AMBA 3.0 AXI/APB3 Bus Protocol

2010-08-18

Kim, Si-Hyun

Design Technology (DT) Part
DIS Group
System IC Business Team
LG Electronics

Please send comments via email to: *sihyun1.kim@lge.com*

This documentation contain confidential and proprietary information that is the property of LG.

LGE CONFIDENTIAL

SYSTEM IC BUSINESS TEAM

Agenda

- Introduction to the AMBA Buses
- AXI protocol
  - Ch. 1 Introduction
  - Ch. 2 Signal Descriptions
  - Ch. 3 Channel Handshake
  - Ch. 4 Addressing Options
  - Ch. 5 Additional Control Information
  - Ch. 6 Atomic Accesses
  - Ch. 7 Response Signaling
  - Ch. 8 Ordering Model
  - Ch. 9 Data Buses
  - Ch. 10 Unaligned Transfers
  - Ch. 11 Clock and Reset
  - Ch. 12 Low-power Interface
- APB3 Protocol
- Appendix, Reference

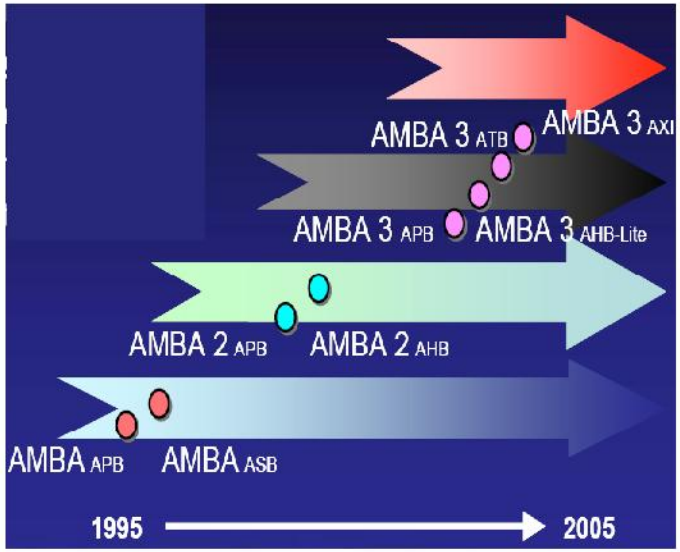
Introduction to the AMBA Buses

❑ Advanced Microprocessor Bus Architecture (AMBA)

- Open Standard Bus
- On-chip bus proposed by ARM

- High bandwidth bus
  - ✓ AHB (Advanced High-performance Bus)
    - – AHB full
    - – AHB-Lite (no req/grant, no split/retry)
  - ✓ AXI (Advanced eXtensible Interface)

- Low power bus
  - ✓ APB (Advanced Peripheral Bus)
    - – Low speed Peri. IPs
  - ✓ APB3 (AXI slave register configuration)



❑ Next generation high performance bus

Agenda

- **AXI protocol**

  - **Ch. 1 Introduction**

    - ✓ About the AXI protocol
    - ✓ Architecture
    - ✓ Basic transactions
    - ✓ Additional features

About the AXI protocol

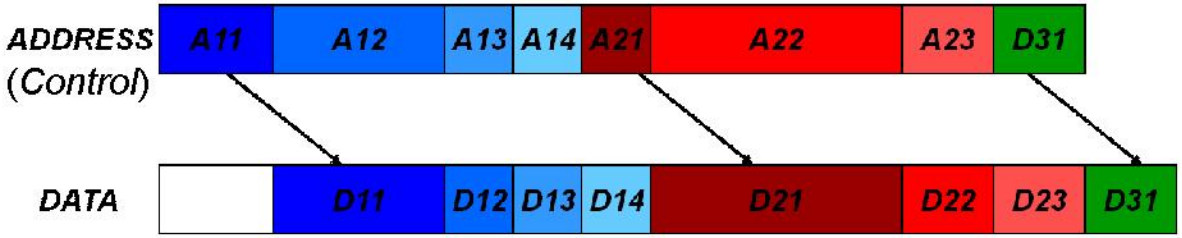
- 3rd Generation Bus architecture from ARM
- Advanced eXtensible Interface
- Feature Comparison: AHB VS. AXI

AMBA 2 AHB Protocol	AMBA 3 AXI Protocol
Separate address for every data item	Burst based - one address per burst (#1)
Fixed pipeline for address and data transfers	Five independent channels for address/data and response (#5)
	Out of order data (#3) and data interleaving (#4)
Only one transaction at a time	Multiple outstanding transactions (#2)
	Simultaneous reads and writes (#6)
Does not natively support the ARM v6 architecture	Native support for unaligned and exclusive accesses

Efficiency Comparison: AHB VS. AXI

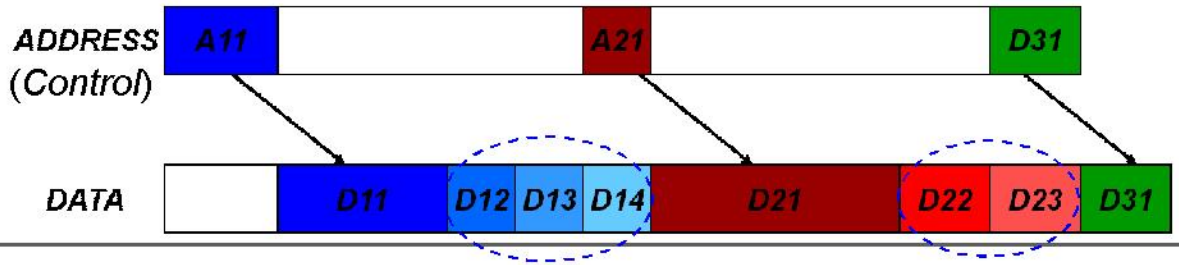
❑ AHB Burst

- Shared Address bus and Multiple Data buses (SAMD)
- Address and Data are locked together (two pipeline stages)
- HREADY controls pipeline operation



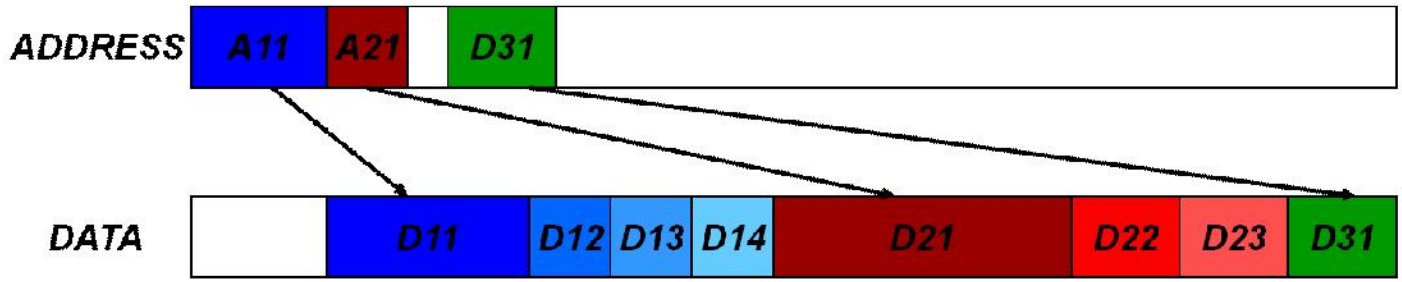
❑ AXI: #1. One Address for entire burst

- Multiple Address buses and Multiple Data buses (MAMD)
- Master provides the start address of burst
  - ✓ slave has to calculate the other addr. for a burst of transfers. (FIXED/INCR/WRAP)



AXI: #2. multiple outstanding addresses

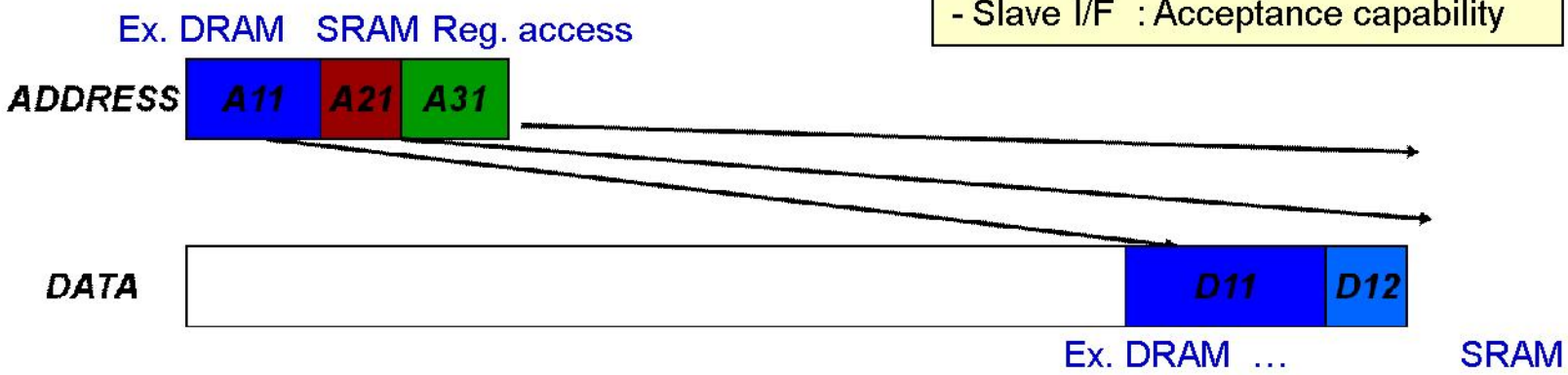
- **Masters can issue transaction addresses without waiting for earlier transactions to complete.**



- **Problem: Slow slave**
  - If one slave is very slow, all data is held up.

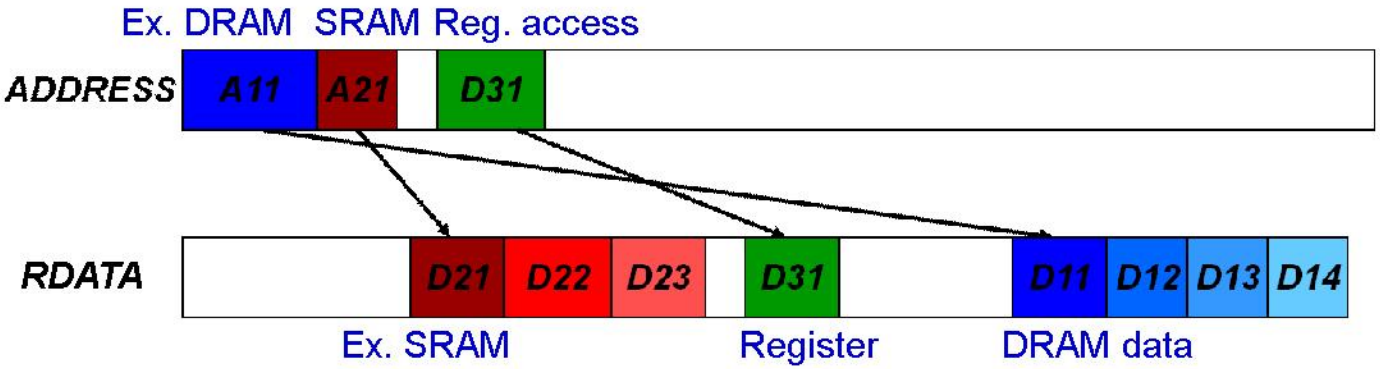
Parameters for multiple outstanding requests

- Master I/F: Issuing capability
- Slave I/F : Acceptance capability



AXI: #3. Out of order transaction

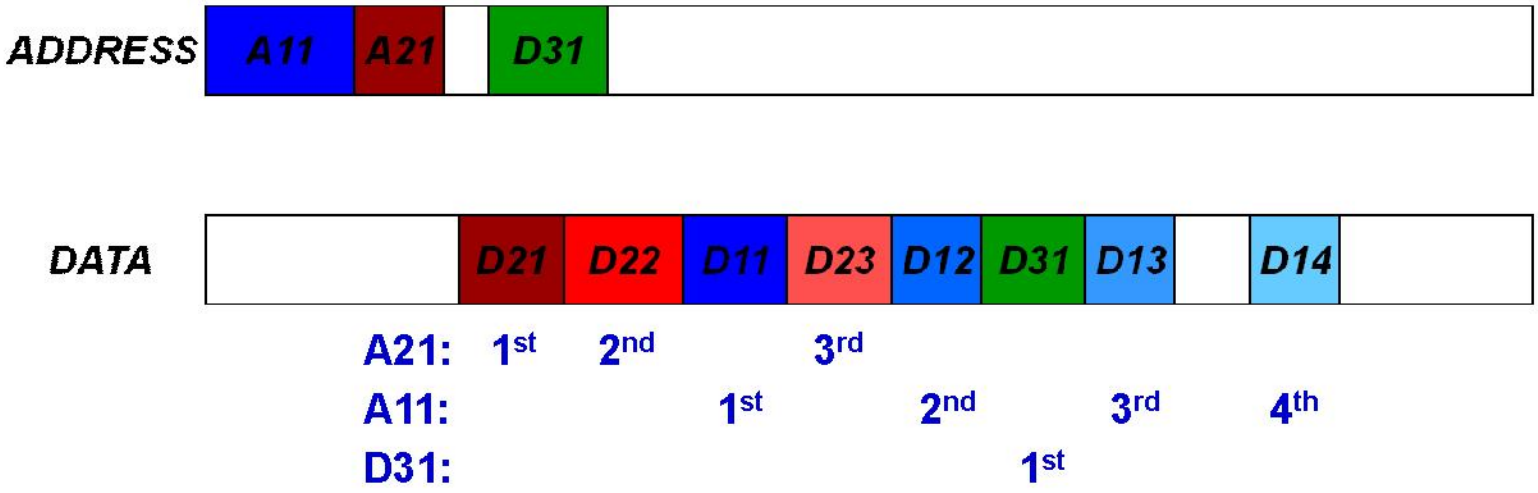
- **Out of order completion allowed fast slaves may return data ahead of slow slaves**



- **Problem: Address-data transaction level ordering**
  - Each Channels has an ID attached : *ARID, AWID, RID, WID, & BID (Up to four bits)*
  - Ordering by transaction ID
    - ✓ Master needs to finish data transfers with the same transaction ID in the order of request issue.
    - ✓ Slave can handle data transfers with different transaction IDs out of order
  - The interconnect in a multi-master system must append another tag to ID to make each master's ID unique

AXI: #4. Data Interleaving

- Returned data can even be interleaved
- Gives maximum use of data bus



Note: Data within a burst is always in order

Channel definition

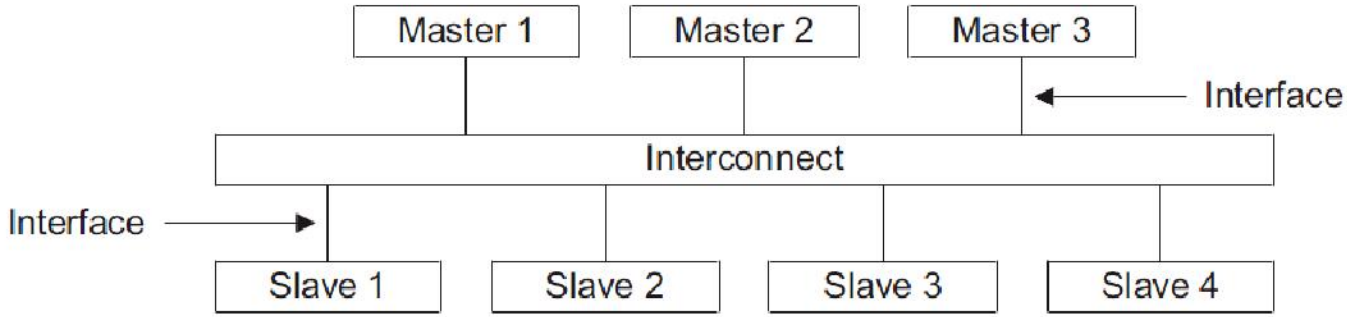
- **Each of the 5 independent channels consists of a set of information (addr, data, control, handshake, response) signals.**

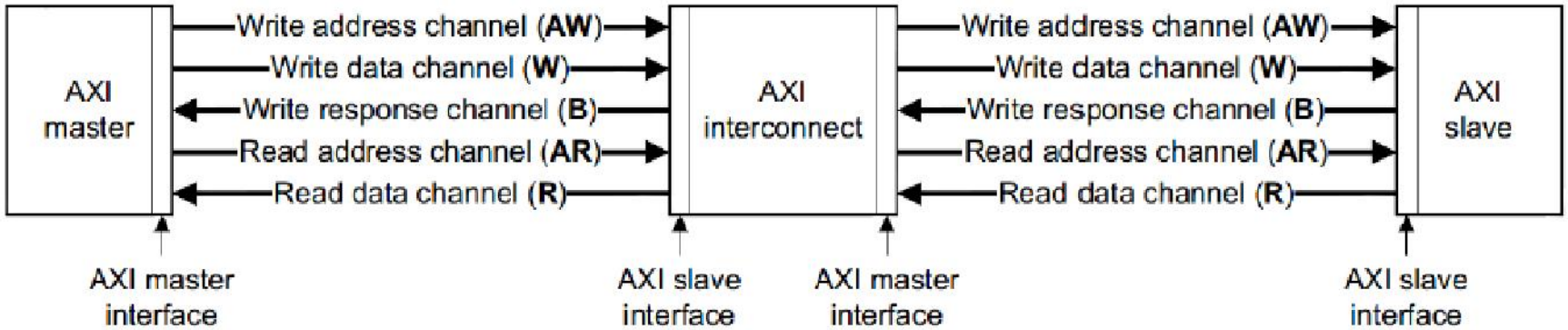
- **Uses a two-way VALID and READY handshake mechanism. (Both the read data operation and the write data operation)**
  - The information source
    - ✓ **VALID** signal to show when valid data or control information is available on the channel.
  - The destination
    - ✓ **READY** signal to show when it can accept the data.
  - Also include a **LAST** signal to indicate when the transfer of the final data item within a transaction takes place.

AXI: #5. Five independent channels

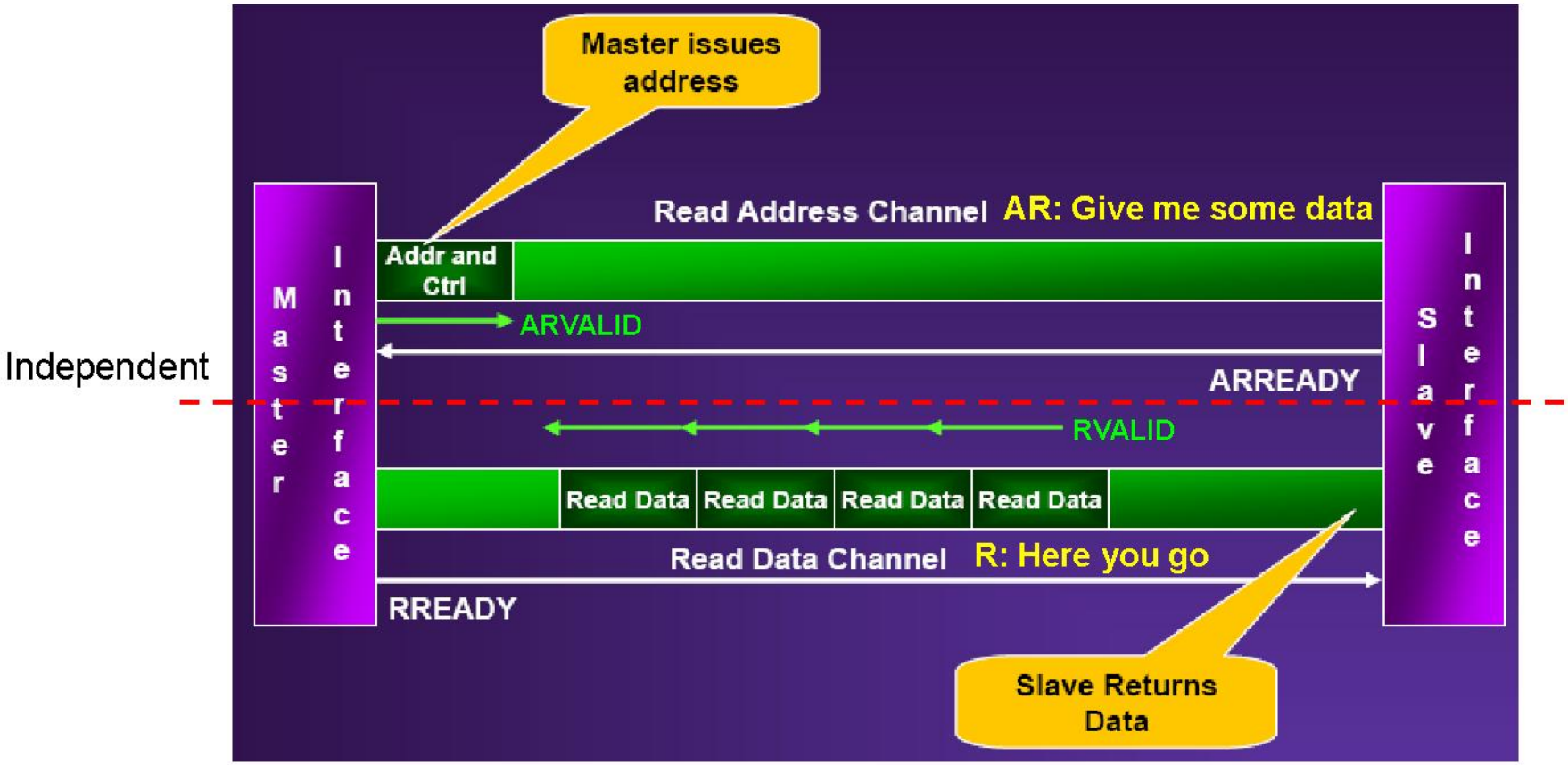
❑ Interface divided into 5 channels

- Parallelization of bus
- Each channel is independent and use two-way flow control (VALID, READY)

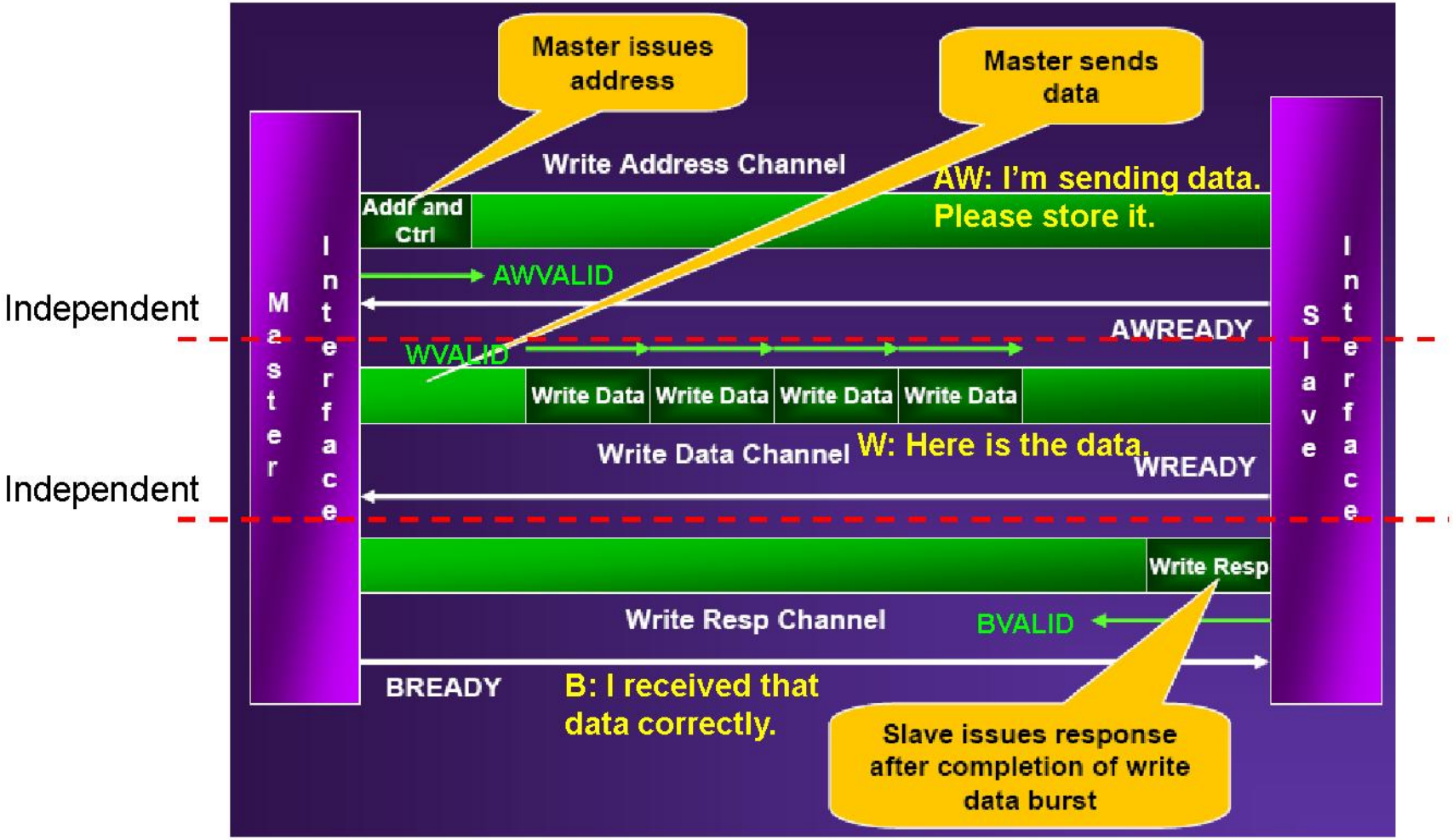




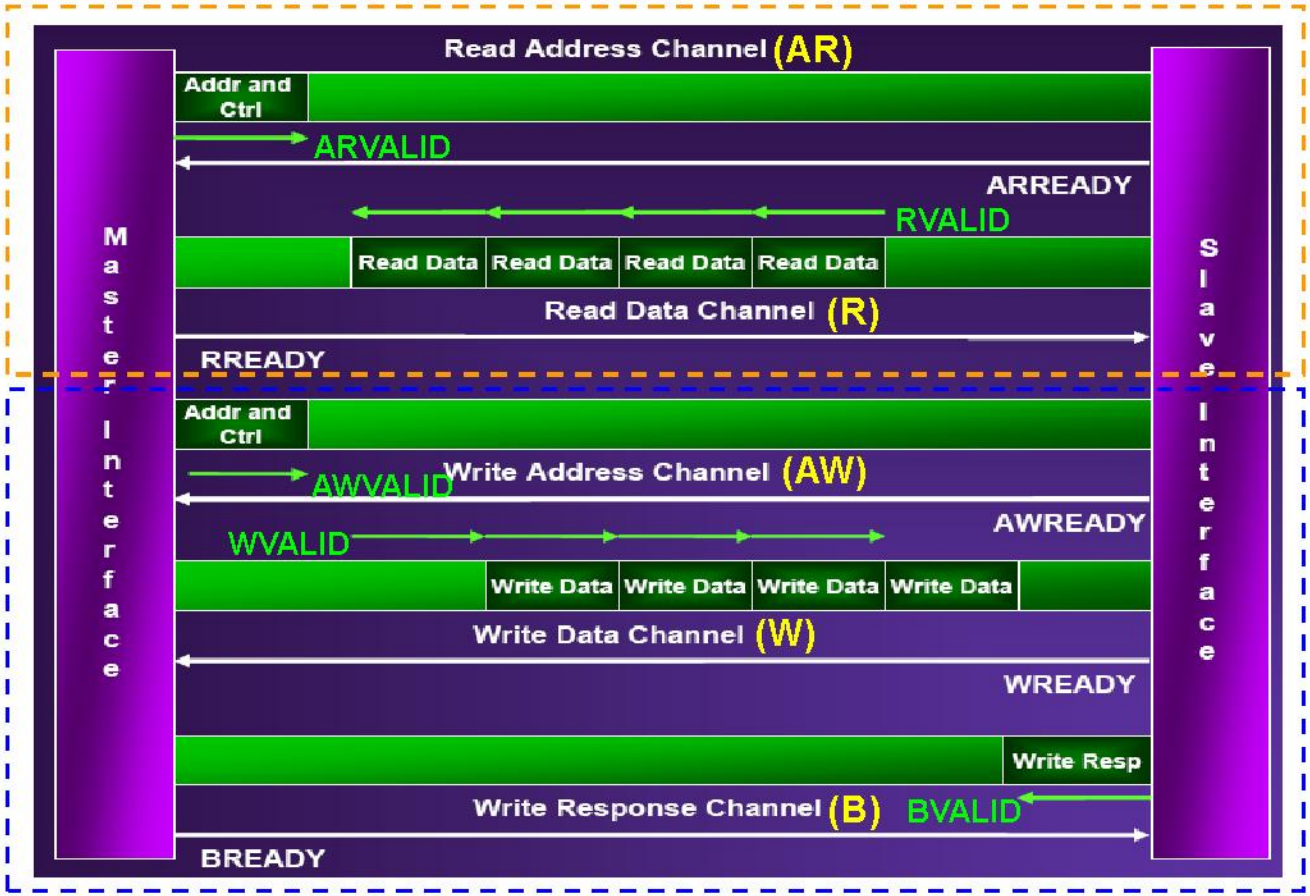
AXI Read Channels

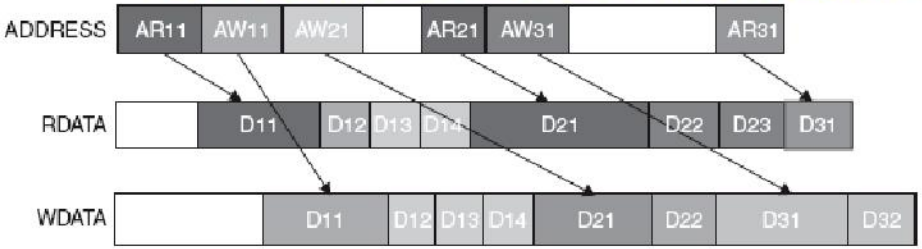


AXI Write Channels



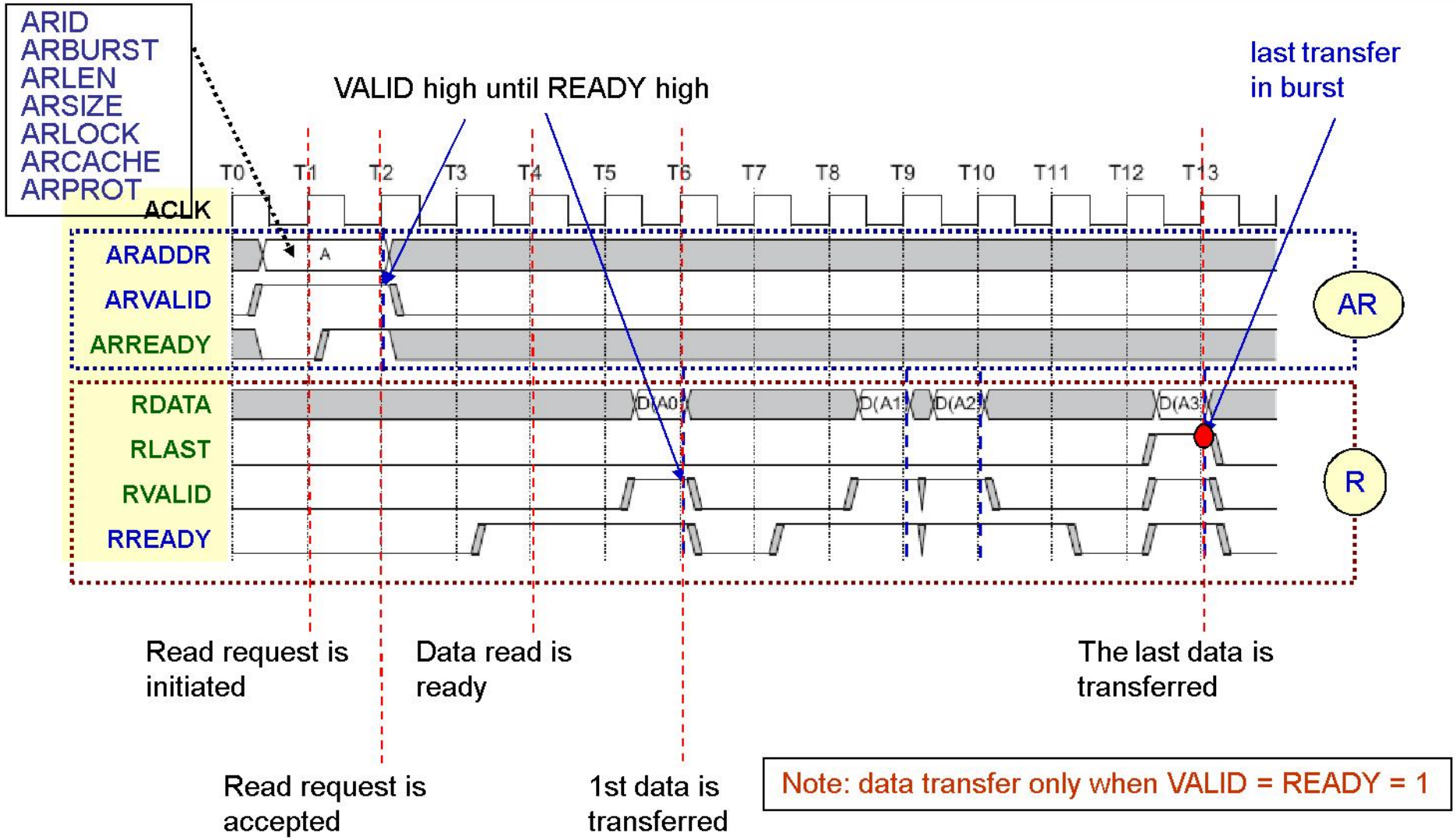
AXI: #6. Simultaneous RDs and WRs





Better bus utilization

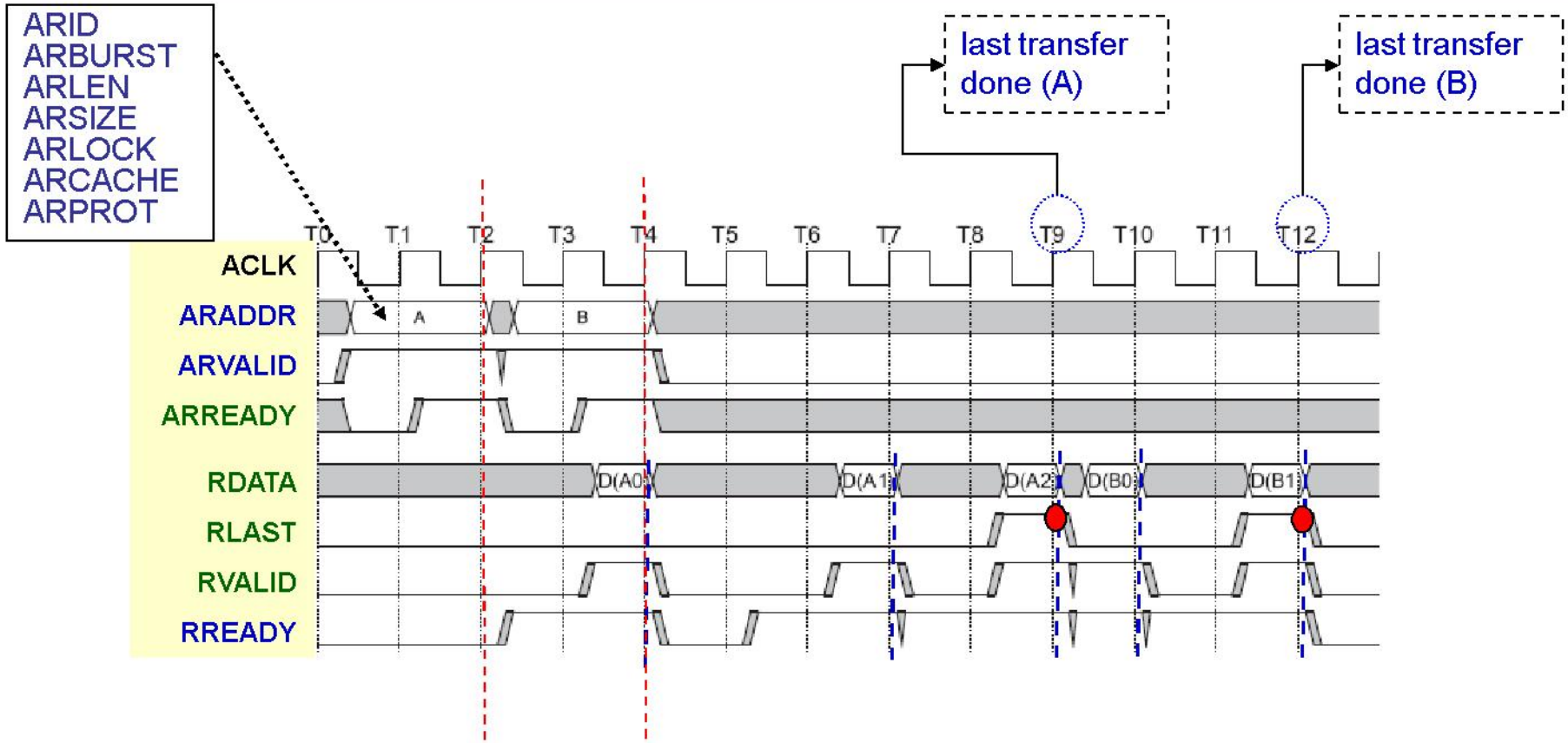
Read burst of 4 transfers



Note: data transfer only when VALID = READY = 1

Source Signal: Master, Slave

Overlapping read burst



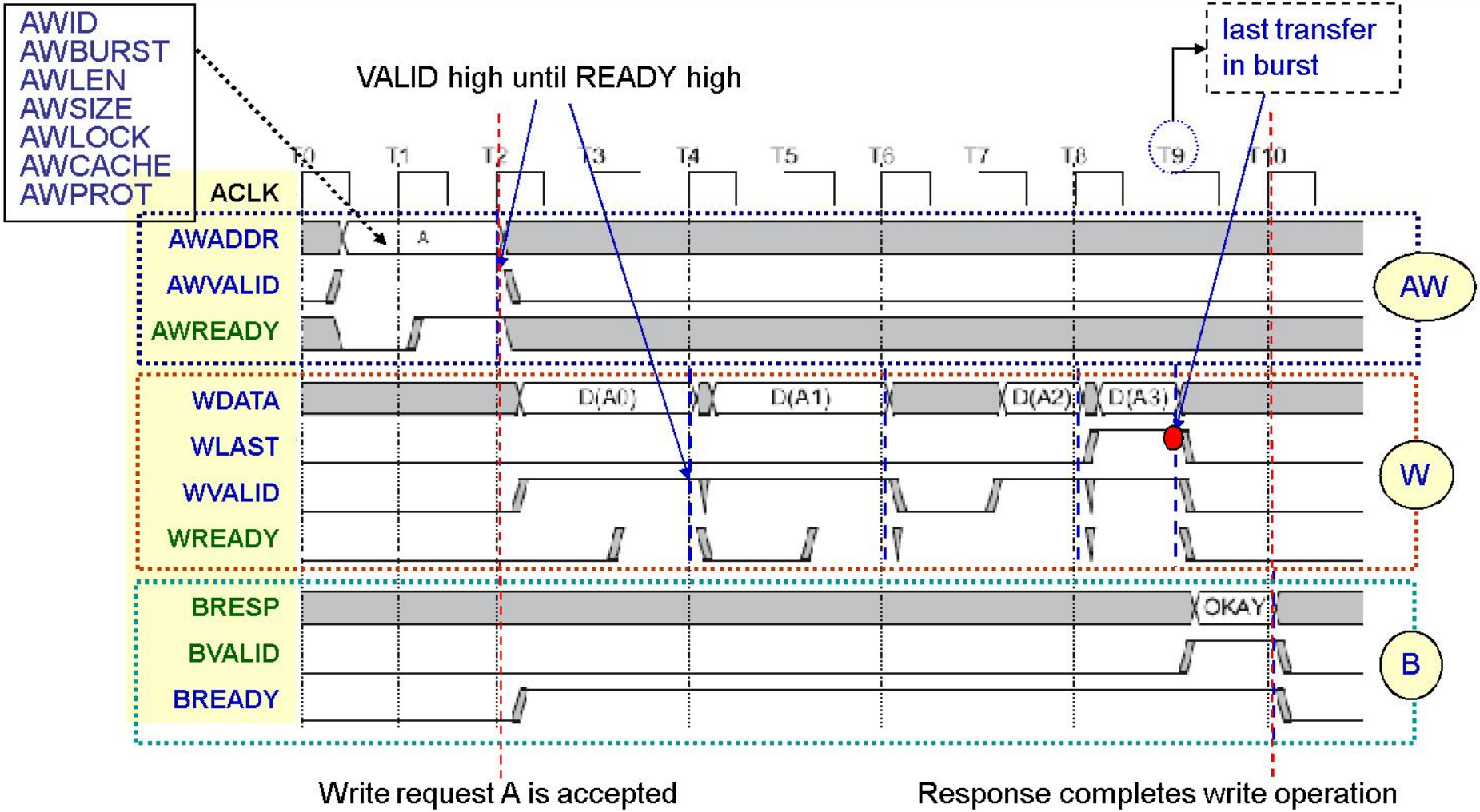
Read request A is accepted

Read request B is accepted via AR channel while data A(0) is transferred via R channel

Ex. Read burst of 3 transfers, 2 transfers

Source Signal: Master, Slave

Write burst of 4 transfers



Source Signal: Master, Slave

Agenda

- **AXI protocol**

  - **Ch. 2 Signal Descriptions**

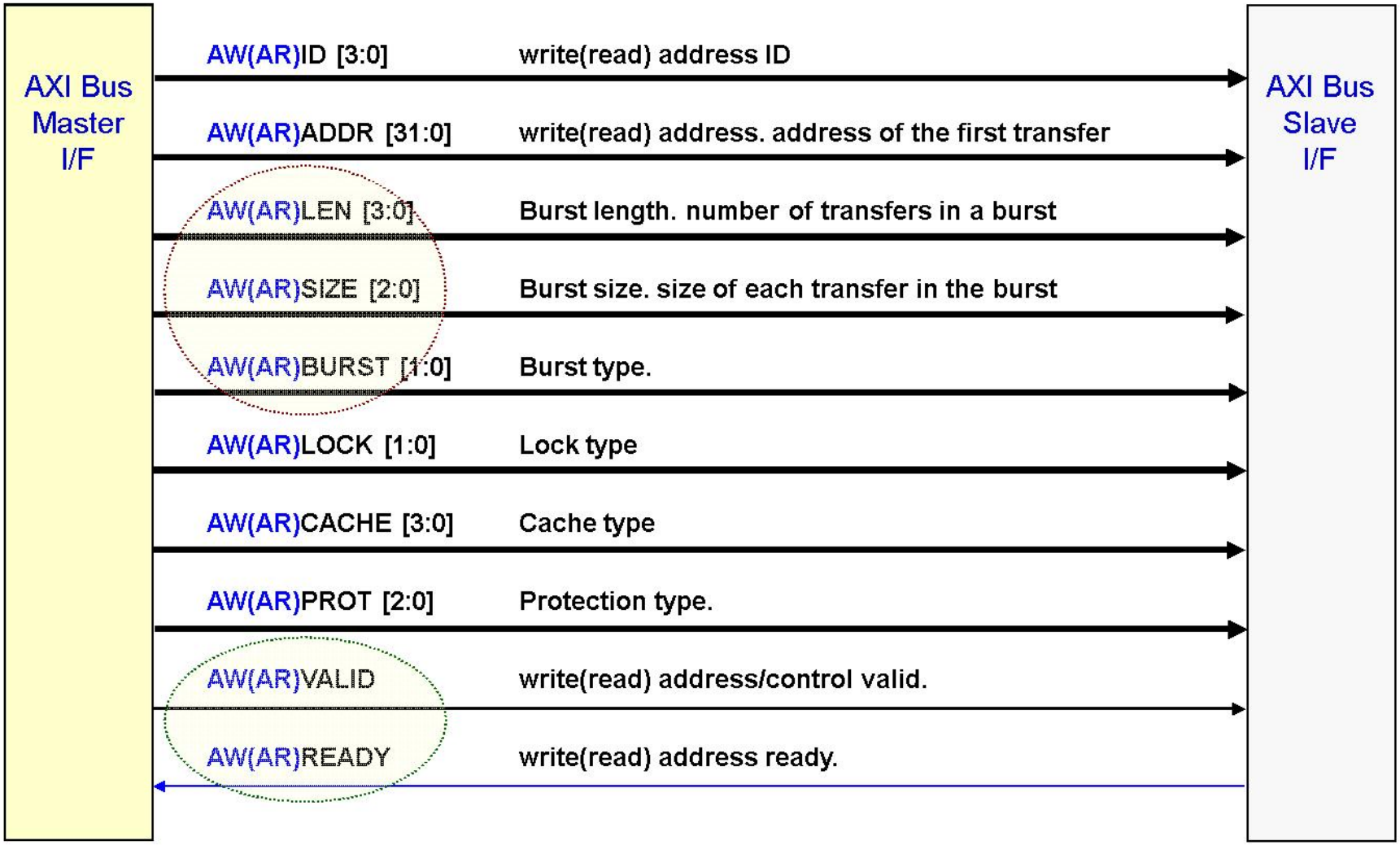
    ✓ Global signals

    – **ACLK** : Global clock signal.
      » All signals are sampled on the rising edge of the global clock.

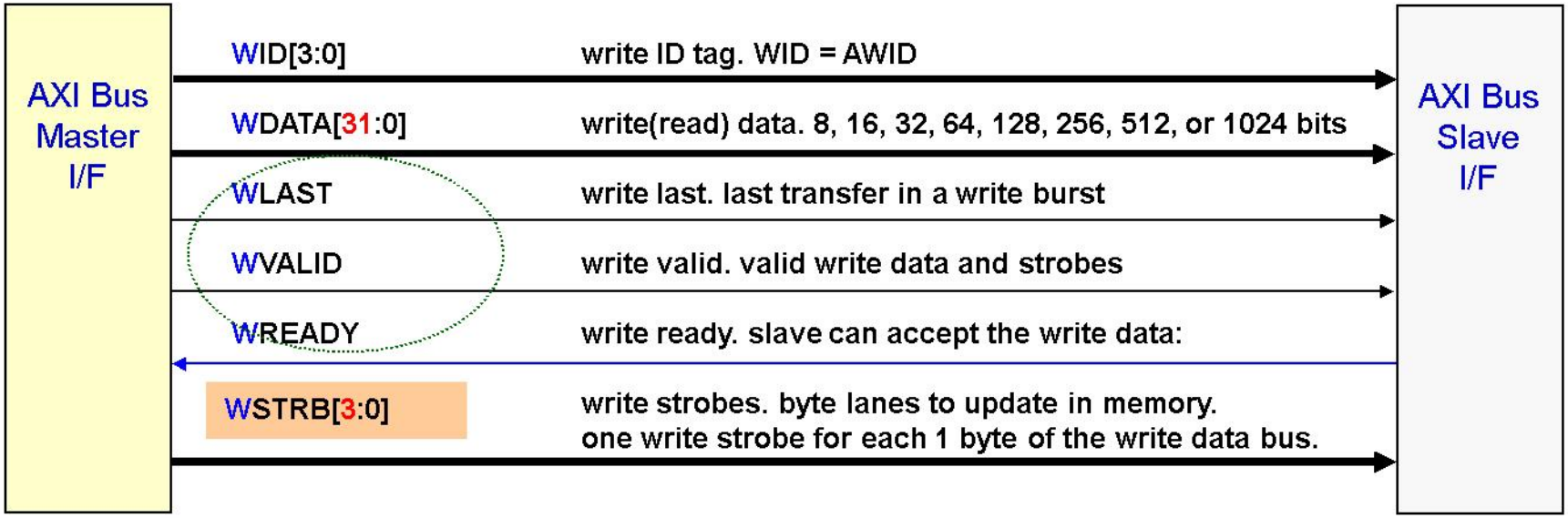
    – **ARESETn** : Global reset signal.
      » This signal is active LOW.

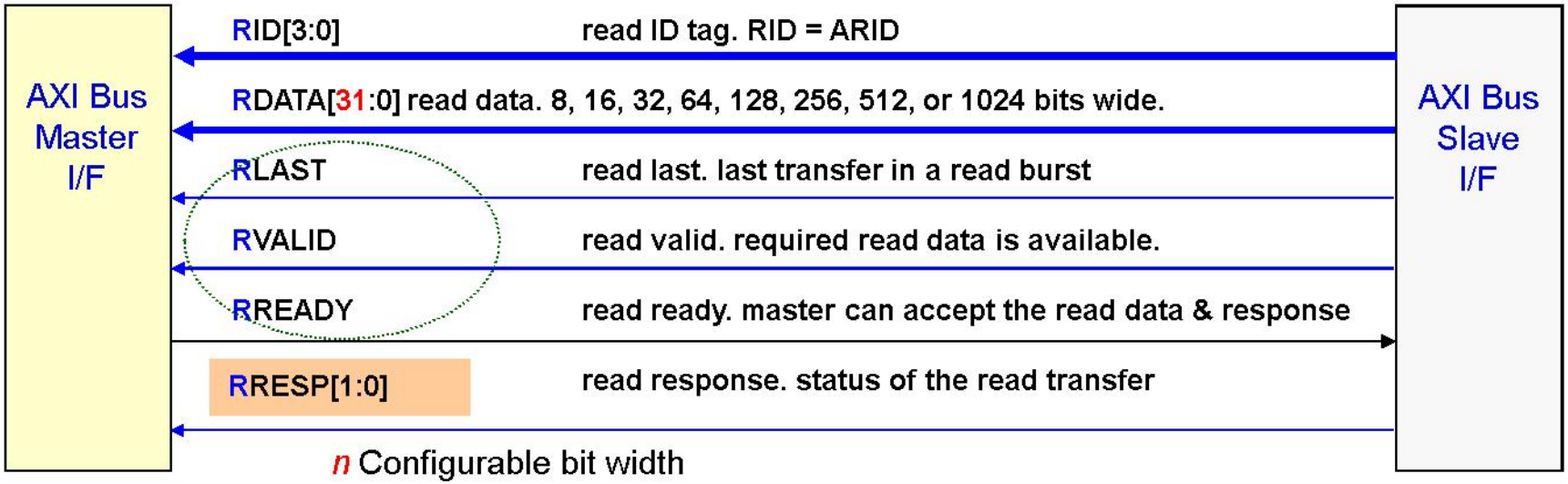
    ✓ Write/Read address channel signals (AW, AR)
    ✓ Write/Read data channel signals (W, R)
    ✓ Write response channel signals (B)

Write/Read address channel signals (AW, AR)



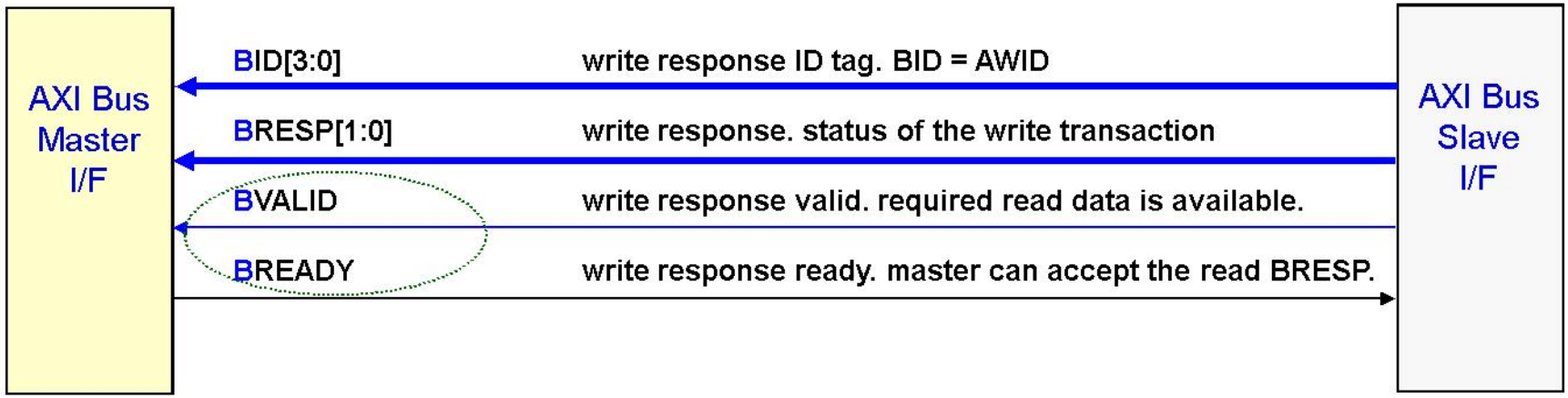
Write/Read data channel signals (W, R)





n Configurable bit width

Write response channel signals (B)



Agenda

- **AXI protocol**

  - **Ch. 3 Channel Handshake**

    - ✓ Handshake process
    - ✓ Relationships between the channels
    - ✓ Dependencies between channel handshake signals

Handshake process

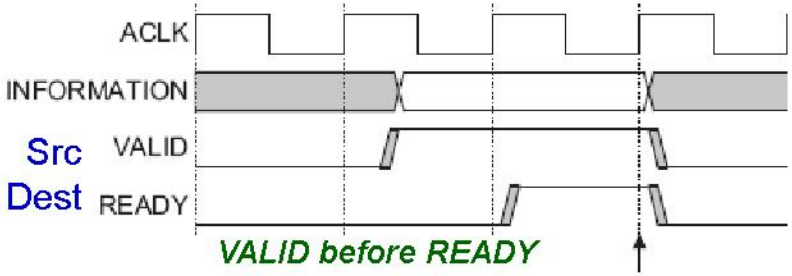
- All five channels use the same **VALID/READY** handshake to transfer data and control info.

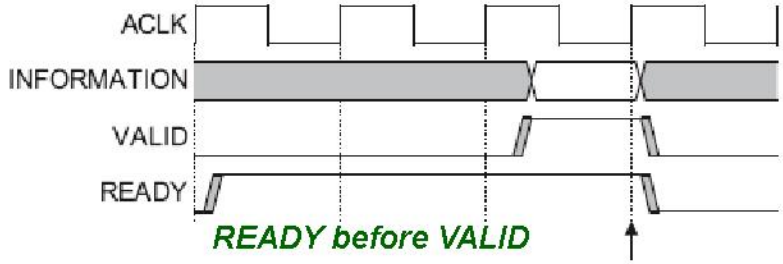
- **Source**
  - generates the **VALID** signal
    - ✓ data or control info. is available

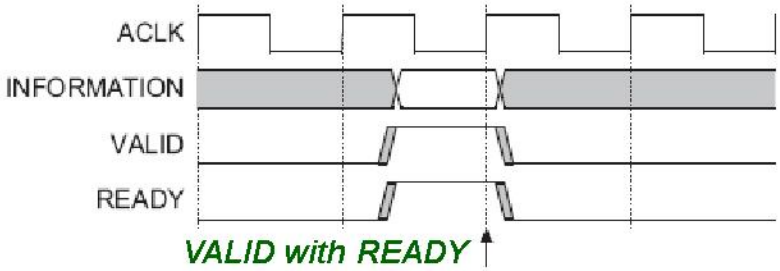
- **Destination**
  - generates the **READY** signal
    - ✓ accepts the data or control info

- Transfer occurs only when both the **VALID** and **READY** signals are HIGH. Arrow !

- The recommended default **READY** value is HIGH.
  - Latency
  - Dead-lock free design



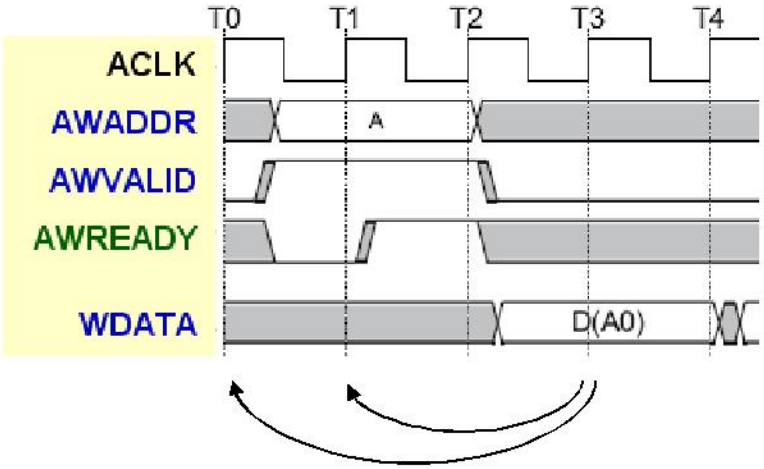


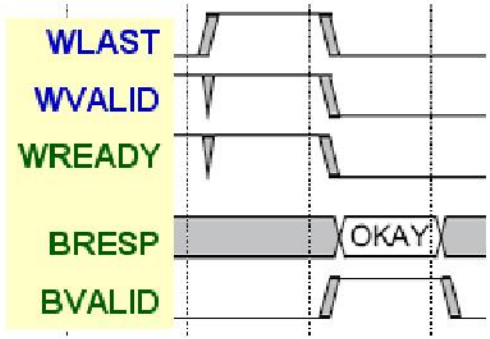


Channel	AW	AR	W	R	B
VALID(Source)	(M)	(M)	(M)	(S)	(S)
READY(Des.)	(S)	(S)	(S)	(M)	(M)

Relationships between the channels

- **The relationship between the address, read, write, and write response channels is flexible.**

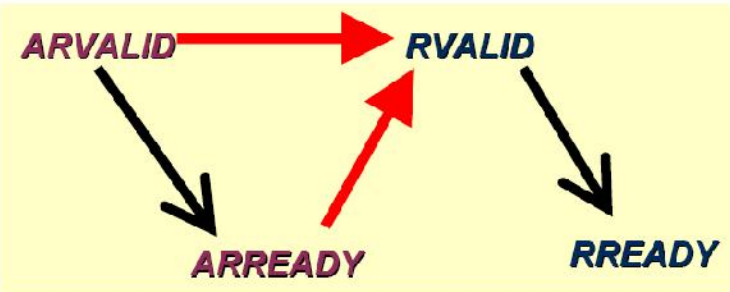




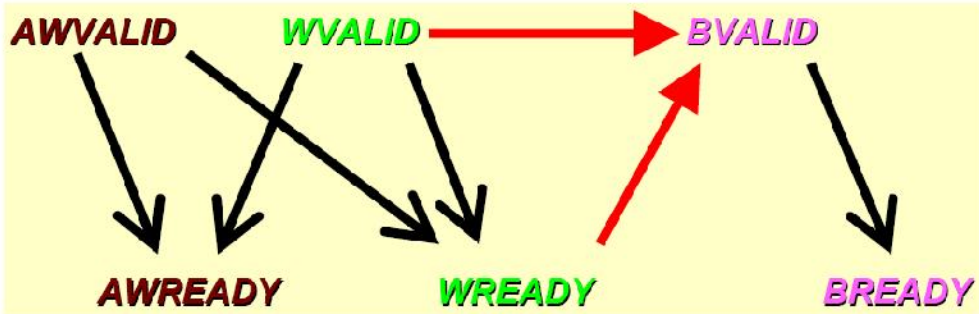
- **Two relationships that must be maintained are:**
  - read data must always follow the address to which the data relates
  - a write response must always follow the last write transfer in the write transaction to which the write response relates.

Dependencies b/w channel handshake signals

- **To prevent a deadlock situation, you must observe the dependencies that exist between the handshake signals.**
  - In any transaction:
    - ✓ **VALID** signal must not be dependent on the **READY** signal.
    - ✓ **READY** signal can wait for assertion of the **VALID** signal.



Read transaction handshake dependencies



Write transaction handshake dependencies

→ arrows point to signals that can be asserted before or after the previous signal is asserted.

→ arrows point to signals that must be asserted only after assertion of the previous signal.

Agenda

- AXI protocol

  - **Ch. 4 Addressing Options**

    - ✓ About addressing options
    - ✓ Burst length
    - ✓ Burst size
    - ✓ Burst type
    - ✓ Burst address

About addressing options

- **Master begins each burst by driving transfer control information and the address of the first byte in the transfer.**
  - A channel: AADDR, ALEN, ASIZE, ABURST
- **slave to calculate the addr. of subsequent transfers in the burst.**
  - Address type : ABURST
  - Address value : ALEN, ASIZE
- **Master to not cross *4KB boundaries* on a single access**
  - prevents the problem of a sequential burst starting in one slave and continuing in the next memory map slave.
  - MMU tiny page region
    - ✓ ARMv5 (ARM9) : 1KB (AHB master)
    - ✓ ARMv6 (ARM11, Cortex): 4KB (AXI master)

Burst Length: AR(W)LEN[3:0]

- **Number of data transfers**
  - AW(R)LEN[3:0] + 1
- **Wrapping burst length**
  - 2, 4, 8, or 16 transfers.
- **No component can terminate a burst early to reduce the number of data transfers.**
  - write burst, the master can disable further writing by deasserting all the write strobes.
    - ✓ Ex. 64 bit data bus, WSTRB[7:0] = 8'b0
  - read burst, the master can discard further read data.

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

Burst Size: AR(W)SIZE[2:0]

- maximum number of data bytes to transfer within a burst.
- The size of any transfer <= data bus width.

Burst size encoding

ARSIZE[2:0] AWSIZE[2:0]		Bytes in transfer
b000	Byte	1
b001	Half-word	2
b010	Word	4
b011	2-Word	8
b100	…	16
b101	…	32
b110	…	64
b111		128

Burst Type: AR(W)BURST[1:0]

- **Selects the burst type.**

Burst type encoding

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	-	-

- **Wrapping burst**
  - address wraps around to a lower address when a wrap boundary is reached.
  - The wrap boundary is the size (*AR(W)SIZE*) of each transfer in the burst multiplied by the total number (*AR(W)LEN*) of transfers in the burst.

Ex. Burst Address Calculation

❑ Incrementing Write Burst : AWBURST[1:0] = 2'b01

- Master

Start Address	AWADDR[31:0] = 32'h0000_1004
Burst Length	AWLEN[3:0] = 4'b0011 (LEN-4)
Burst Size	AWSIZE[2:0] = 3'b010 (Word, 4bytes)

- Slave

1st	2nd	3rd	4th
+0	+4	+8	+C
32'h0000_1004	32'h0000_1008	32'h0000_100C	32'h0000_1010

❑ Wrapping Read Burst : ARBURST[1:0] = 2'b10

- Master

Start Address	ARADDR[31:0] = 32'h0000_1004
Burst Length	ARLEN[3:0] = 4'b0011 (LEN-4)
Burst Size	ARSIZE[2:0] = 3'b010 (Word 4bytes)

- Slave

wrap boundary : 4 byte x LEN-4 = 16-byte boundaries. (5'b1_0000)

1st	2nd	3rd	4th
+0	+4 (4 bytes)	+8	+C
32'h0000_1004	32'h0000_1008	32'h0000_100C	32'h0000_1000

❑ AXI protocol

- **Ch. 5 Additional Control Information**

  ✓ Cache support
  ✓ Protection unit support

Cache support: AR(W)CACHE[3:0]

- **Bufferable bit (B): AWCACHE[0]**
  - Write delay can be an arbitrary one
- **Cacheable bit (C): AR(W)CACHE[1]**
  - Read: prefetch or read cache is possible
  - Write: write merging is possible
- **Read Allocate bit (RA): ARCACHE[2]**
  - If read cache miss, fetch the data to cache
  - If C=low, RA=low
- **Write Allocate bit (WA): AWCACHE[3]**
  - If write cache miss, fetch the data to cache, and then write to the cache (and through the memory)
  - If C=low, WA=low

Cache encoding

ARCACHE[3:0]
AWCACHE[3:0]

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

- **Normal or privileged: AR(W)PROT[0]**
  - High : privileged
  - privileged processing mode typically has a greater level of access within a system.
- **Secure or non-secure: AR(W)PROT[1]**
  - Low : secure
- **Instruction or data, AR(W)PROT[2]**
  - High / low : instruction / data
  - not accurate in all cases.
    - ✓ a transaction contains a mix of I$, D$.
  - default
    - ✓ data access

	Protection encoding
ARPROT[2:0] AWPROT[2:0]	Protection level
[0]	1 = privileged access 0 = normal access
[1]	1 = nonsecure access 0 = secure access
[2]	1 = instruction access 0 = data access

❑ AXI protocol

● Ch. 6 Atomic Accesses

- ✓ About atomic accesses
- ✓ Locked access
- ✓ Exclusive access

Atomic Accesses: AR(W)LOCK[1:0]

- **Normal access**
- **Locked access**
  - Start with 2'b10, and end with 2'b00
  - During the period, only the lock initiating master can access the address region (not the slave!!!)
  - swap instruction

Atomic access encoding

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

Exclusive access

❑ Exclusive access, AR(W)LOCK[1:0]=b01

- Exclusive read … Exclusive write
- If no intervening write to the address region, EXOKAY response. If not, OKAY response.
- Usually used for read-modify-write
- The advantage
  - ✓ semaphore type operations do not impact either the critical bus access latency or the maximum achievable bandwidth.

❑ AXI protocol

- **Ch. 7 Response Signaling**

  ✓ About response signaling
  ✓ Response types

AXI Response Signaling

❑ Write Transaction

- One response given for the entire burst

❑ Read Transaction

- Slave can give different responses for different transfers within a burst

RRESP[1:0] and BRESP[1:0] encoding

RRESP[1:0] BRESP[1:0]	Response	Meaning
b00	OKAY	Normal access successful or Exclusive access failure or an exclusive access to a slave that doesn't support exclusive accesses
b01	EXOKAY	Exclusive access successful
b10	SLVERR	Slave error (FIFO error, unsupported transfer size, write to read only location, timeout in slave, slave power down, unmapped address location in slave)
b11	DECERR	Decode error (when there is no slave at transaction address), *default slave* that generates R(B)VALID and signals DECERR on R(B)RESP.

NOTE: The burst is not cancelled if the slave gives a single error response

❑ AXI protocol

● Ch. 8 Ordering Model

- ✓ About the ordering model
- ✓ Transfer ID fields
- ✓ Read ordering
- ✓ Normal write ordering
- ✓ Write data interleaving
- ✓ Read and write interaction
- ✓ Interconnect use of ID fields
- ✓ Recommended width of ID fields

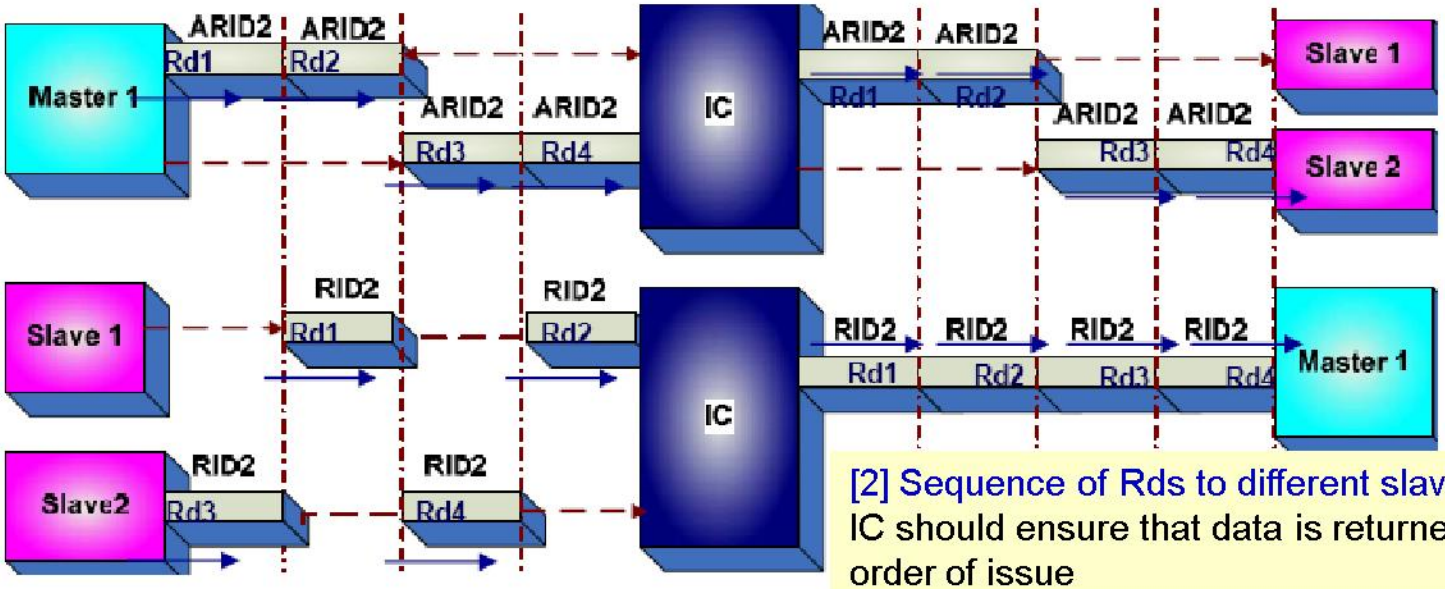
Ordering Model

- **High performance, maximizing data throughput**
  - Out of order transaction completion
  - Issue of multiple outstanding addresses
  - 5 transaction IDs (*AWID, WID, BID, ARID, RID*)

- **Same ID transactions must be ordered**
  - No restriction on transactions with different IDs.

Read Ordering

[1] Sequence of Rds to same slave with same ARID
slave should return data in same order as order of issue

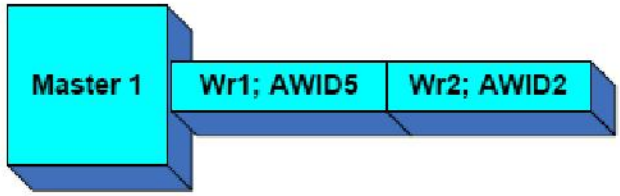


[2] Sequence of Rds to different slave with same ARID
IC should ensure that data is returned in same order as order of issue

Write Ordering

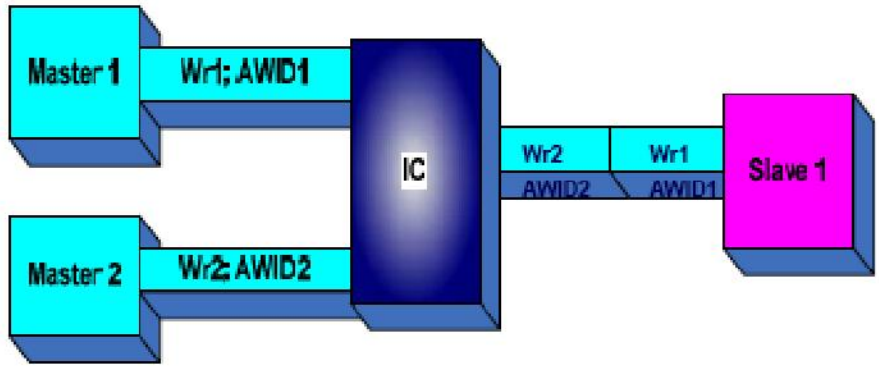
[1] Same Masters with different AWID:

can complete in any order



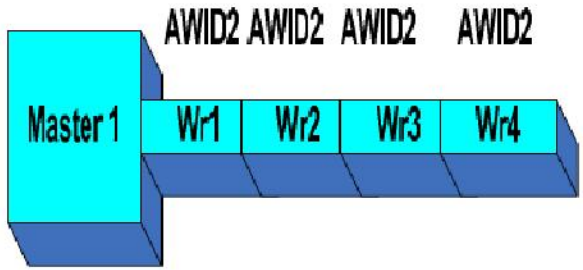
[2] Different Masters with different AWID:

can complete in any order

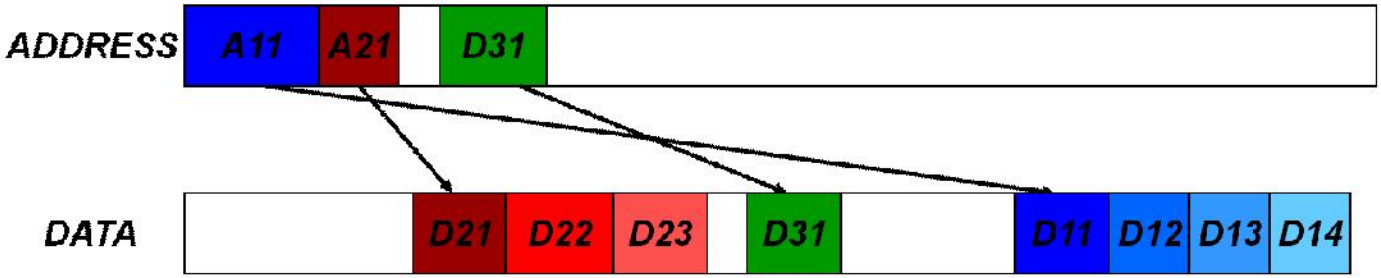


[3] Sequence of Wrs from same master with same AWID:

must complete in order of issue



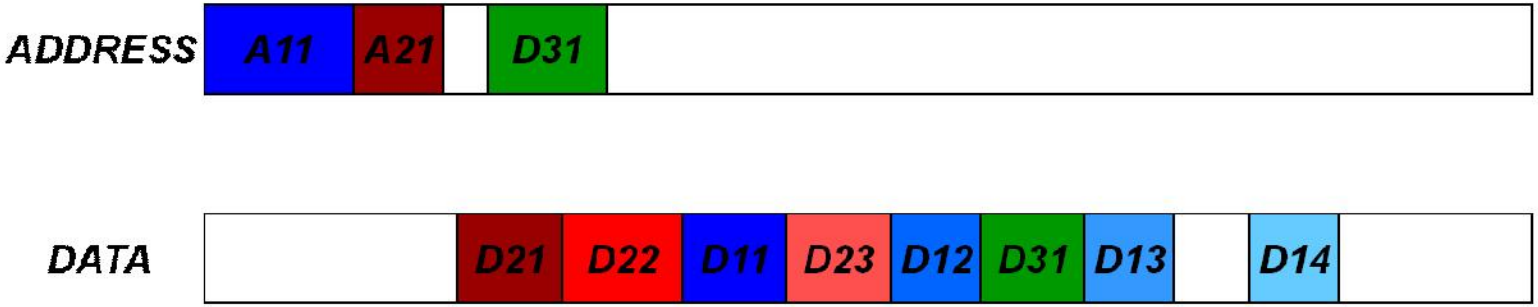
Out of order transaction



❑ Out of order rule

- Each transaction has an ID attached (given by the master IF)
- Each Channels have ID signals
  - ✓ – *ARID, AWID, RID, WID, & BID (Up to four bits)*
- Ordering by transaction ID
  - ✓ Master needs to finish data transfers with the same transaction ID in the order of request issue.
  - ✓ Slave can handle data transfers with different transaction IDs out of order
- The interconnect in a multi-master system must append another tag to ID to make each master's ID unique

Data Interleaving



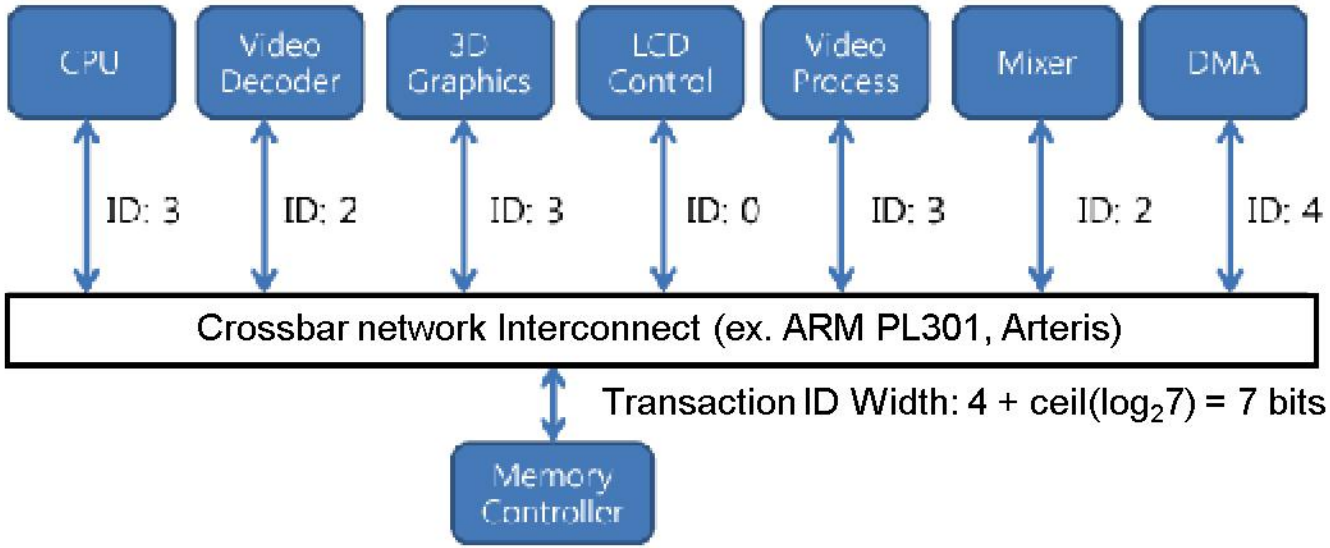
❑ Interleaving rule

- Data with different ID can be interleaved.
- The order within a single burst is maintained

Transaction ID Implementation

❑ Real implementation

- Transaction ID = <master ID, channel ID>
- Channel ID = original AXI transaction id
- Master ID is needed to identify the initiating master among all the masters



❑ AXI protocol

- **Ch. 9 Data Buses**

  ✓ Write strobes on page
  ✓ Narrow transfers

Byte Lane mapping

❑ Data buses

- 2 independent data buses (W, R), read data and write data
- their own individual handshake signals
  - ✓ It is possible for data transfers to occur on both buses at the same time.

❑ Write Strobes : W channel, WSTRB [n:0], n = (data bus width/8) - 1

- Each write strobe indicates corresponding byte lane has valid data

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

❑ Narrow transfers (transfer narrower than data bus width)

5 beat burst

Start addr[3:0] = 32'h100

8 bit transfer

32 bit data bus

Byte lane used

			DATA[7:0]	1st transfer	WSTRB[3:0] = 4'b0001, 32'h100
		DATA[15:8]		2nd transfer	WSTRB[3:0] = 4'b0010, 32'h101
	DATA[23:16]			3rd transfer	WSTRB[3:0] = 4'b0100, 32'h102
DATA[31:24]				4th transfer	WSTRB[3:0] = 4'b1000, 32'h103
			DATA[7:0]	5th transfer	WSTRB[3:0] = 4'b0001, 32'h104

❑ AXI protocol

● Ch. 10 Unaligned Transfers

- ✓ About unaligned transfers
- ✓ Examples

About unaligned transfers

❑ Burst-based addressing

- Transaction consists of a number of data transfers.
- Typically, each data transfer is aligned to the size of the transfer.
  - ✓ Ex. 32-bit wide transfer is aligned to four-byte
    - *32'h…0, 32'h…4, 32'h…8, 32'h…C*
  - ✓ Ex. 64-bit wide transfer is aligned to eight-byte
    - *32'h…0, 32'h…8*

❑ Begin a burst at an unaligned address.

- it is possible that the first bytes that have to be accessed do not align with the natural data width boundary.
  - ✓ Ex. 32-bit (four-byte) data that starts at address of *32'h…**2***
- software designers to request transfers to/from any address, without having to ensure that the address is aligned to the width of the transfer.

Ex. Unaligned transfers on a 32-bit bus

Address: 0x01
Transfer size: 32 bits
Burst type: incrementing
Burst length: 4 transfers

3	2	1	0	1st transfer
7	6	5	4	2nd transfer
B	A	9	8	3rd transfer
F	E	D	C	4th transfer

WSTRB[3:0] = 4'b1110

WSTRB[3:0] = 4'b1111

AW(AR)ADDR = 32'h….1
AW(AR)SIZE = 3'b101 (WORD)
AW(AR)BURST = 2'b01 (INCR)
AW(AR)LEN = 4'h3 (LEN4)

Address: 0x07
Transfer size: 32 bits
Burst type: incrementing
Burst length: 5 transfers

7	6	5	4	1st transfer
B	A	9	8	2nd transfer
F	E	D	C	3rd transfer
13	12	11	10	4th transfer
17	16	15	14	5th transfer

WSTRB[3:0] = 4'b1000

WSTRB[3:0] = 4'b1111

AW(AR)ADDR = 32'h….7
AW(AR)SIZE = 3'b101 (WORD)
AW(AR)BURST = 2'b01 (INCR)
AW(AR)LEN = 4'h4 (LEN5)

NOTE: shaded cells indicate bytes that are not transferred

❑ AXI protocol

- **Ch. 11 Clock and Reset**

ACLK, ARESETn

❑ Clock

- single clock signal, ACLK.
- All input signals are sampled on the rising edge of ACLK.
- All output signal changes must occur after the rising edge of ACLK.

❑ Reset

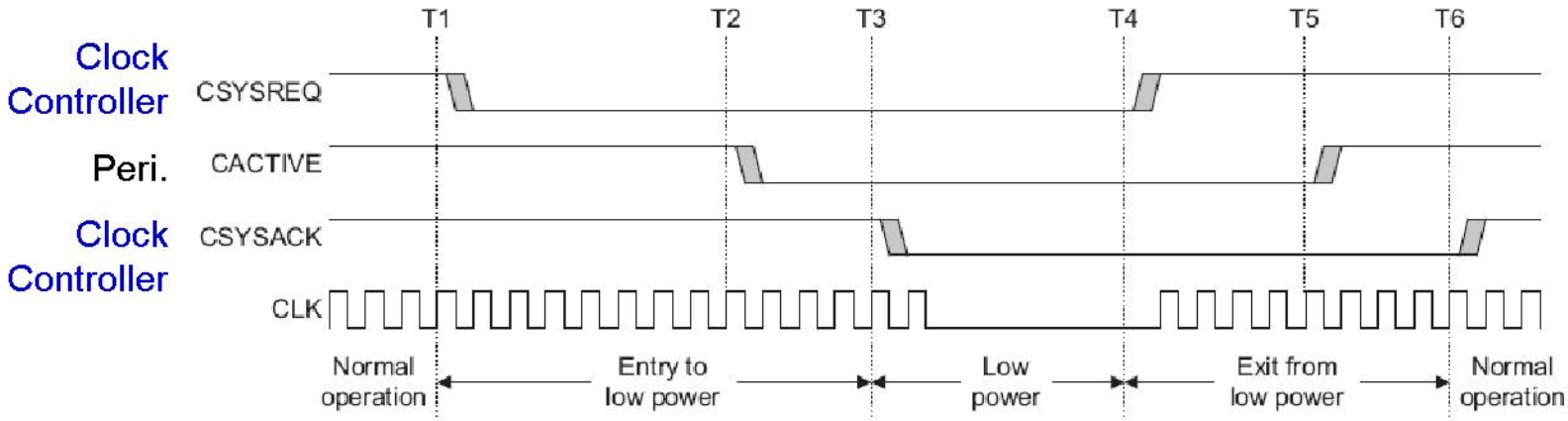
- single active LOW reset signal, ARESETn.
- The reset signal can be asserted asynchronously
- Reset value
  - ✓ Master: ARVALID, AWVALID, and WVALID *LOW*
  - ✓ Slave : RVALID and BVALID *LOW*.
  - ✓ All other signals can be driven to any value.

❑ AXI protocol

- **Ch. 12 Low-power Interface**

Acceptance of a low-power request

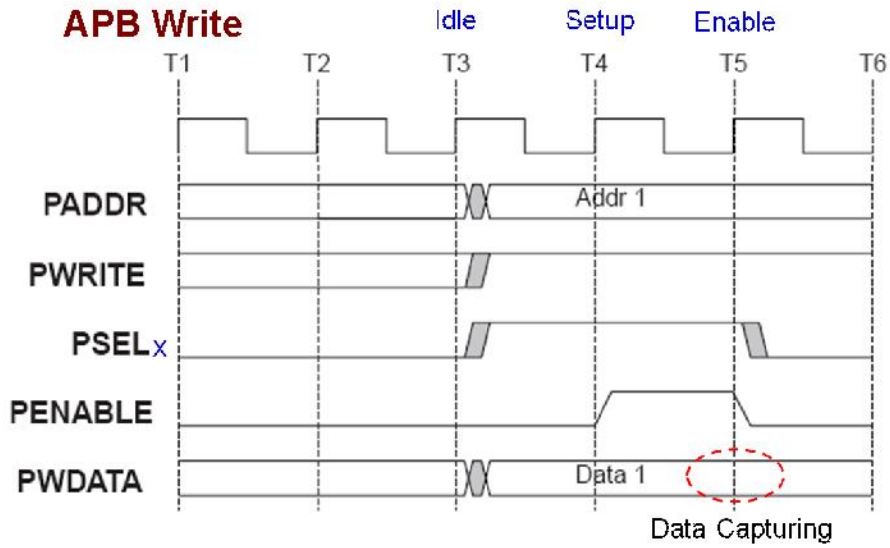
- **CSYSREQ**
  - To request that the peripheral enter a low-power state, the system clock controller drives the CSYSREQ signal LOW. During normal operation, CSYSREQ is HIGH.
- **CACTIVE**
  - The peripheral asserts CACTIVE to indicate that it requires the clock, and the system clock controller must enable the clock immediately.
- **CSYSACK**
  - The peripheral uses the CSYSACK signal to acknowledge both the low-power state request and the exit from the low-power state.

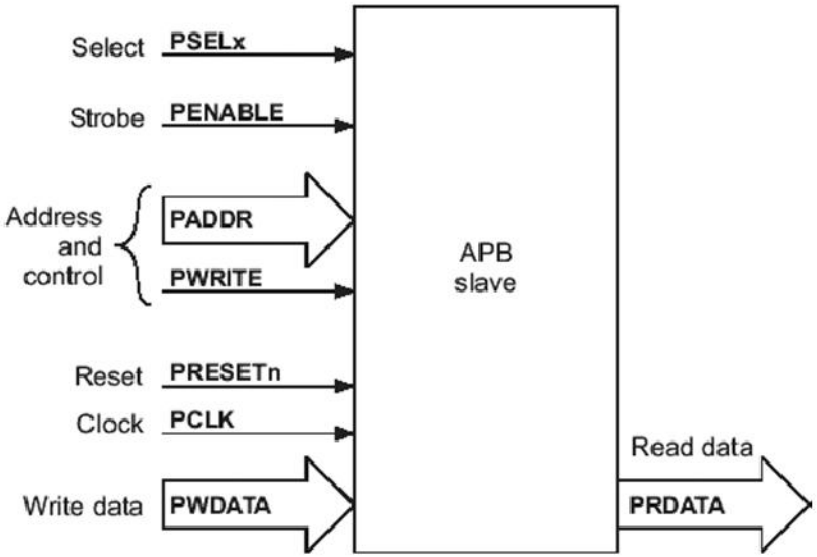


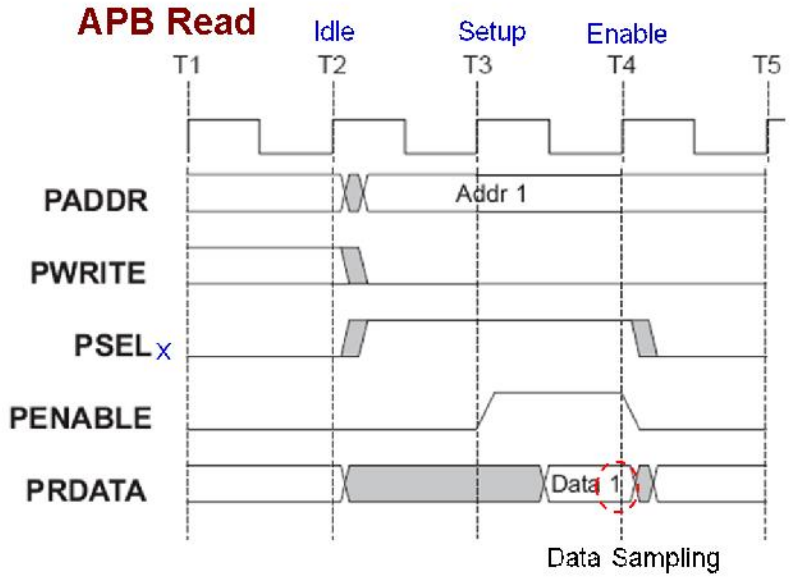
❑ APB3 Protocol

AMBA 2.0 APB Slave Protocol

Name	Description
PCLK	Bus clock (Rising edge)
PRESETn	APB reset (Active low)
PADDR[31:0]	APB address bus
PSELx	APB select (secondary decorder, within peripheral bridge)
PENABLE	APB strobe
PWRITE	APB transfer direction
PRDATA	APB read data bus
PWDATA	APB write data bus

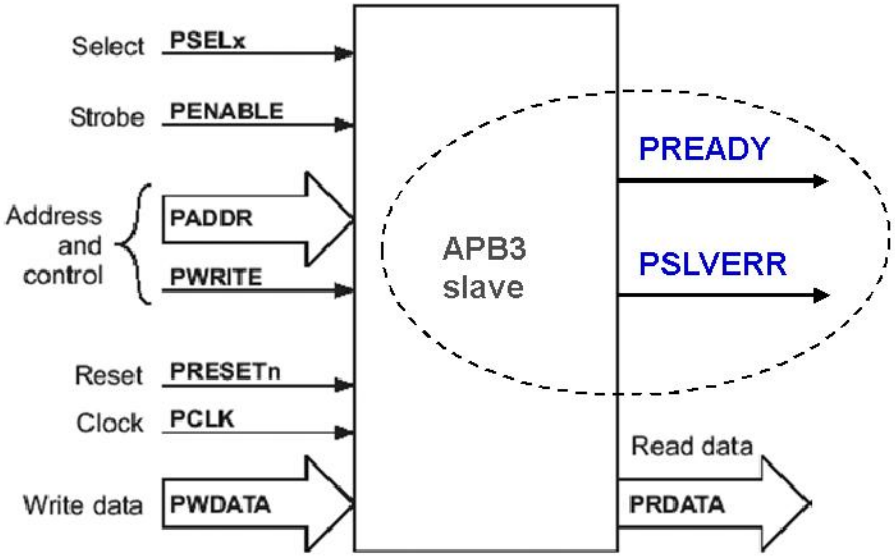


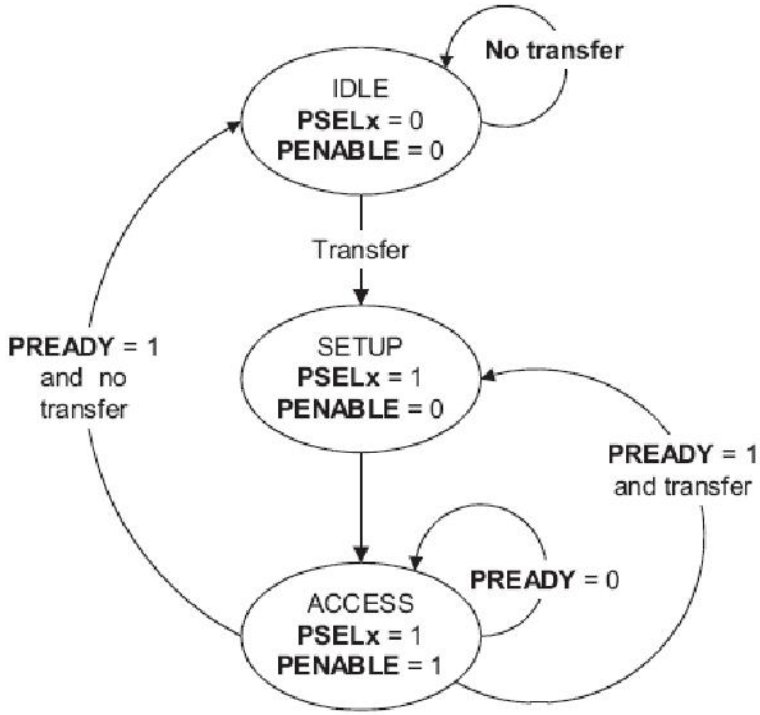




Changes for APB3 Slave Protocol Spec.

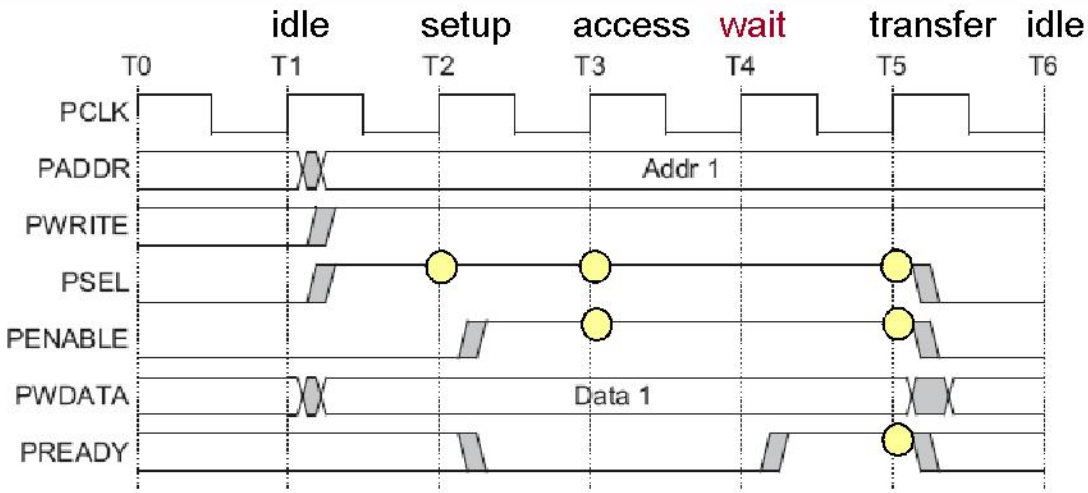
- A ready signal, **PREADY**, to extend an APB transfer.
- An error signal, **PSLVERR**, to indicate the failure of a transfer.



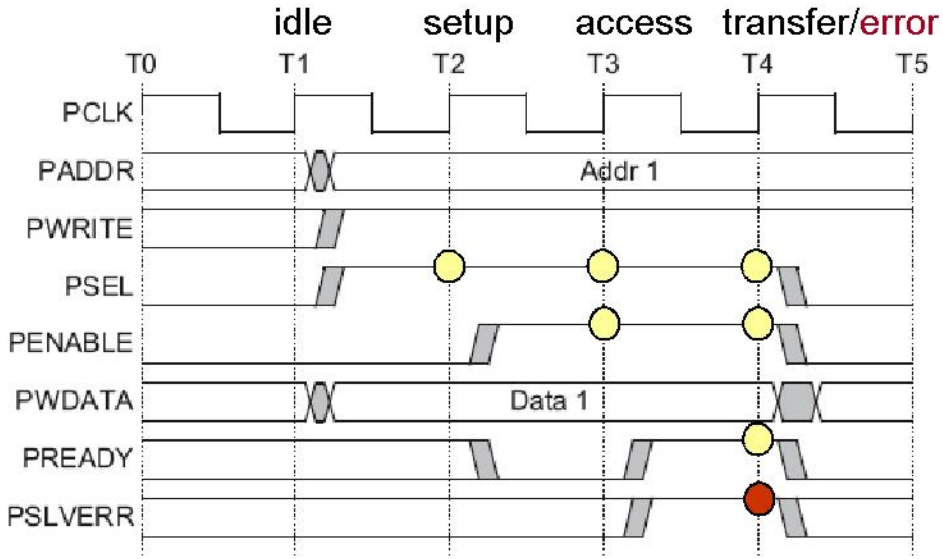


Ex. Crossbar to/from APB3 Slave Transfers

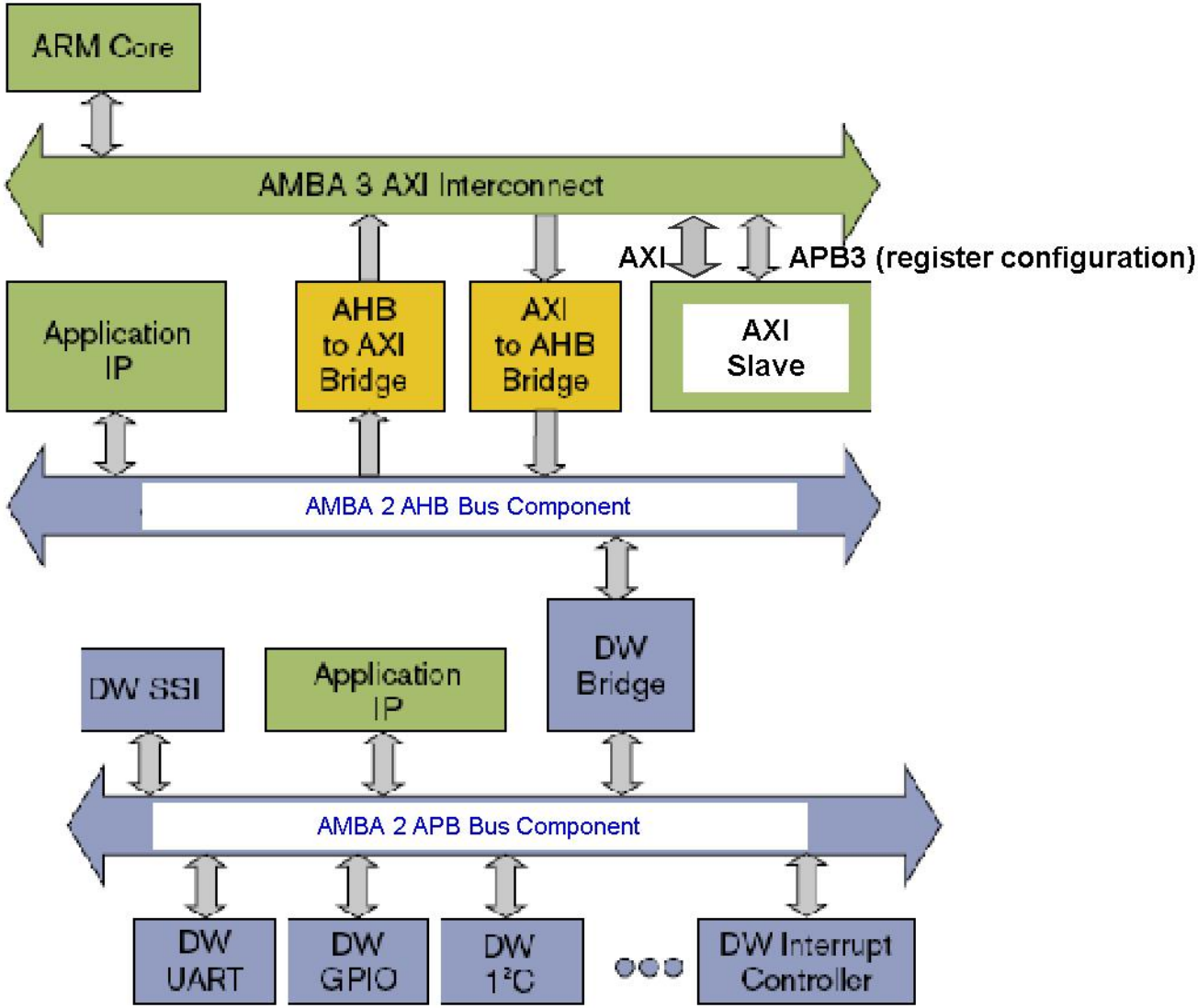
Write transfer with wait states



Failing write transfer



Ex: AXI-based SoC Platform



Appendix

❑ AMBA 2.0 AHB/APB Bus Protocol Seminar

e-SIC 2.0 System IC Business Team | BOARD | PROJECT

공통게시판 그룹게시판 커뮤니티

교육/세미나

AMBA 2.0 bus protocol/component 및 검증

Writer	김시현
Date	2010년 7월 5일 10시 39분
File #1	ambaBusSpec_20100702_shkim.pdf
File #2	ambaBusVerify_20100702_shkim.pdf

공통 게시판

- 공지사항
- 뉴스

❑ AMBA Spec., Design and Integration Guide:

- ARM Information Center

http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.amba/index.html