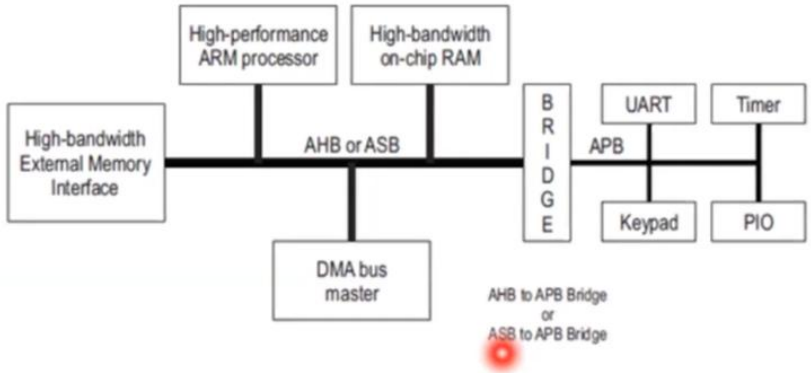
Evolution of AMBA (cont...)

Initially in SoC designs, AHB (Advanced High Performance Bus) used for high bandwidth interconnect and
APB (Advanced Peripheral Bus) for low bandwidth peripheral interconnect

With increasing number of functional blocks (IP) integrating into SoC design,
the shared bus protocol started hitting limitation and then in 2003 AMBA3 introduced a new point-to-point connectivity protocol – AXI (Advanced Extensible Interface)

Later in 2010, enhanced version AXI4 was introduced



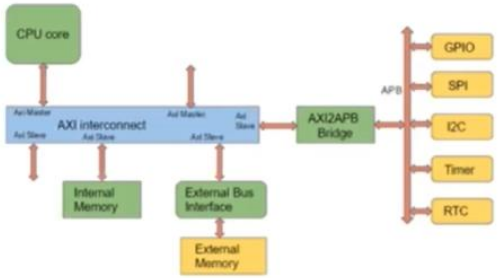
Evolution of AMBA (cont...)

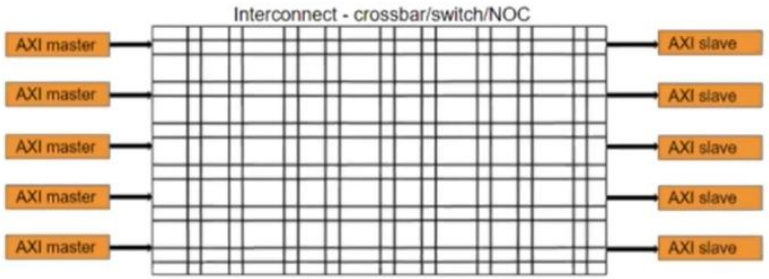
Further evolution happened in the era of mobile and smartphones with SoCs having dual/quad/octa core processors with shared caches integrated and the need for hardware managed coherency across the memory subsystem, this lead to AMBA4 ACE

Lastly, in the current era of heterogeneous computing for HPC and data center markets, the integration trend continues with increasing number of processor cores along with several heterogeneous computing elements like GPU, DSP, FPGAs, memory controllers and IO sub systems.

In 2013, AMBA5 CHI was introduced as a re-design of the AXI/ACE protocol.

The signal based AXI/ACE protocol was replaced with the new packet based CHI layered protocol that can scale very well for near term future.

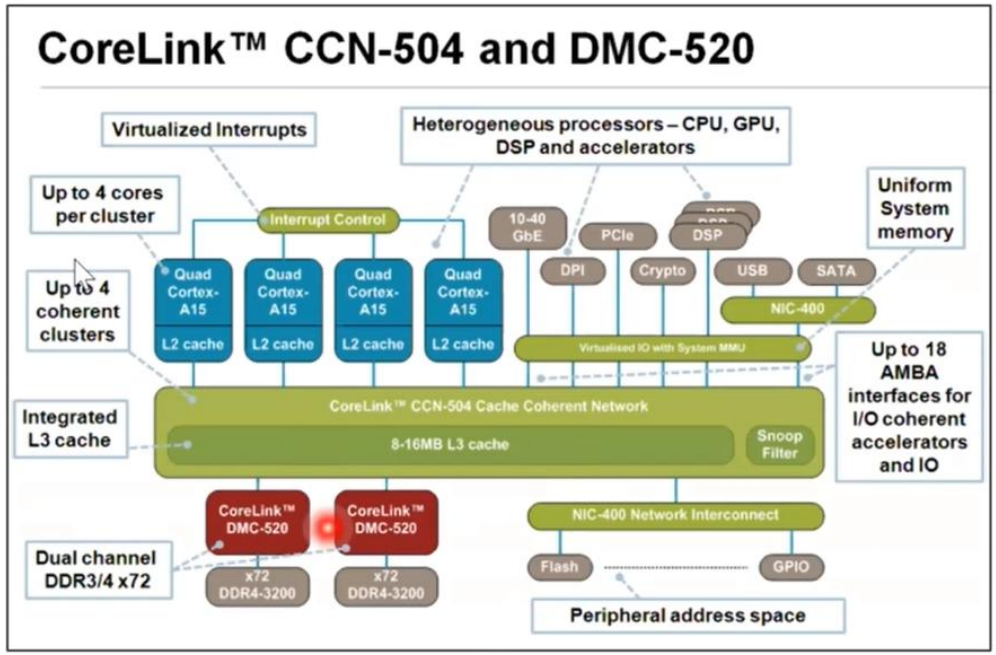


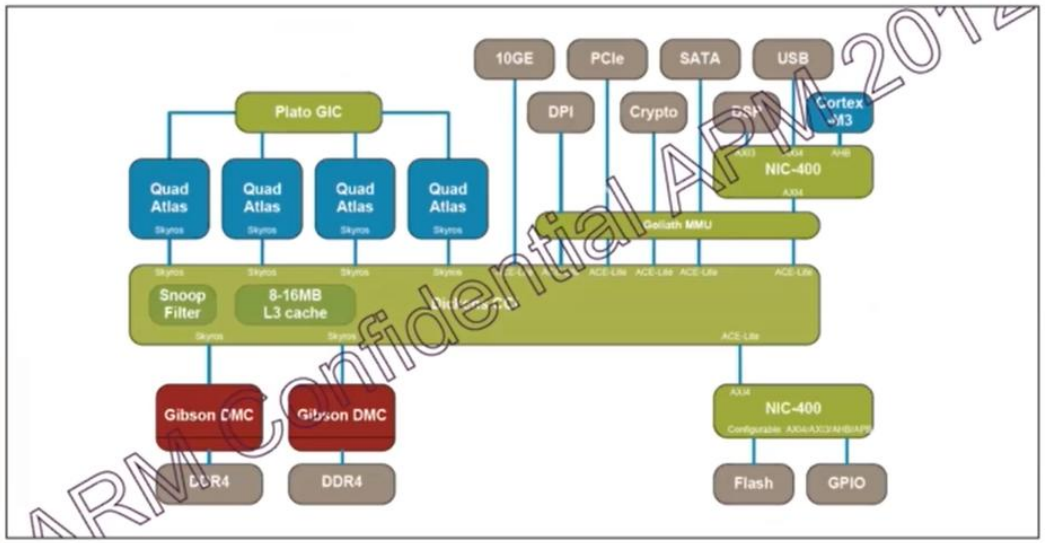


AXI interconnect can be used to build SoC with various functional blocks talking through a master-slave protocol.

Interconnect helps in scaling connectivity compared to AHB bus.
AXI2APB bridge is used to communicate to a set of peripherals shared on APB bus.

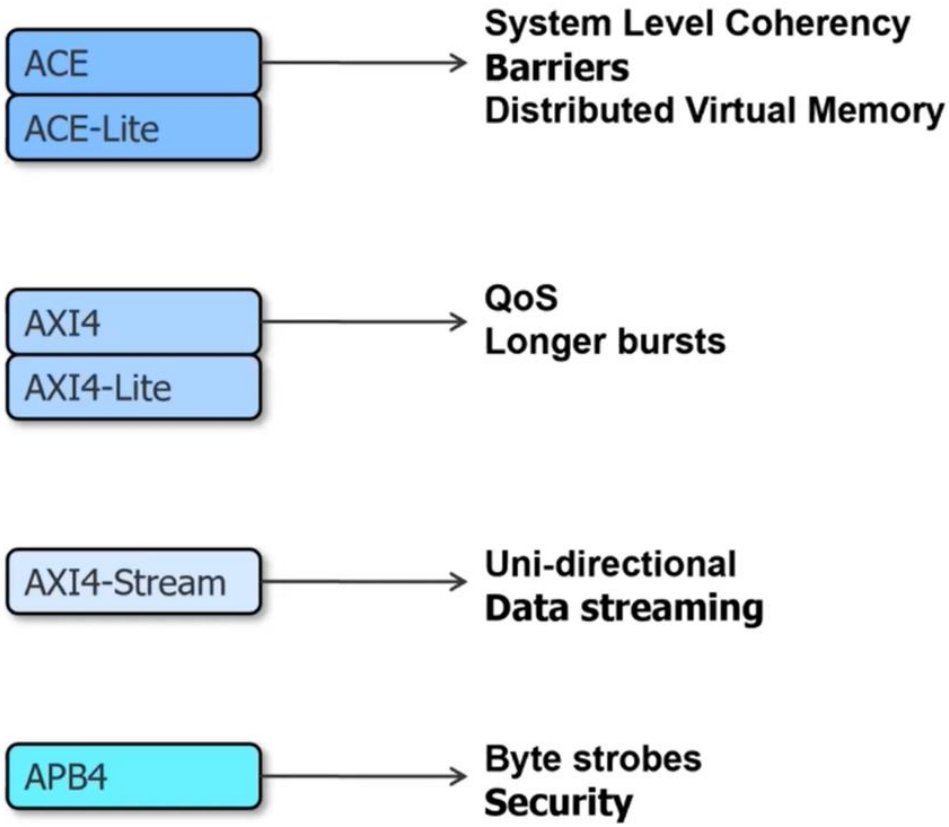
ARM-AMBA based systems





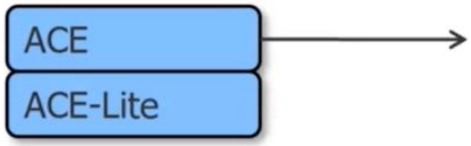
Source – Based on ARM Corelink CCN-504 diagram

Headlines for AMBA 4

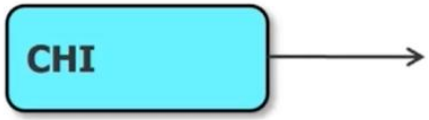


Cache Coherency Protocols

ARM



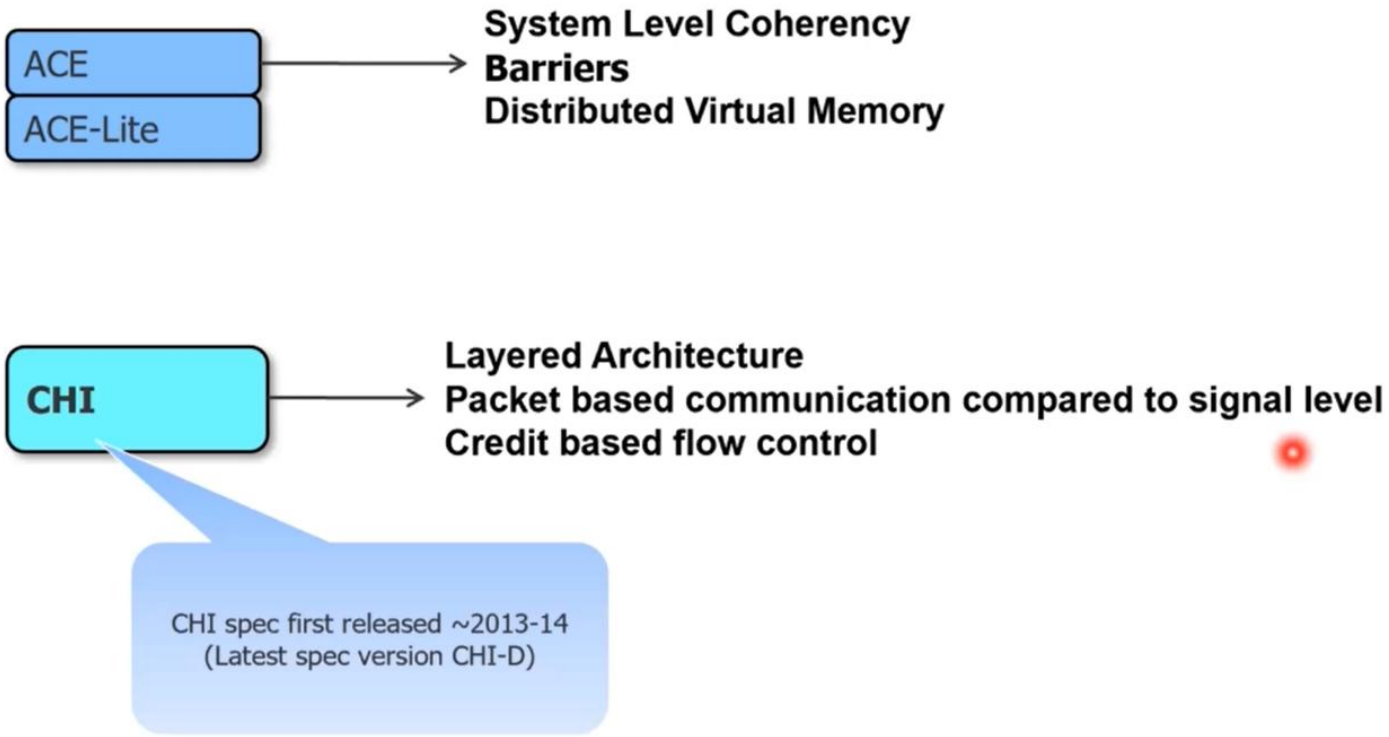
System Level Coherency
Barriers
Distributed Virtual Memory



Layered Architecture
Packet based communication compared to signal level
Credit based flow control

Cache Coherency Protocols

ARM



1. Part 1 - AMBA protocol evolution and APB2/3/4 protocol concepts overview

APB
- Used for connecting low bandwidth peripherals
- Simple non-pipelined protocol
- Shared bus from single master to multiple slaves

AHB
- Used for connecting components that need higher bandwidth on shared bus
- This is shared bus with multiple masters and slaves
- Maximum concurrency of transactions is 2
- Higher bandwidth is possible through Burst transfers

AHB-lite
- Simplified version of AHB
- Single master and multiple slave
  - Removes need for arbitration, retry and split transactions

AXI
- Used for high bandwidth, low latency interconnects
- Point-to-point connection interface, overcomes limitations of shared bus protocol in terms of number of agents that can be connected
- Support for
  - Multiple outstanding data transfers (pipe-lined)
  - Burst data transfers
  - Out of order responses, data interleaving
  - Separate RD-WR paths

AXI4-lite
- Simplified version of AXI4 full
- No support of burst data transfers

AXI4-Stream
- Single direction data transfer from Master to Slave
- No separate read-write channels
- Useful for designs such as video streaming applications

ACE
- AXI coherency extension over AXI4 protocol
- Supports hardware based coherency
- Extends AXI read, write channels and introduces new snoop request, data, response channels

ACE-lite
- Simplified version of ACE for agents that does not have its own cache
- Typically used for IO coherent agents

CHI
- ACE, as extension over AXI, used signal level communication between master-slave
  - Increased wires with additional snoop channel
  - Worked well for small coherent clusters
- With increased number of coherent clusters, AMBA5 CHI was introduced as a complete re-design of the ACE protocol.
- Uses packet based layered protocol

APB

- Used for connecting low bandwidth peripherals
- Simple non-pipelined protocol
- Shared bus from single master to multiple slaves

AHB

- Used for connecting components that need higher bandwidth on shared bus
- This is shared bus with multiple masters and slaves
- Maximum concurrency of transactions is 2
- Higher bandwidth is possible through Burst transfers

AHB-lite

- Simplified version of AHB
- Single master and multiple slave
  - Removes need for arbitration, retry and split transactions

AXI

- Used for high bandwidth, low latency interconnects
- Point-to-point connection interface, overcomes limitations of shared bus protocol in terms of number of agents that can be connected
- Support for
  - Multiple outstanding data transfers (pipe-lined)
  - Burst data transfers
  - Out of order responses, data interleaving
  - Separate RD-WR paths

AXI4-lite

- Simplified version of AXI4 full
- No support of burst data transfers

AXI4-Stream

- Single direction data transfer from Master to Slave
- No separate read-write channels
- Useful for designs such as video streaming applications

ACE

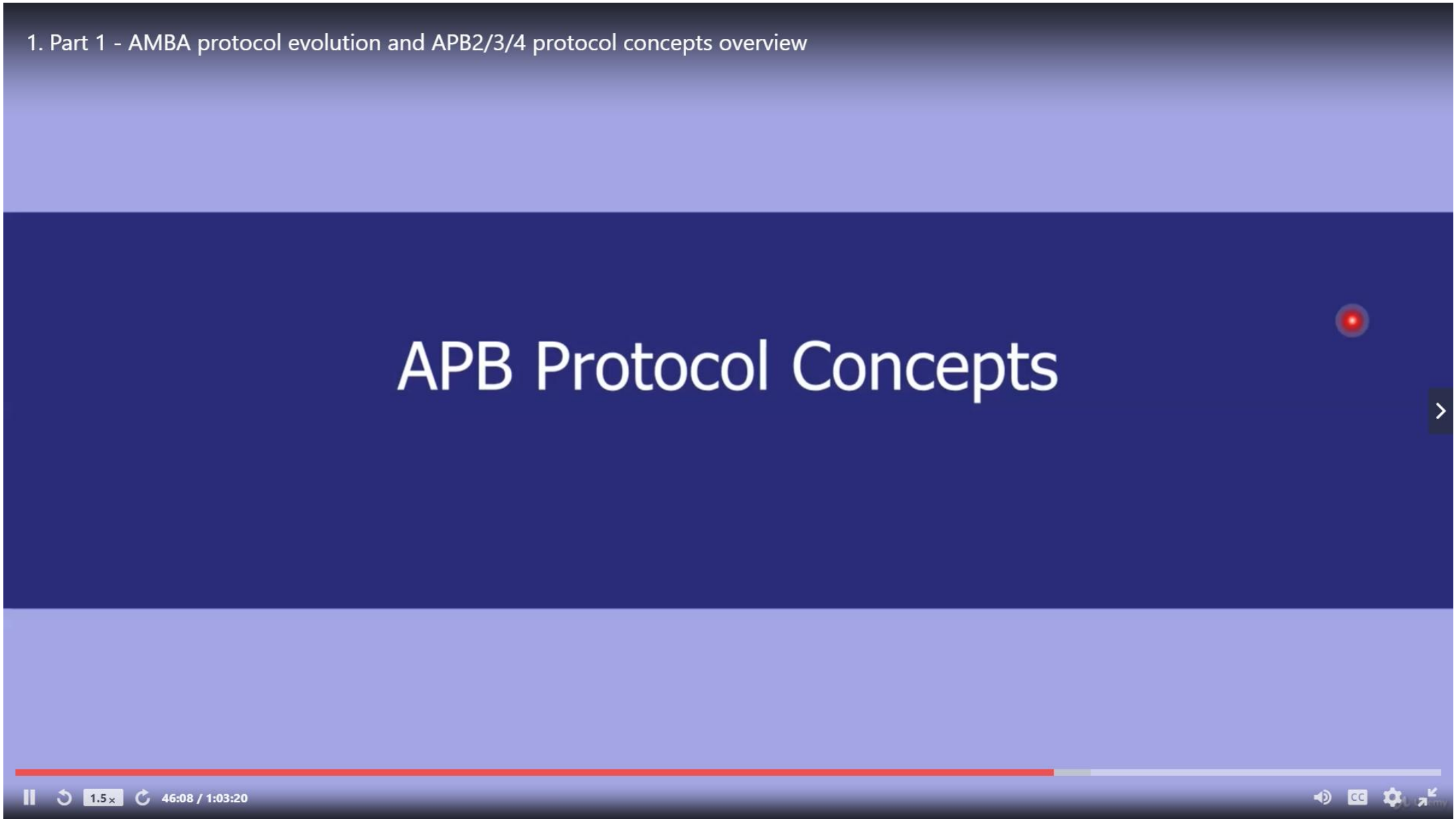
- AXI coherency extension over AXI4 protocol
- Supports hardware based coherency
- Extends AXI read, write channels and introduces new snoop request, data, response channels

ACE-lite

- Simplified version of ACE for agents that does not have its own cache
- Typically used for IO coherent agents

CHI

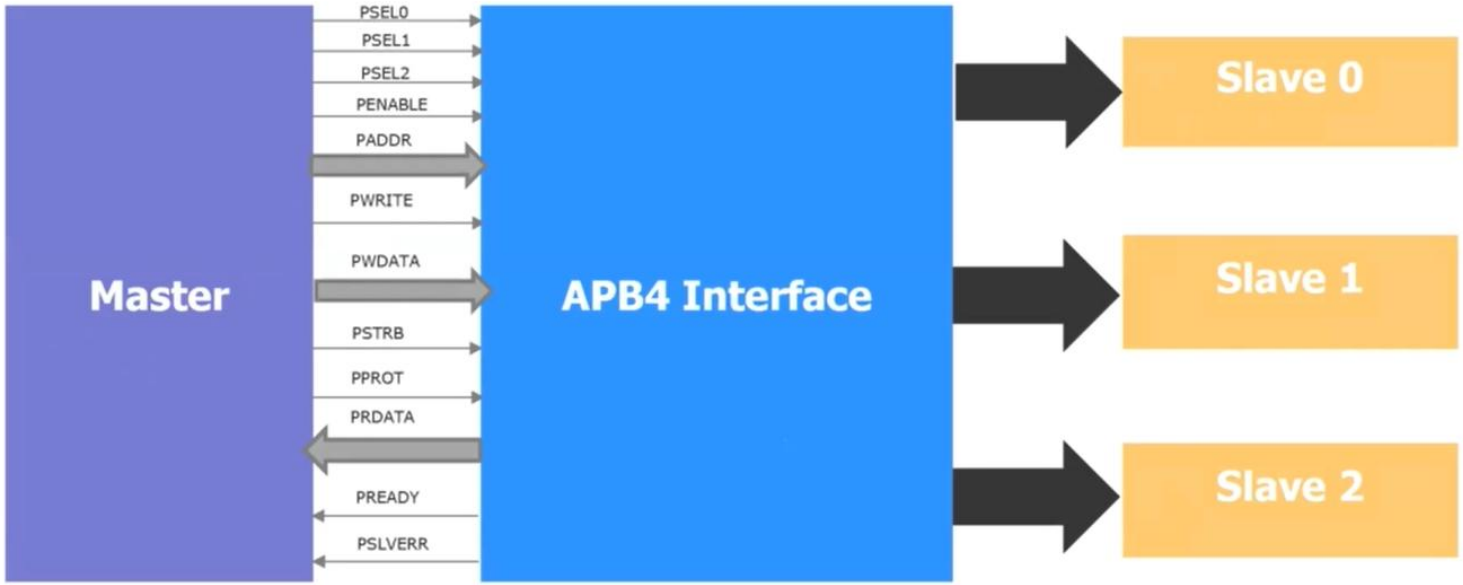
- ACE, as extension over AXI, used signal level communication between master-slave
  - Increased wires with additional snoop channel
  - Worked well for small coherent clusters
- With increased number of coherent clusters, AMBA5 CHI was introduced as a complete re-design of the ACE protocol.
- Uses packet based layered protocol


1. Part 1 - AMBA protocol evolution and APB2/3/4 protocol concepts overview
APB Protocol Concepts
1.5
46:08 / 1:03:20
CC

AMBA APB Versions

- AMBA2 APB (Called as APB2)
  - Basic read and write transfers
- AMBA3 APB 1.0 (called as APB3)
  - Wait states - A ready signal, 'PREADY', to extend an APB transfer.
  - Error reporting - An error signal, 'PSLVERR', to indicate the failure of a transfer.
- AMBA3 APB 2.0 (called as APB4 also)
  - Transaction protection – Protection signal to support secure and non-secure transactions on APB
  - Sparse data transfer – Write strobe signal to enable sparse data transfer on the write data bus.

AMBA APB4 Interface Architecture



APB Protocol Introduction

- APB stands for ‘Advanced Peripheral Bus’.
- The APB is part of Advanced Microcontroller Bus Architecture (AMBA) protocol family.
- Mainly used as general purpose register based peripheral such as timers, interrupt controllers, UARTs, IO ports etc.
- Optimized for minimal power consumption and reduced interface complexity.
- Non-pipelined protocol.
- APB interfaces with any peripherals that are low bandwidth and do not require the high performance of a pipelined bus interface.
- It is connected to the system bus via a bridge, helps in reducing system power consumption.
- Easy to interface.
- Provides a low cost interface that is optimized for minimal power consumption and reduced interface complexity.

APB Protocol Introduction (Cont..)

- APB protocol has 2 independent data buses, one for write data and one for read data
- Buses can be up to 32-bits wide.
- Buses do not support individual handshake signals, so not possible to perform write and read operations on both buses at same time.

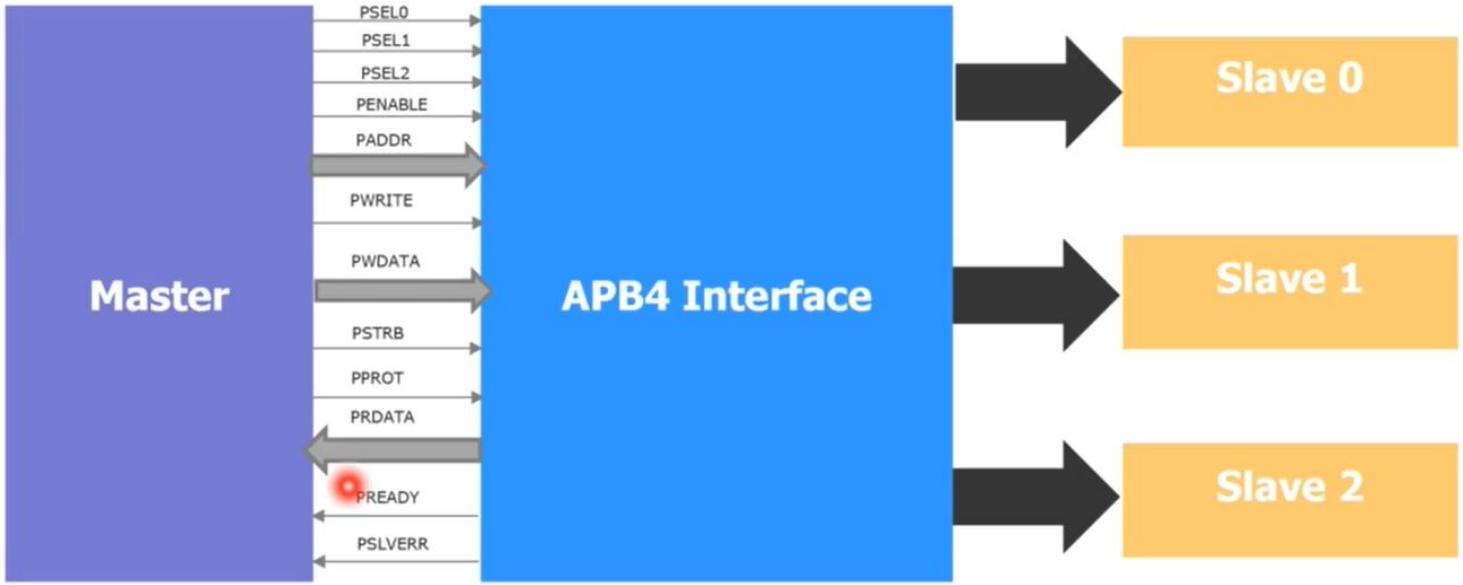
APB Protocol Introduction (Cont..)

- APB protocol has 2 independent data buses, one for write data and one for read data
- Buses can be up to 32-bits wide.
- Buses do not support individual handshake signals, so not possible to perform write and read operations on both buses at same time.
- Every transfer takes at least 2 cycles (one cycle which is called as Setup phase and second cycle which is called as access phase).

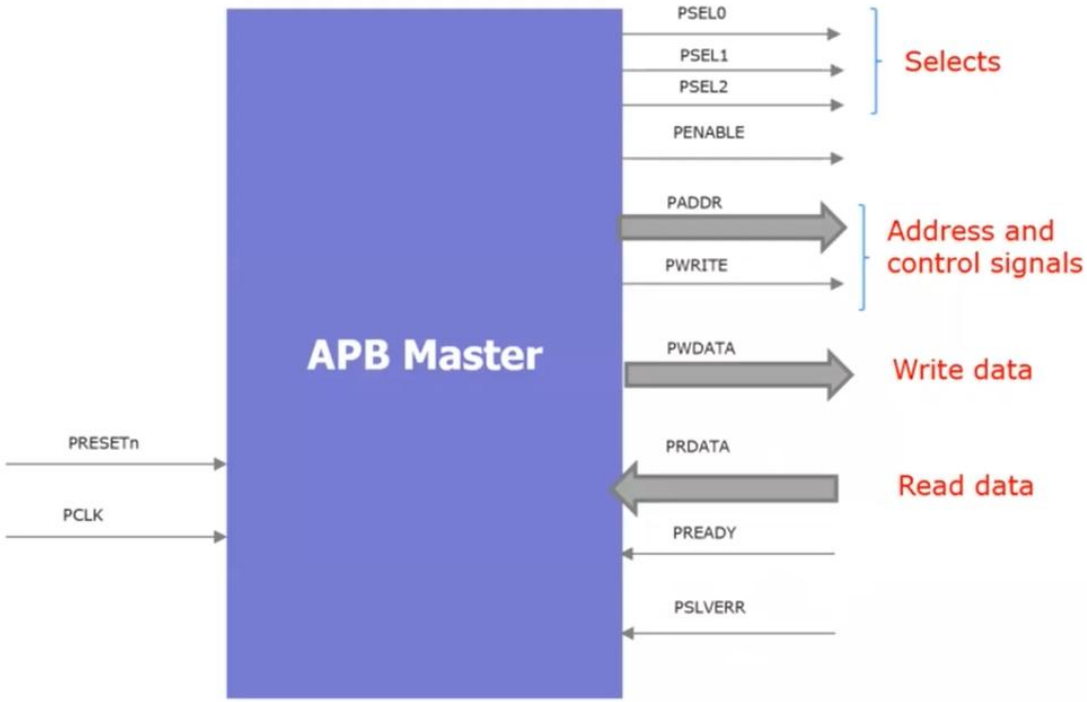
APB Interface signals

Signals	Source	Description
PCLK	Clock source	Clock
PRESETn	System bus equivalent	Reset. The APB reset signal is active LOW
PADDR	APB bridge	APB address bus, 32 bits wide
PPROT	APB bridge	Protection type. Indicates the normal/privileged, or secure/non-secure or data/instruction access.
PSELx	APB bridge	Slave select signal
PENABLE	APB bridge	Enable signal
PWRITE	APB bridge	Direction. When 1, write access else read acce
PWDATA	APB bridge	Write data bus, 32-bits wide.
PSTRB	APB bridge	Write strobes. One write strobe bit for each 8-bits or 1-byte of the write data bus. Write strobes must not be active during read transfer.
PREADY	Slave interface	Ready signal, slave used this signal to extend an APB transfer
PRDATA	Slave interface	Read data, 32-bits wide
PSLVERR	Slave interface	Indicates transfer failure, slave error response.

AMBA APB4 Interface Architecture



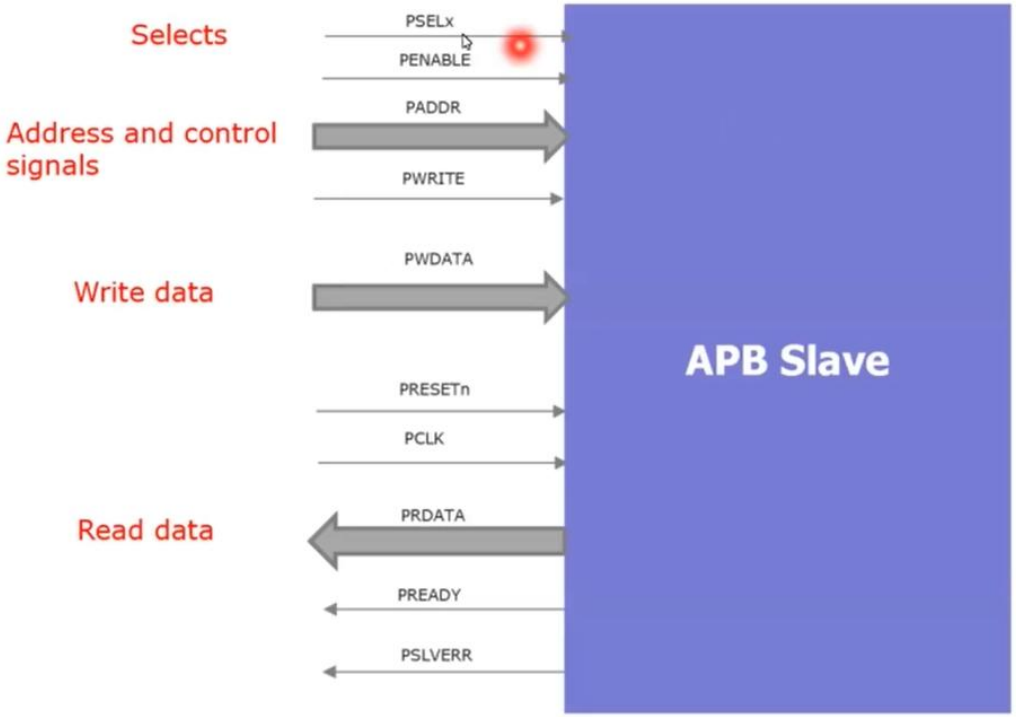
APB Master


APB Master
PSEL0
PSEL1
PSEL2
PENABLE
PADDR
PWRITE
PWDATA
PRDATA
PREADY
PSLVERR
PRESETn
PCLK
Selects
Address and control signals
Write data
Read data

APB Master

- If there is a single master on the APB, no need for an arbiter.
- Master drives the address and write buses and also performs a combinational decode of the address to decide which PSELx signal to activate.
- It is also responsible for driving the PENABLE to time the transfer.
- It drives APB data onto the system bus during a read transfer.

APB Slave

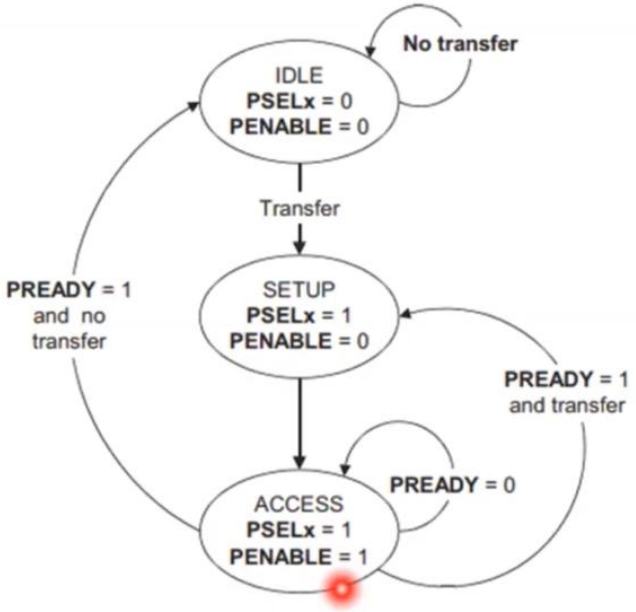

Selects
PSELx
PENABLE
Address and control signals
PADDR
PWRITE
Write data
PWDATA
APB Slave
PRESETn
PCLK
Read data
PRDATA
PREADY
PSLVERR

APB Slave

- APB slave have a very simple and flexible interface.
- The interface will be dependent on the design style employed and many different options are available.
- ‘PSLVERR’ and ‘PREADY’ are the two main signals which protect the lost data while transfer of data is taking place.

APB Operating States

- IDLE Phase
  - Default state of the APB
- SETUP Phase
  - When a transfer is required the bus moves into SETUP state, where the appropriate select signals PSELx is asserted.
  - Bus only remains in the SETUP state for one clock cycle and always moves to ACCESS state on the next rising edge of the clock.
  - Address, write, select and write data signals must remain stable during the transition from SETUP to ACCESS state.



State diagram

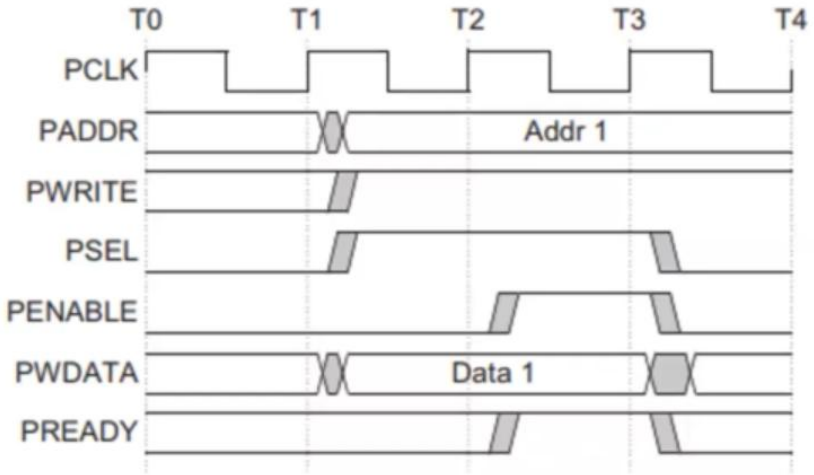
APB Operating States (Cont..)

- ACCESS Phase
  - The enable signal, PENABLE is asserted in ACCESS state.
  - Exit from the ACCESS state is controlled by the PREADY signal from the slave:
    - During 1st cycle of ACCESS phase, If 'PREADY' is held low by the slave then the peripheral bus remains in the ACCESS state.
    - During 2nd cycle of ACCESS phase, If 'PREADY' is driven high by the slave then the ACCESS state is exited and the bus returns to the IDLE state if no more transfers are required.
    - If there are another transfer followed, the bus moved directly into SETUP state.

APB Operating States (Cont..)

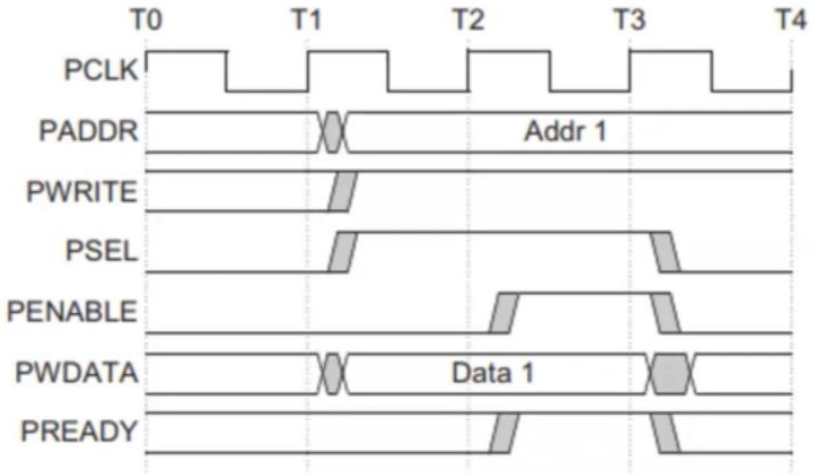
- ACCESS Phase
  - The enable signal, PENABLE is asserted in ACCESS state.
  - Exit from the ACCESS state is controlled by the PREADY signal from the slave:
    - During 1st cycle of ACCESS phase, If 'PREADY' is held low by the slave then the peripheral bus remains in the ACCESS state.
    - During 2nd cycle of ACCESS phase, If 'PREADY' is driven high by the slave then the ACCESS state is exited and the bus returns to the IDLE state if no more transfers are required.
    - If there are another transfer followed, the bus moved directly into SETUP state.

APB Write transfer with no wait states



Basic write transfer with no wait states

APB Write transfer with no wait states



Basic write transfer with no wait states

APB Write transfer with no wait states

- Write transfer starts with address, write data, write signal and select signals and all will be changing during the rising edge of the clock.
- The first clock cycle of the transfer is called the SETUP phase.

APB Write transfer with no wait states

- Write transfer starts with address, write data, write signal and select signals and all will be changing during the rising edge of the clock.
- The first clock cycle of the transfer is called the SETUP phase.
- After following clock edge, the enable signal PENABLE is asserted and it indicates that the ACCESS phase is taking place.
- The address, data and control signals all remain valid throughout the ACCESS phase. Transfer completes at the end of this cycle.
- The enable signal PENABLE, is de-asserted at the end of the transfer.

APB Write transfer with no wait states

- Write transfer starts with address, write data, write signal and select signals and all will be changing during the rising edge of the clock.
- The first clock cycle of the transfer is called the SETUP phase.
- After following clock edge, the enable signal PENABLE is asserted and it indicates that the ACCESS phase is taking place.
- The address, data and control signals all remain valid throughout the ACCESS phase. Transfer completes at the end of this cycle.
- The enable signal PENABLE, is de-asserted at the end of the transfer.
- The select signal PSELx, also goes low unless the transfer is to be followed immediately by another transfer to the same peripheral.

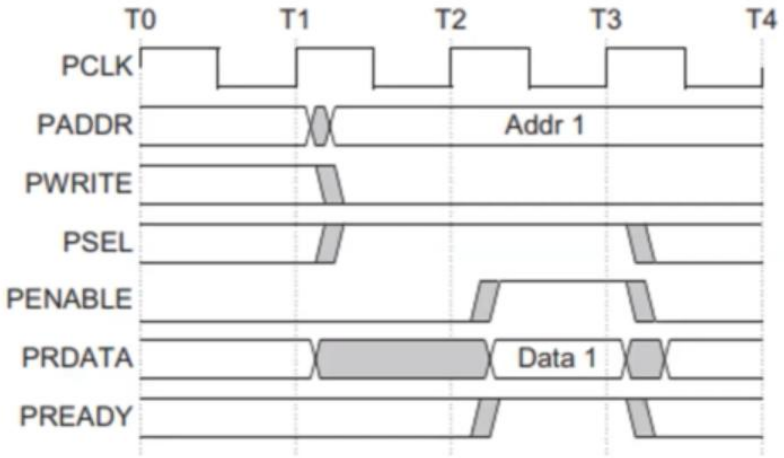
APB Write transfer with no wait states

- Write transfer starts with address, write data, write signal and select signals and all will be changing during the rising edge of the clock.
- The first clock cycle of the transfer is called the SETUP phase.
- After following clock edge, the enable signal PENABLE is asserted and it indicates that the ACCESS phase is taking place.
- The address, data and control signals all remain valid throughout the ACCESS phase. Transfer completes at the end of this cycle.
- The enable signal PENABLE, is de-asserted at the end of the transfer.
- The select signal PSELx, also goes low unless the transfer is to be followed immediately by another transfer to the same peripheral.

APB Write transfer with wait states

Write transfer with wait states

APB Read transfer with no wait states



Basic read transfer with no wait states

APB Read transfer with no wait states

- Similarly read transfer starts with address, write signal and select signals and all will be changing during the rising edge of the clock.
- The first clock cycle of the transfer is called the SETUP phase.
- After following clock edge, the enable signal PENABLE is asserted and it indicates that the ACCESS phase is taking place.
- The address, data and control signals all remain valid throughout the ACCESS phase. Transfer completes at the end of this cycle.

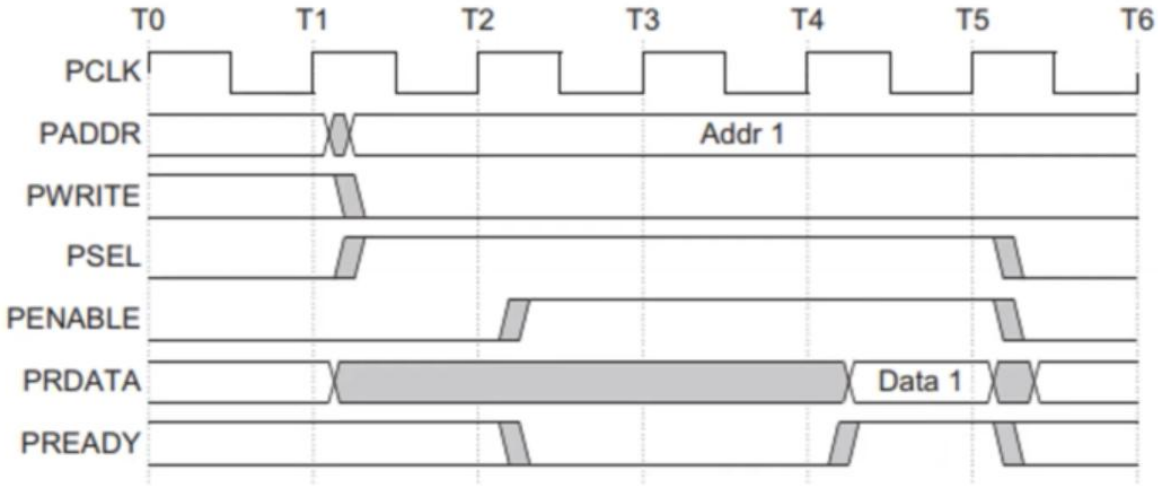
APB Read transfer with no wait states

- Similarly read transfer starts with address, write signal and select signals and all will be changing during the rising edge of the clock.
- The first clock cycle of the transfer is called the SETUP phase.
- After following clock edge, the enable signal PENABLE is asserted and it indicates that the ACCESS phase is taking place.
- The address, data and control signals all remain valid throughout the ACCESS phase. Transfer completes at the end of this cycle.
- The enable signal PENABLE, is de-asserted at the end of the transfer.

APB Read transfer with no wait states

- Similarly read transfer starts with address, write signal and select signals and all will be changing during the rising edge of the clock.
- The first clock cycle of the transfer is called the SETUP phase.
- After following clock edge, the enable signal PENABLE is asserted and it indicates that the ACCESS phase is taking place.
- The address, data and control signals all remain valid throughout the ACCESS phase. Transfer completes at the end of this cycle.
- The enable signal PENABLE, is de-asserted at the end of the transfer.
- The select signal PSELx, also goes low unless the transfer is to be followed immediately by another transfer to the same peripheral.

APB Read transfer with wait states



Read transfer with wait states

APB Read transfer with wait states

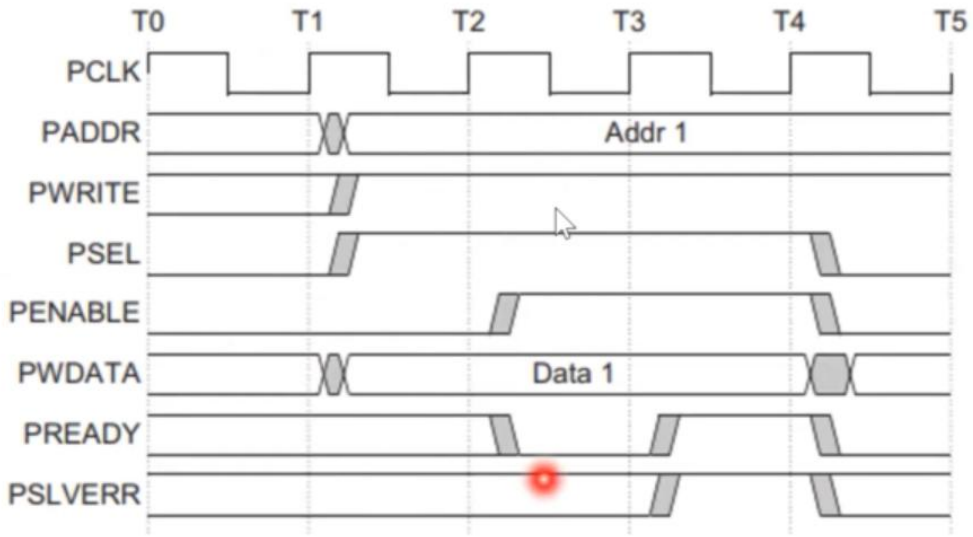
- As demonstrated in the Figure, how the 'PREADY' signal can extend the transfer.
- During an ACCESS phase, when PENABLE is high, the transfer can be extended by driving PREADY low.
- Following signals remains same for the additional cycles of ACCESS phase:
  - PADDR, address signal
  - PSEL, select signal
  - PWRITE, write signal
  - PENABLE, enable signal
  - PPROT, protection type

APB Error Response

- Use ‘PSLVERR’ to indicate an error condition on an APB transfer.
- Error condition can occur on both read and write transactions.
- ‘PSLVERR’ is only considered valid during the last cycle of APB transaction, when PSEL, PENABLE and PREADY all signals are high.
- Transactions that receive an error, might or might not have changed the state of the peripheral. It is peripheral specific.
- APB peripherals are not required to support the ‘PSLVERR’ pin. If peripheral does not include this pin then input to APB bridge can be tied to low.

APB Write transfer with Error Response

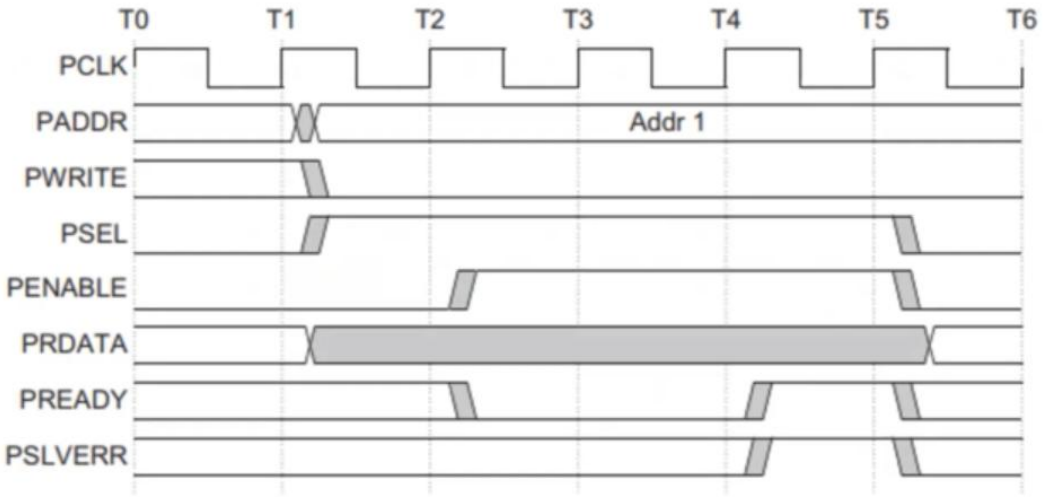
- When a write transaction receives an error, it does not mean that the register within the peripheral has not been updated.



Failing write transfer

APB Read transfer with Error Response

- Read transactions that receive an error can return invalid data. There is no requirement for the peripheral to drive the data bus to all 0s for a read error.



Failing read transfer

APB Write Strobes

- Represented by 'PSTRB' signal, Enables sparse data transfer on the write data bus.
- One write strobe bits for each 8-bit or 1-byte of write data bus.
- Bus master must drive all the bits of PSTRB to low in case of read transfer.

Byte lane mapping

APB Protection Unit

- Represented by 'PPROT' signal, mostly used for interconnect and other devices both in the system to provide protection against illegal transactions.
- 3-bit wide, PPROT[2:0] signal
- Main use of PPROT is for secure or non-secure transactions.
- Can be considered interpretations of PPROT[0] & PPROT[2] bits differently for different peripherals.

PPROT[2:0]	Protection Level
[0]	1 = Privileged access 0 = normal access
[1]	1 = non-secure access 0 = secure access
[2]	1 = instruction access 0 = data access

Protection encoding

Advantages

- Low power
- Latched address and control
- Simple interface
- Suitable for many peripherals

Disadvantages

- Single master – Limits parallelism
- Scalability – Performance suffers as bus is loaded.
- Single outstanding request – Poor throughput and multi threading performance bottleneck.

APB Sample Functional Coverage Plan

Title	Description	Covergroup:Coverpoint/Cross
Transfer direction	This cover point covers the possible values of PWRITE signal. There are two bins of corresponding values [APB3_READ] and [APB3_WRITE]	apb3_cov:xfer_type
Slave error response	This covers the PSLVERR signal values LOW and HIGH. Cover point slverr contains the two different bins, LOW and HIGH.	apb3_cov:slverr
Write transfers	This covers the different PSLVERR signal values LOW and HIGH for write transfer. Cross cover point slverr with xfer_type contains the different bins values.	apb3_cov:slverrXxfer_type
Read transfers	This covers the different PSLVERR signal values LOW and HIGH for read transfer. Cross cover point slverr with xfer_type contains the different bins values.	apb3_cov:slverrXxfer_type
PWRITE	This cover point covers the possible values of PWRITE signal. There are two bins of corresponding values [READ] and [WRITE]	apb3_cov:xfer_type
PSLVERR	This covers the PSLVERR signal values LOW and HIGH. Cover point slverr contains the two different bins, LOW and HIGH.	apb3_cov:slverr
PENABLE	This cover point covers the possible values of PENABLE signal. There are two bins of corresponding values [0] and [1].	apb3_cov:penable
PREADY	This covers the PREADY signal values LOW and HIGH. Cover point pready contains the two different bins, LOW and HIGH.	apb3_cov:pready
PSELx	This cover point covers all possible values of PSELx signal. Cover point slv_id have different bins corresponding to each slave id.	apb3_cov:slv_id
Transfer direction and slave error responses for different Slave ID	This cross cover point covers all possible values of PSELx signal with PSLVERR and PWRITE.	apb3_cov:slv_idXslverrXxfer_type

Outline

- AHB Bus
  - Introduction
  - Typical AMBA AHB Based System
  - Bus Interconnection
- AHB Components
- AHB Operation
  - Overview
  - Pipelined transfers
  - Transfer type
  - Other control signals
  - Slave Responses
    - Slave transfer responses
    - Two cycle response
    - Split and retry response

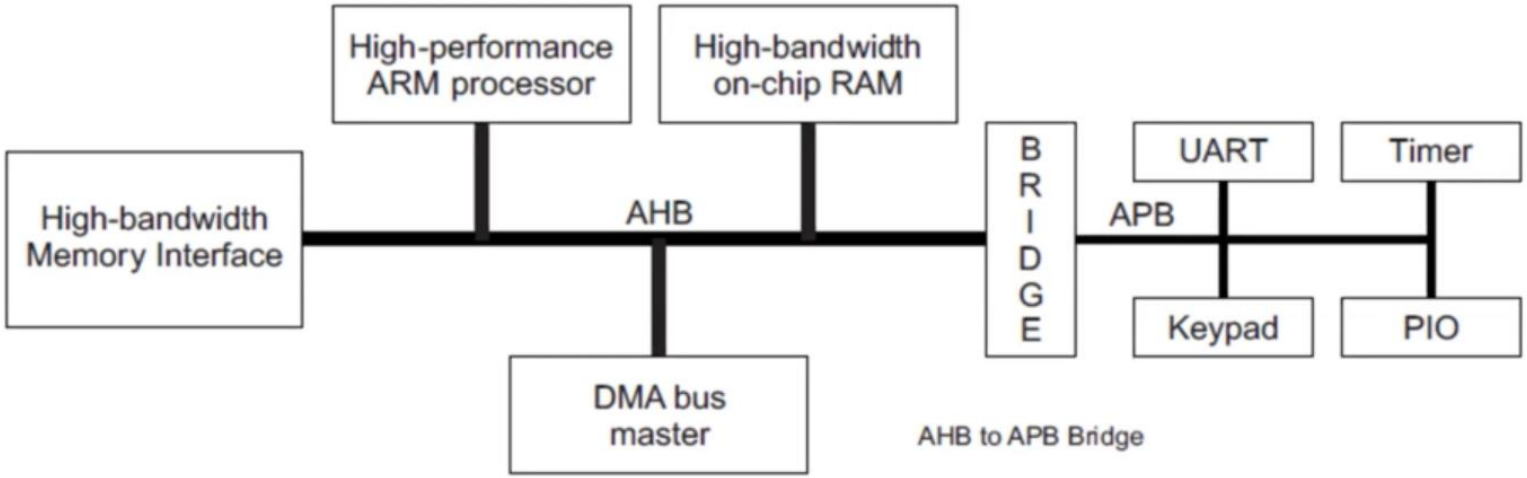
Outline

- AHB Bus
  - Introduction
  - Typical AMBA AHB Based System
  - Bus Interconnection
- AHB Components
- AHB Operation
  - Overview
  - Pipelined transfers
  - Transfer type
  - Other control signals
  - Slave Responses
    - Slave transfer responses
    - Two cycle response
    - Split and retry response
- AHB Arbitration
- AHB-Lite/AHB5 differences
- AHB Verification Perspective
  - Verification Environment
  - Assertions
  - Coverage

Introduction

- AHB is a generation of AMBA bus
  - — Intended to address requirements of high-performance synthesizable designs
  - — Supports multiple bus masters & slaves
  - — Provides high-bandwidth operation
- AMBA AHB implements the following features required for high-performance:
  - — Burst transfers
  - — Split transactions
  - — Single-cycle bus master handover
  - — Single-clock edge operation
  - — Wider data bus configurations (64/128 bits).
  - — Non tri state implementation

Typical AMBA AHB Based system

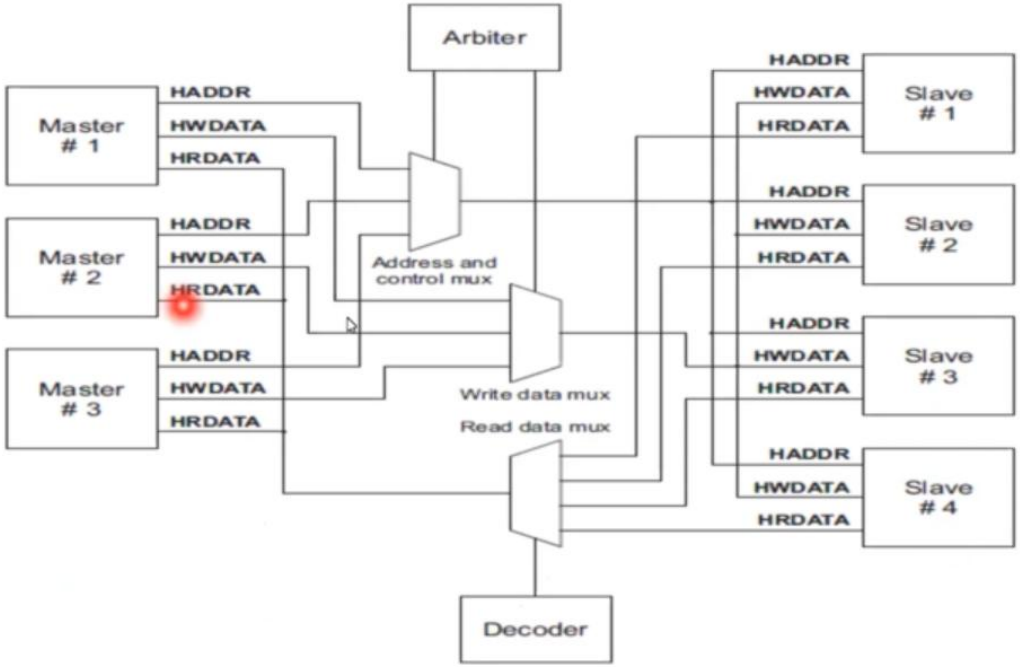


AHB to APB Bridge

AMBA Advanced High-performance Bus (AHB)
* High performance
* Pipelined operation
* Burst transfers
* Multiple bus masters
* Split transactions

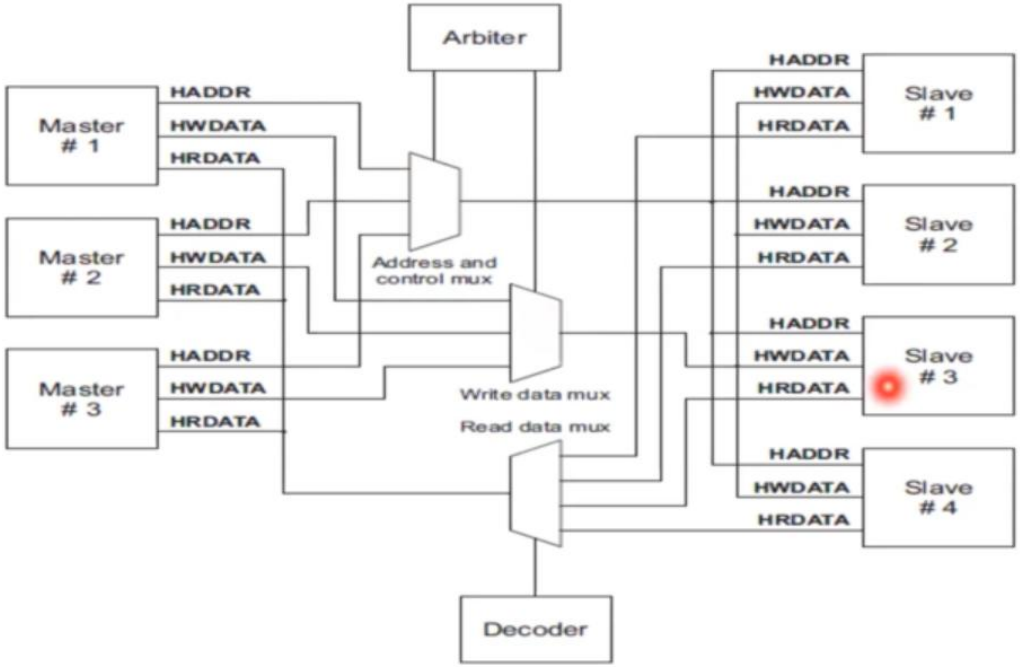
AMBA Advanced Peripheral Bus (APB)
* Low power
* Latched address and control
* Simple interface
* Suitable for many peripherals

Bus Interconnection



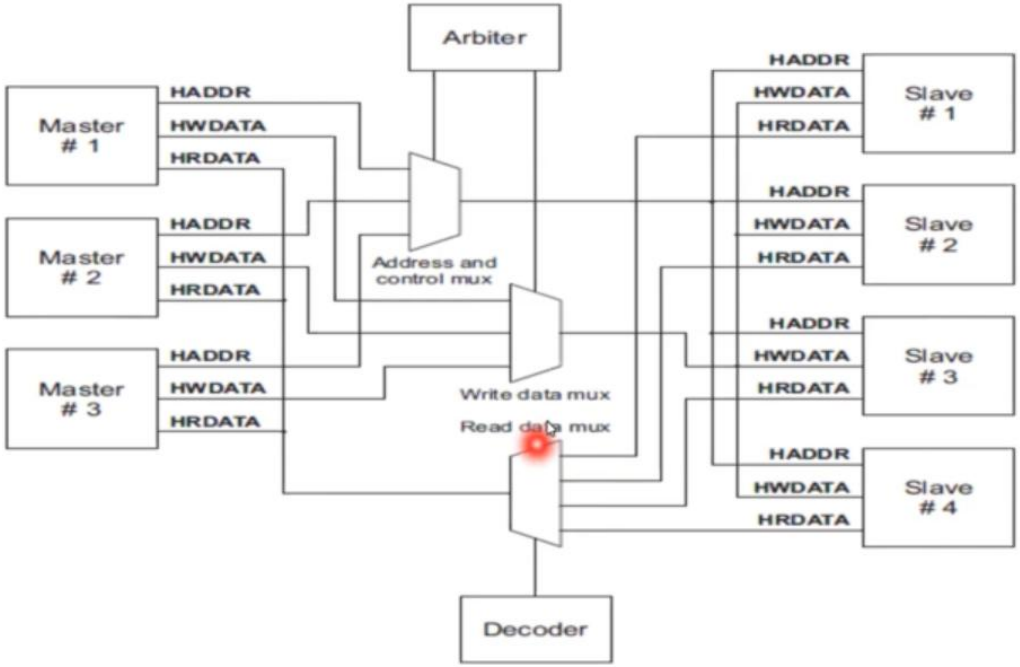
Bus Interconnection

- Two MUXs controlled by arbiter for Write Data and Address



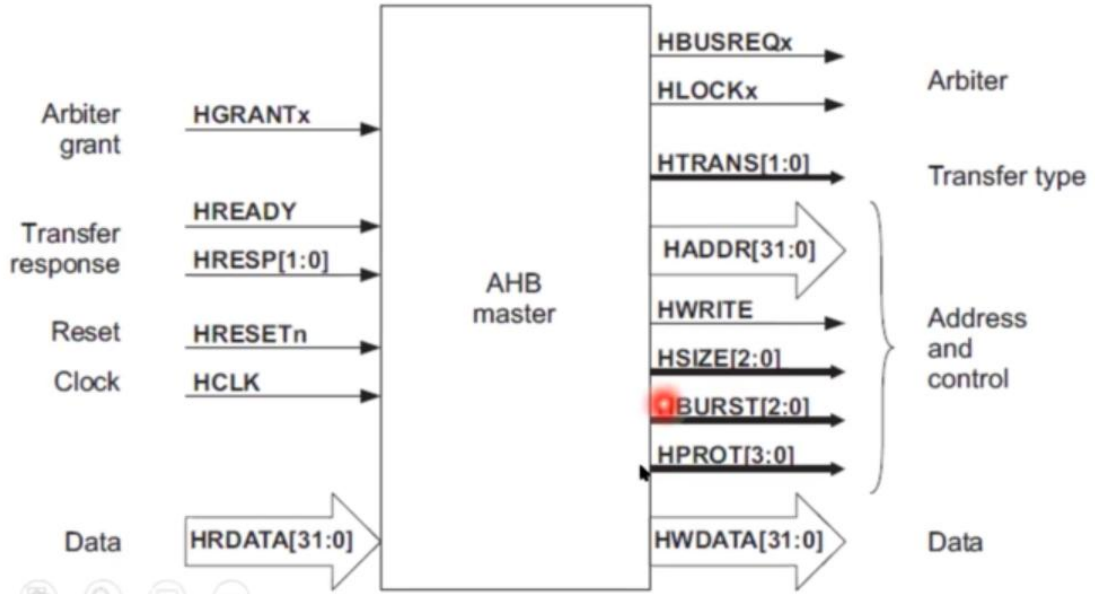
Bus Interconnection

- Two MUXs controlled by arbiter for Write Data and Address
- One MUX controlled by central decoder for Read Data



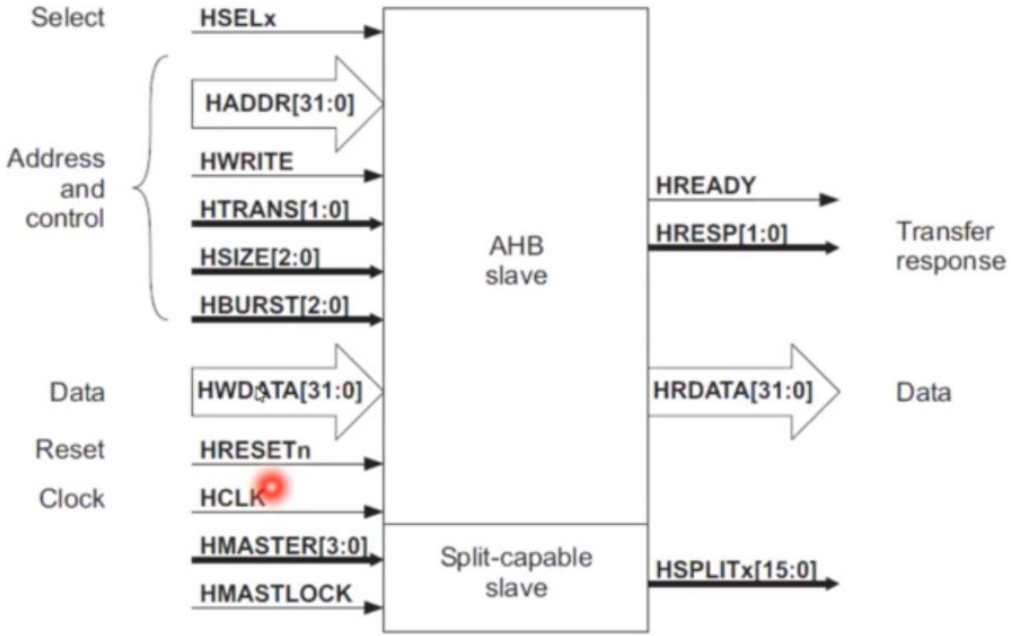
Components in AHB

- Master
  - — Initiates read/write operations by providing address and control information
  - — Only 1 bus master allowed to actively use the bus at any one time
  - — Max. of 16 masters allowed



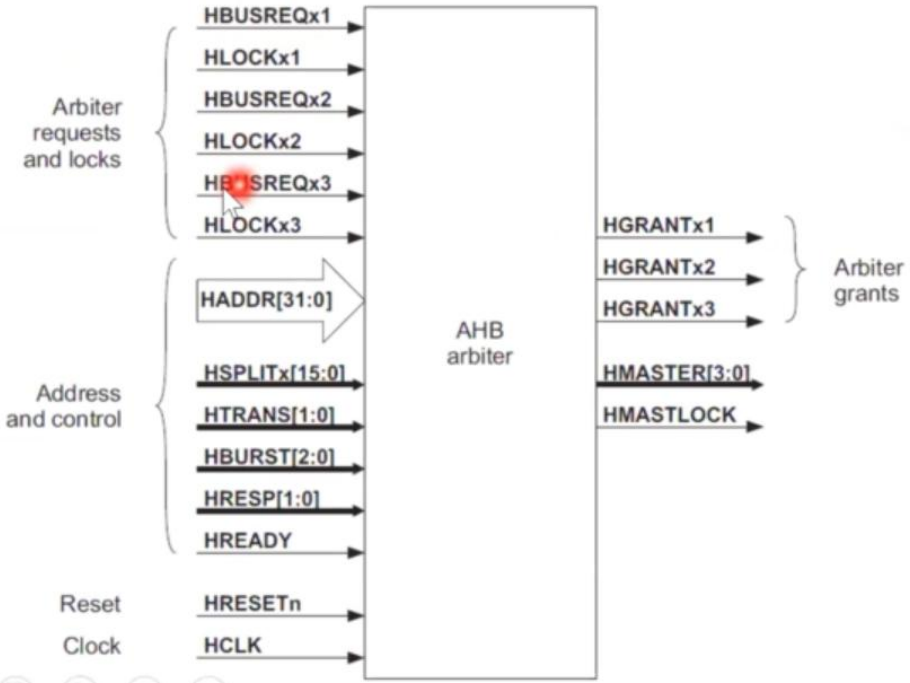
Components in AHB (contd...)

- Slave
  - — Responds to read/write operation
  - — Signals back to active master success, failure or waiting of data transfer.



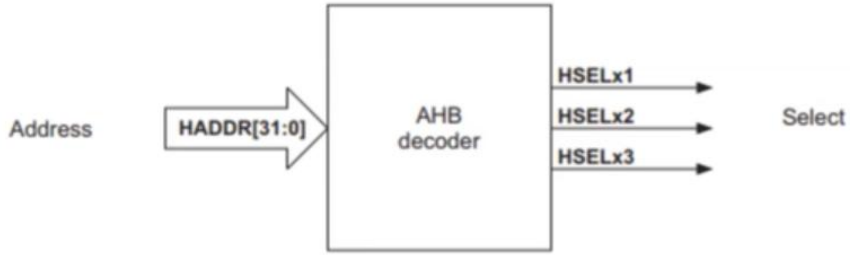
Components in AHB (contd...)

- Arbiter
  - Controls which bus master has access to bus
  - Selects active master based on the used prioritization scheme (Default Round-robin)



Components in AHB (contd…)

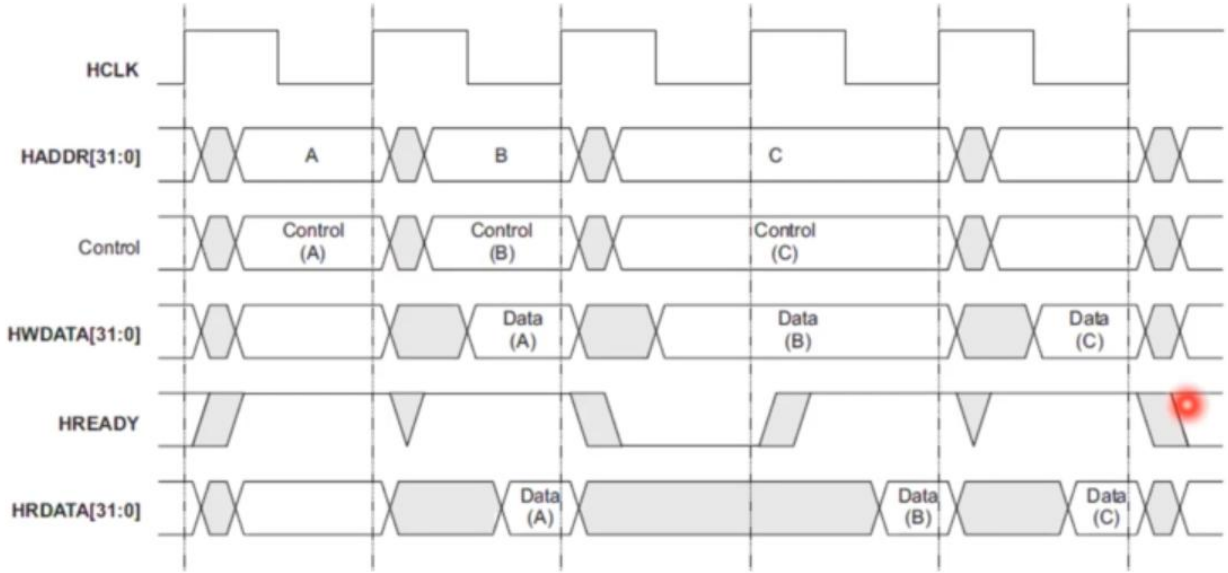
- Decoder
  - Centralized address decoding function.



Overview of operation

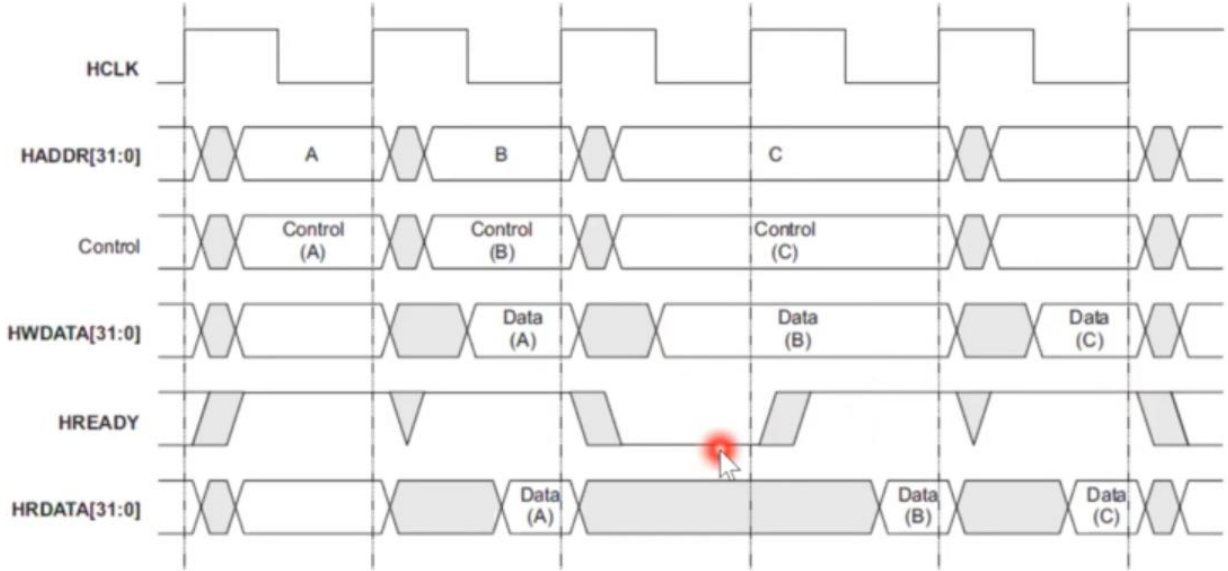
- Master requires bus access grant for a transfer
  - It asserts request signals to request grant from arbiter
  - Starts the transfer after grant is provided by arbiter
- AHB Bus transfer
  - Address and Data phases of different transactions overlapped
  - Address Phase
    - Single cycle for master to drive address and control signals
    - Provide information related address, direction, width of transfer and burst type
  - Data Phase
    - One or more cycles
    - Controlled by HREADY from slave
    - Write/Read data transfer to/from slave
    - Response provided by slave using HRESP[1:0]
      - OKAY
      - ERROR
      - RETRY OR SPLIT

AHB pipelined transfers



AHB pipelined transfers

- Overlap between Address and data phases of consecutive transfers
- A & C – zero wait state
- B – 1 wait state (extending C's address phase)



Transfer type

- 4 possible transfer types indicated by HTRANS[1:0]:
  - IDLE (00)
    - No data transfer is required
    - Transfer ignored by slave
    - It always provides zero wait state OKAY response
  - BUSY (01)
    - Allow master to insert IDLE cycle in the middle of bursts of transfers
    - Transfer ignored by slave
    - It always provides zero wait state OKAY response
  - NONSEQ (10)
    - Indicate the first transfer of a burst or a single transfer
  - SEQ (11)
    - Indicate a remaining transfer in a burst
    - Address related to previous transfer
    - Control information identical to previous transfer

Other Control signals

- Burst types: HBURST [2:0] signal
  - — 4/8/16 beat bursts are defined along with undefined length bursts and single transfer.
  - — INCR and WRAP burst are supported.

HBURST[2:0]	Type	Description
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing burst

  - — Incrementing Bursts
    - Access sequential location and address of each transfer in the burst is just an increment of previous address
    - For example: A 4-beat increment burst of word size (32-bit or 4-byte) access, If starts address is 0x20, further address would be 0x24, 0x28, 0x2C

Other Control signals

- Burst types: HBURST [2:0] signal
  - 4/8/16 beat bursts are defined along with undefined length bursts and single transfer.
  - INCR and WRAP burst are supported.

HBURST[2:0]	Type	Description
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing burst

  - Incrementing Bursts
    - Access sequential location and address of each transfer in the burst is just an increment of previous address
    - For example: A 4-beat increment burst of word size (32-bit or 4-byte) access, If starts address is 0x20, further address would be 0x24, 0x28, 0x2C
    - An INCR burst can be of any length, but the upper limit is set by the fact that the address must not cross 1KB boundary.

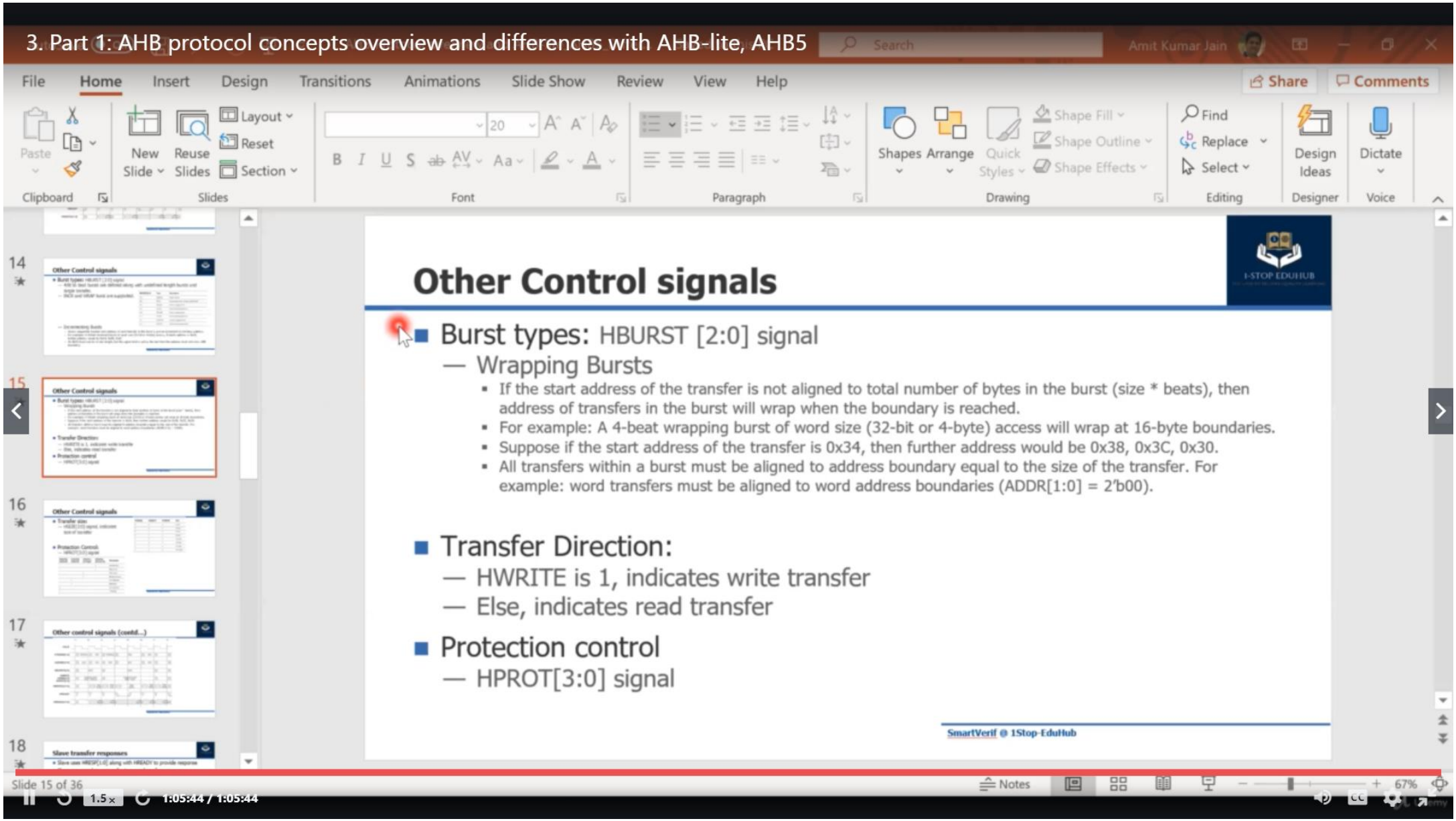
Other Control signals

- Burst types: HBURST [2:0] signal
  - 1KB = 1024B = 2*10B
  - 10 addr bits
  - 0 - 3FF
  - 0x20 (align this address with 1 KB)
  - 0 - 3FF
  - 3F0
  - INCR8
  - 3F0 3F4 3F8 3FC 400
  - ined along with undefined length bursts and
  - supported.

HBURST[2:0]	Type	Description
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing burst

- Incrementing Bursts
  - Access sequential location and address of each transfer in the burst is just an increment of previous address
  - For example: A 4-beat increment burst of word size (32-bit or 4-byte) access, If starts address is 0x20, further address would be 0x24, 0x28, 0x2C
  - An INCR burst can be of any length, but the upper limit is set by the fact that the address must not cross 1KB boundary.

3. Part 1: AHB protocol concepts overview and differences with AHB-lite, AHB5



Other Control signals

- Burst types: HBURST [2:0] signal
  - — Wrapping Bursts
    - If the start address of the transfer is not aligned to total number of bytes in the burst (size * beats), then address of transfers in the burst will wrap when the boundary is reached.
    - For example: A 4-beat wrapping burst of word size (32-bit or 4-byte) access will wrap at 16-byte boundaries.
    - Suppose if the start address of the transfer is 0x34, then further address would be 0x38, 0x3C, 0x30.
    - All transfers within a burst must be aligned to address boundary equal to the size of the transfer. For example: word transfers must be aligned to word address boundaries (ADDR[1:0] = 2'b00).

WRAP4, 0x34, size=4 bytes,
Total bytes = 4*4 = 16
Aligned address = (0x34 - (0x34%16)) = 0x30
Wrap Boundary = Aligned address + total bytes
= 0x30 + 16 bytes = 0x3F
0x34 0x38 0x3C 0x30

Other Control signals

- Burst types: HBURST [2:0] signal
  - — Wrapping Bursts
    - If the start address of the transfer is not aligned to total number of bytes in the burst (size * beats), then address of transfers in the burst will wrap when the boundary is reached.
    - For example: A 4-beat wrapping burst of word size (32-bit or 4-byte) access will wrap at 16-byte boundaries.
    - Suppose if the start address of the transfer is 0x34, then further address would be 0x38, 0x3C, 0x30.
    - All transfers within a burst must be aligned to address boundary equal to the size of the transfer. For example: word transfers must be aligned to word address boundaries (ADDR[1:0] = 2'b00).

WRAP4, 0x34, size=4 bytes,
Total bytes = 4*4 = 16
Aligned address = (0x34 - (0x34%16)) = 0x30
Wrap Boundary = Aligned address + total bytes
= 0x30 + 16 bytes = 0x3F
0x34 0x38 0x3C 0x30

Other Control signals

- Burst types: HBURST [2:0] signal
  - — Wrapping Bursts
    - If the start address of the transfer is not aligned to total number of bytes in the burst (size * beats), then address of transfers in the burst will wrap when the boundary is reached.
    - For example: A 4-beat wrapping burst of word size (32-bit or 4-byte) access will wrap at 16-byte boundaries.
    - Suppose if the start address of the transfer is 0x34, then further address would be 0x38, 0x3C, 0x30.
    - All transfers within a burst must be aligned to address boundary equal to the size of the transfer. For example: word transfers must be aligned to word address boundaries (ADDR[1:0] = 2'b00).
- Transfer Direction:
  - — HWRITE is 1, indicates write transfer
  - — Else, indicates read transfer
- Protection control

Other Control signals

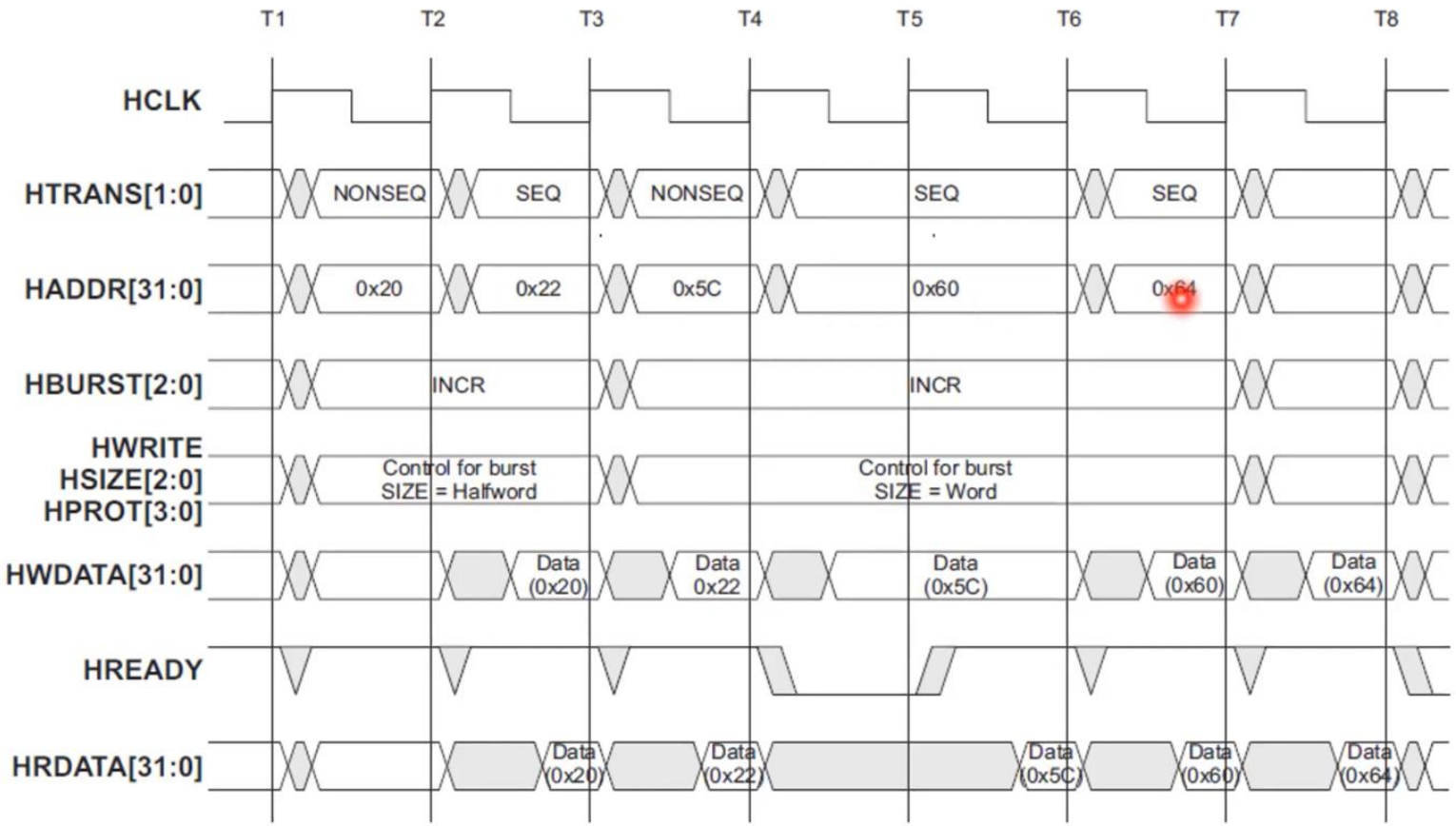
- Transfer size:
  - HSIZE[2:0] signal, indicates size of transfer

HSIZE[2]	HSIZE[1]	HSIZE[0]	Size
0	0	0	8 bits
0	0	1	16 bits
0	1	0	32 bits
0	1	1	64 bits
1	0	0	128 bits
1	0	1	256 bits
1	1	0	512 bits
1	1	1	1024 bits

- Protection Control:
  - HPROT[3:0] signal

HPROT[3] cacheable	HPROT[2] bufferable	HPROT[1] privileged	HPROT[0] data/opcode	Description
-	-	-	0	Opcode fetch
-	-	-	1	Data access
-	-	0	-	User access
-	-	1	-	Privileged access
-	0	-	-	Not bufferable
-	1	-	-	Bufferable
0	-	-	-	Not cacheable
1	-	-	-	Cacheable

Other control signals (contd…)

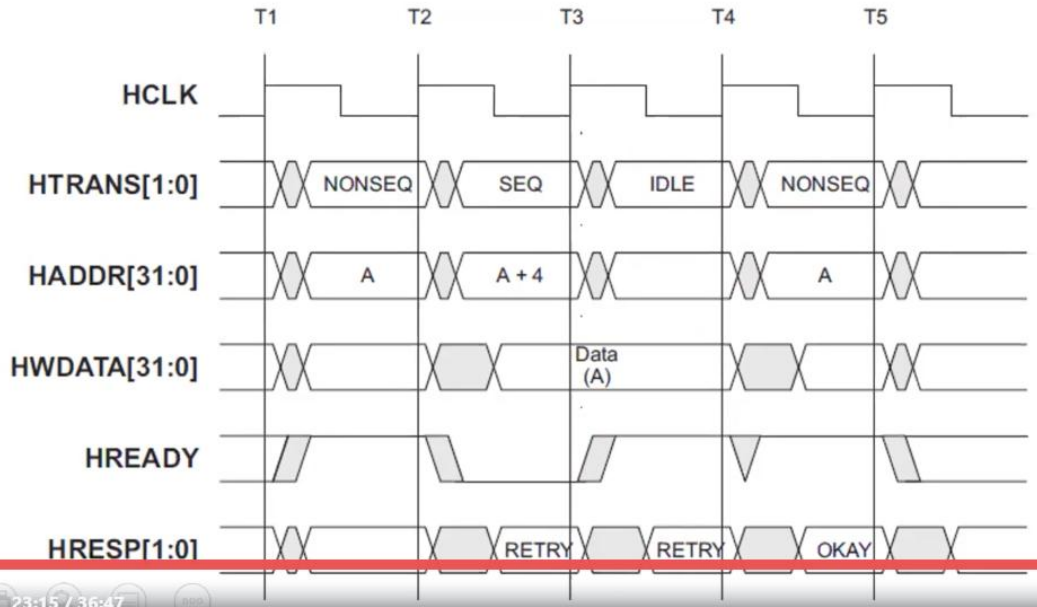


Slave transfer responses

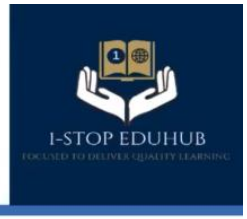
- Slave uses HRESP[1:0] along with HREADY to provide response
- Slave can complete a transfer in a number of ways:
  - Complete transfer immediately
    - OKAY response provided (HRESP = 2'b00)
    - HREADY high and okay response
  - Insert one or more wait states
    - HREADY low used to introduce wait states
  - Signal an error
    - ERROR response provided (HRESP = 2'b01)
    - Two cycle response
  - Delay completion but allow bus to be released
    - SPLIT/RETRY response provided
    - Two cycle response

Two cycle response

- ERROR, SPLIT and RETRY use a two cycle response
  - Allows master to cancel the following transfer
  - HREADY low in first and high in second cycle
  - HRESP[1:0] indicating required ERROR, SPLIT or RETRY response in both cycles



Split and retry responses

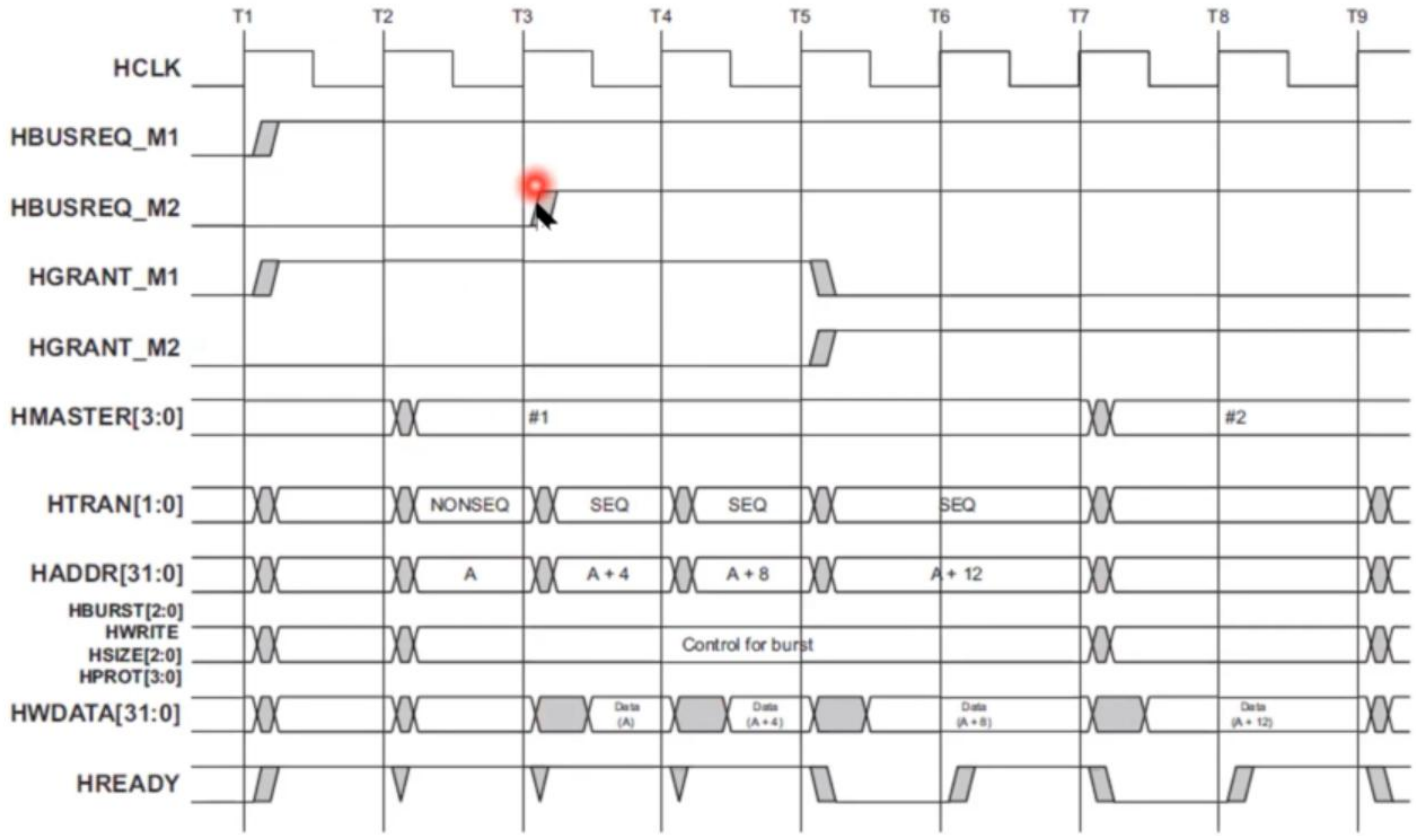


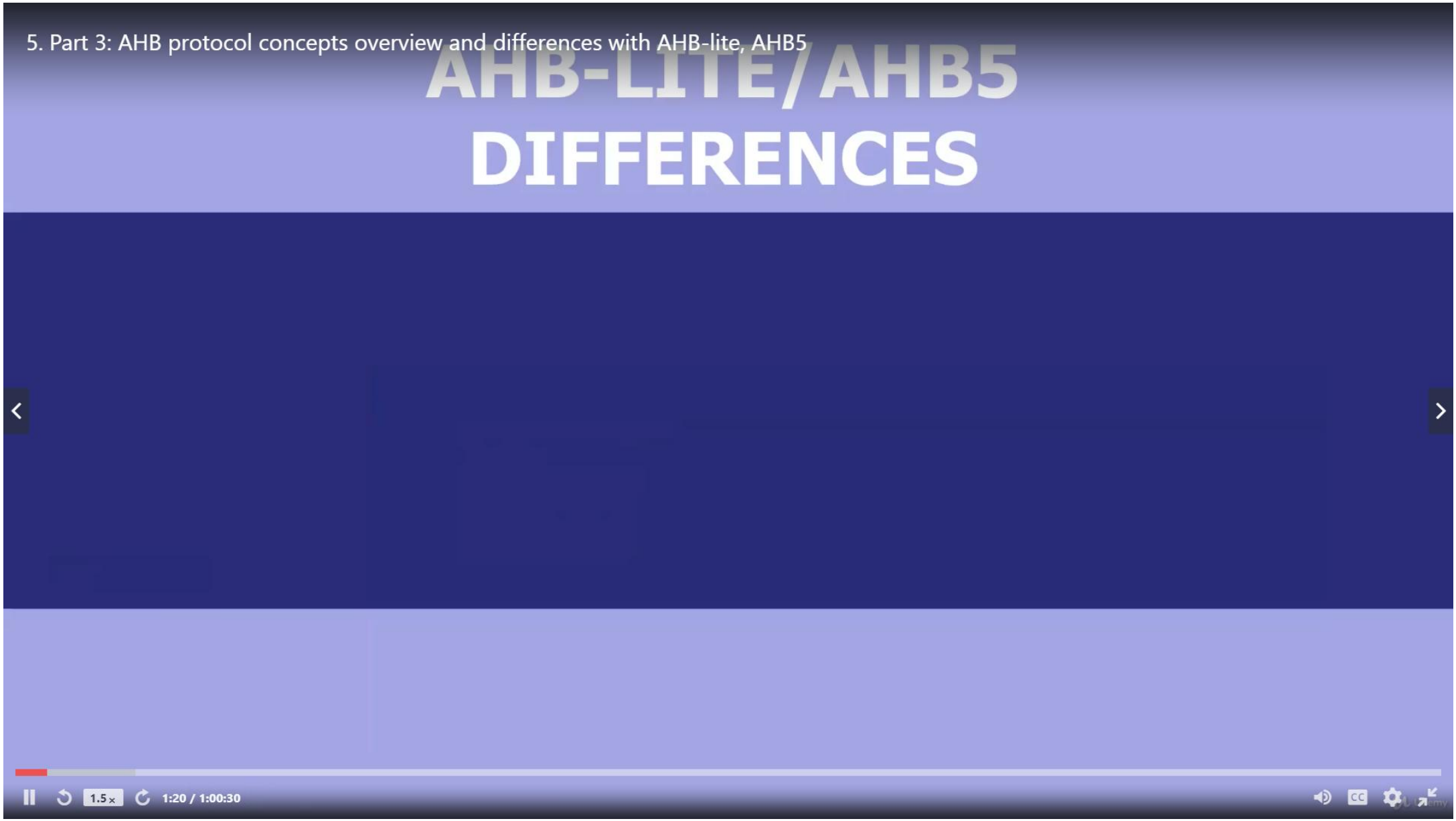
- Provide mechanism for slave to release bus for other transfers
- Both allow transfer to finish and therefore allow a higher priority master to gain access
- Difference is in the way arbiter allocates bus after these
  - RETRY
    - Arbiter continues to use normal priority scheme after this
  - SPLIT
    - Any other master will gain access
    - This is true even if it is lower in priority than the master for which SPLIT was issued
    - Transfer started again only after corresponding HSPLITx signal from the slave
    - Slave thus records the HMASTER[3:0] for each of the SPLIT responses

Arbitration

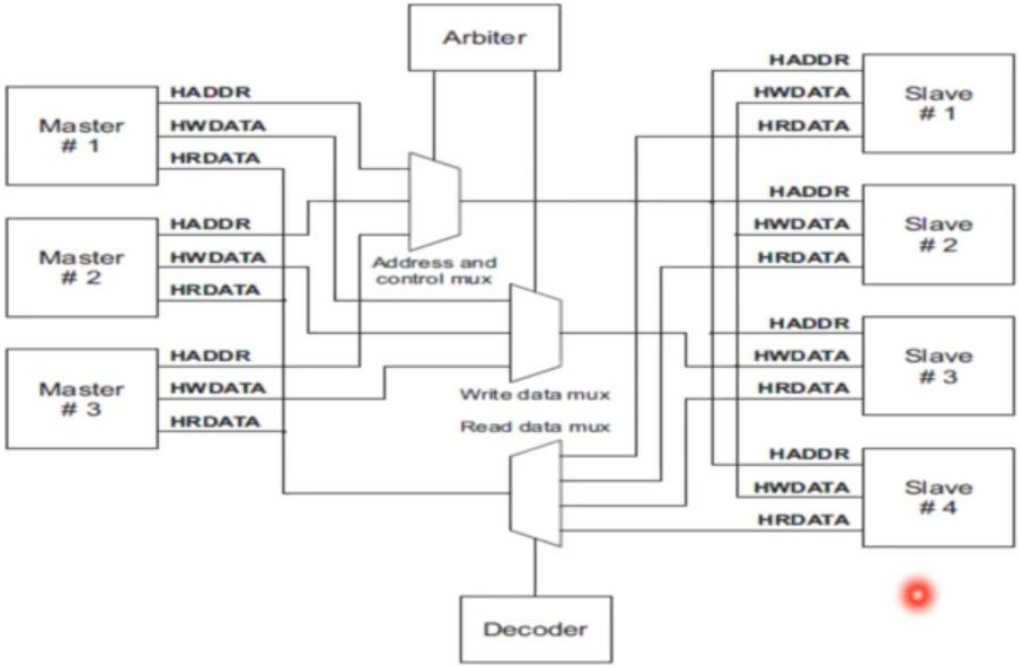
- Used to ensure only one master has bus access
- Master uses HBUSREQx to request bus access
- HLOCKx with this indicates exclusive transfer and arbiter shall not grant bus to another master in this case
- Master not required to hold HBUSREQx high after grant
  - Though required in case of undefined length bursts
- HMASTER[3:0] used by arbiter to indicate which master has bus access
- In case bus is granted but not requested then "IDLE" transfer should be done

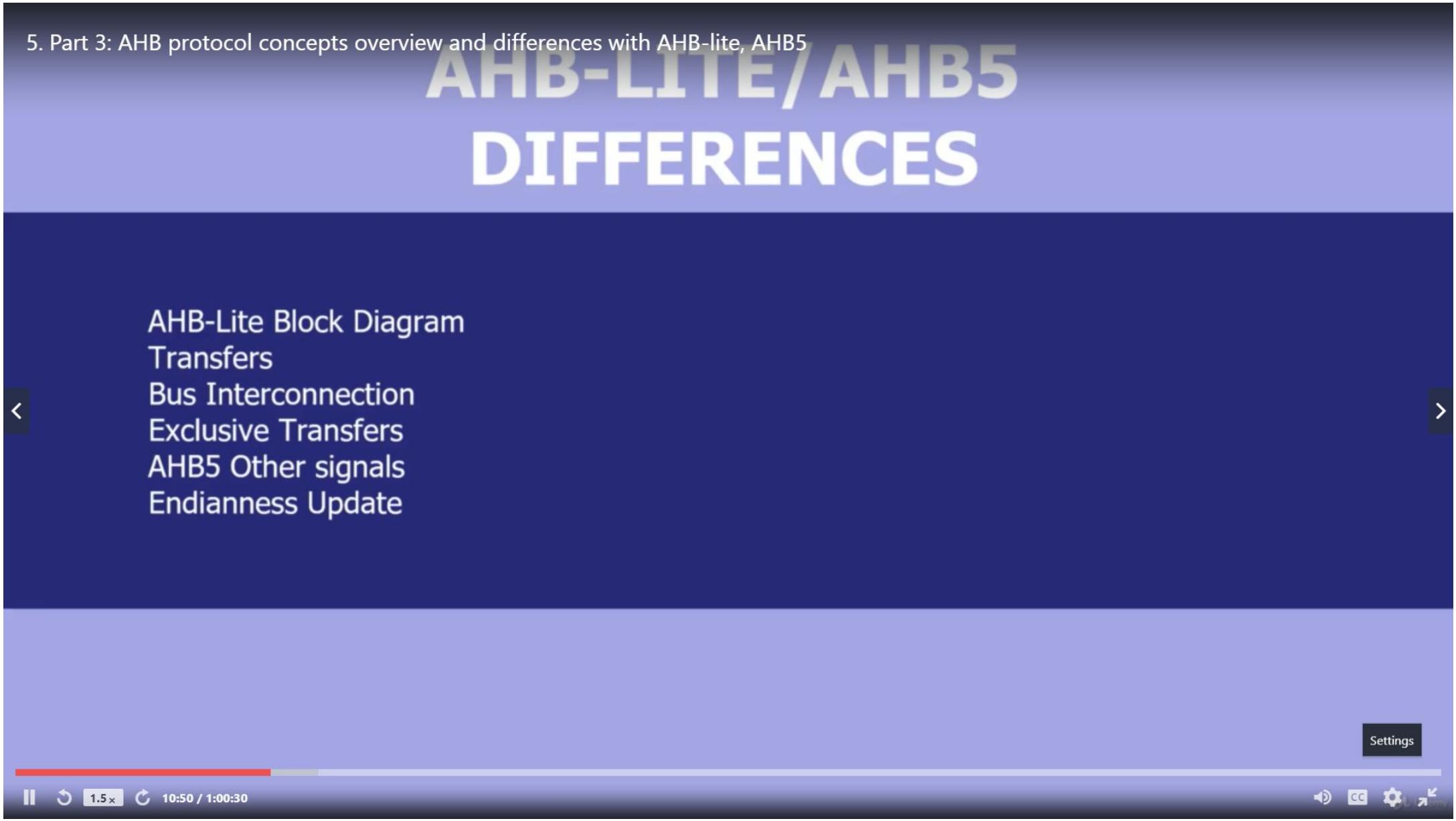
Arbitration (contd...)




5. Part 3: AHB protocol concepts overview and differences with AHB-lite, AHB5
AHB-LITE/AHB5
DIFFERENCES
1.5×
1:20 / 1:00:30
CC

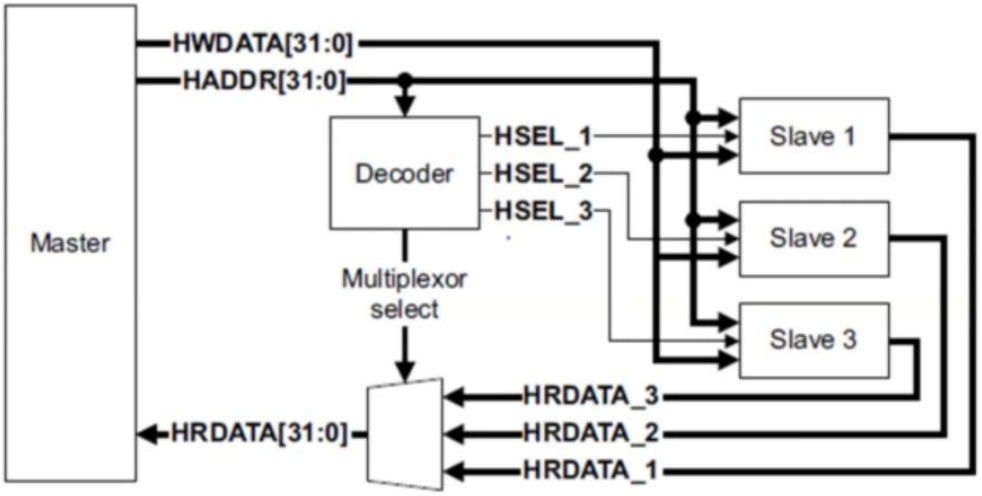
Bus Interconnection




5. Part 3: AHB protocol concepts overview and differences with AHB-lite, AHB5
AHB-LITE/AHB5
DIFFERENCES
AHB-Lite Block Diagram
Transfers
Bus Interconnection
Exclusive Transfers
AHB5 Other signals
Endianness Update
Settings
1.5×
10:50 / 1:00:30
CC

AHB-Lite block diagram

- Defines system for a single master and multiple slaves
- Decoder provides select signal to slaves and control signals to MUX.
- Mux used to route response from slaves to master

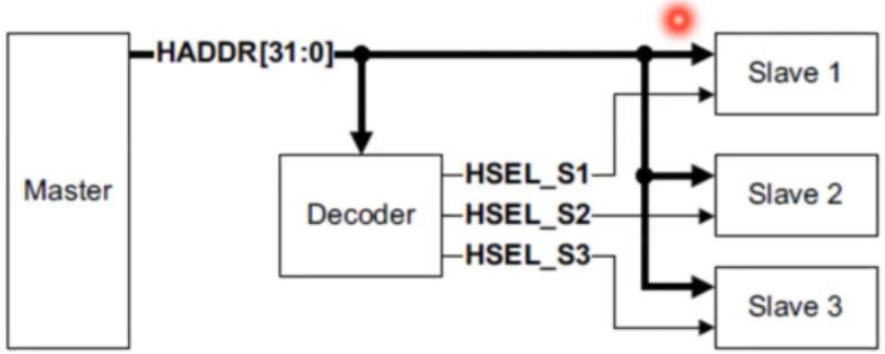


Transfers

- Similar to transfers done in AHB
- No Request/Grant handshake required, as only single master is there
- All sizes, transfer types and burst types are same as in AHB
- Locked transfers can be done through HMASTLOCK signal
  - Indicates slave that current transfer is indivisible
  - All transfer in a locked sequence required to be in same slave address region
- Slave uses HREADYOUT to insert wait states
- Only OKAY and ERROR responses are supported by slave
  - Error response has the same two-cycle timing as in AHB

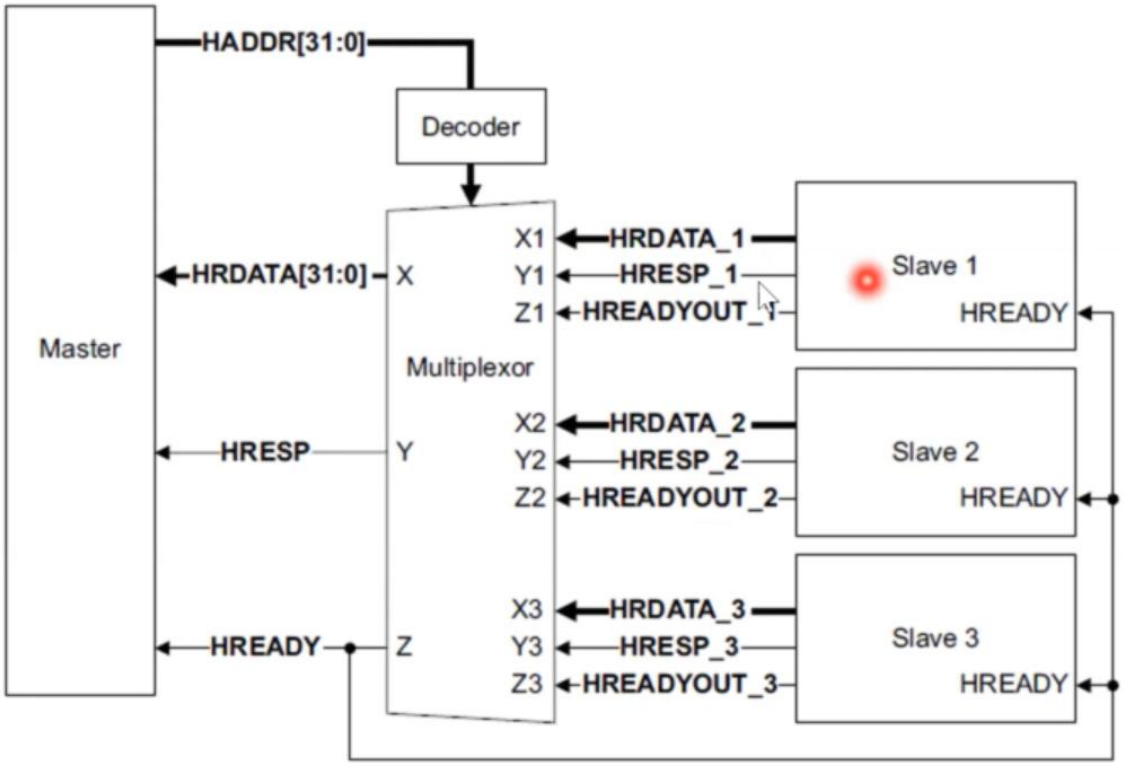
Bus Interconnection

- Address decoding
  - — Select signal
    - A decode of high order address bits
    - Only when HREADY is active
  - — Default slave
    - Provides response when non-existent memory location accessed
    - Error response for SEQ & NON-SEQ transfers
    - Zero wait state OKAY response for IDLE or BUSY transfers



Bus Interconnection (contd...)

- Read data and response mux



Exclusive transfers(Added in AHB5)

- Performs read-modify-write operations

Addr = 0
D1
D1^D2 = D3
Write D3 to address 0

T1 - Read address 0
Read data is D1, modify it to make it D3
T2 - Write back modified data D3 to addr 0

Between T1 and T2, other master modified address 0 location to make data from D1 -> D4

Exclusive transfers(Added in AHB5)

- Performs read-modify-write operations
- Memory location should not be updated between exclusive read and write
  - Exclusive transfer fails if this happens
- Response to Exclusive write indicates pass or fail
  - Memory location only updated if passed
- Exclusive additional signals
  - HEXCL
    - Indicates if transfer is an exclusive transfer
    - Timing same as address phase

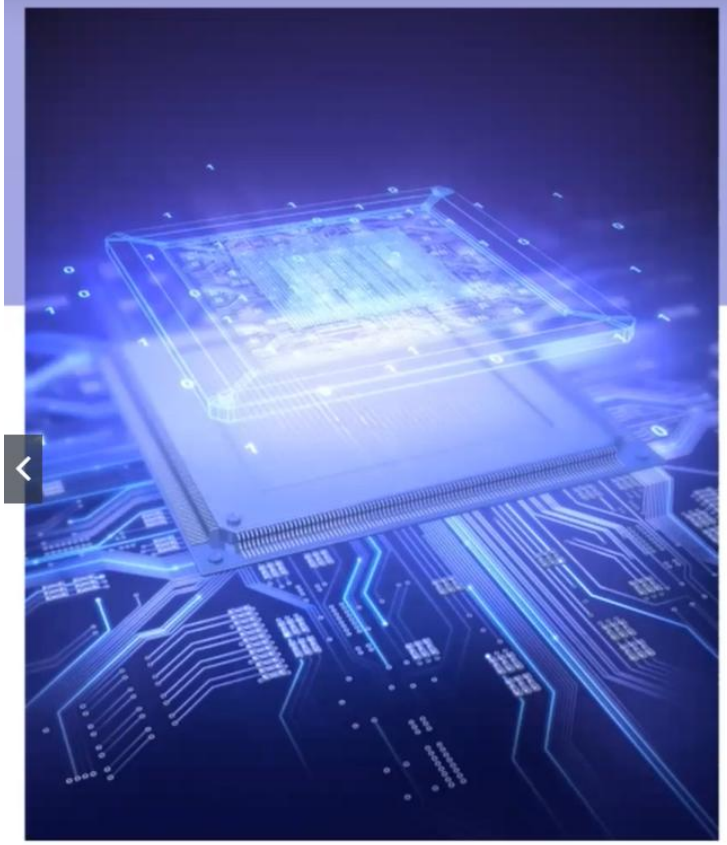
Exclusive transfers(Added in AHB5)

- Performs read-modify-write operations
- Memory location should not be updated between exclusive read and write
  - — Exclusive transfer fails if this happens
- Response to Exclusive write indicates pass or fail
  - — Memory location only updated if passed
- Exclusive additional signals
  - — HEXCL
    - – Indicates if transfer is an exclusive transfer
    - – Timing same as address phase
  - — HMASTER[m:0] - Master Identifier
    - – Used by master to differentiate multiple Exclusive threads that it can isuue

AHB5 other signals

- HPROT[5:0]
  - — Extended from a 4 bit to a 6 bit signal
- HNONSEC
  - — Indicates Non-secure transfer access if asserted
  - — Timing same as address and control signals
- User signaling
  - — User defined signals for each channel are added
    - – HAUSER : address channel user signals
    - – HWUSER : write data channel user signals
    - – HRUSER : read data channel user signals

6. Part 1: AXI3 protocol concepts overview and differences with AXI4lite, AXI4



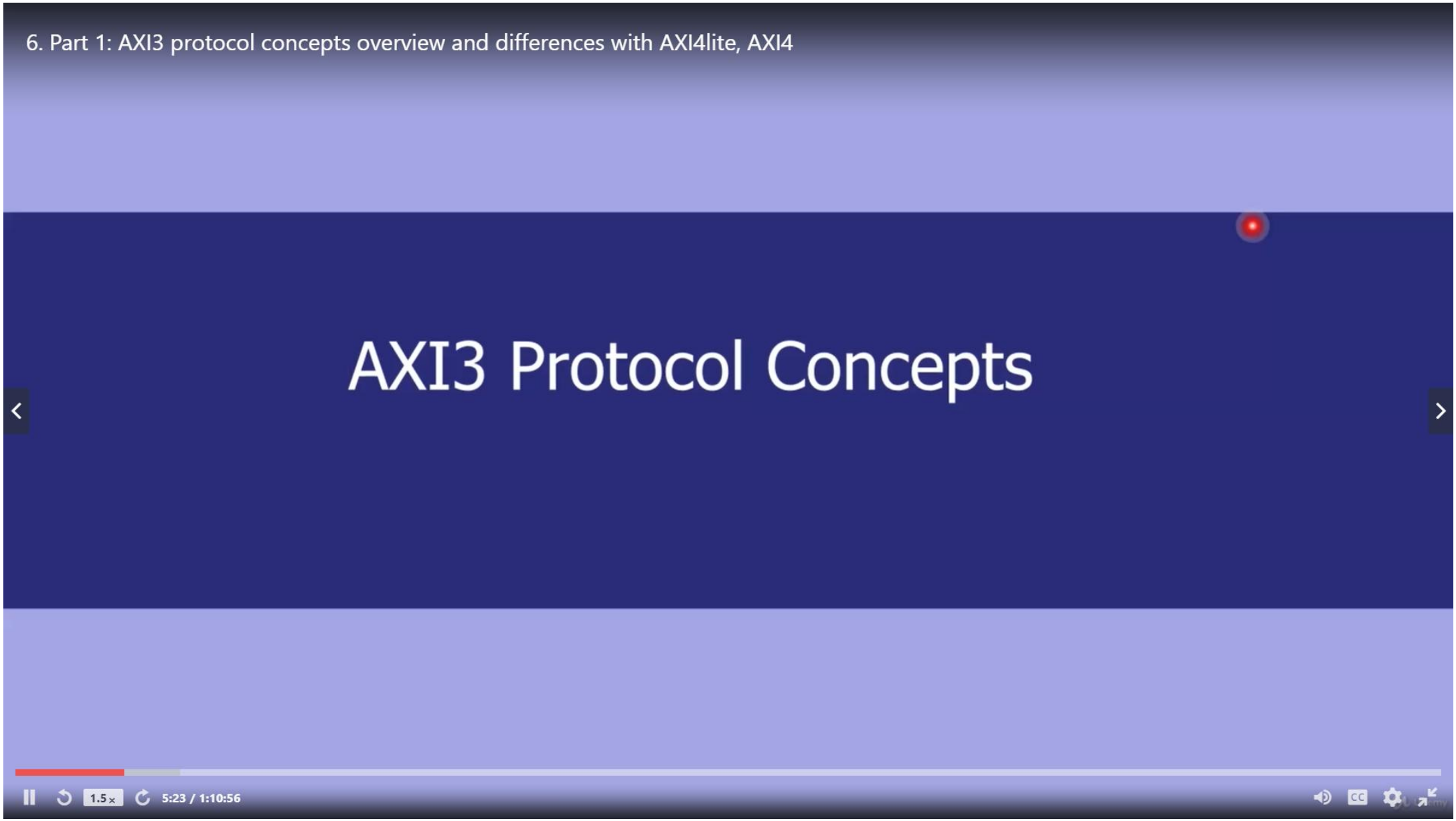
AXI3 protocol overview and differences with AXI4, AXI4-lite

Team SmartVerif @ 1Stop-EduHub

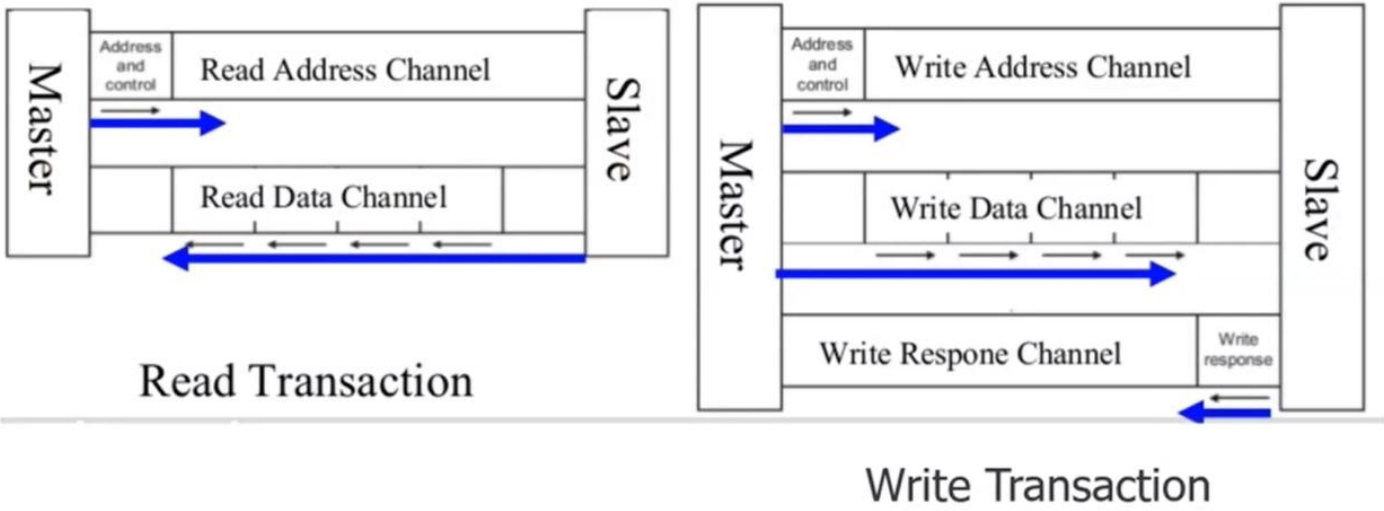
Jan 26, 2020

Outline

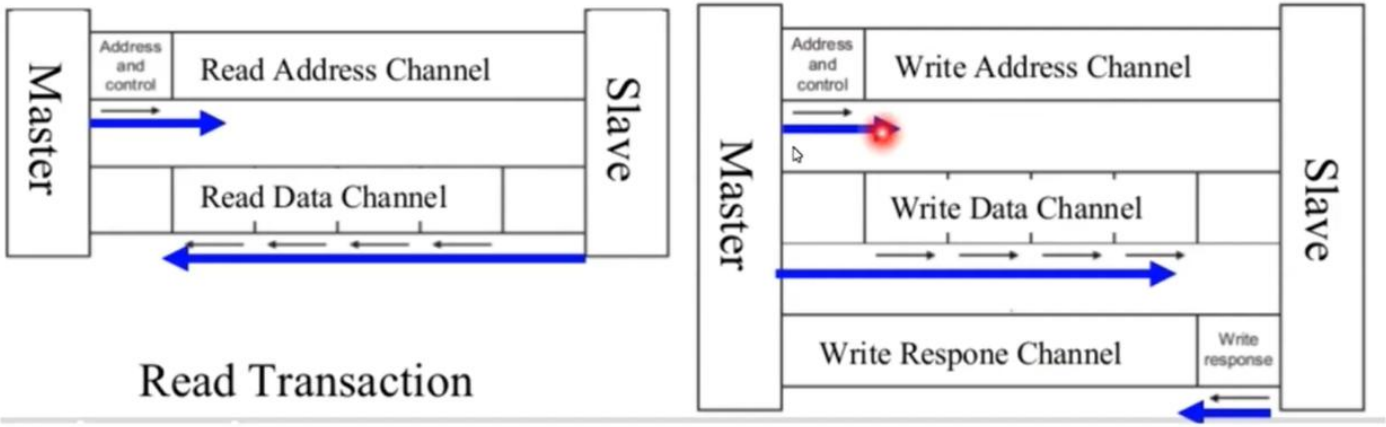
- AXI3
  - Introduction to AXI channels
  - Read channel ordering rules
  - Write channel ordering rules
  - Valid-Ready handshake mechanism
  - Transaction attributes
  - Locked, Exclusive transactions
  - AXI responses
- AXI4 – differences with AXI3
- AXI4-lite – differences with AXI4
- Verification perspective
  - Category of coverage items
  - Sample assertions


6. Part 1: AXI3 protocol concepts overview and differences with AXI4lite, AXI4
AXI3 Protocol Concepts
1.5×
5:23 / 1:10:56
CC

AXI Channels



AXI Channels



AXI Interface Signal List

- Global Signals

Signals	Source	Description
ACLK	Clock source	Global clock signal
ARESETn	System bus equivalent	Global reset signal, active low

AXI Interface Signal List

- Write address channel signals

Signals	Source	Description
AWID	Master	Write address ID. Identification tag for the write address group of signals
AWADDR	Master	Write address
AWLEN	Master	Burst length, Indicates the exact number of transfers in a burst
AWSIZE	Master	Burst size, Indicates the size of each transfer in the burst
AWBURST	Master	Burst type, burst type and the size information, determine how the address for each transfer within the burst is calculated
AWLOCK	Master	Lock type. Provides additional information about the atomic characteristics of the transfer.
AWCACHE	Master	Memory type. This signal indicates how transactions are required to progress through a system.
AWPROT	Master	Protection type. This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
AWVALID	Master	Write address valid. This signal indicates that the channel is signaling valid write address and control information.
AWREADY	Slave	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals.

AXI Interface Signal List

- Write data channel signals

Signals	Source	Description
WID	Master	Write ID tag. This signal is the ID tag of the write data transfer.
WDATA	Master	Write data
WSTRB	Master	Write strobes. This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
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.
WREADY	Slave	Write ready. This signal indicates that the slave can accept the write data.

AXI Interface Signal List

- Write response channel signals

Signals	Source	Description
BID	Slave	Response ID tag. This signal is the ID tag of the write response.
BREP	Slave	Write response. This signal indicates the status of the write transaction.
BVALID	Slave	Write response valid. This signal indicates that the channel is signaling a valid write response.
BREADY	Master	Response ready. This signal indicates that the master can accept a write response.

AXI Interface Signal List

- Read address channel signals

Signals	Source	Description
ARID	Master	Read address ID. Identification tag for the read address group of signals
ARADDR	Master	Read address
ARLEN	Master	Burst length, Indicates the exact number of transfers in a burst
ARSIZE	Master	Burst size, Indicates the size of each transfer in the burst
ARBURST	Master	Burst type, burst type and the size information, determine how the address for each transfer within the burst is calculated
ARLOCK	Master	Lock type. Provides additional information about the atomic characteristics of the transfer.
ARCACHE	Master	Memory type. This signal indicates how transactions are required to progress through a system.
ARPROT	Master	Protection type. This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
ARVALID	Master	Read address valid. This signal indicates that the channel is signaling valid read address and control information.
ARREADY	Slave	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals.

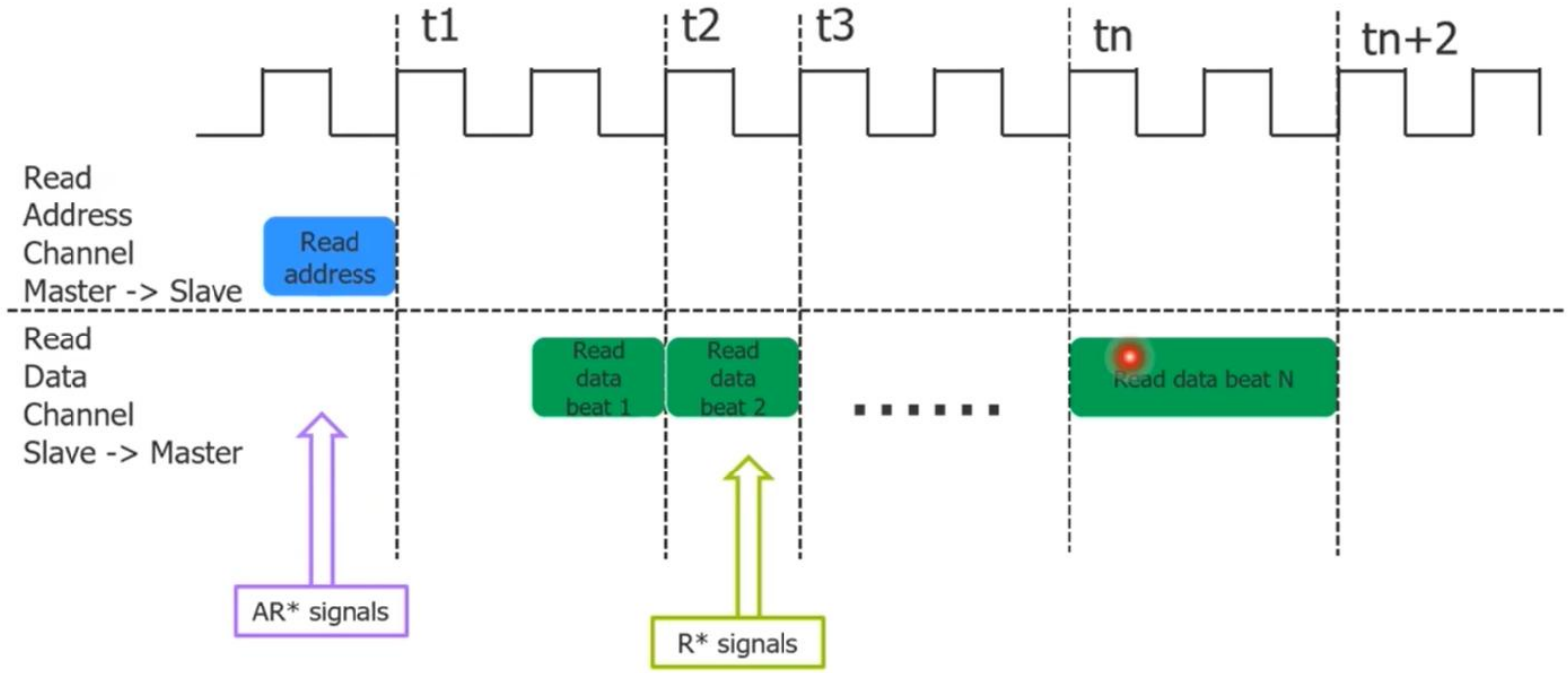
AXI Interface Signal List

- Read data channel signals

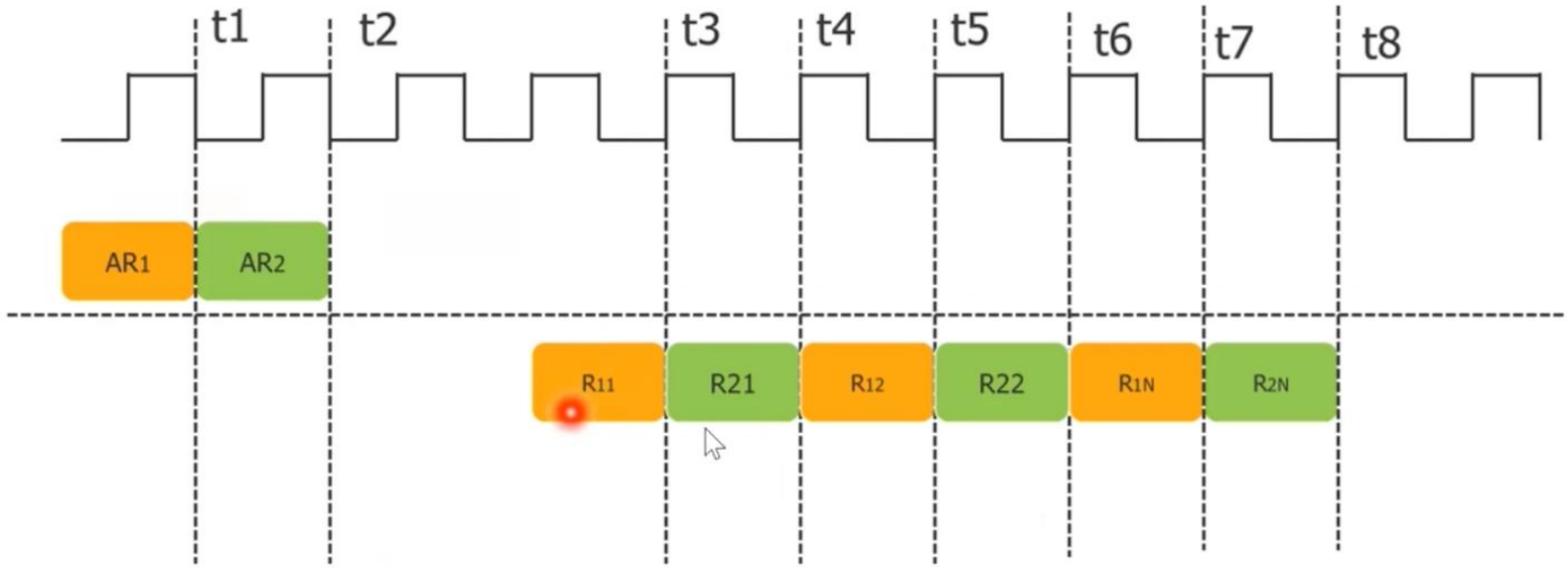
Signals	Source	Description
RID	Slave	Read ID tag. This signal is the ID tag of the read data transfer.
RDATA	Slave	Read data
RRESP	Slave	Read response. This signal indicates the status of the read transfer.
RLAST	Slave	Read last. This signal indicates the last transfer in a read burst.
RVALID	Slave	Read valid. This signal indicates that the channel is signaling the required read data.
RREADY	Master	Read ready. This signal indicates that the master can accept the read data and response information.

AXI Channels (cont...

- Read channels in AXI
  - Read address
  - Read data

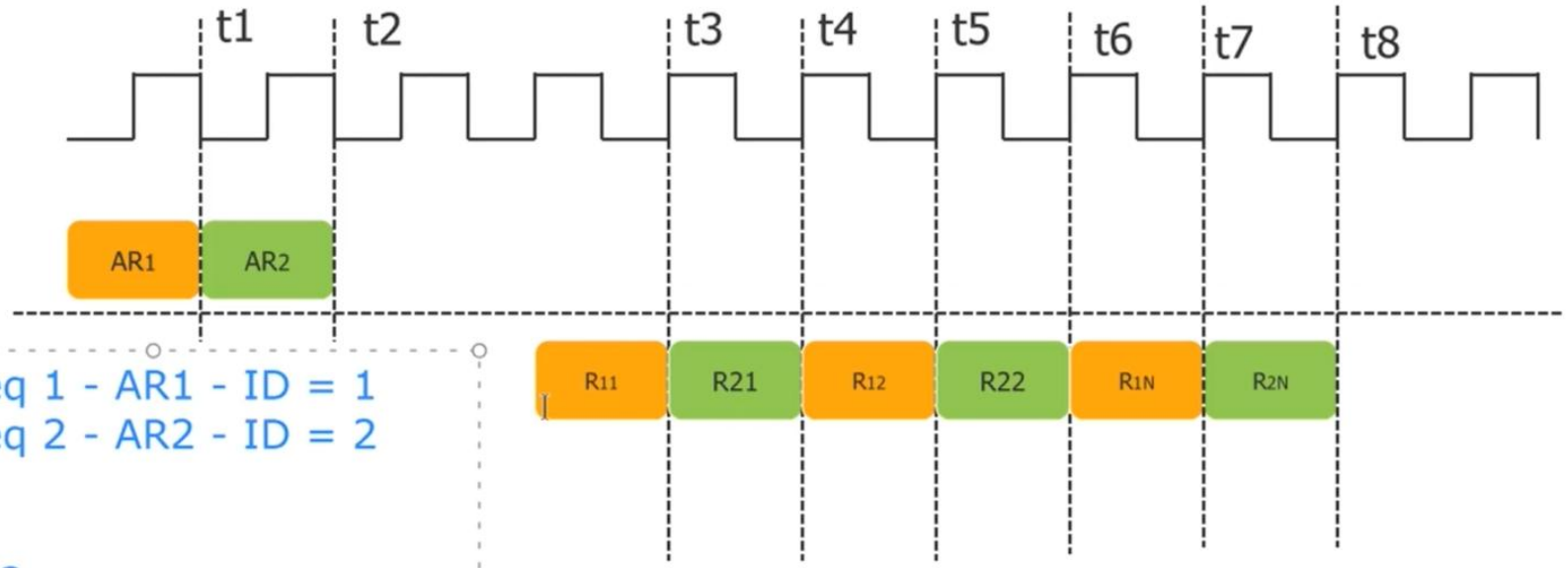


Read Channel ordering



The read data can be interleaved

Read Channel ordering



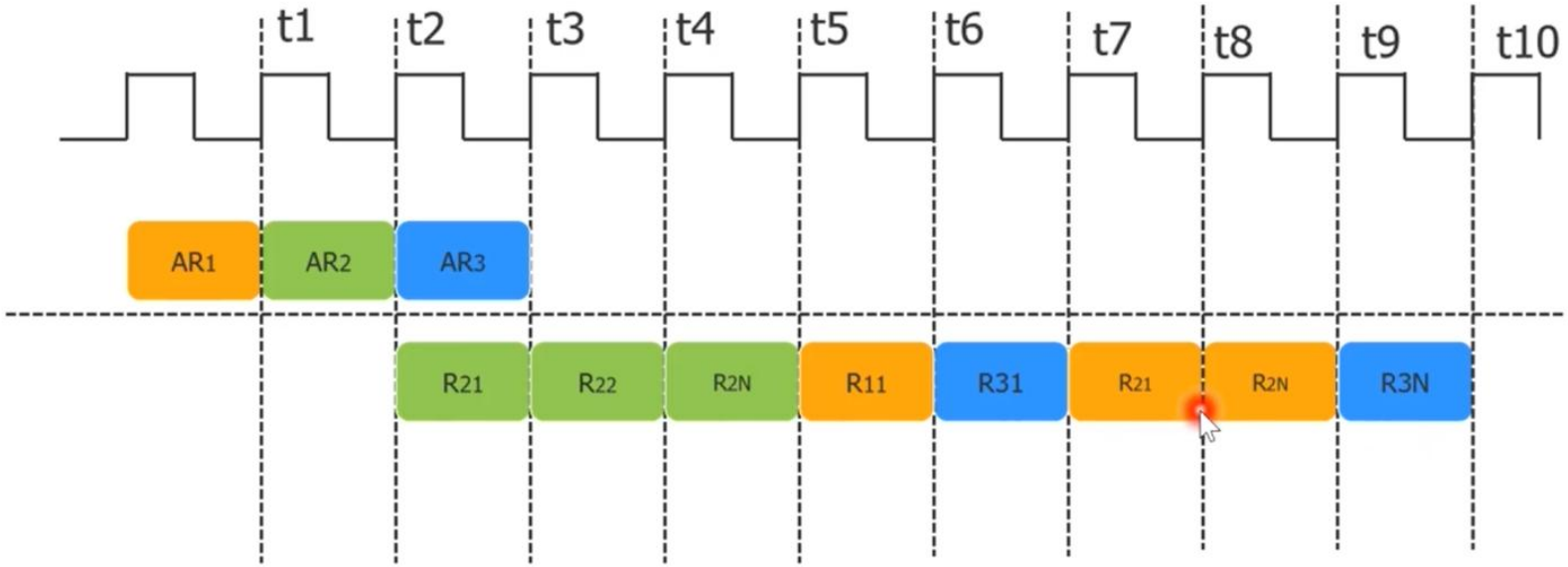
Read Req 1 - AR1 - ID = 1
Read Req 2 - AR2 - ID = 2
ARID

B1 B2 B3
C1 C2 C3

n be interleaved

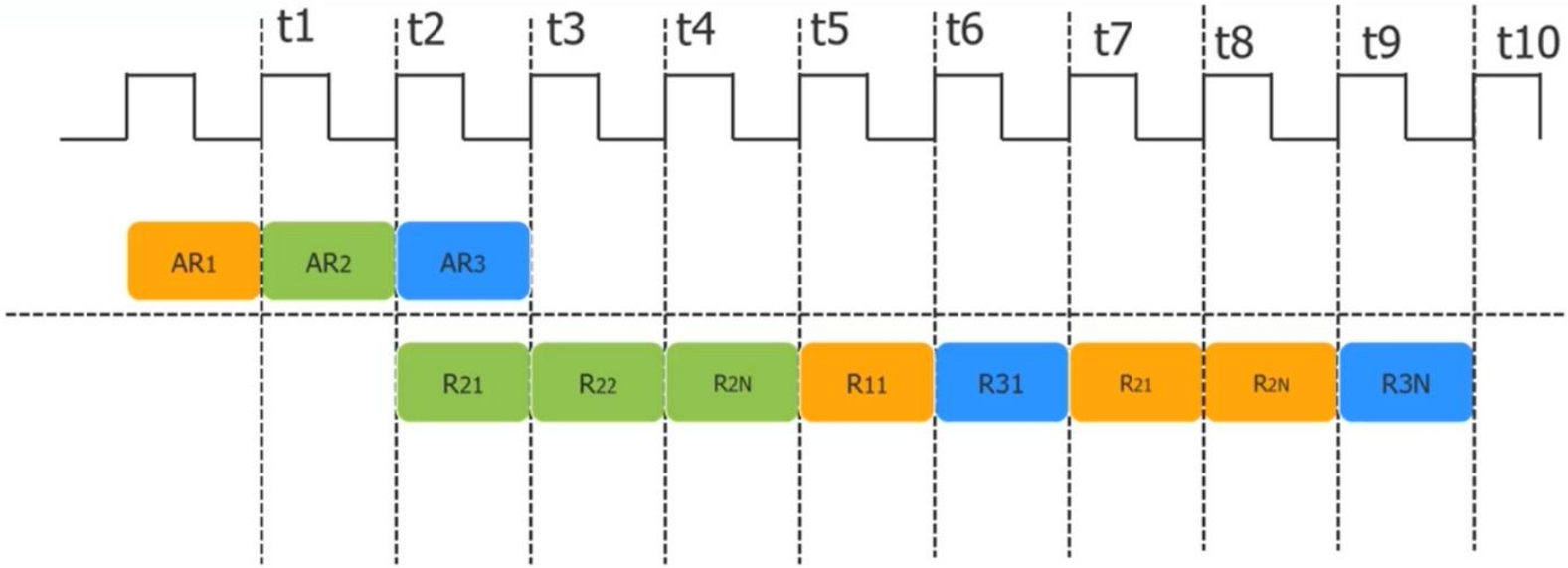
	B1	C1	C2	C3	B2	B3
RID	1	2	2	2	1	1
RLAST	0	0	0	1	0	1

Read channel ordering (cont.)



The read data can be out of order
Or/and a combination of them

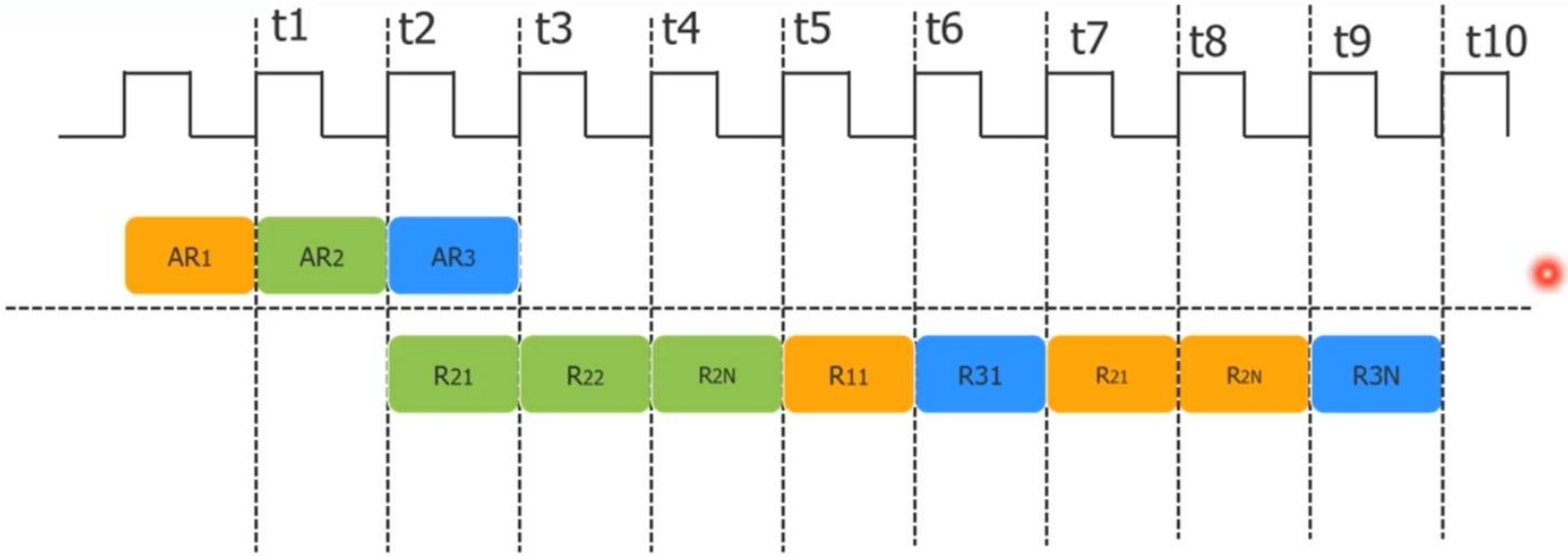
Read channel ordering (cont.)



The read data can be out of order
Or/and a combination of them

ARID-RID matching identifies the read data corresponding to a read address

Read channel ordering (cont.)

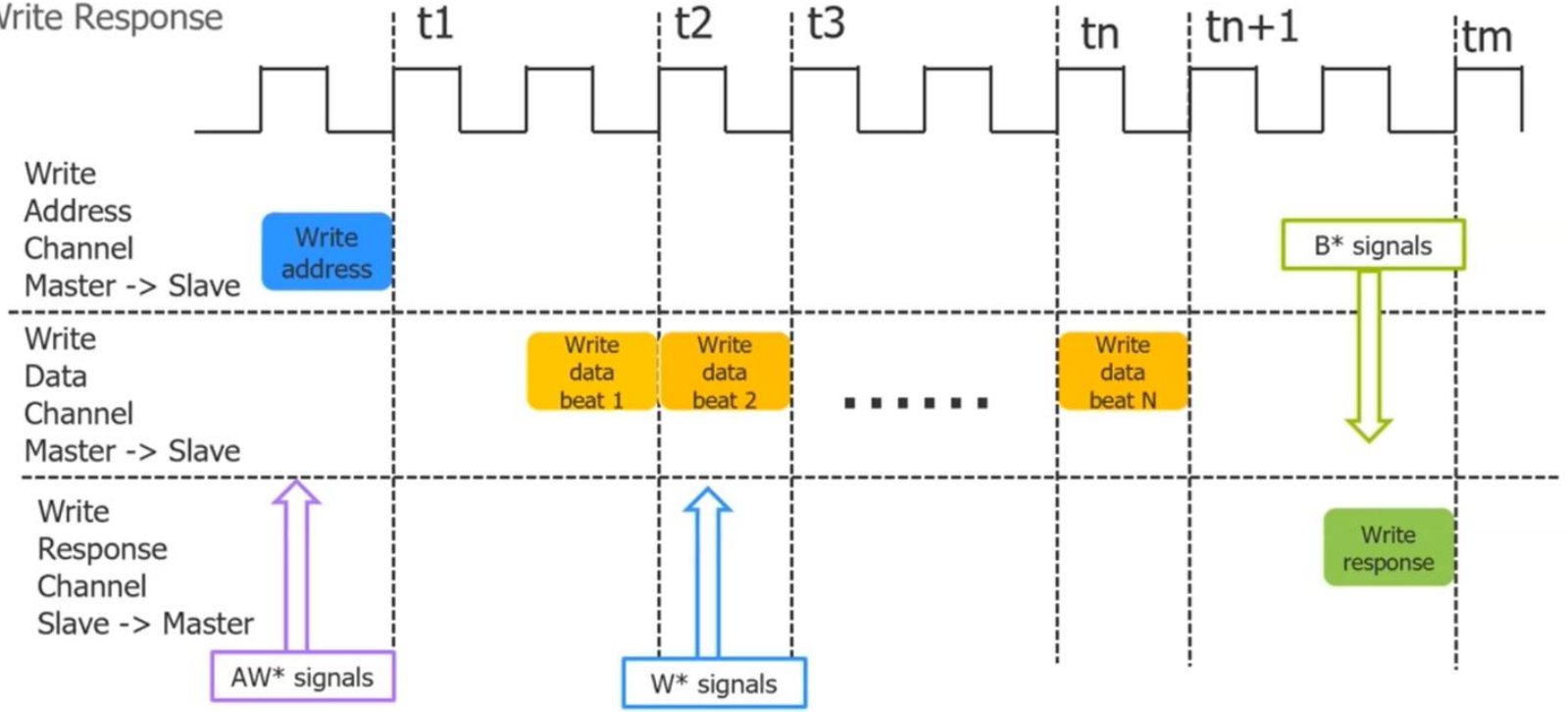


The read data can be out of order
Or/and a combination of them

ARID-RID matching identifies the read data corresponding to a read address

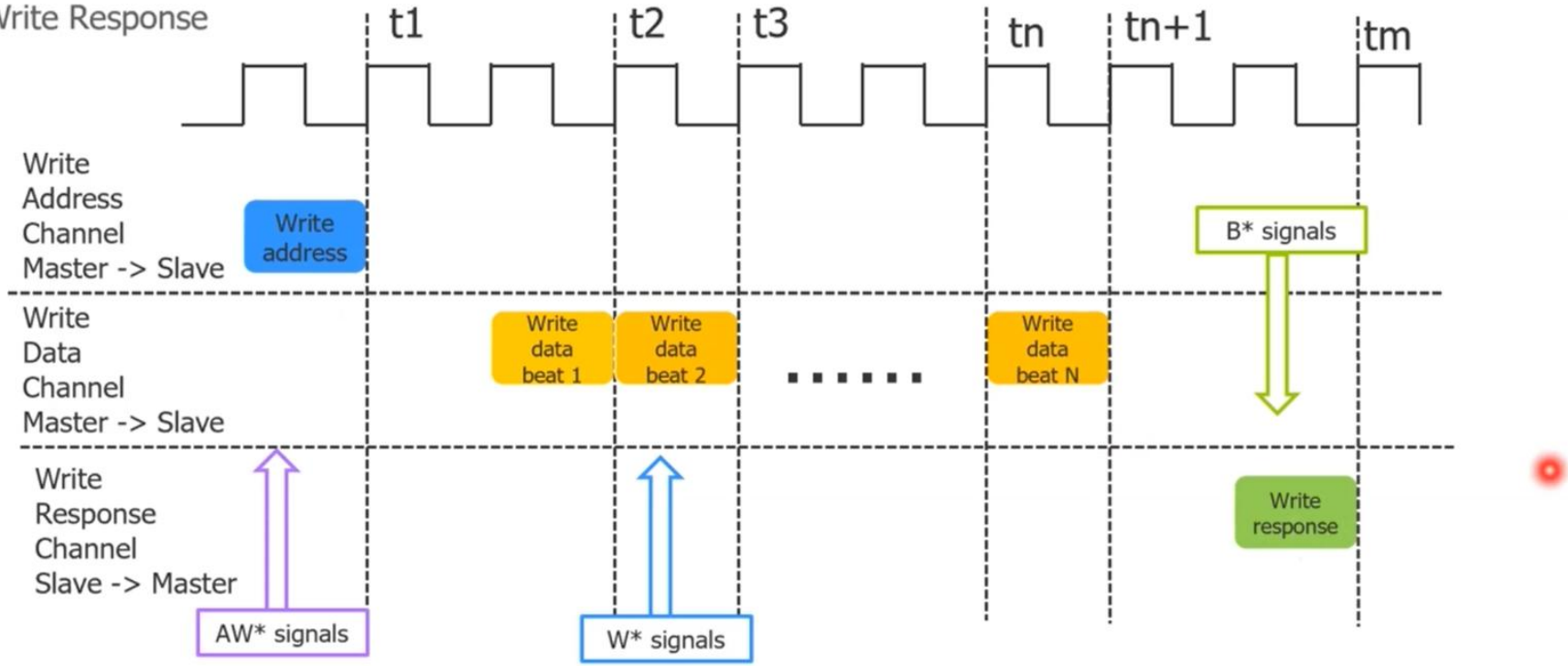
Channel descriptors (Contd.)

- Write Channels
  - Write Address
  - Write Data
  - Write Response

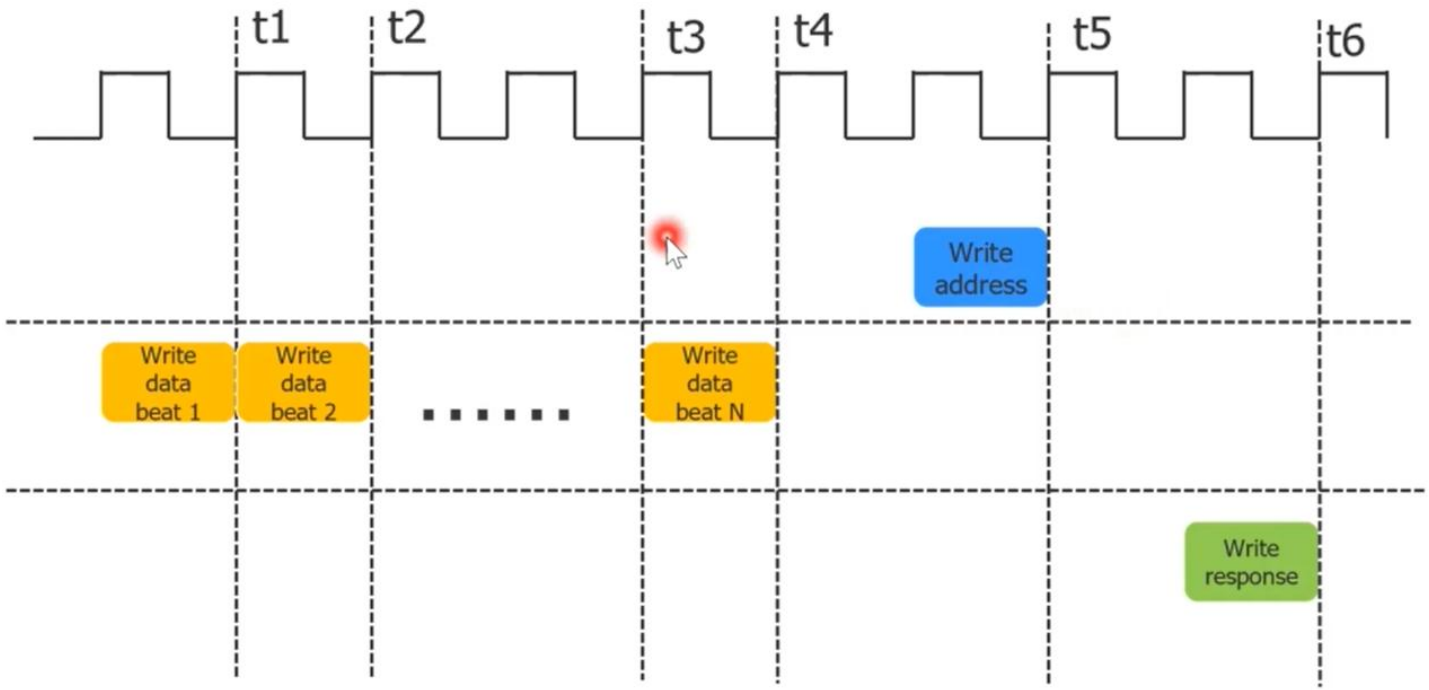


Channel descriptors (Contd.)

- Write Channels
  - Write Address
  - Write Data
  - Write Response

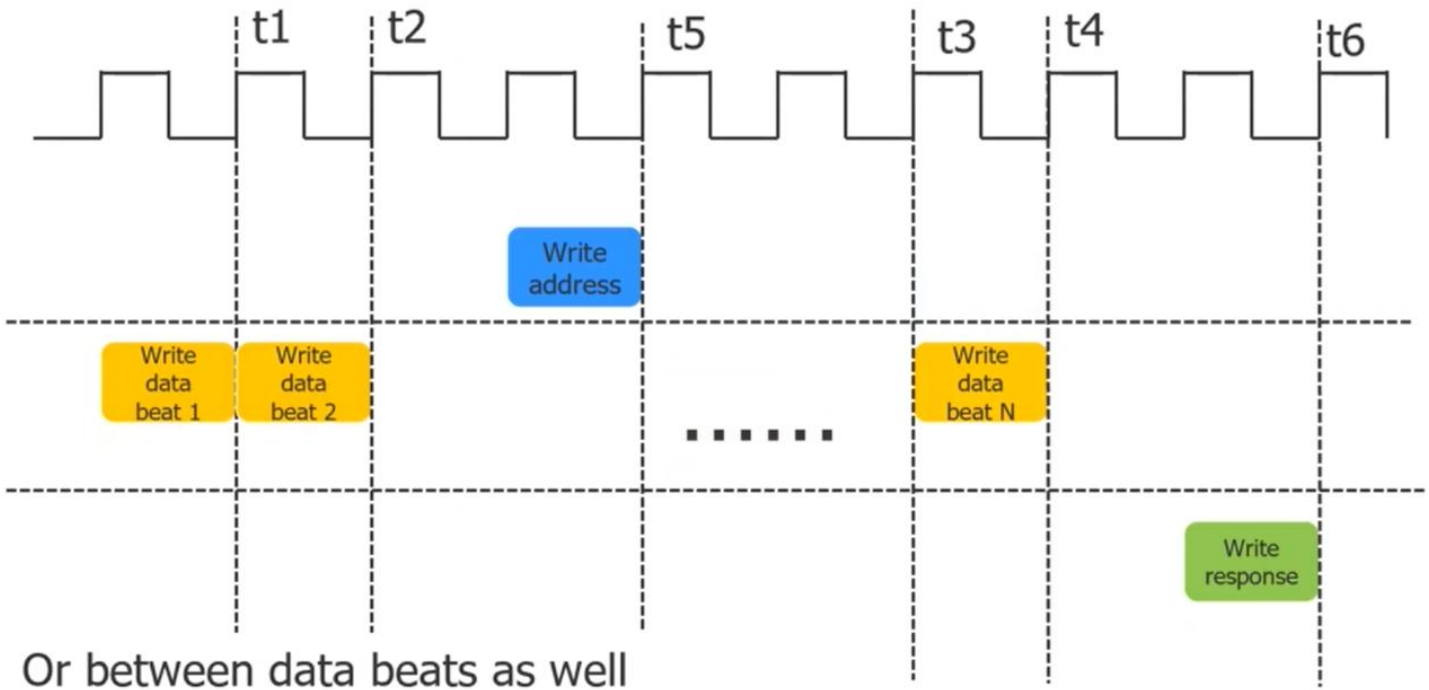


Write channel ordering



Address could be after data beats

Channel descriptors (Contd.)

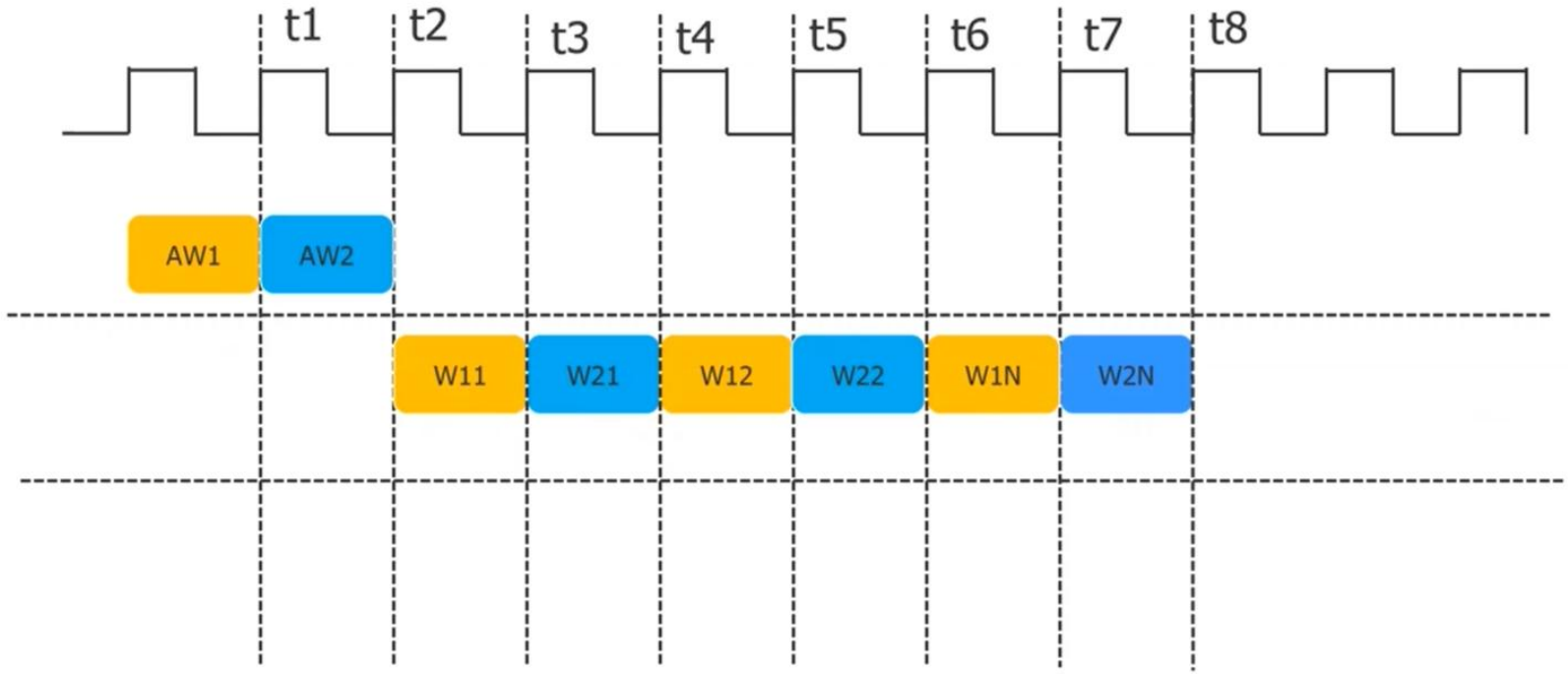


Or between data beats as well

They must not wait for each other

Only the response must be after the write data

Write Channel ordering

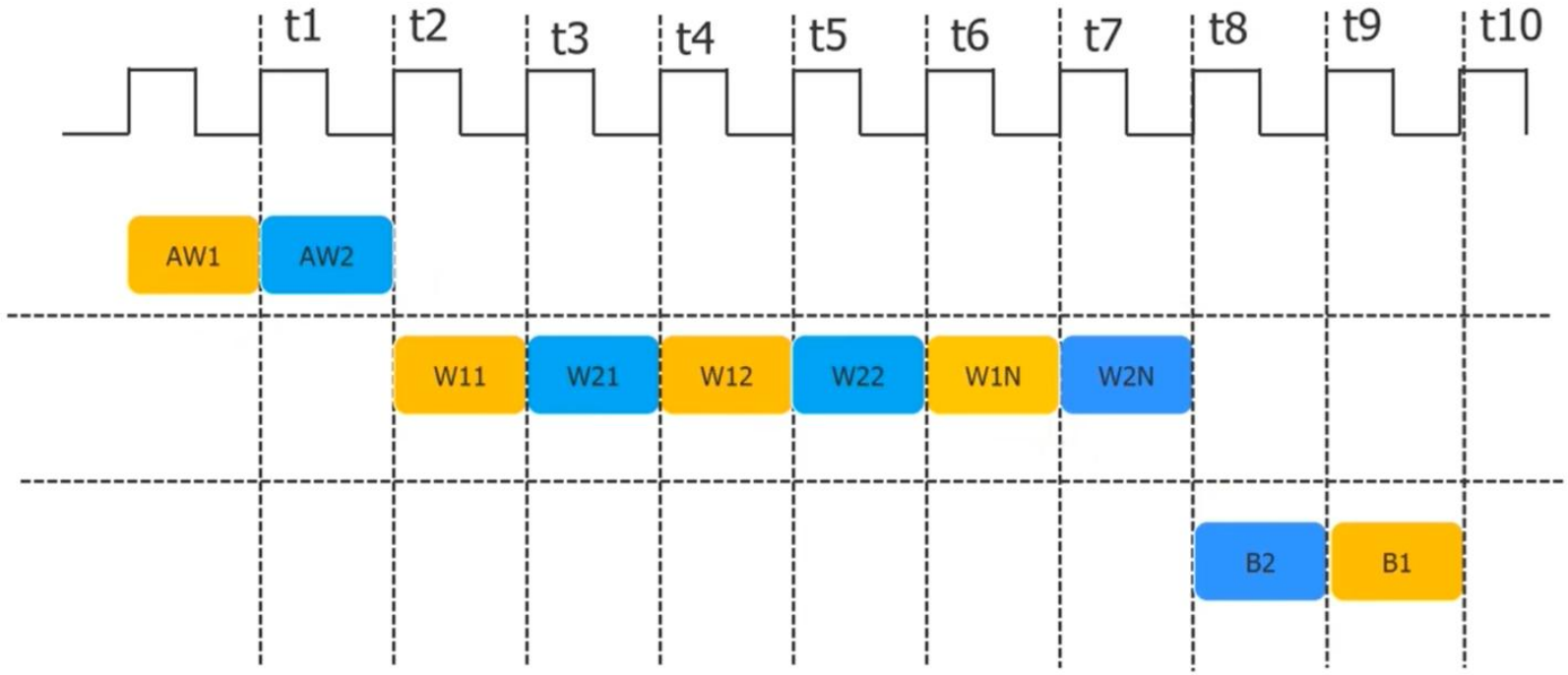


Write data can be interleaved

But not out of order

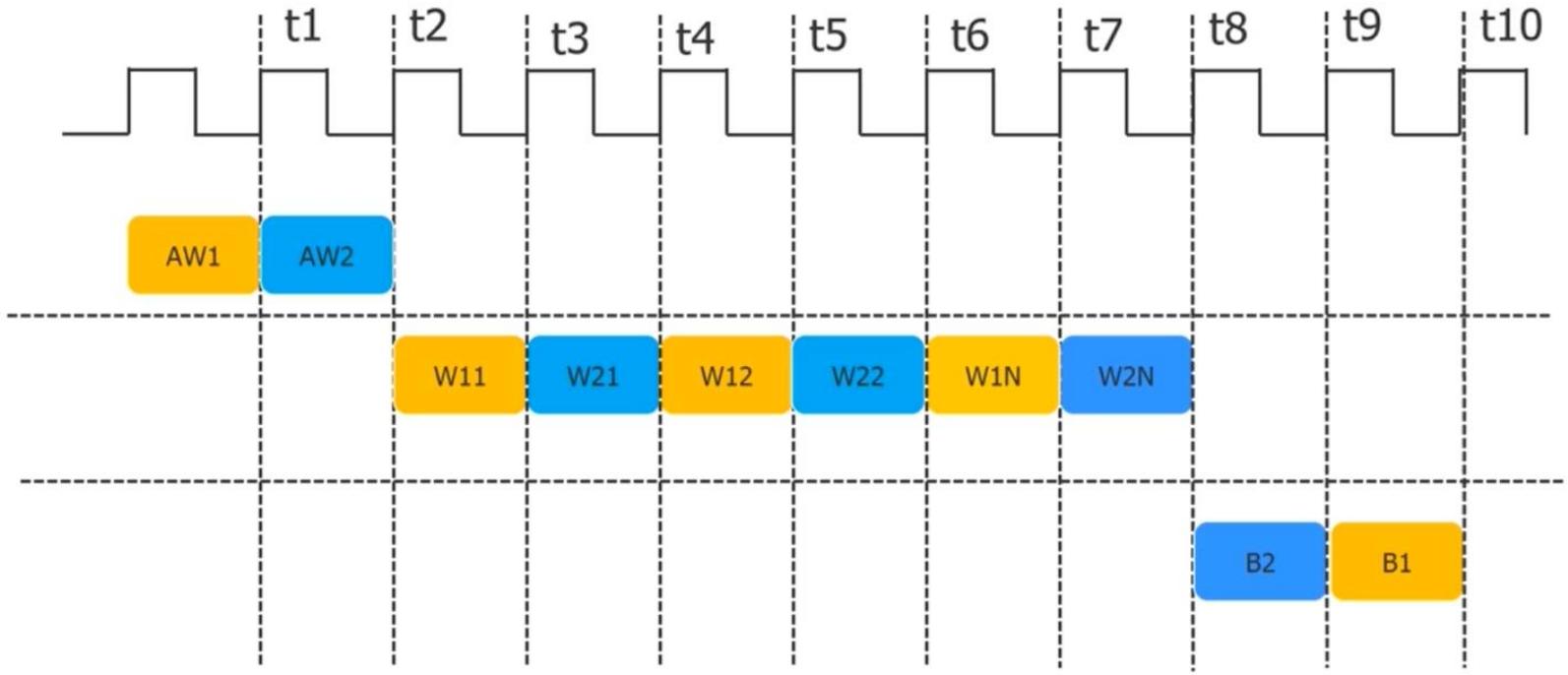
AWID-WID matching identifies the data corresponding to an address

Write Channel ordering



Write response can be out of order

Write Channel ordering

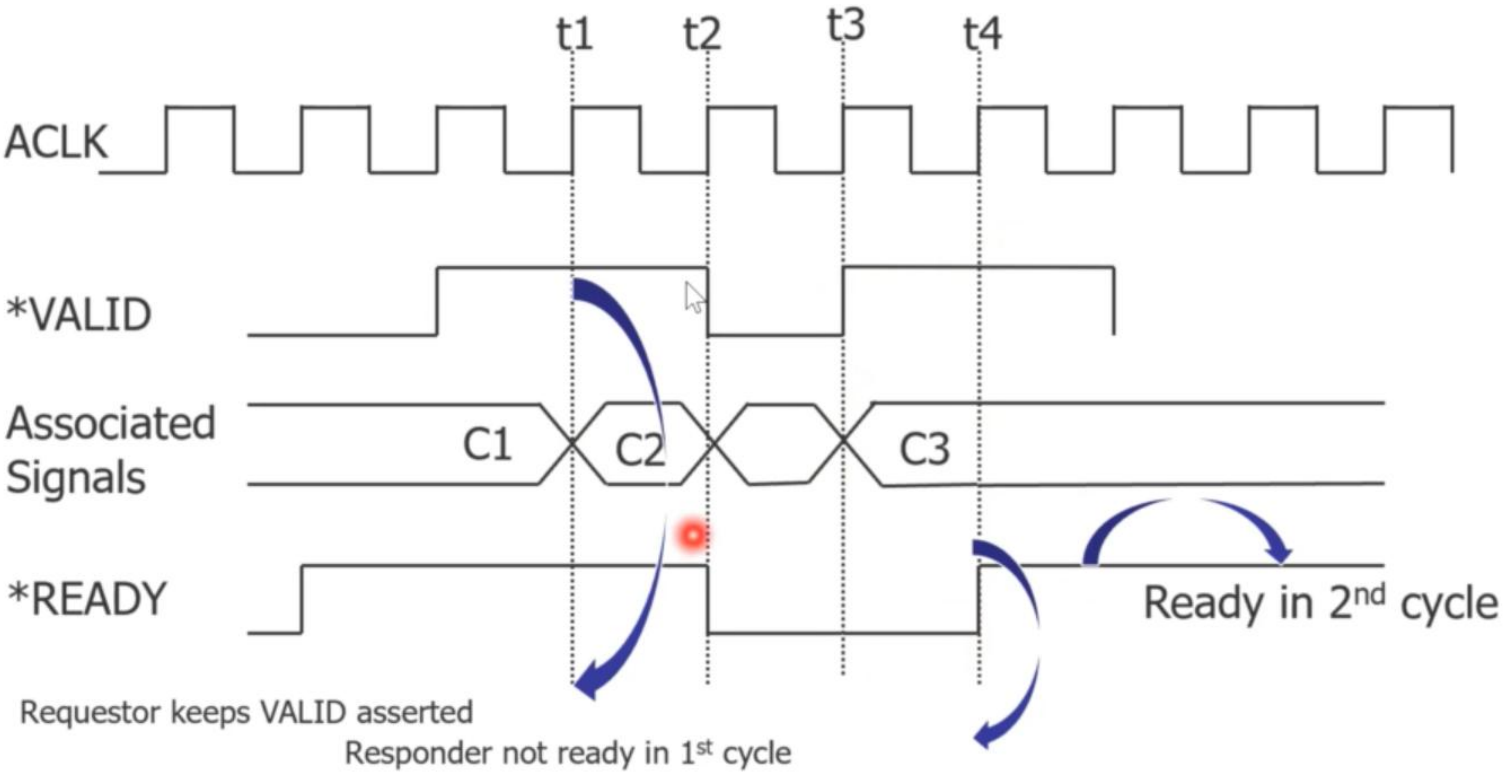


Write response can be out of order

AWID(WID)-BID matching identifies the response corresponding to an addr

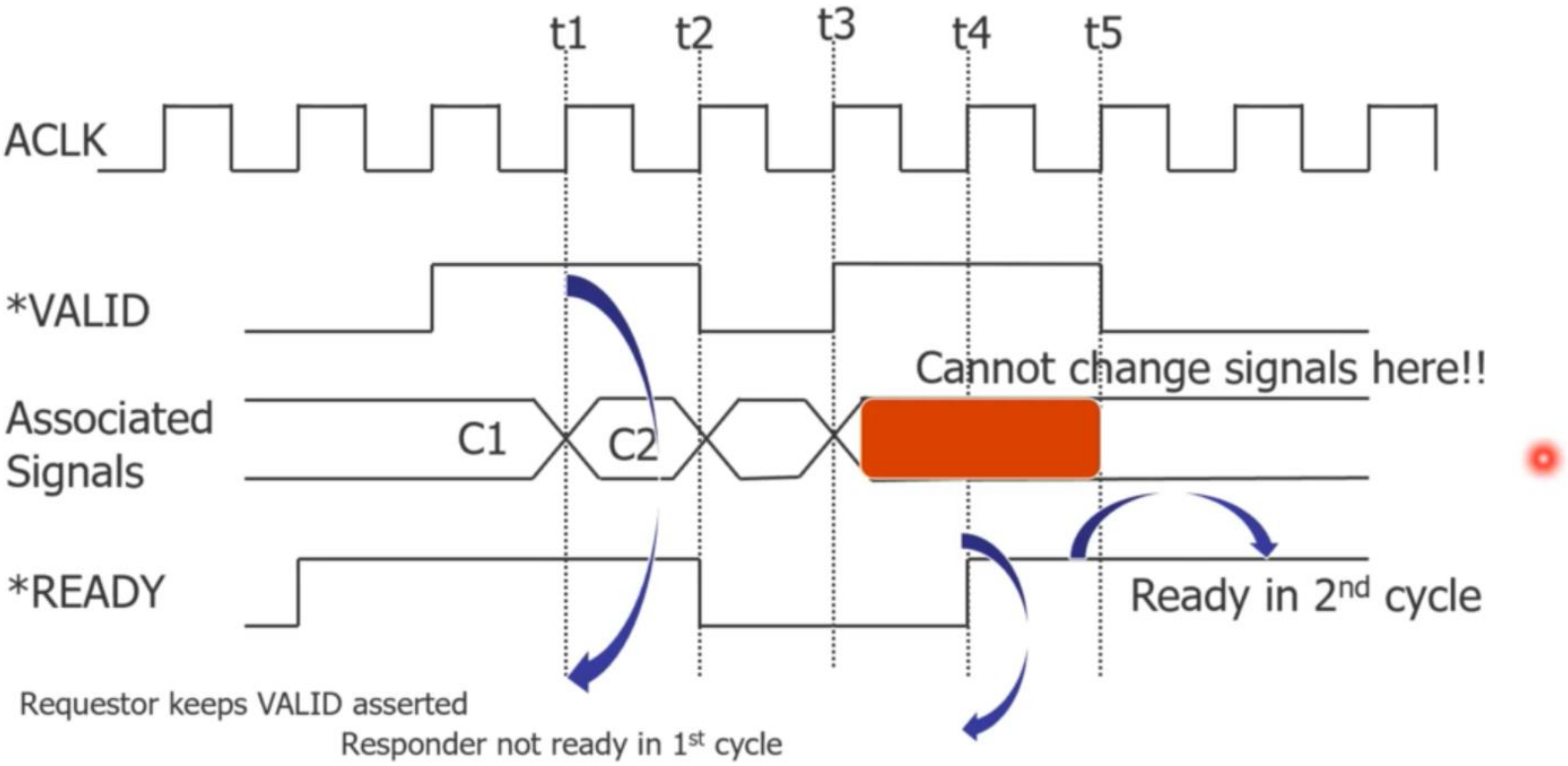
VALID – READY Handshake mechanism

- All these above 5 channels have VALID-READY handshake mechanisms
- Any channel's signals only valid when that channel's VALID asserted



VALID – READY Handshake mechanism

- All these above 5 channels have VALID-READY handshake mechanisms
- Any channel's signals only valid when that channel's VALID asserted



Transaction attributes

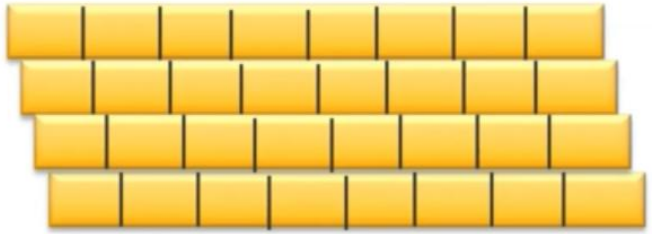
- Burst length
  - — ARLEN, AWLEN signals
  - — Actual number of beats = AxLEN+1
- Burst size
  - — ARSIZE, AWSIZE signals
  - — Actual number of bytes transferred in a beat = 2^AxSIZE

WR, start_addr=0
Data_Bus_Width = 8B
Size = 4B
32B

Transaction attributes

- Burst length
  - ARLEN, AWLEN signals
  - Actual number of beats = AxLEN+1
- Burst size
  - ARSIZE, AWSIZE signals
  - Actual number of bytes transferred in a beat =
- Burst type
  - ARBURST, AWBURST signals
    - FIXED - Memory access to same address. Example FIFO

WR, start_addr=0, WDB=4B, size=4B, FIXED, BL=3 (4 beats)



Transaction attributes

- Burst length
  - ARLEN, AWLEN signals
  - Actual number of beats = AxLEN+1
- Burst size
  - ARSIZE, AWSIZE signals
  - Actual number of bytes transferred in a beat = 2^AxSIZE
- Burst type
  - ARBURST, AWBURST signals
    - FIXED - Memory access to same address. Example FIFO
    - INCR - Normal memory accesses

Transaction address = 0x04, Axsize = 4 bytes, burst type = INCR, AxLEN = 1

Transaction attributes

- Burst length
  - ARLEN, AWLEN signals
  - Actual number of beats = AxLEN+1
- Burst size
  - ARSIZE, AWSIZE signals
  - Actual number of bytes transferred in a beat = 2^AxSIZE
- Burst type
  - ARBURST, AWBURST signals
    - FIXED - Memory access to same address. Example FIFO
    - INCR - Normal memory accesses

Transaction address = 0x04, Axsize = 4 bytes, burst type = INCR, AxLEN = 1

Transaction attributes

- Burst length
  - ARLEN, AWLEN signals
  - Actual number of beats = AxLEN+1
- Burst size
  - ARSIZE, AWSIZE signals
  - Actual number of bytes transferred in a beat = 2^AxSIZE
- Burst type
  - ARBURST, AWBURST signals
    - FIXED - Memory access to same address. Example FIFO
    - INCR - Normal memory accesses
    - WRAP - Cache memory type accesses. Address wraps to lower boundary on reaching a max address

**Note: Wrapping bursts restrictions
1. The start address must be aligned to the size of each transfer.
2. The length of WRAP burst must be 2, 4, 8 or 16 transfers.

Transaction address = 0x04, Axsize = 4 bytes, burst type = INCR, AxLEN = 1

Transaction attributes

- Burst length
  - ARLEN, AWLEN signals
  - Actual number of beats = AxLEN+1
- Burst size
  - ARSIZE, AWSIZE signals
  - Actual number of bytes transferred in a beat = 2^AxSIZE
- Burst type
  - ARBURST, AWBURST signals
    - FIXED - Memory access to same address. Example FIFO
    - INCR - Normal memory accesses
    - WRAP - Cache memory type accesses. Address wraps to lower boundary on reaching a max address
- Lock type
  - ARLOCK, AWLOCK signals
  - Tell about transaction's atomicity

**Note: A burst must not cross a 4KB address boundary.
It prevents a burst from crossing a boundary between two slaves.

Click to add title

WR
Start addr = 4
BL = 4 beats
Burst Type – INCR
Data Bus Width = 64-bits (8B)
Size = 4B

DB1 – 4, 5, 6, 7
DB2 – 8, 9, 10, 11
DB3 – 12, 13, 14, 15
DB4 – 16, 17, 18, 19

4, 5, 6, 7
8, 9, a, b
C, d, e, f
0, 1, 2, 3

Total_TXN_SIZE = (BL*size) = 16Bytes
Lower wrap boundary = start_addr – (start_addr % Total_TXN_SIZE)
= 0x4 - (0x4 % 16)
= 0x4 – 0x4 = 0

Upper wrap boundary = Lower wrap boundary + Total TXN size
= 0 + 16 = 0xF

Click to add title

WR
Start addr = 4
BL = 4 beats
Burst Type – INCR
Data Bus Width = 64-bits (8B)
Size = 4B

DB1 – 4, 5, 6, 7
DB2 – 8, 9, 10, 11
DB3 – 12, 13, 14, 15
DB4 – 16, 17, 18, 19

4, 5, 6, 7
8, 9, a, b
C, d, e, f
0, 1, 2, 3

Total_TXN_SIZE = (BL*size) = 16Bytes
Lower wrap boundary = start_addr – (start_addr % Total_TXN_SIZE)
= 0x4 - (0x4 % 16)
= 0x4 – 0x4 = 0

Upper wrap boundary = Lower wrap boundary + Total TXN size
= 0 + 16 = 0xF

AutoSave Off | AXI3 Protocol overview and AXI4, AXI4-lite differ... - Saved to this PC | Search | Amit Kumar Jain

File Home Insert Design Transitions Animations Slide Show Review View Help | Share | Comments

Paste | New Slide | Reuse Slides | Layout | Reset | Section | Clipboard | Slides | Font | Paragraph | Shapes | Arrange | Quick Styles | Shape Fill | Shape Outline | Shape Effects | Drawing | Find | Replace | Select | Editing | Design Ideas | Designer | Dictate | Voice

23 

24 

25 

[No Title]

26 

27 

Click to add title

WR
Start addr = 3
BL = 4 beats
Burst Type – INCR
Data Bus Width = 64-bits (8B)
Size = 4B

DB1 : - - - - 1 - - -
DB2 : 1 1 1 1 - - - -
DB3 : - - - - 1 1 1 1
DB4 : 1 1 1 1 - - - -

SmartVerif @ 1Stop-EduHub

Slide 25 of 44 | Notes | 67%

Udemy

AutoSave Off | AXI3 Protocol overview and AXI4, AXI4-lite differ... - Saved to this PC | Search | Amit Kumar Jain

File Home Insert Design Transitions Animations Slide Show Review View Help | Share | Comments

Paste | New Slide | Reuse Slides | Layout | Reset | Section | Clipboard | Slides | Font | Paragraph | Shapes | Arrange | Quick Styles | Shape Fill | Shape Outline | Shape Effects | Drawing | Find | Replace | Select | Editing | Design Ideas | Designer | Dictate | Voice

23 Transaction attributes

24 Transaction attributes

25

26 Transaction attributes (contd.)

27 Cache support

Click to add title

WR
Start addr = 3
BL = 4 beats
Burst Type – INCR
Data Bus Width = 64-bits (8B)
Size = 4B

DB1 : - - - - 1 - - -
DB2 : 1 1 1 1 - - - -
DB3 : - - - - 1 1 1 1
DB4 : 1 1 1 1 - - - -

SmartVerif @ 1Stop-EduHub

Slide 25 of 44 | Notes | 67%

Udemy

Transaction attributes (contd.)

- Burst address
  - ARADDR, AWADDR signals
  - The address to which transaction sent
- Memory attribute signalling
  - ARCACHE, AWCACHE signals
  - Bufferable, Cacheable, and Allocate attributes of the transaction.
- Protection type
  - ARPROT, AWPROT signals
  - Provides access permissions
- Write data strobes
  - WSTRB signal
  - De-asserted for all invalid lanes
  - May or may not be asserted for valid lanes
  - Provides option to not write particular bytes to memory

Cache support

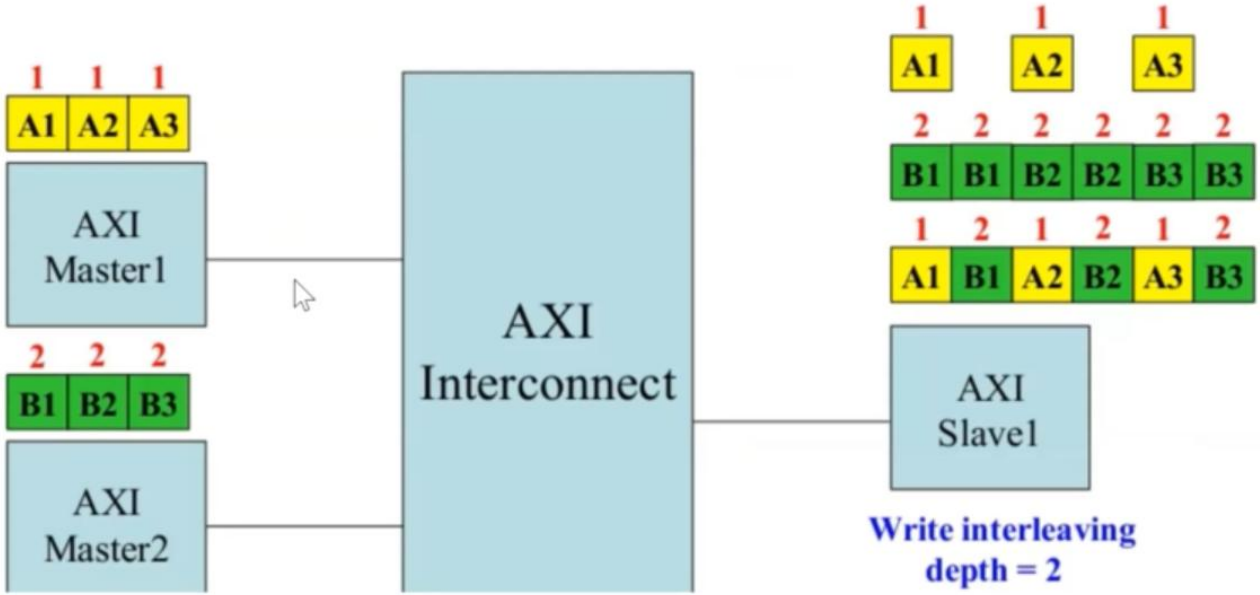
- Support for system level caches and other performance enhancing components
  - Bufferable (B) bit, ARCACHE[0] and AWCACHE[0]
  - Cacheable (C) bit, ARCACHE[1] and AWCACHE[1]
  - Read Allocate (RA) bit, ARCACHE[2] and AWCACHE[2]
  - Write Allocate (WA) bit, ARCACHE[3] and AWCACHE[3]

Cache support

- 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]

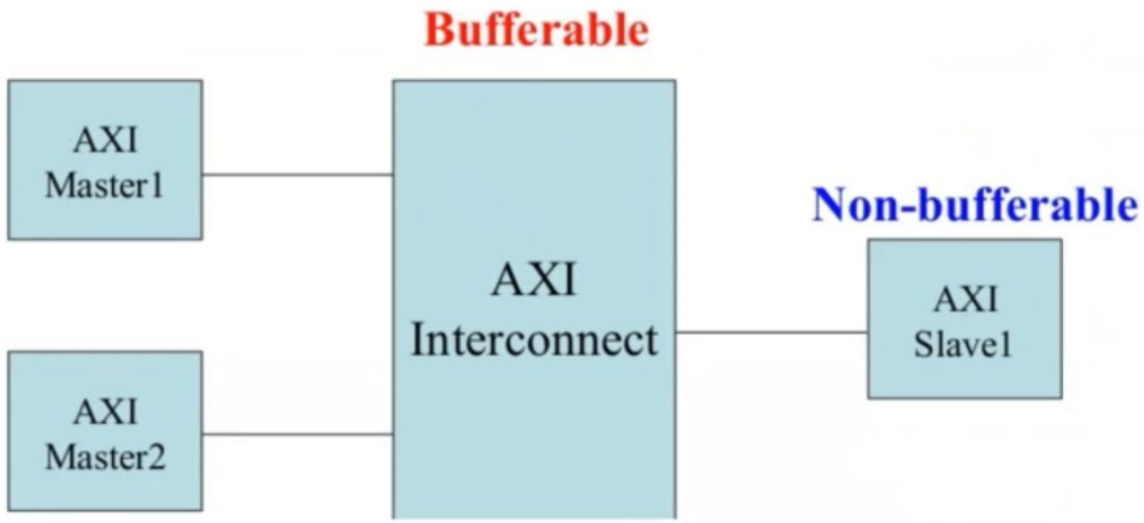
Bufferable bit

- The interconnect or any component can delay the transaction for an arbitrary number of cycles, usually only relevant to writes



Bufferable bit (cont...)

- IF a transaction is bufferable
  - It is acceptable for a bridge or system level cache to provide write response
- If non-bufferable
  - Final destination to provide response



Cache support

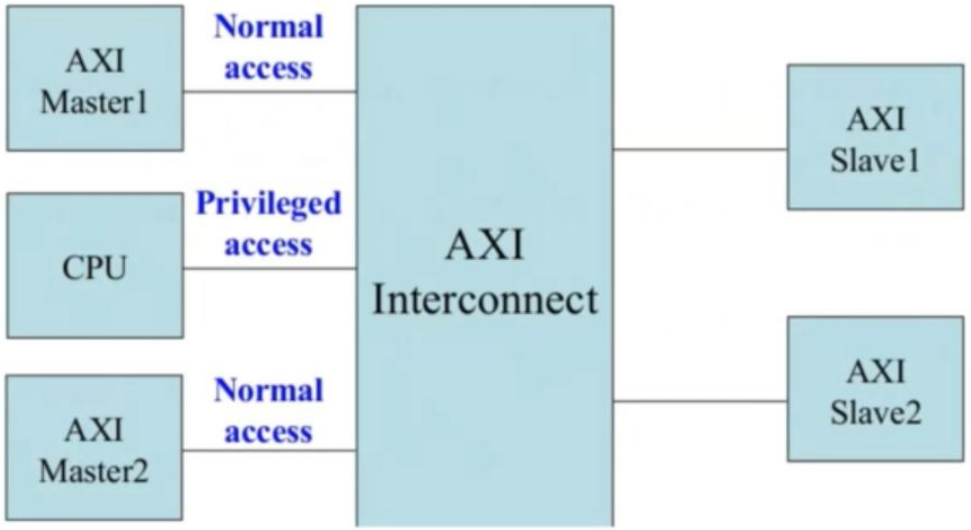
- Cacheable Bit
  - Write : a number of different writes can be merged together
  - Read : a location can be pre-fetched or can be fetched just once for multiple read transactions
- Read Allocate Bit
  - If the transfer is a read and it misses in the cache then it should be allocated
- Write Allocate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

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] and AWPROT[0]
    Low High
  - Secure or non-secure, ARPROT[1] and AWPROT[1]
    Low High
  - Data or instruction, ARPROT[2] and AWPROT[2]
    Low High

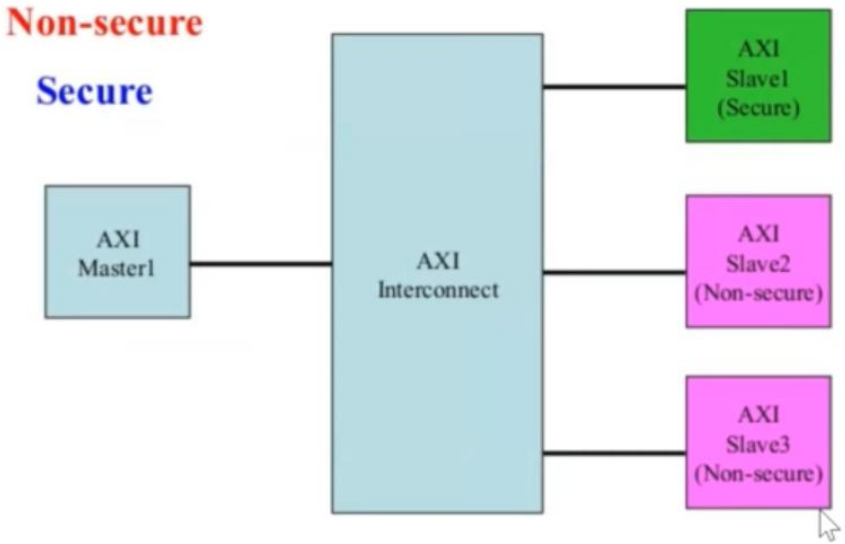
Normal/Privileged

- This is used by some masters to indicate their processing mode. A privileged processing mode typically has a greater lever of access within a system



Secure/Non-secure

- This is used in systems where a greater degree of differentiation between processing modes is required



SLVERR response

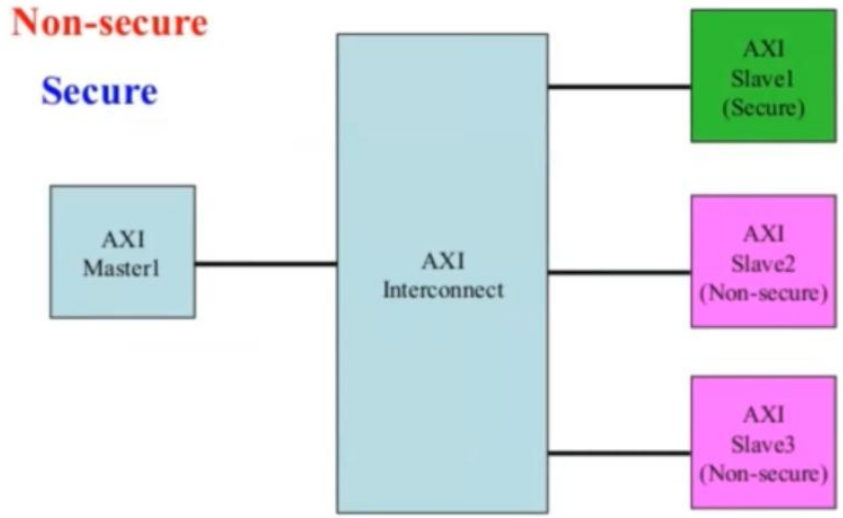
OKAY response

DECERR response

Non-secure slave disappears from the memory map during secure accesses

Secure/Non-secure

- This is used in systems where a greater degree of differentiation between processing modes is required



SLVERR response

OKAY response

DECERR response

Non-secure slave disappears from the memory map during secure accesses

Data/Instruction

- This bit gives an indication if the transaction is an instruction or a data access

When a transaction contains a mix of instruction and data items

It's recommended that, by default, an access is marked as a data access unless specifically known to be an instruction access

Locked transfers

- Specified when AxLOCK = 2'b10
- Used when an AXI master requires special accesses for bus.
- Before starting ensure no other unlocked transfer pending
- Start a locked read or write
- Finish by issuing an unlocked read or write – Removes lock

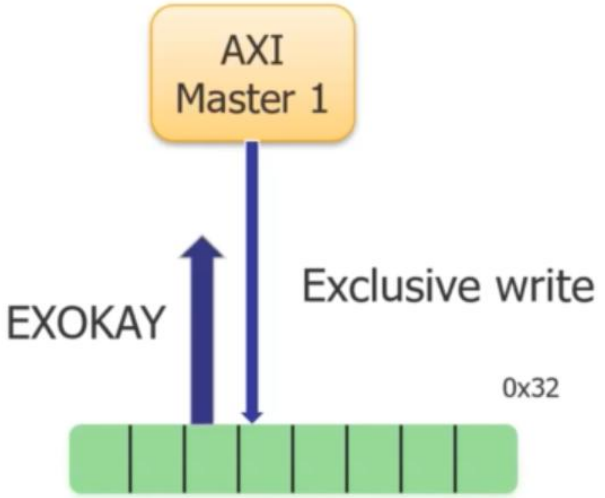
Exclusive transaction

- Read modify write kind of transactions with special care
    1. Issue an exclusive read txn to some address
    2. Monitor the address for exclusivity
    3. Issue matching exclusive write txn to addr
    4. Exclusive fail/pass
        - If addr updated by other txn/master
            - Respond with OKAY

Exclusive transaction

- Read modify write kind of transactions with special care

1. Issue an exclusive read txn to some address
2. Monitor the address for exclusivity
3. Issue matching exclusive write txn to addr
4. Exclusive fail/pass
   - If addr not updated by any other txn/master
     - Respond with EXOKAY

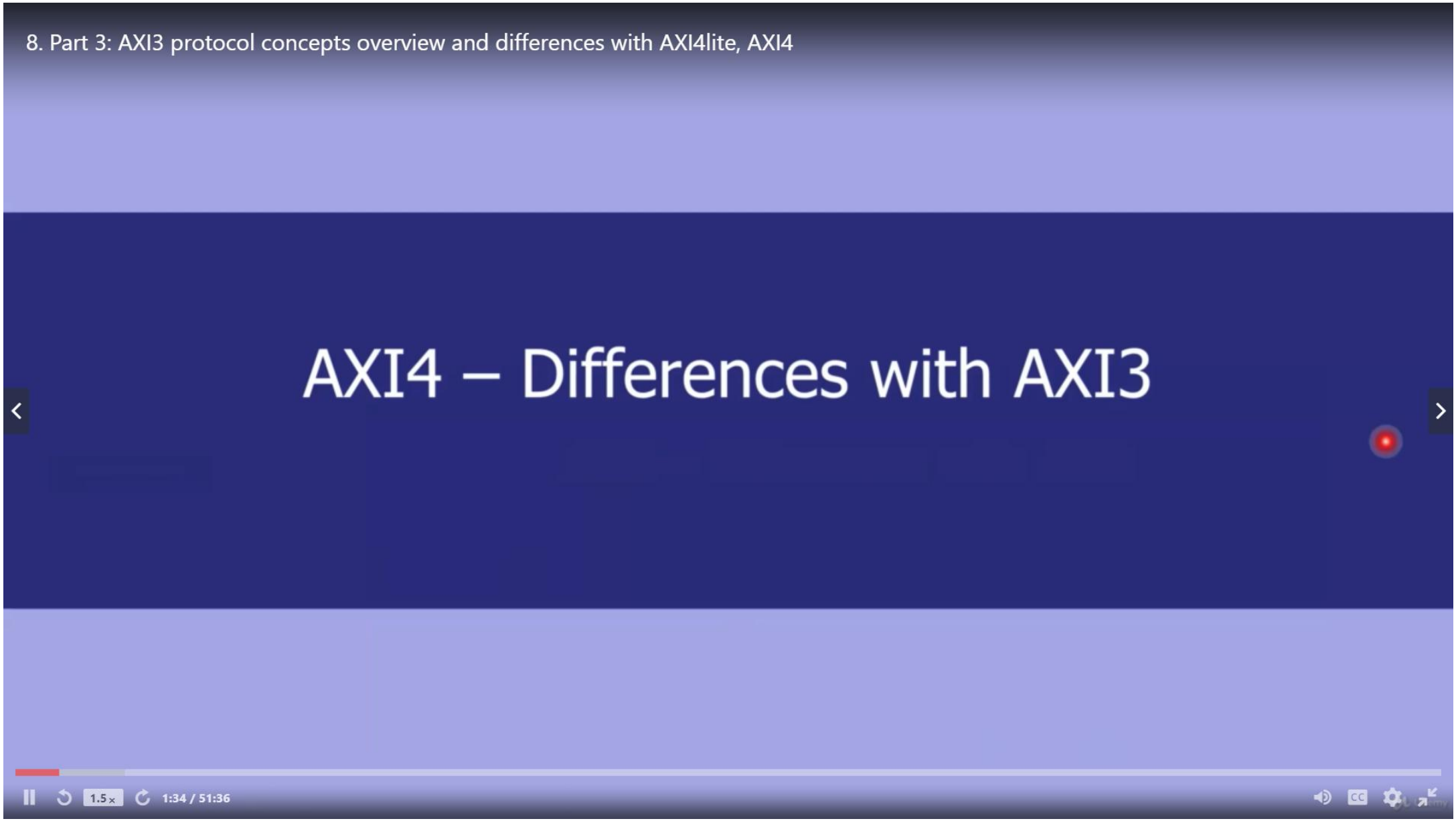


AXI responses

- BRESP[1:0]/RRESP[1:0]
  - OKAY, normal access successes
    - Success of a normal access
    - Failure of an exclusive access
    - An exclusive access to a slave that does not support exclusive
  - EXOKAY, exclusive access successes
    - Success of an exclusive access
  - SLVERR, slave error
    - FIFO or buffer overflow or underflow condition
    - Unsupported transfer size attempt
    - Write access attempt to RO location
  - DECERR, decode error
    - Returns DECERR in case interconnect can not successfully decode a slave access

AXI responses

- BRESP[1:0]/RRESP[1:0]
  - OKAY, normal access successes
    - Success of a normal access
    - Failure of an exclusive access
    - An exclusive access to a slave that does not support exclusive
  - EXOKAY, exclusive access successes
    - Success of an exclusive access
  - SLVERR, slave error
    - FIFO or buffer overflow or underflow condition
    - Unsupported transfer size attempt
    - Write access attempt to RO location
  - DECERR, decode error
    - Returns DECERR in case interconnect can not successfully decode a slave access


8. Part 3: AXI3 protocol concepts overview and differences with AXI4lite, AXI4
AXI4 – Differences with AXI3
1.5×
1:34 / 51:36
CC

AXI Interface: AXI4

- AWLEN/ARLEN now 8 bits wide to support INCR bursts of up to 256 transfer means burst up to 256 data beats.
- AWLOCK/ARLOCK now single bit as support for "locked" transfers dropped.
- No WID signal now as interleaving of write data transfers no longer supported.
- Additional handshake rule describing that B channel response can only be returned after both the AW and final W channel transfers have completed.
- Burst up to 256 data beats
- The AXI4 read interfaces allows for data phase transfer up to 256 beats as opposed to the 16 beats that were supported for AXI3.
- New AxQOS, AxREGION and AxUSER signals added.

AXI4: QoS (Quality of Service)

- AXI4 extended to support the two 4-bit Qos signaling.
  - AWQOS: Associated with write address channel and 4-bit signal is sent for each write transaction.
  - ARQOS: Associated with read address channel and 4-bit signal is sent for each read transaction.
- Qos signals are used as priority indicator for associated write or read transaction.
- Higher value indicates higher priority transactions.
- Default value: 4'b0000 of it indicates that interface is not participating in any Qos scheme.
- The AXI4 protocol does not include an exact interpretation of the priority signals, instead each actual implementation can define how these signal are used to provide quality of service criteria, like max. access time etc.

AXI4: Region signaling

- AWREGION/ARREGION:
  - AWREGION: A region identifier, sent on the write address channel for each write transaction.
  - ARREGION: A region identifier, sent on the read address channel for each read transaction.
- These signals provide an address decode of the existing address space that can be used by slaves to remove the need for an address decode function.
- It does not create new independent address space.

AXI4: User defined signaling

- User defined signals:
  - AWUSER: Write address channel user signals
  - ARUSER : Read address channel user signals
  - WUSER : Write data channel user signals
  - RUSER : Read data channel user signals
  - BUSER : Write response channel user signals
- Width of user defined signals is Implementation defined and can be different for each of the channels.

AXI4-lite – Differences with AXI4

AXI4-Lite

- No burst supported.
- All transactions are burst length of 1.
- All data accesses are the same size as the width of the data bus.
- Support for data bus width of 32 or 64 bits.
- All accesses are equivalent to AWCACHE, or ARCACHE equal to b0000. (Non-modifiable, Non-bufferable)
- Exclusive accesses are not supported.
- AXI4-Lite does not support AXI IDs. This means all transactions must be in order, and all accesses use a single fixed ID value.

AXI4-Lite

- No burst supported.
- All transactions are burst length of 1.
- All data accesses are the same size as the width of the data bus.
- Support for data bus width of 32 or 64 bits.
- All accesses are equivalent to AWCACHE, or ARCACHE equal to b0000. (Non-modifiable, Non-bufferable)
- Exclusive accesses are not supported.
- AXI4-Lite does not support AXI IDs. This means all transactions must be in order, and all accesses use a single fixed ID value.

AXI4-Lite

- No burst supported.
- All transactions are burst length of 1.
- All data accesses are the same size as the width of the data bus.
- Support for data bus width of 32 or 64 bits.
- All accesses are equivalent to AWCACHE, or ARCACHE equal to b0000. (Non-modifiable, Non-bufferable)
- Exclusive accesses are not supported.
- AXI4-Lite does not support AXI IDs. This means all transactions must be in order, and all accesses use a single fixed ID value.

AXI4-Lite

- No burst supported.
- All transactions are burst length of 1.
- All data accesses are the same size as the width of the data bus.
- Support for data bus width of 32 or 64 bits.
- All accesses are equivalent to AWCACHE, or ARCACHE equal to b0000. (Non-modifiable, Non-bufferable)
- Exclusive accesses are not supported.
- AXI4-Lite does not support AXI IDs. This means all transactions must be in order, and all accesses use a single fixed ID value.
- AXI4-Lite does not support data interleaving, the burst length is defined as 1.

AXI Lite Signal list

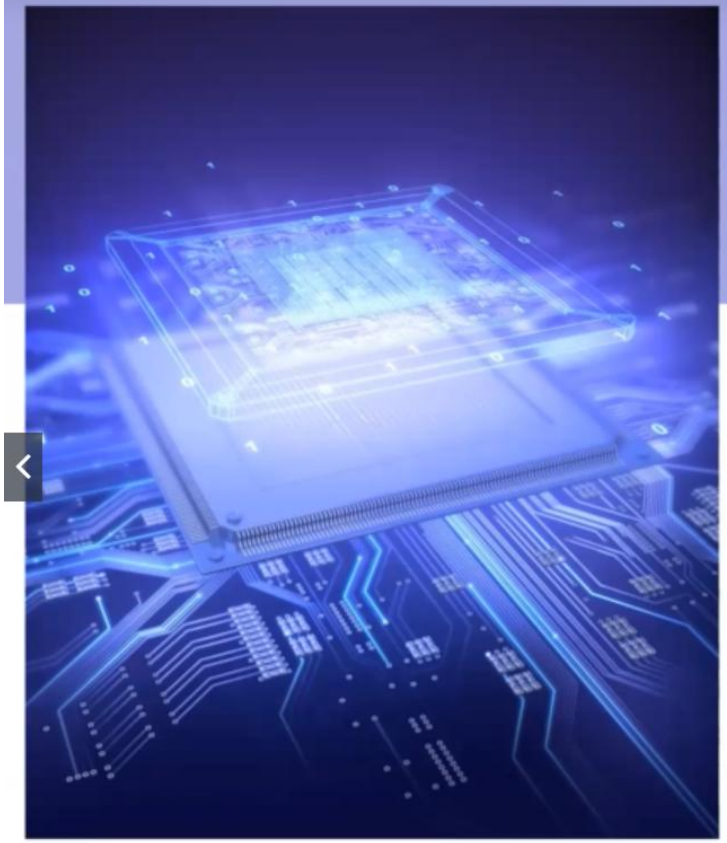
- Subset of AXI signal set
- Simple traditional signaling
- Targeted applications: simple, low-performance peripherals
  - GPIO
  - UART

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 Signal list

- AXI4-Lite interface does not support the following signals:
  - RRESP, BRESP:
    - EXOKAY response is not supported on the read data and write response channels.
  - AWLEN, ARLEN:
    - burst length is defined to be 1, equivalent to an AxLEN value of 0.
  - AWSIZE, ARSIZE:
    - All accesses are defined to be the width of the data bus.
  - AWBURST, ARBURST:
    - Burst type has no meaning because the burst length is 1.
  - AWLOCK, ARLOCK:
    - All accesses are defined as Normal accesses, equivalent to an AxLOCK value of o.
  - AWCACHE, ARCACHE:
    - All accesses are defined as Non-modifiable, Non-bufferable, equivalent to an AxCACHE value of 0.
  - WLAST, RLAST:
    - All bursts are defined to be of length 1, equivalent to a WLAST or RLAST value of 1.

9. Part 1: I2C protocol concepts overview



I2C (Inter Integrated Circuit) Protocol Concepts Overview

Team SmartVerif @ 1Stop-EduHub

Feb 1, 2020

Outline

- I2C
  - Introduction
  - Terminology
  - I2C Bus Applications
  - Features
    - Start-Stop condition
    - ACK, NACK
    - Clock Synchronization
    - Arbitration
    - Clock Stretching
    - Data Transfer Formats
      - 7-bit addressing
      - 10-bit addressing
  - HS Mode
  - Ultra Fast Mode
- Verification perspective
  - Category of coverage items

Introduction

- I^2C was originally developed in 1982 by Philips for various Philips chips
- The original spec allowed for only 100kHz standard mode communications, and provided only for 7-bit addresses
- In 1992, the first public specification was published, adding a 400kHz fast-mode as well as an expanded 10-bit address space.
- There are three additional modes specified:
  - Fast-mode plus, at 1MHz
  - High-speed mode, at 3.4MHz
  - Ultra-fast mode, at 5MHz(Unidirectional Bus)

Introduction (Contd..)

I2C
Multi-master Multi-slave system
All of them are connected through only 2 pins - SDA and SCL

These 2 pins are shared among all master and slaves in the system

These 2 pins use wired-AND mechanism

P A S Sanj all 4 connected through same SDA
P - 0
A - 0
S - 0
Sanj - 1
What will be resultant SDA ??? - 0

Introduction (Contd..)

- Two wires carry information between a number of devices
  - One wire use for the data (SDA, high when bus is free.)
  - One wire used for the clock(SCL, high when bus is free.)
- Unique start and stop condition
- Bi-directional data transfer(Except UFM)
- Acknowledgement after each transferred byte
- No fixed length of transfer
- True multi-master capabilit
  - Clock synchronization
  - Arbitration procedure

D0 and D1
D0 - 1100001110101010 1
D1 - 1100010001010101

Introduction (Contd..)

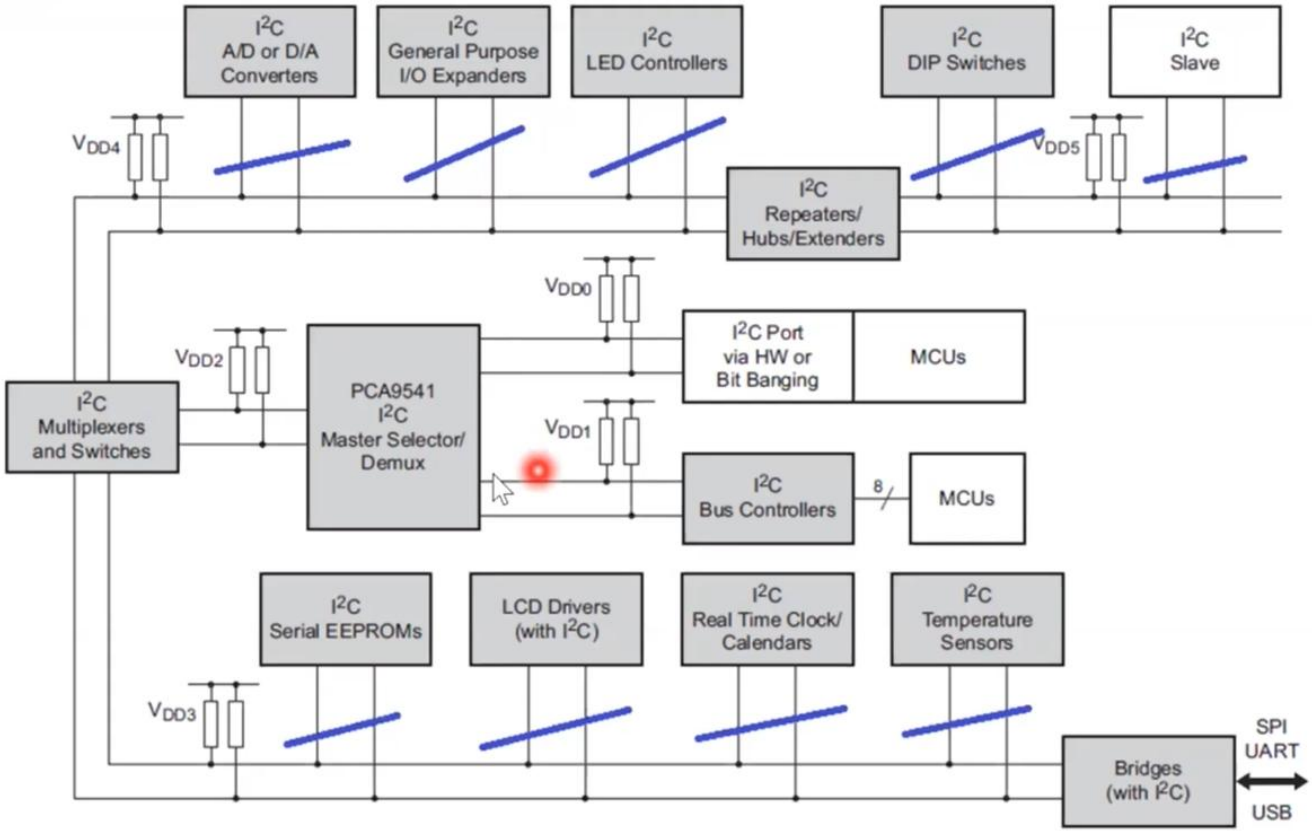
- Two wires carry information between a number of devices
  - One wire use for the data (SDA, high when bus is free.)
  - One wire used for the clock(SCL, high when bus is free.)
- Unique start and stop condition
- Bi-directional data transfer(Except UFM)
- Acknowledgement after each transferred byte
- No fixed length of transfer
- True multi-master capabilit
  - Clock synchronization
  - Arbitration procedure

D0 and D1
D0 - 1100001110101010 1
D1 - 1100011111111111 1

Terminology

Term	Description
Transmitter	the device which sends data to the bus
Receiver	the device which receives data from the bus
Master	the device which initiates a transfer, generates clock signals and terminates a transfer
Slave	the device addressed by a master
Multi-master	more than one master can attempt to control the bus at the same time without corrupting the message
Arbitration	procedure to ensure that, if more than one master simultaneously tries to control the bus, only one is allowed to do so and the winning message is not corrupted
Synchronization	procedure to synchronize the clock signals of two or more devices

I2C Bus Applications



Features

M = mandatory; O = optional; n/a = not applicable.

Feature	Configuration		
	Single master	Multi-master	Slave[1]
START condition	M	M	M
STOP condition	M	M	M
Acknowledge	M	M	M
Synchronization	n/a	M	n/a
Arbitration	n/a	M	n/a
Clock stretching	O[2]	O[2]	O
7-bit slave address	M	M	M
10-bit slave address	O	O	O
General Call address	O	O	O
Software Reset	O	O	O
START byte	n/a	O[3]	n/a
Device ID	n/a	n/a	O

[1] Also refers to a master acting as a slave.

[2] Clock stretching is a feature of some slaves. If no slaves in a system can stretch the clock (hold SCL LOW), the master need not be designed to handle this procedure.

[3] 'Bit banging' (software emulation) multi-master systems should consider a START byte. See Section 3.1.15.

Bit Transfer

- Data on the SDA line must be stable during the high period of clock.
- High or low state of SDA line can only changed when SCL is low.
- One clock pulse is generated for each data bit transferred.

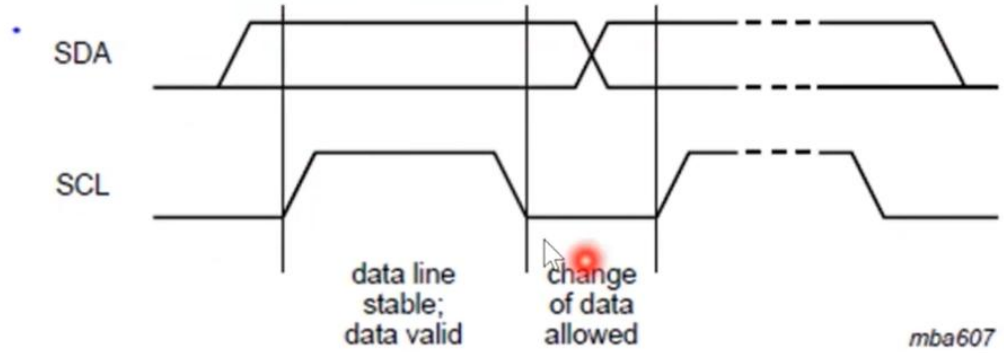


Fig 4. Bit transfer on the I^2C-bus

START and STOP conditions

- Start and stop conditions are always generated by master.
- All transactions begin with START and are terminated by STOP.

- SCL - 1, SDA – 1->0 ➔Start condition
- SCL - 1, SDA – 0->1 ➔Stop condition

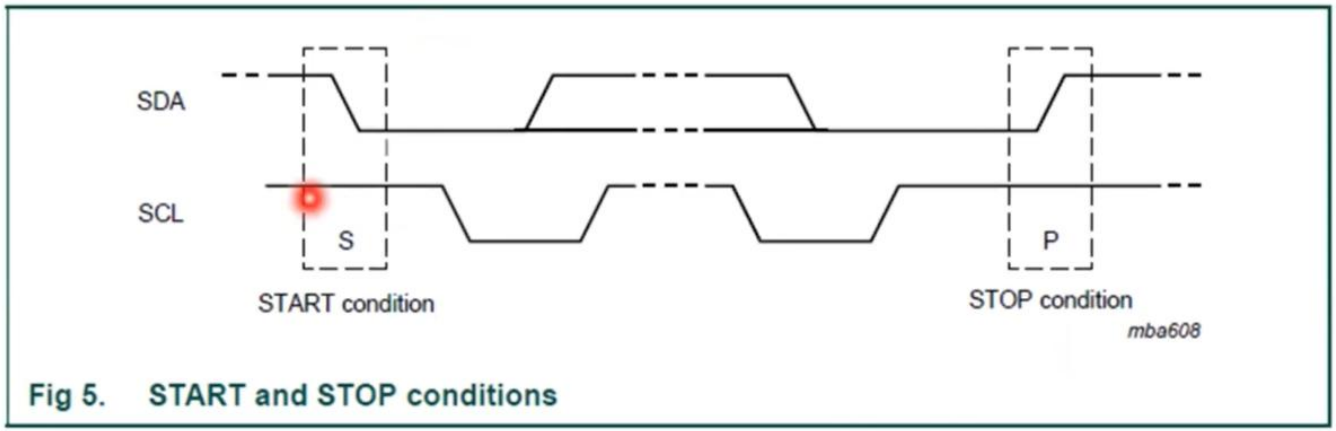


Fig 5. START and STOP conditions

Byte format

- Each byte put on SDA line must be 8-bits long.
- Number of bytes transmitted per transfer is unrestricted.
- Each byte must be followed by acknowledge bit.

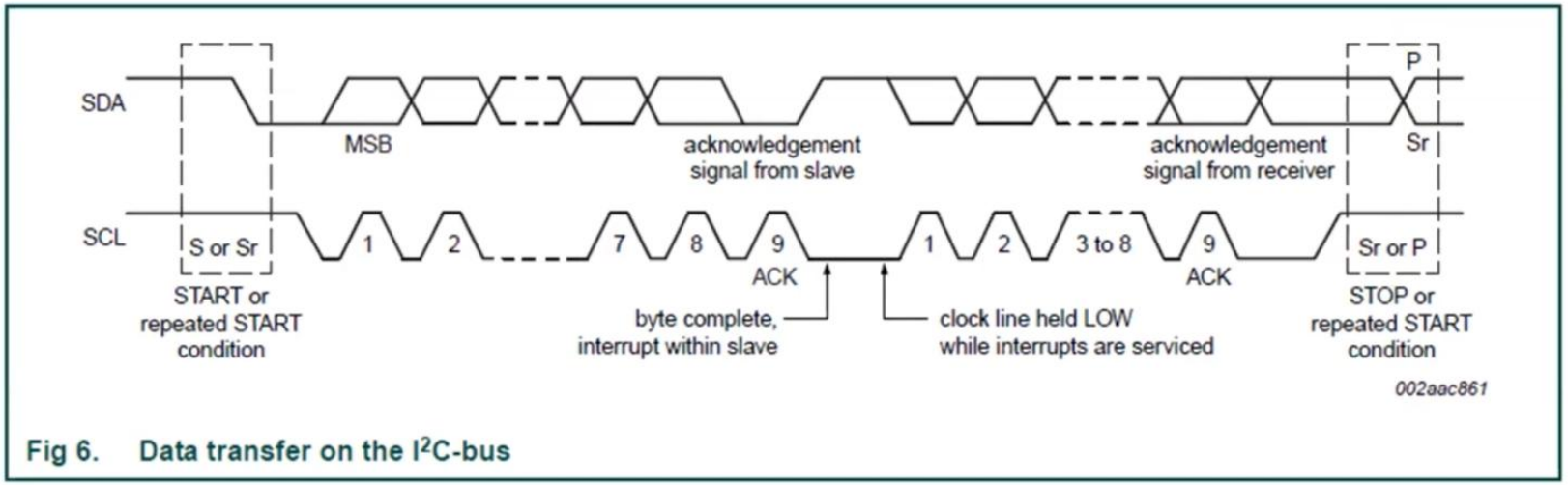


Fig 6. Data transfer on the I²C-bus

Acknowledge (ACK) and Not Acknowledge (NACK)

- ACK bit takes place after every byte.
- ACK bit allows the receiver to signal the transmitter that the byte was successfully received and another byte may be sent.
- ACK signal:
  - Transmitter releases the SDA line during the ACK clock pulse so the receiver can pull SDA line low and remains stable low during the HIGH period of SCL clock pulse
- If SDA remains high during 9th clock pulse:
  - Defined as not acknowledge signal.
  - Master can generate either STOP condition to abort the transfer or repeated START condition to start a new transfer.

Acknowledge (ACK) and Not Acknowledge (NACK)

- Following condition that may lead the generation of NACK:
  - If No receiver is present on the bus with the transmitted address.
  - If receiver is unable to receive or transmit because performing some real time functions and it is not ready to start communication with master.
  - During the transfer, if receiver gets data or command that it does not understand.
  - During transfer, if receiver can not receive any more data bytes.

Clock synchronization

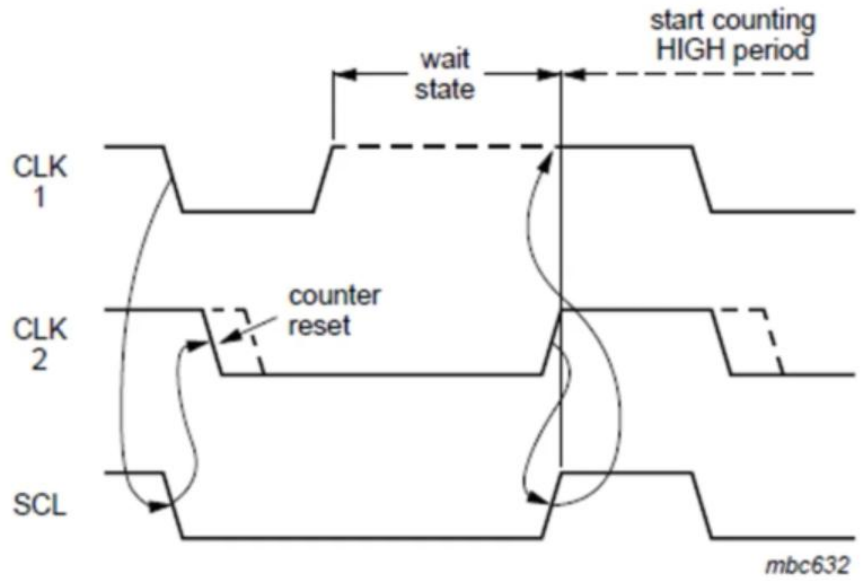


Fig 7. Clock synchronization during the arbitration procedure

Arbitration

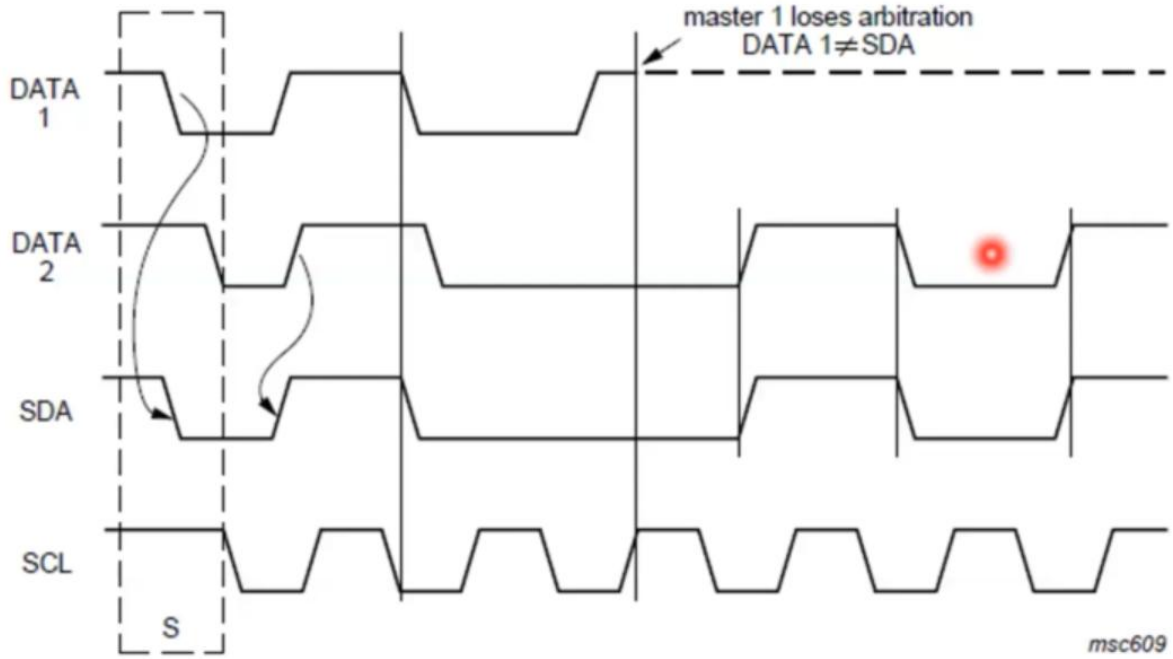


Fig 8. Arbitration procedure of two masters

Clock stretching

- Clock stretching pauses a transaction by holding the SCL line LOW.
- Its an optional feature
- Clock stretching is defined at following levels
  - On the byte level
  - On the bit level

Data transfer formats

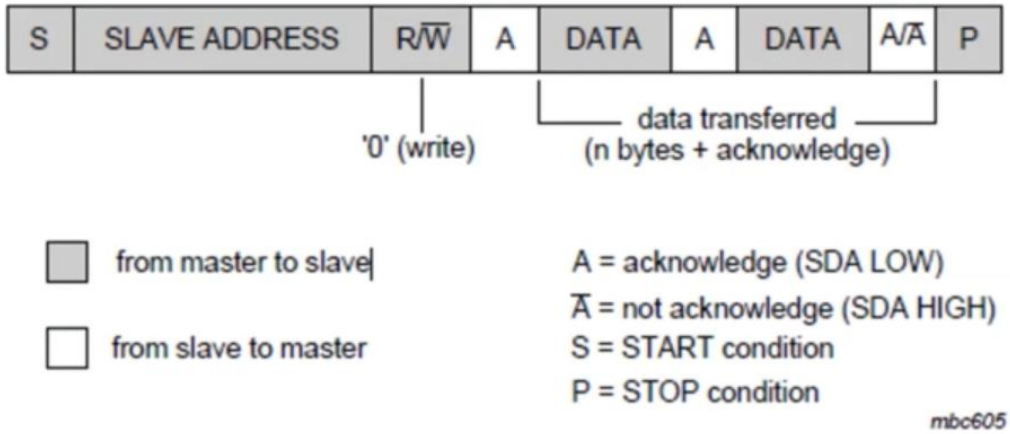


Fig 11. A master-transmitter addressing a slave receiver with a 7-bit address (the transfer direction is not changed)

Data transfer formats (Contd..)

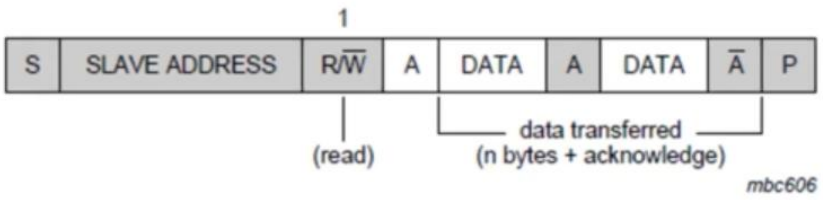


Fig 12. A master reads a slave immediately after the first byte

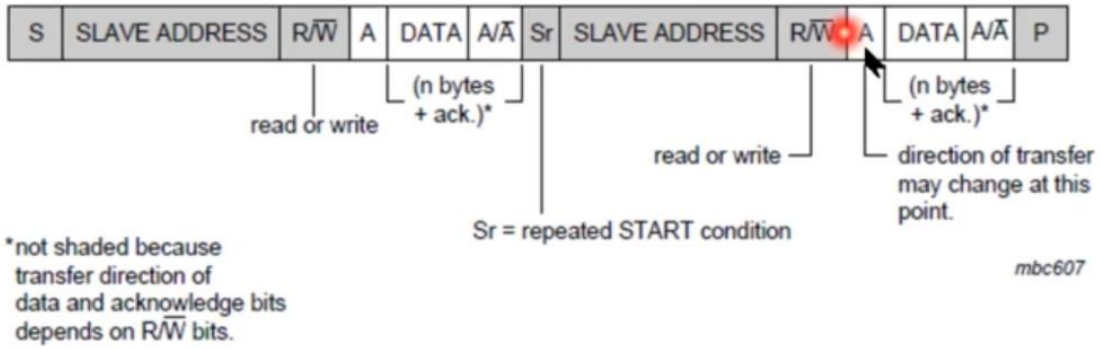


Fig 13. Combined format

10-bit addressing

- It expands the number of possible addresses
- Devices with 7-bit and10-bit addresses can be connected to the same I2C-bus, and both 7-bit and 10-bit addressing can be used in all bus speed modes.
- The 10-bit slave address is formed from the first two bytes following a START condition(S) or a repeated START condition(Sr).
- The first seven bits of the first byte are the combination 1111 0XX of which the last two bits(XX) are the two Most-Significant Bits (MSB) of the 10-bit address; the eighth bit of the first byte is the R/W bit that determines the direction of the message.

10-bit addressing (Contd..)

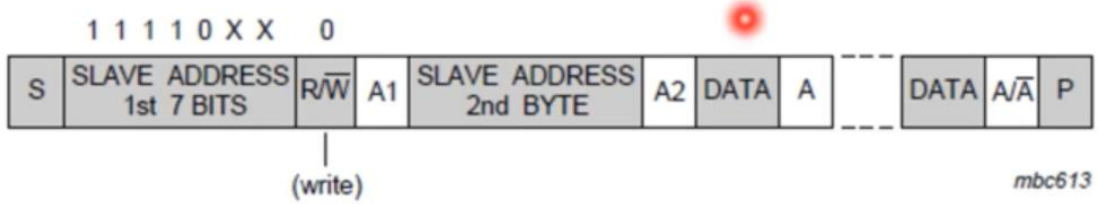


Fig 14. A master-transmitter addresses a slave-receiver with a 10-bit address

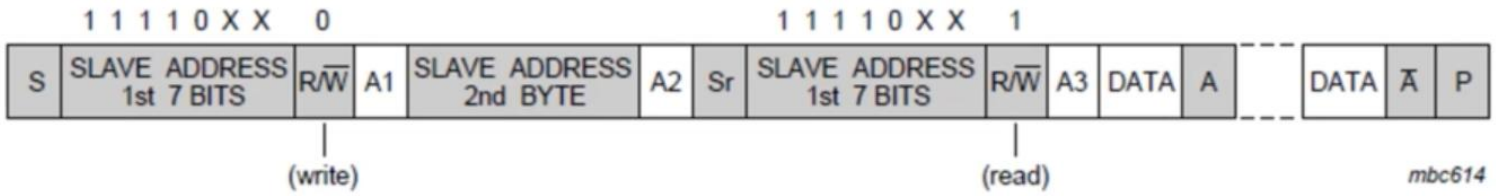


Fig 15. A master-receiver addresses a slave-transmitter with a 10-bit address

Quick recap

- Multi-master 2 wire bus
  - One data line – SDA
  - One clock line – SCL
  - Master controls clock for slaves
- Complete I2C Transfer

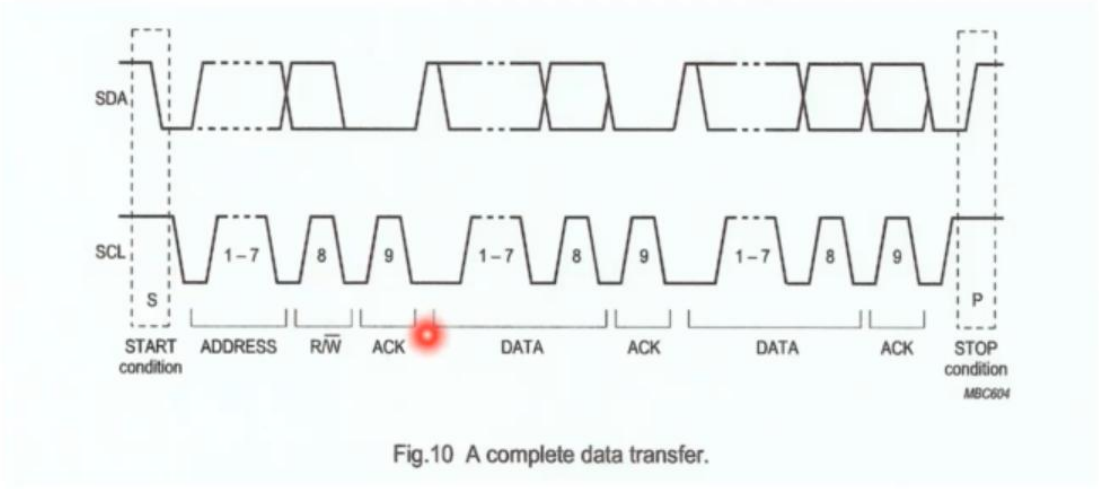
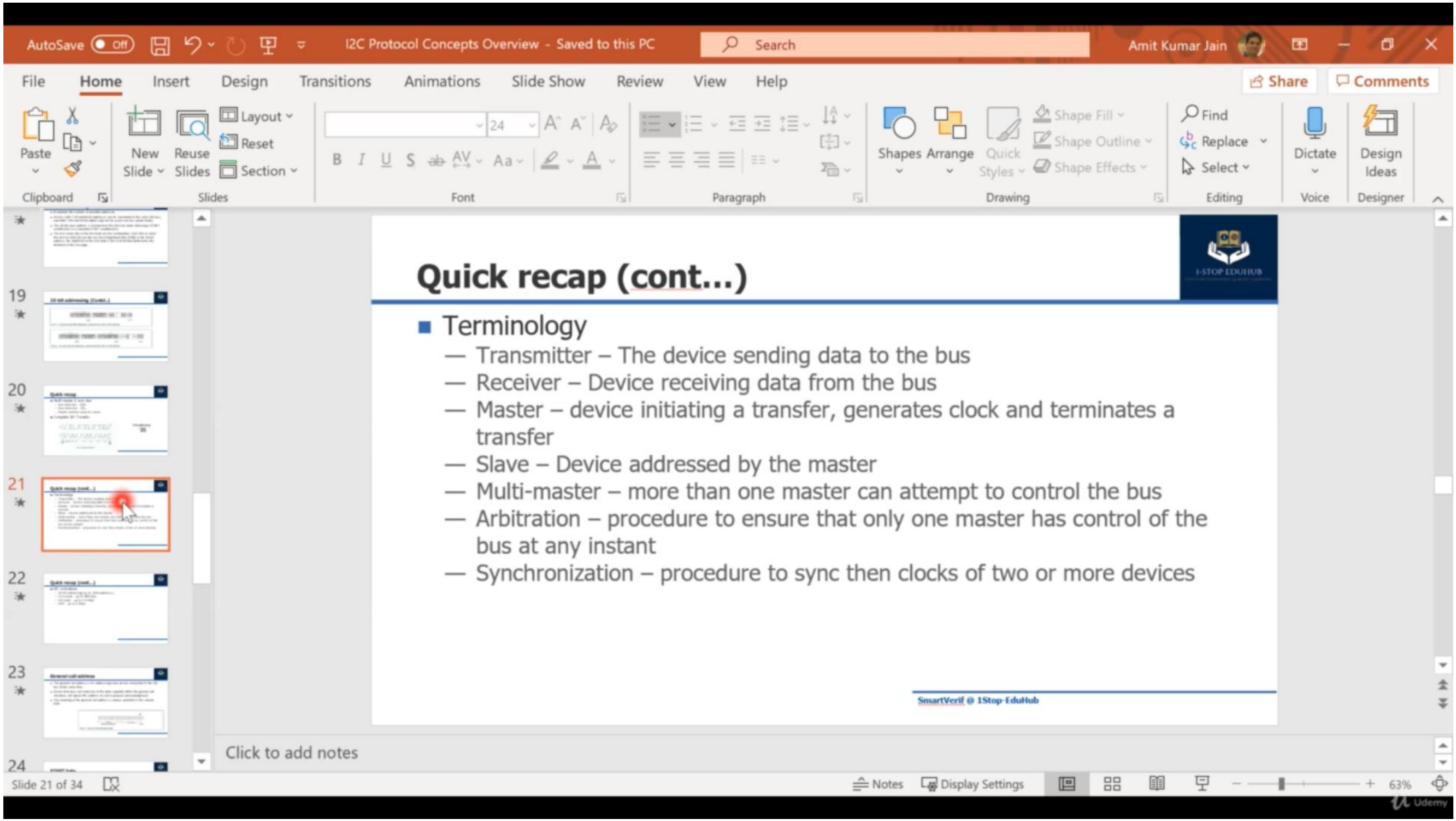


Fig.10 A complete data transfer.

7-bit addressing
- Read
- Write


AutoSave Off
I2C Protocol Concepts Overview - Saved to this PC
Search
Amit Kumar Jain
File Home Insert Design Transitions Animations Slide Show Review View Help
Share Comments
Paste New Slide Reuse Slides Layout Reset Section
Clipboard Slides Font Paragraph Drawing Editing Voice Designer
Shapes Arrange Quick Styles Shape Fill Shape Outline Shape Effects
Find Replace Select Dictate Design Ideas
Quick recap (cont...)
Terminology
— Transmitter – The device sending data to the bus
— Receiver – Device receiving data from the bus
— Master – device initiating a transfer, generates clock and terminates a transfer
— Slave – Device addressed by the master
— Multi-master – more than one master can attempt to control the bus
— Arbitration – procedure to ensure that only one master has control of the bus at any instant
— Synchronization – procedure to sync then clocks of two or more devices
SmartVerif @ 1Stop-EduHub
Click to add notes
Slide 21 of 34
Notes Display Settings
63%
Udemy

Quick recap (cont...)

- I2C extensions
  - 10-bit addressing (up to 1024 addresses)
  - Fast mode – up to 400 Kbps
  - HS mode – up to 3.4 Mbps
  - UFM – up to 5 Mbps

General call address

- The general call address is for addressing every device connected to the I2C-bus at the same time.
- Device that does not need any of the data supplied within the general call structure, can ignore this address by not issuing an acknowledgment.
- The meaning of the general call address is always specified in the second byte.

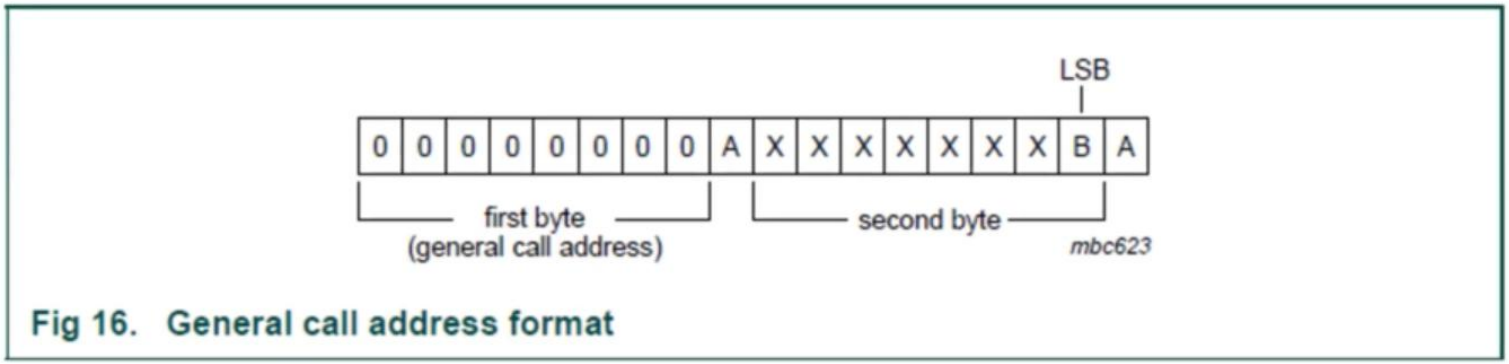


Fig 16. General call address format

START byte

- An interrupt request on which I2C-bus interface can be programmed.
- A START byte (0000 0001) acts as an interrupt
- The start procedure consists of:
  - A START condition (S)
  - A START byte (0000 0001)
  - An acknowledge clock pulse (ACK)
  - A repeated START condition (Sr).

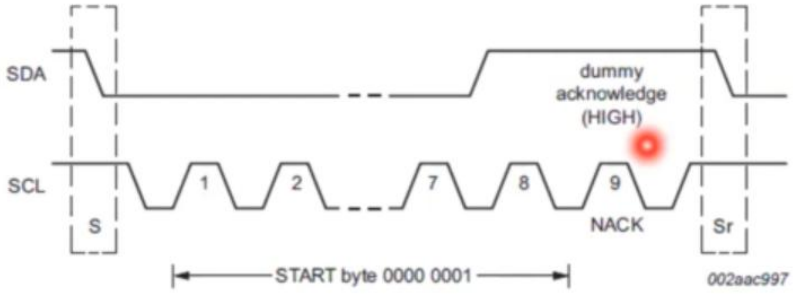


Fig 19. START byte procedure

Device ID

- The Device ID field (see Figure 20) is an optional 3-byte read-only (24 bits) word giving the following information:
- To access device ID master uses 10-bit addressing mode
- The master sends the Reserved Device ID I2C-bus address followed by the R/W bit set to '0' (write): '1111 1000' to indicate transfer is for Device ID.
- The master sends the I2C-bus slave address of the slave device it must identify. The LSB is a 'Don't care' value.
- After Sr the master sends the Reserved Device ID I2C-bus address followed by the R/W bit set to '1' (read): '1111 1001'.

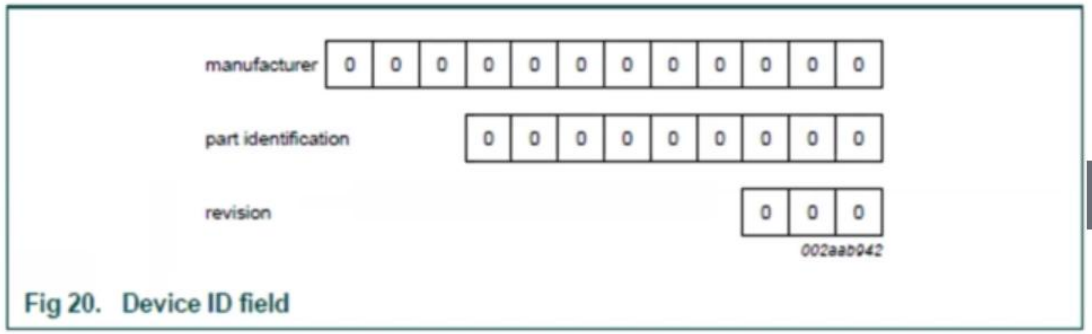
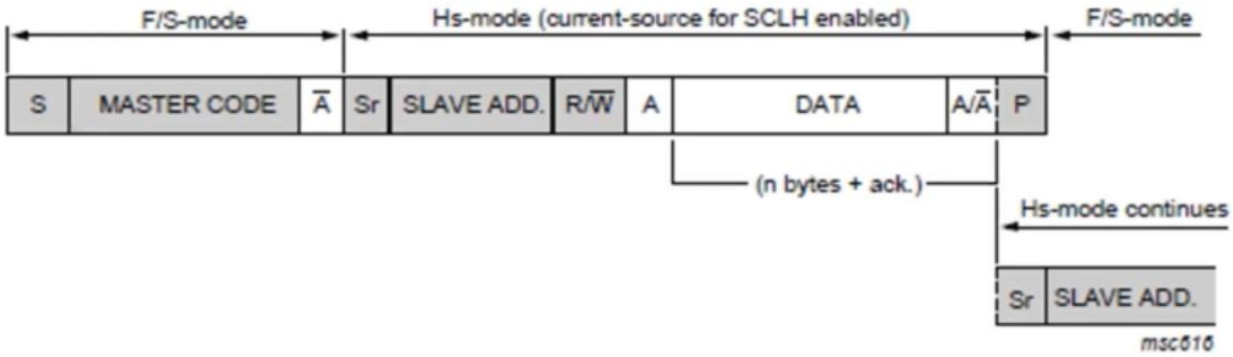


Fig 20. Device ID field

Data transfer format in Hs-mode

- Serial data transfer format in Hs-mode meets the Standard-mode I2C-bus specification.
- Hs-mode can only commence after the following conditions (all of which are in F/S-mode):
  - START condition (S)
  - 8-bit master code (0000 1XXX)
  - Not-acknowledge bit (A)



Ultra Fast-mode I2C-bus protocol

- 2-wire push-pull serial bus that operates from DC to 5 MHz transmitting data in one direction.
- Based on the standard I2C-bus protocol that consists of a START, slave address, command bit, ninth clock, and a STOP bit.
- The command bit is a 'write' only, and the data bit on the ninth clock is driven HIGH, ignoring the ACK cycle due to the unidirectional nature of the bus.
- Slave devices contain a unique address (whether it is a microcontroller, LCD driver, LED controller, GPO) and operate only as receivers. An LED driver may be only a receiver and can be supported by UFm

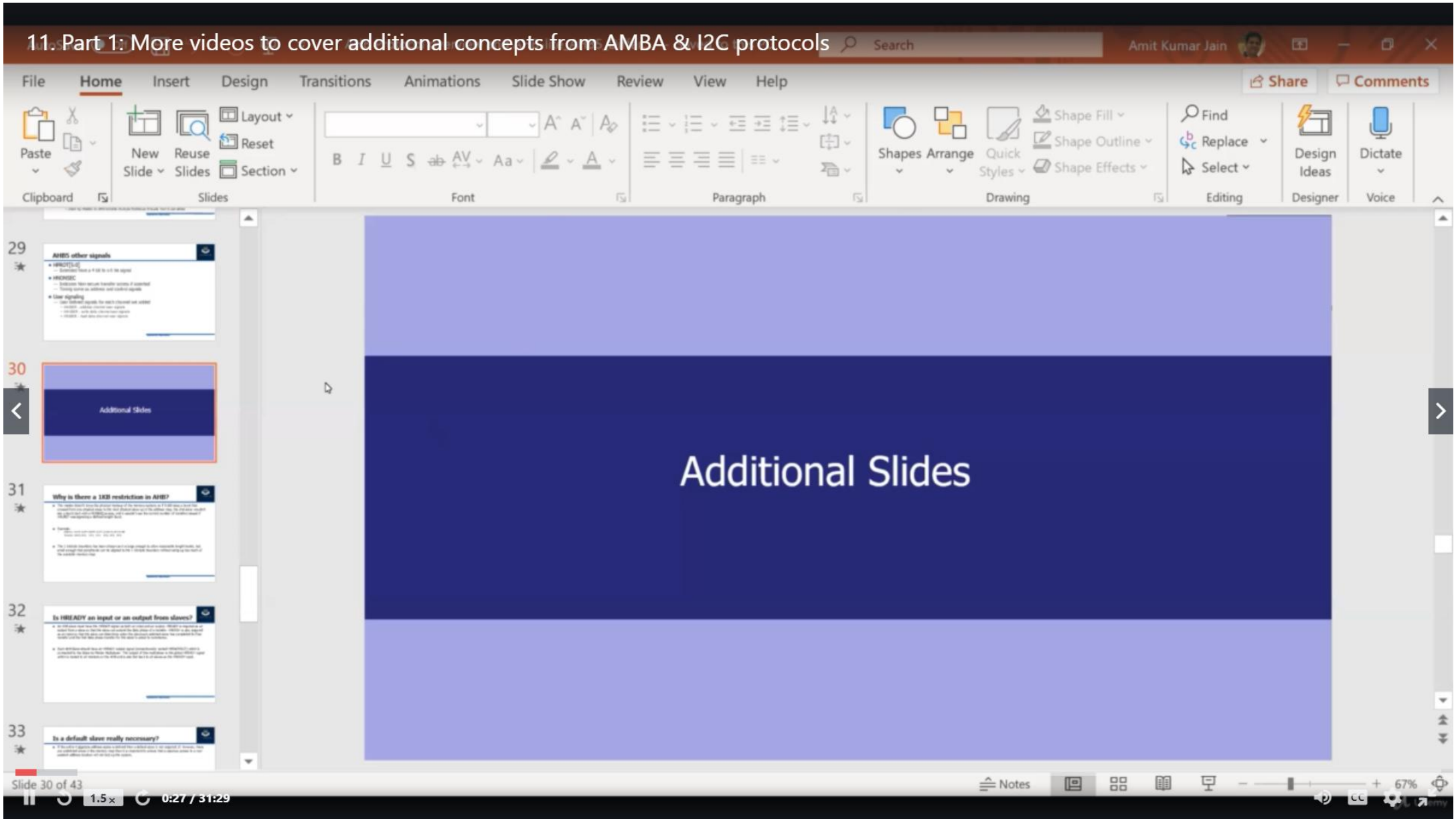
Ultra Fast-mode I2C-bus protocol (Contd..)

Feature	Configuration
	Single master
START condition	M
STOP condition	M
Acknowledge	n/p
Synchronization	n/p
Arbitration	n/p
Clock stretching	n/p
7-bit slave address	M
10-bit slave address	O
General Call address	O
Software Reset	O
START byte	O
Device ID	n/p

I2C Sample Assertions (Protocol Check)

1. All I2C interface signals SCLK and SDATA must not be x or z.
2. SCL and SDA must be high and SCL should not toggle, when the bus is IDLE.
3. Except Start byte transfer, every start signal must be followed by at least 1 data phase.
4. On the second byte of General call (GCALL) address, 8'h00 is not allowed.
5. No device is allowed to acknowledge the start byte or generate NACK.
6. Start byte transfer must be followed by repeated start condition as it does not contain any data phase.
7. Master must restart/stop the transaction if slave issues NACK.
8. Every data phase in the serial transfer must be only 8-bits wide.
9. Reserved address (HS mode master code), 8'b0000_1000 is not allowed in HS mode.
10. In 10-bit addressing, the R/W bit in 1st part of starting address must be zero.
11. Ultra fast mode can only have R/W bit set to zero except for start byte.

11. Part 1: More videos to cover additional concepts from AMBA & I2C protocols



Why is there a 1KB restriction in AHB?

- The master doesn't know the physical makeup of the memory system, so if it did issue a burst that crossed from one physical slave to the next physical slave up in the address map, the 2nd slave wouldn't see a burst start with a NONSEQ access, and it wouldn't see the correct number of transfers issued if HBURST was signaling a defined length burst.

- Example:
  - Address: 0x3F0 0x3F4 0x3F8 0x3FC 0x400 0x404 0x408
    Transfer: NSEQ SEQ SEQ SEQ SEQ SEQ SEQ

- The 1 kilobyte boundary has been chosen as it is large enough to allow reasonable length bursts, but small enough that peripherals can be aligned to the 1 kilobyte boundary without using up too much of the available memory map.

Is HREADY an input or an output from slaves?

- An AHB slave must have the HREADY signal as both an input and an output. HREADY is required as an output from a slave so that the slave can extend the data phase of a transfer. HREADY is also required as an input so that the slave can determine when the previously selected slave has completed its final transfer and the first data phase transfer for this slave is about to commence.

- Each AHB Slave should have an HREADY output signal (conventionally named HREADYOUT) which is connected to the Slave-to-Master Multiplexer. The output of this multiplexer is the global HREADY signal which is routed to all masters on the AHB and is also fed back to all slaves as the HREADY input.

Is a default slave really necessary?

- If the entire 4 gigabyte address space is defined then a default slave is not required. If, however, there are undefined areas in the memory map then it is important to ensure that a spurious access to a non-existent address location will not lock up the system.

Protection Control bits in AHB5

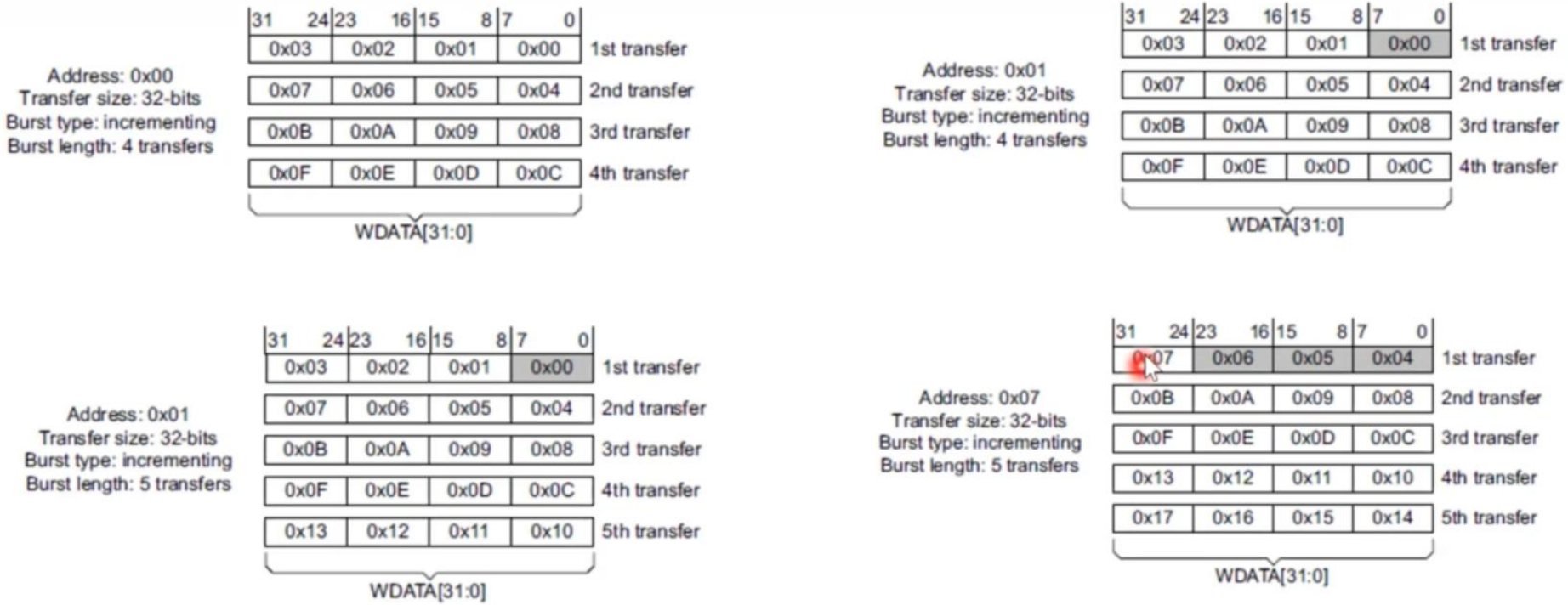
Bit	Name	Description
HPROT[0]	Data/Inst	When asserted, this bit indicates the transfer is a data access. When deasserted this bit indicates the transfer is an instruction fetch.
HPROT[1]	Privileged	When asserted, this bit indicates the transfer is a privileged access. When deasserted this bit indicates the transfer is an unprivileged access.
HPROT[2]	Bufferable	If both of **HPROT[4:3]** are deasserted then, when this bit is: • Deasserted, the write response must be given from the final destination. • Asserted, the write response can be given from an intermediate point, but the write transfer is required to be made visible at the final destination in a timely manner.
HPROT[3]	Modifiable	When asserted, the characteristics of the transfer can be modified. When deasserted the characteristics of the transfer must not be modified.
HPROT[4]	Lookup	When asserted, the transfer must be looked up in a cache. When deasserted, the transfer does not need to be looked up in a cache and the transfer must propagate to the final destination.
HPROT[5]	Allocate	When asserted, for performance reasons, this specification recommends that this transfer is allocated in the cache. When deasserted, for performance reasons, this specification recommends that this transfer is not allocated in the cache.
HPROT[6]	Shareable	When asserted, indicates that this transfer is to a region of memory that is shared with other masters in the system. A response for the transfer must not be provided until the transfer is visible to other masters. When deasserted, indicates that this transfer is Non-shareable and the region of memory is not shared with other masters in the system. A response for the transfer does not guarantee the transfer is visible to other masters.

HPROT[6] Shareable	HPROT[5] Allocate	HPROT[4] Lookup	HPROT[3] Modifiable	HPROT[2] Bufferable	Memory Type
0	0	0	0	0	Device-nE
0	0	0	0	1	Device-E
0	0	0	1	0	Normal Non-cacheable, Non-shareable
0	0 or 1	1	1	0	Write-through, Non-shareable
0	0 or 1	1	1	1	Write-back, Non-shareable
1	0	0	1	0	Normal Non-cacheable, Shareable
1	0 or 1	1	1	0	Write-through, Shareable
1	0 or 1	1	1	1	Write-back, Shareable

AXI Unaligned transfers

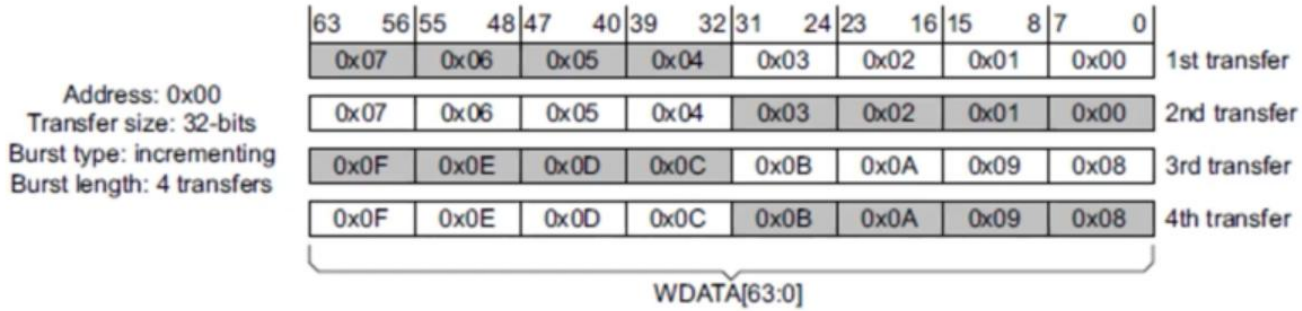
- For burst that is made up of data transfers wider than one byte, 1st byte might be unaligned with the natural address boundary.
- Example: 32-bit data packet that starts at a byte address of 0x1002 is not aligned to the natural 32-bit address boundary.

AXI Unaligned transfers (Examples)



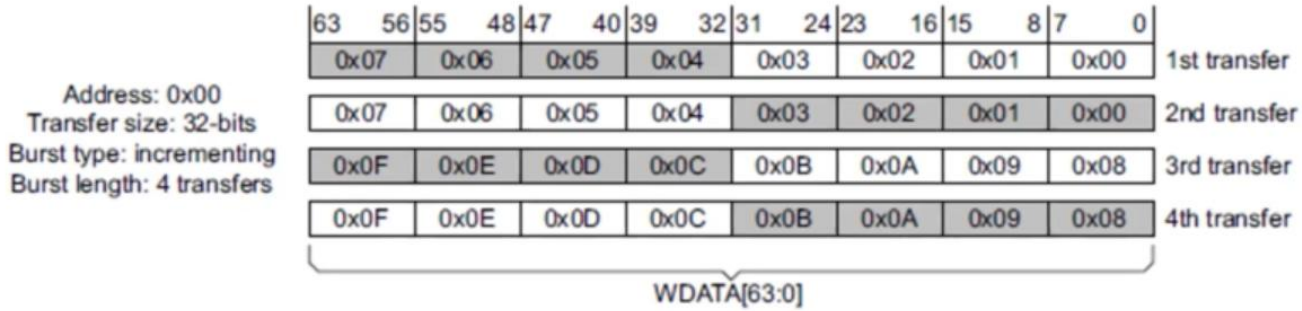
Incrementing bursts, Aligned and unaligned transfers on 32-bit bus

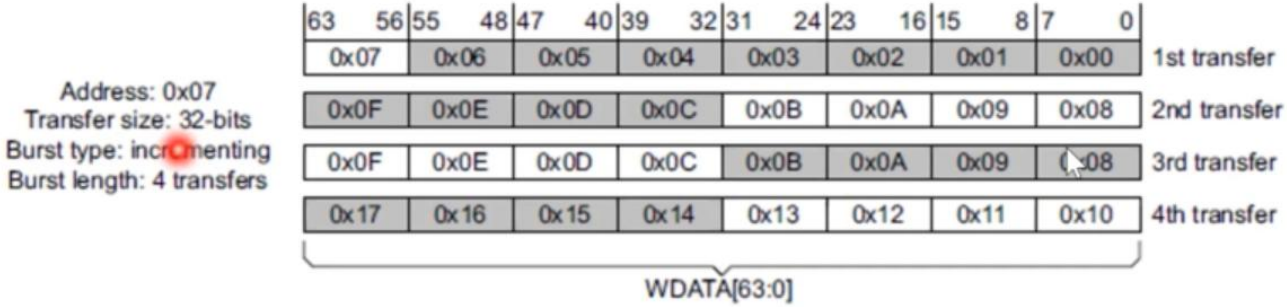
AXI Unaligned transfers (Examples)



Incrementing bursts, Aligned and unaligned transfers on 64-bit bus

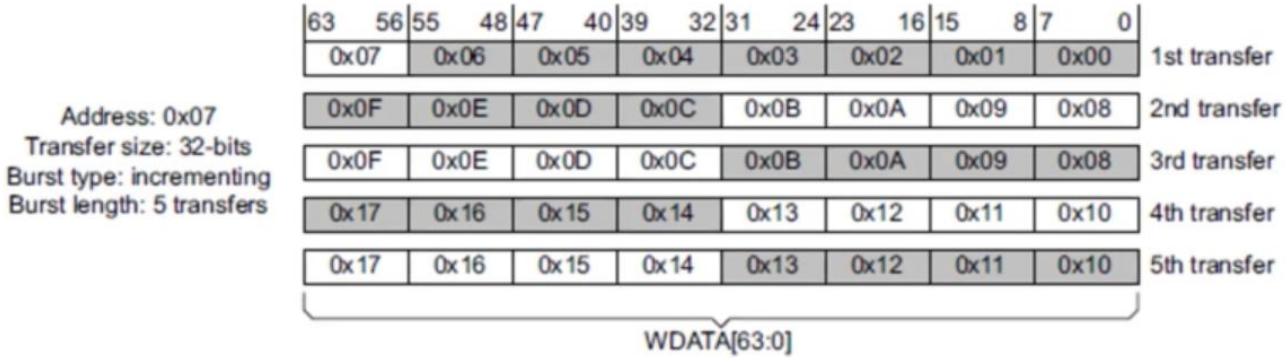
AXI Unaligned transfers (Examples)



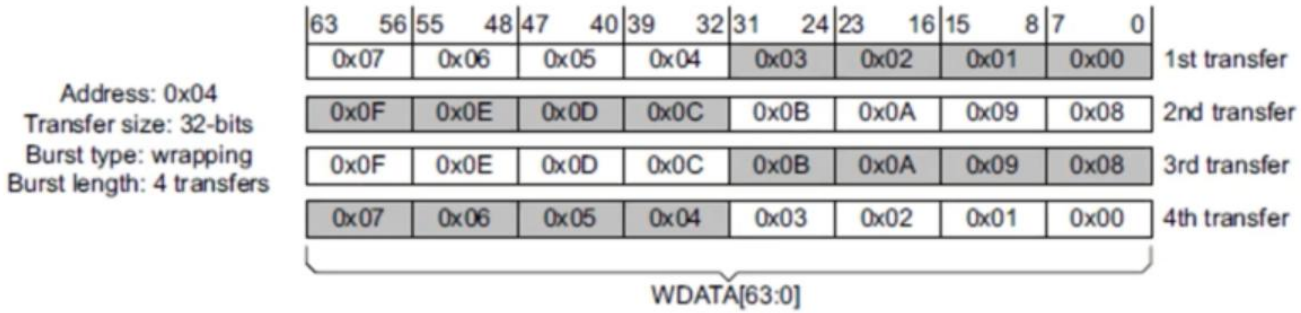


Incrementing bursts, Aligned and unaligned transfers on 64-bit bus

AXI Unaligned transfers (Examples)



Incrementing bursts, Aligned and unaligned transfers on 32-bit bus



Wrapping bursts, Aligned transfers on 64-bit bus

Functional Verification

Burst Transfer and Types – AHB & AXI

Team SmartVerif @ 1Stop-EduHub

Apr 2, 2020

I-STOP EDUHUB

FOCUSED TO DELIVER QUALITY LEARNING

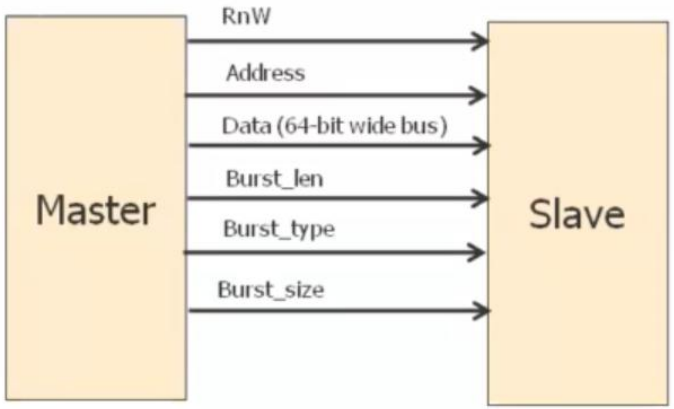
Udemy

Outline

- Single & Burst Transfers
- Address calculations in burst – Burst Types
- Single & Burst transfer examples
  - AHB
  - AXI
- WRAP Burst – Example and usage application
- FIXED Burst – Example and usage application

Single and Burst Transfers

Bus Interface



Request --- =RnW=WR...Addr=0
Write data --- Data(8 Byte)
Response

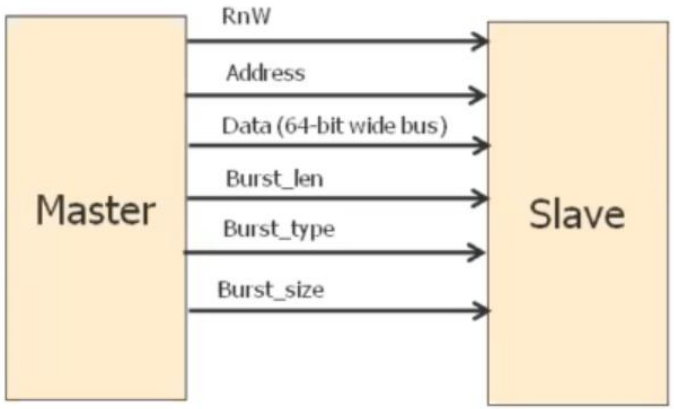
Single transfer

Request --- RnW=WR ... Addr=0
Write Data D0 (8B)
D1 (8B)
D2 (8B)
D3 (8B)
Response

Burst transfer

Single and Burst Transfers

Bus Interface



Single Transfer??

Burst Transfer??

Request --- =RnW=WR...Addr=0
Write data --- Data(8 Byte)
Response

Single transfer

Starting from address 0, 8B data is written

0 1 2 3 4 5 6 7

Request --- RnW=WR ... Addr=0
Write Data D0 (8B)
D1 (8B)
D2 (8B)
D3 (8B)
Response

Burst transfer

Start address - 0

0 1 7
8 9 a....f
10 ... 17
18 .... 1f

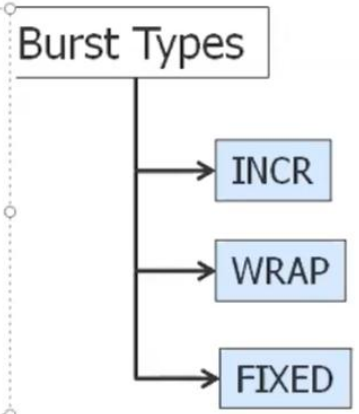
Because address is incrementing serially, we calll it INCR burst
This is burst type....

Address calculations in burst – Burst Types

A simple example:
Transfer type – Write
Start address – 0x01
Data bus width – 4 bytes (32-bit)
Burst Length – 4

D1 - d13 d12 d11 d10 - 3B
D2 - d23 d22 d21 d20 - 4B
D3 - d33 d32 d31 d30 - 4B
D4 - d43 d42 d41 d40 - 4B

Starting from address 1, 31 bytes of data is written
1 2 3 4 5 6 7 8 9 a b c d e f
10 11 12 13 .....1f
INCR



Address calculations in burst – Burst Types

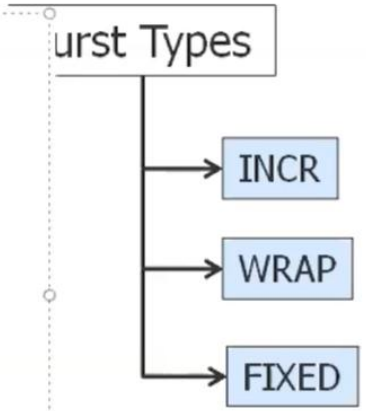
A simple example:
Transfer type – Write
Start address – 0x01
Data bus width – 4 bytes (32-bit)
Burst Length – 4

1. What does burst lengt
2. What is burst size?

Data bus width - 4 bytes
Out of total bytes in a beat, how many bytes are valid...Burst size...

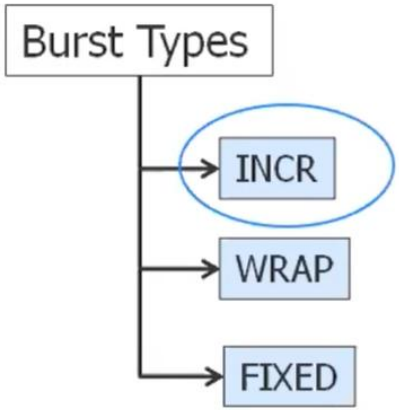
Burst size - 2 bytes

Start addr = 0
D1 - d13 d12 d11 d10
D2
D3
D4
2 + 2 + 2 + 2 = 8B
0 1 2 3 7



Address calculations in burst – Burst Types

A simple example:
Transfer type – Write
Start address – 0x01
Data bus width – 4 bytes (32-bit)
Burst Length – 4

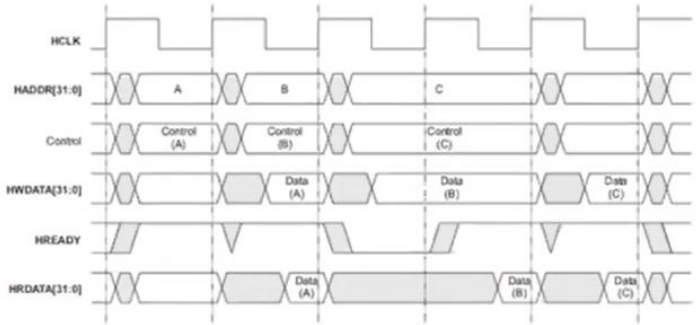


1. What does burst length represent?
2. What is burst size?
3. What is burst type in above example?
4. What are other burst types?
5. What is the usage application for INCR burst type?

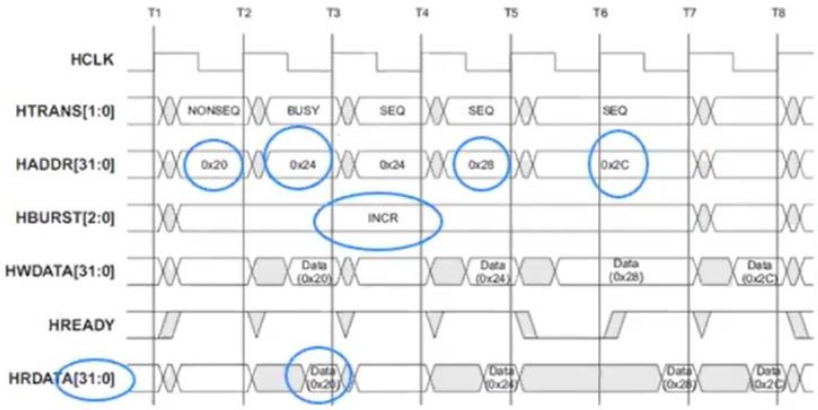
- Normal physical DRAM memory
- Say byte addressable

Single & Burst transfer example – AHB & AXI

AHB

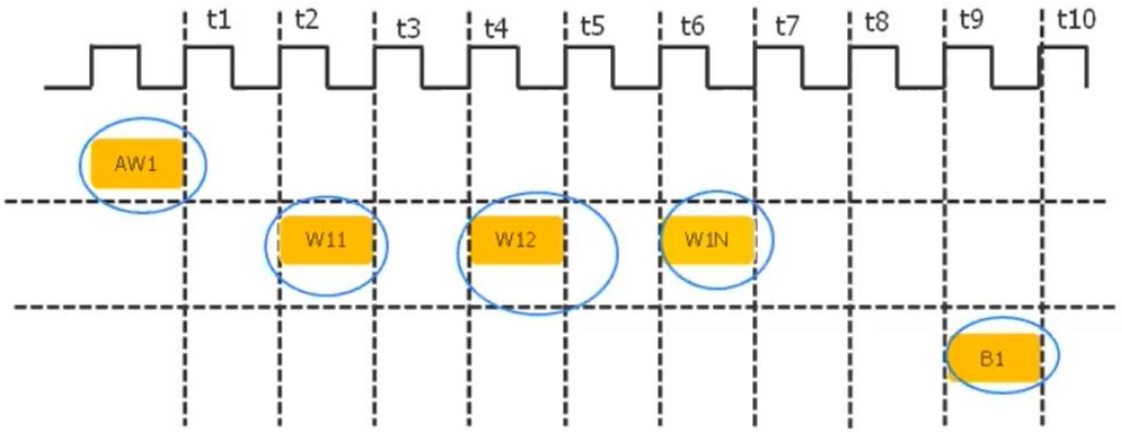


Single Transfer



Burst Transfer

Single & Burst transfer example (cont...)



Burst WR transfer - AXI

WRAP Burst – example and usage application

Example 1 –
WR, Addr=0, BL=4,
Data Bus Width = 4 Bytes
Burst Size = 4 Bytes
Burst Type = WRAP

Example 2 –
WR, Addr=4, BL=4,
Data Bus Width = 4 Bytes
Burst Size = 4 Bytes
Burst Type = WRAP

D1 -
D2 -
D3 -
D4 -

16 bytes transferred from addr 0

0 1 2 3 4 5 6 7 8 9 a b c d e f

16 bytes transferred from addr 4

4 5 6 7 8 9 a b c d e f 0 1 2 3

WRAP Boundary - Start and End
To calculate End = (start_addr%Total_data_byte)+Total_data_byte
Start = start_addr-(start_addr%Total_data_byte)
Total_data_byte = 4*(4) = 16
End Boundary = (0%16)+16 = F
Start Boundary = 0-(0%16) = 0

WRAP Burst – example and usage application

Example 1 –
WR, Addr=0, BL=4,
Data Bus Width = 4 Bytes
Burst Size = 4 Bytes
Burst Type = WRAP

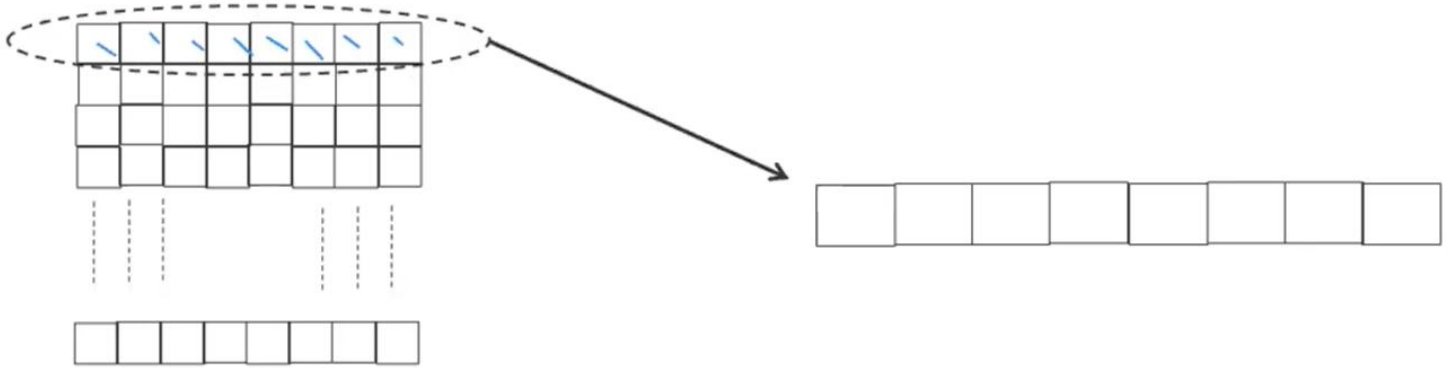
Example 2 –
WR, Addr=4, BL=4,
Data Bus Width = 4 Bytes
Burst Size = 4 Bytes
Burst Type = WRAP

WRAP addr boundary calculations

Start address alignment to Burst Size??

1. How do we calculate WRAP address boundary – START and END?
2. Starting address – It should always be aligned to size of the transfer. Explain?

WRAP Burst – Usage in cache memory access



FIXED Burst – example and usage application

Example 1 –
WR, Addr=0, BL=4,
Data Bus Width = 4 Bytes
Burst Size = 4 Bytes
Burst Type = FIXED

D1
D2
D3
D4

1. How does address increment across the beats in a burst?
2. How does address increment within a beat?

Usage application:

1. Used in FIFO memory
2. Every address location has FIFO with say each location equal to data bus width
3. Next beat in burst cause rd/wr pointer to increment in FIFO