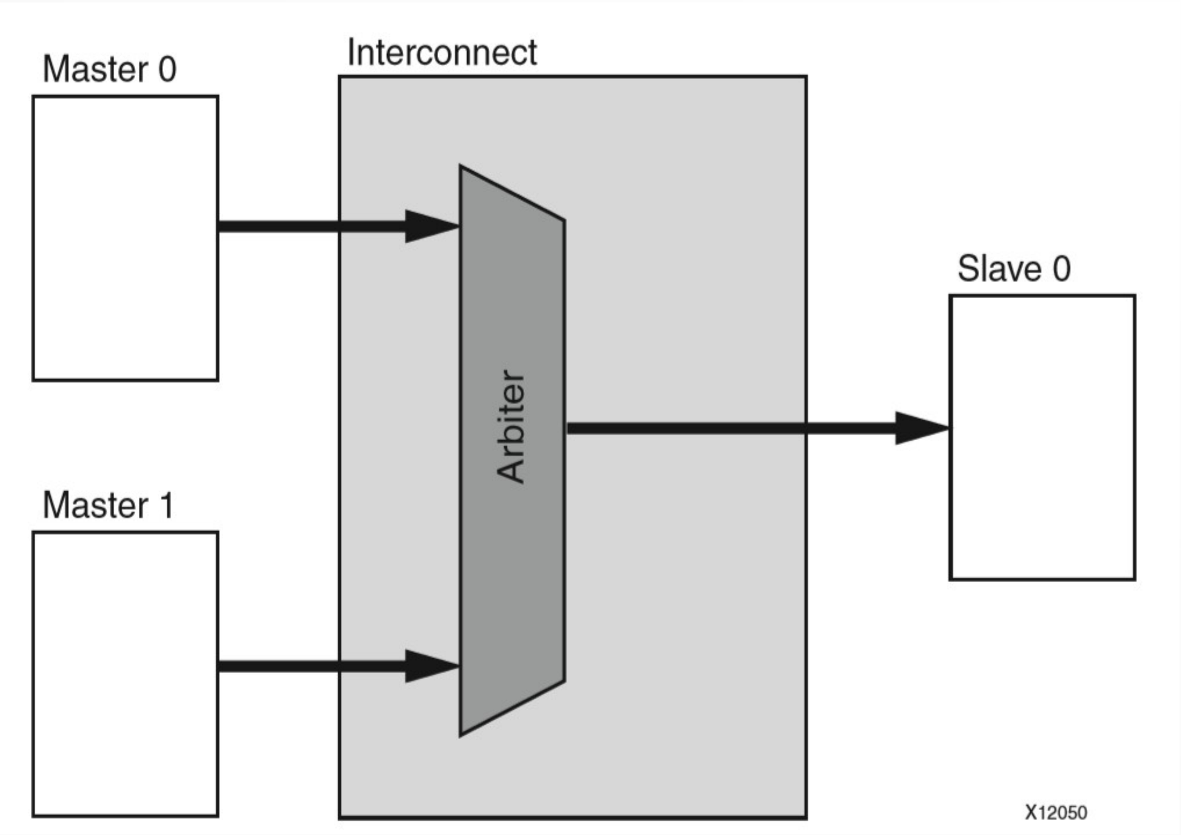
N-to-1 and 1-to-N Interconnect

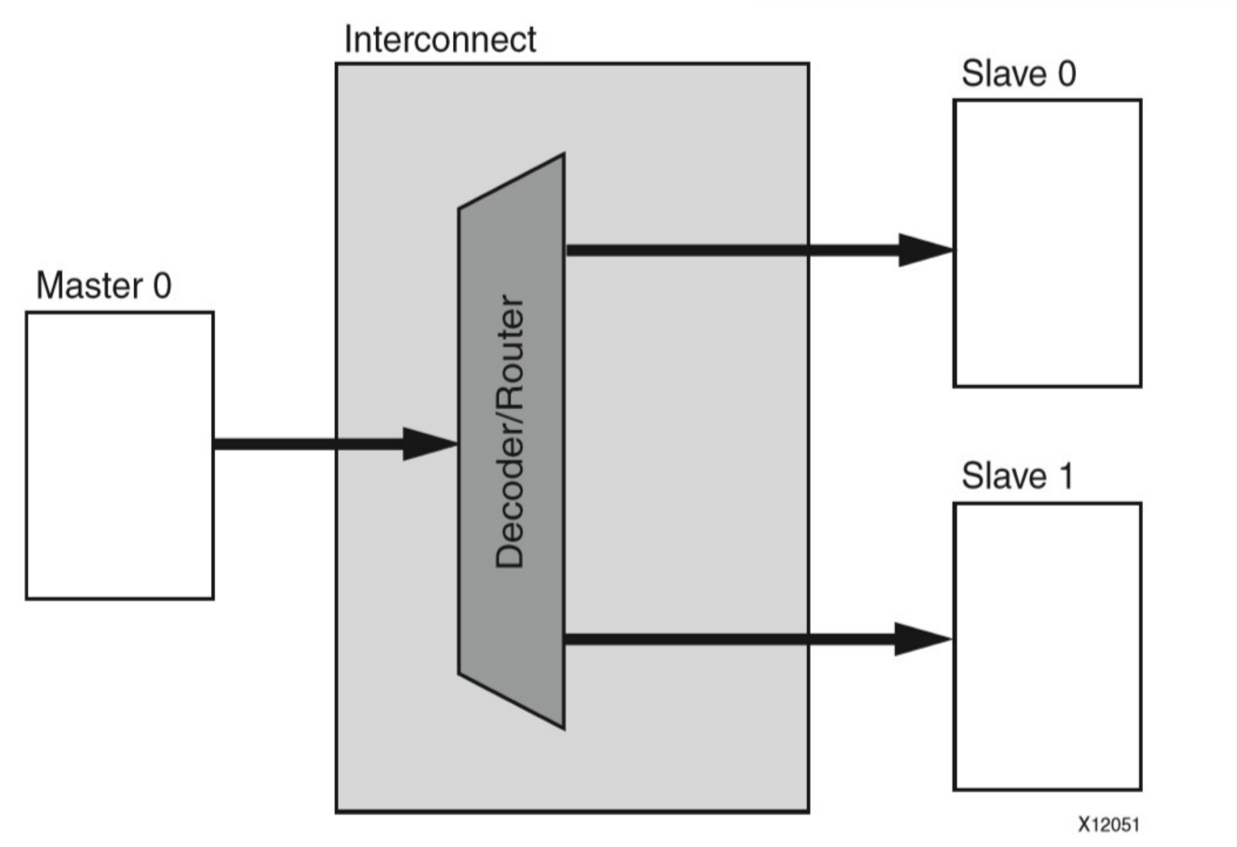
Note :

A bus arbiter is a device used in a multi-master bus system to decide which bus master will be allowed to control the bus for each bus cycle. The most common kind of bus arbiter is the memory arbiter in a system bus system.

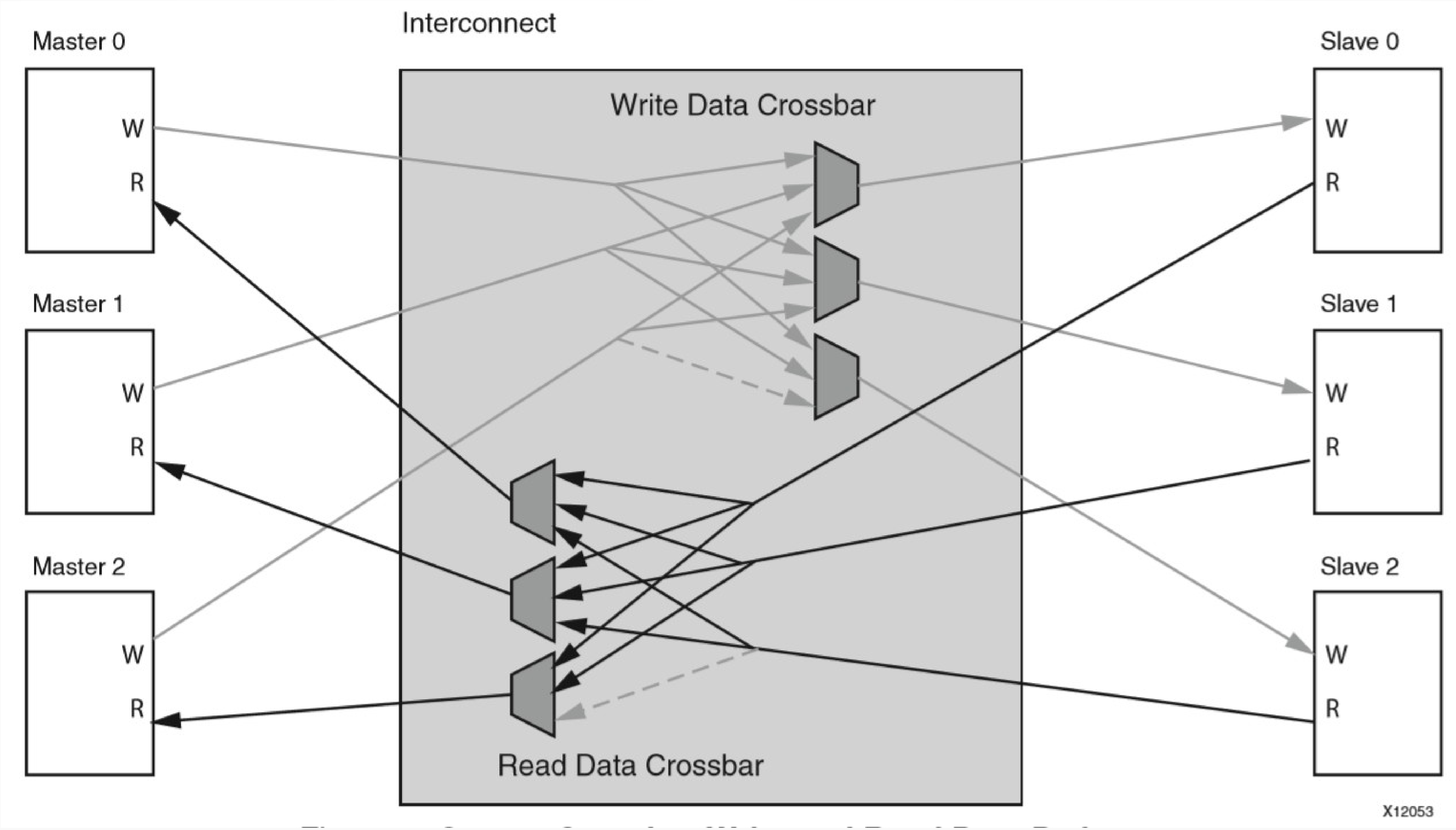
- A memory arbiter is a device used in a shared memory system to decide, for each memory cycle, which CPU will be allowed to access that shared memory.

N-to-1 and 1-to-N Interconnect





N-to-M Interconnect (Crossbar Mode)

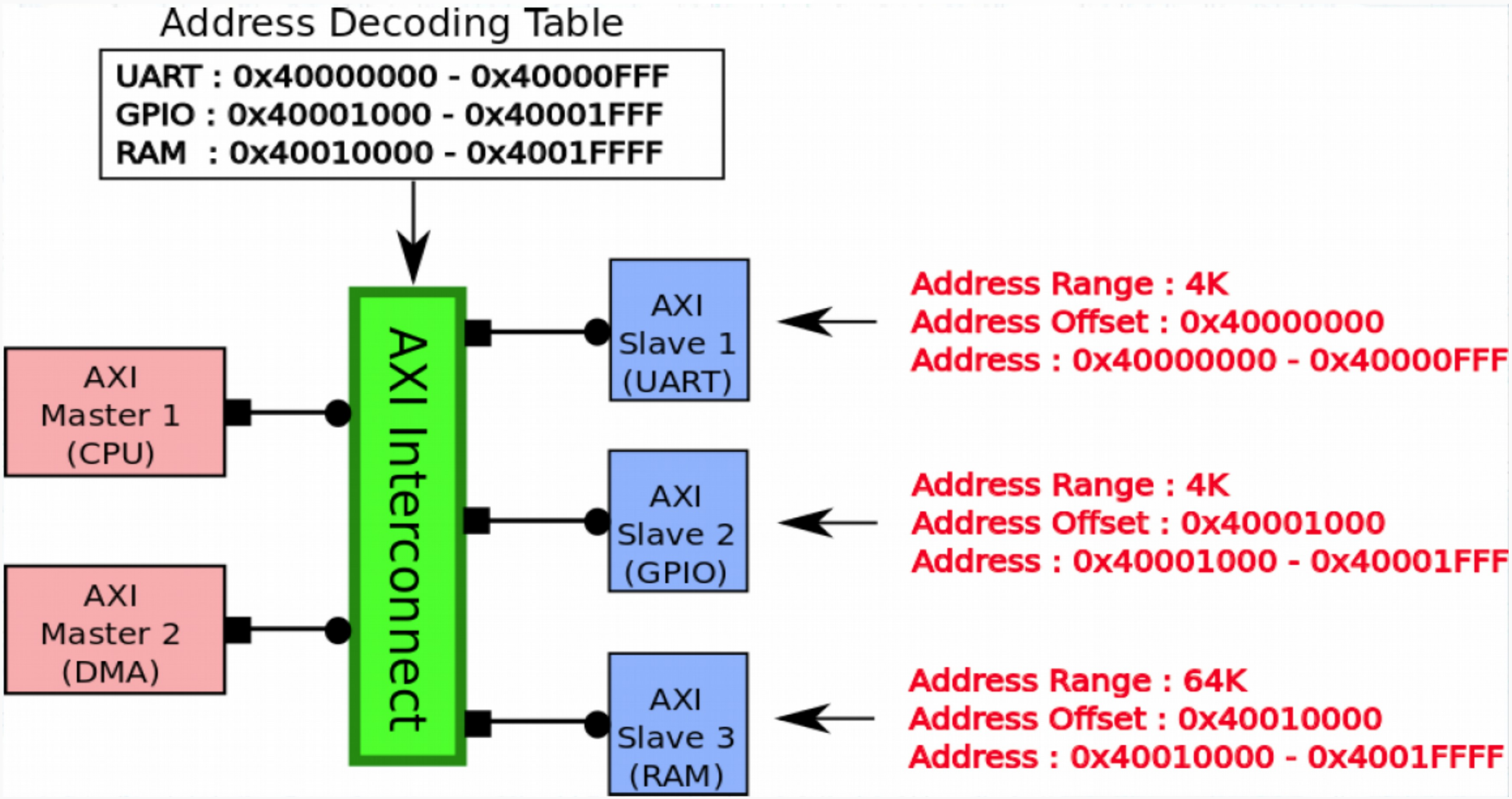


AXI interconnect

- ID Reflection mechanism :

  additional ID bits would not be seen by the AXI master. So these "routing" bits are generated by the AXI interconnect, not the AXI master.

AXI Address mapping



AXI 4 and AXI Lite

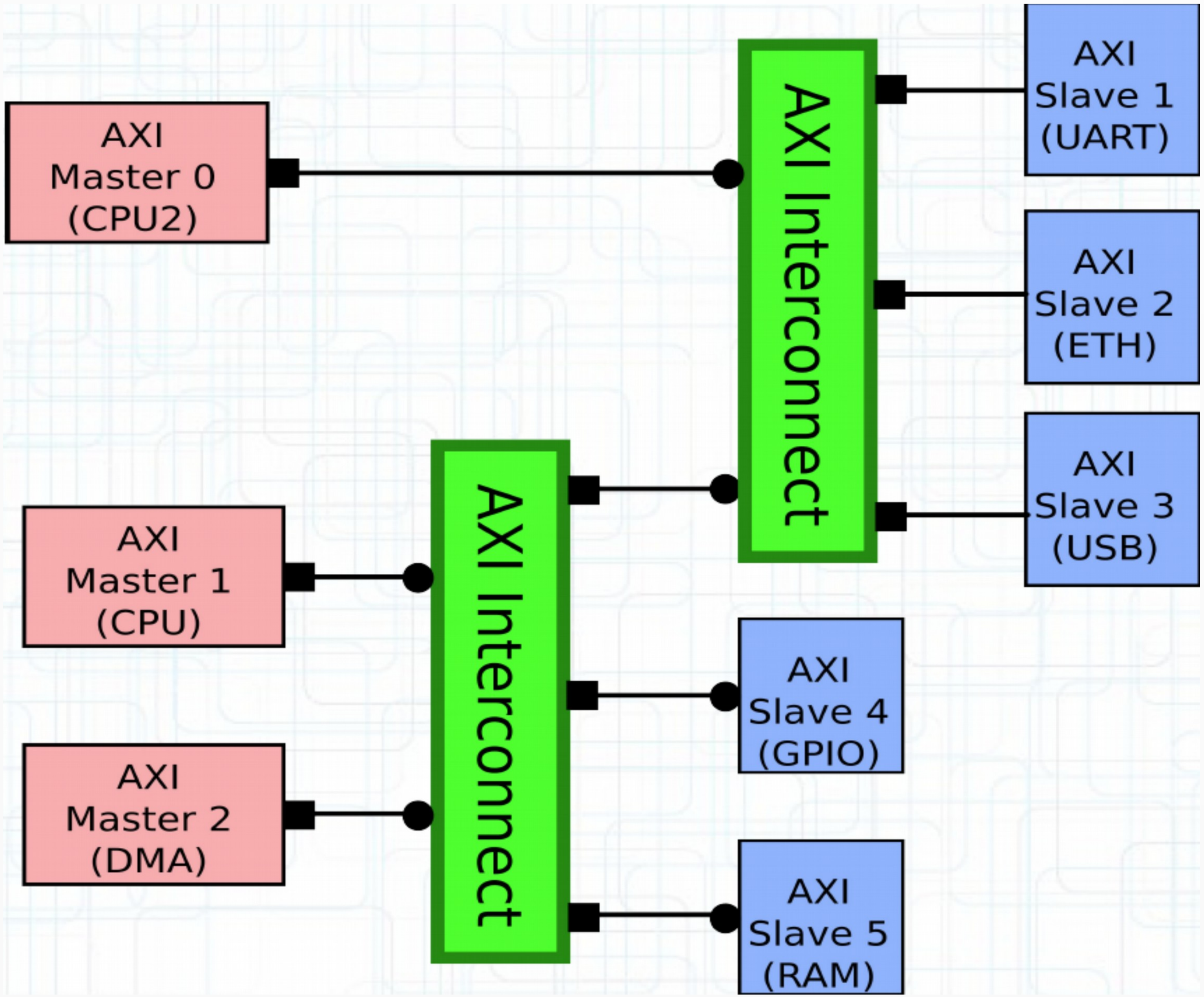
<table>
<tr><th></th><th>AXI4</th><th>AXI4-Lite</th></tr>
<tr><td rowspan="2">Glbl</td><td colspan="2">ACLK</td></tr>
<tr><td colspan="2">ARESETN</td></tr>
<tr><td rowspan="15">Write Address</td><td>AWID</td><td></td></tr>
<tr><td colspan="2">AWADDR</td></tr>
<tr><td>AWLEN</td><td></td></tr>
<tr><td>AWSIZE</td><td></td></tr>
<tr><td>AWBURST</td><td></td></tr>
<tr><td>AWLOCK</td><td></td></tr>
<tr><td colspan="2">AWCACHE</td></tr>
<tr><td>AWPROT</td><td></td></tr>
<tr><td>AWQOS</td><td></td></tr>
<tr><td>AWSIZE</td><td></td></tr>
<tr><td>AWREGION</td><td></td></tr>
<tr><td>AWLOCK</td><td></td></tr>
<tr><td>AWUSER</td><td></td></tr>
<tr><td colspan="2">AWVALID</td></tr>
<tr><td colspan="2">AWREADY</td></tr>
</table>

<table>
<tr><th></th><th>AXI4</th><th>AXI4-Lite</th></tr>
<tr><td rowspan="6">Write Data</td><td>WDATA</td><td>WDATA</td></tr>
<tr><td>WSTRB</td><td>WSTRB</td></tr>
<tr><td>WLAST</td><td></td></tr>
<tr><td>WUSER</td><td></td></tr>
<tr><td colspan="2">WVALID</td></tr>
<tr><td colspan="2">WREADY</td></tr>
<tr><td rowspan="5">Write Resp.</td><td>BID</td><td></td></tr>
<tr><td>BRESP</td><td>BRESP</td></tr>
<tr><td>BUSER</td><td></td></tr>
<tr><td colspan="2">BVALID</td></tr>
<tr><td colspan="2">BREADY</td></tr>
</table>

<table>
<tr><th></th><th>AXI4</th><th>AXI4-Lite</th></tr>
<tr><td rowspan="13">Read Address</td><td>ARID</td><td></td></tr>
<tr><td colspan="2">ARADDR</td></tr>
<tr><td>ARLEN</td><td></td></tr>
<tr><td>ARSIZE</td><td></td></tr>
<tr><td>ARBURST</td><td></td></tr>
<tr><td>ARLOCK</td><td></td></tr>
<tr><td>ARCACHE</td><td>ARCACHE</td></tr>
<tr><td>ARPROT</td><td>ARPROT</td></tr>
<tr><td>ARQOS</td><td></td></tr>
<tr><td>ARREGION</td><td></td></tr>
<tr><td>ARUSER</td><td></td></tr>
<tr><td colspan="2">ARVALID</td></tr>
<tr><td colspan="2">ARREADY</td></tr>
</table>

<table>
<tr><th></th><th>AXI4</th><th>AXI4-Lite</th></tr>
<tr><td rowspan="7">Read Data</td><td>RID</td><td></td></tr>
<tr><td>RDATA</td><td>RDATA</td></tr>
<tr><td>RRESP</td><td>RRESP</td></tr>
<tr><td>RLAST</td><td></td></tr>
<tr><td>RUSER</td><td></td></tr>
<tr><td colspan="2">RVALID</td></tr>
<tr><td colspan="2">WREADY</td></tr>
</table>

AXI interconnect



AXBURST signal

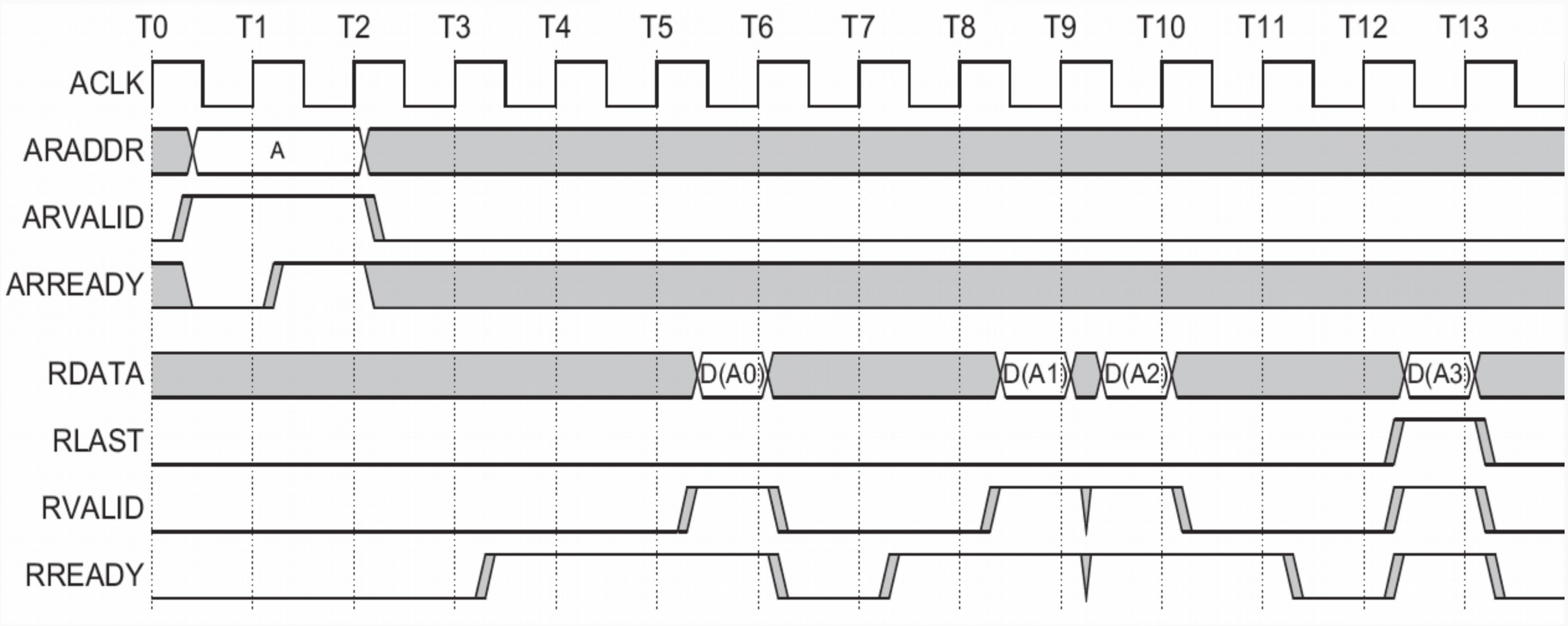
ARBURST[1:0] AWBURST[1:0]	Burst type	Description	Access
b00	FIXED	Fixed-address burst	FIFO-type
b01	INCR	Incrementing-address burst	Normal sequential memory
b10	WRAP	Incrementing-address burst that wraps to a lower address at the wrap boundary	Cache line
b11	Reserved	-	-

WRAP Burst mode

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

Read Burst



T0
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
T13
ACLK
ARADDR
A
ARVALID
ARREADY
RDATA
D(A0)
D(A1)
D(A2)
D(A3)
RLAST
RVALID
RREADY

THANK YOU:)