VCI Interface and AMBA Bus

Outline

- VCI Interface Standards
- AMBA - On Chip Buses
- AMBA 3.0 - AXI

Outline

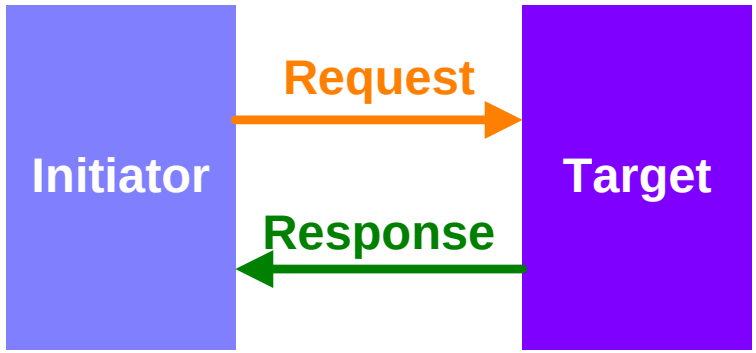
- VCI Interface Standards
- AMBA - On Chip Buses
- AMBA 3.0 - AXI

Virtual Component Interface - VCI

- What is VCI
  - A request-response protocol, contents and coding, for the transfer of requests and responses
- Why VCI
  - Other IP blocks not available ‘wrapped’ to the on-chip communications may work with IP wrappers. VCI is the best choice to start with for an adaptation layer
- VCI specifies
  - Thee levels of protocol, compatible each other
    - Advanced VCI (AVCI),
    - Basic VCI (BVCI)
    - Peripheral VCI (PVCI)
  - Transaction language

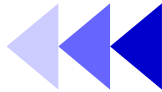
VCI Point-to-Point Usage

- Simplicity: small footprint and high bandwidth
  - Initiator only request
  - Target only respond
  - If a VC needs both, implement parallel initiator and target interfaces
- Star topology

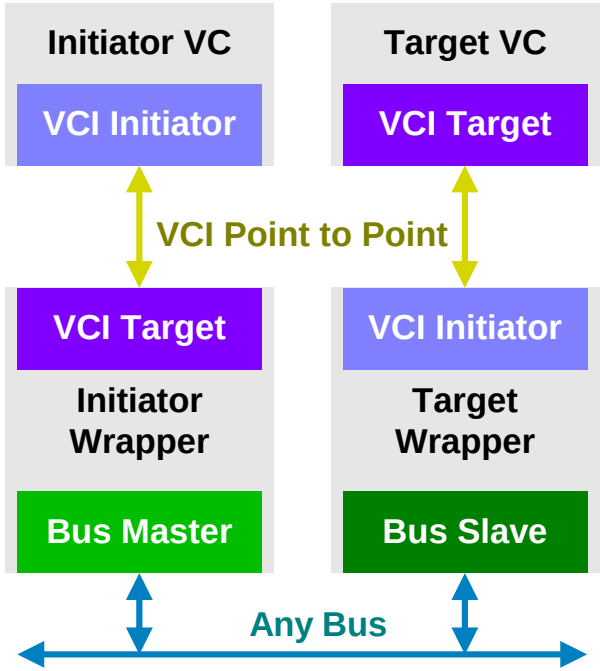


Point-to-point usage

VCI Usage with a Bus



- Used as the interface to a wrapper (a connection to a bus)
  - OCB suppliers provide VCI wrappers.
  - EDA vendors provide tools to create wrapper automatically



VCI usage with a bus

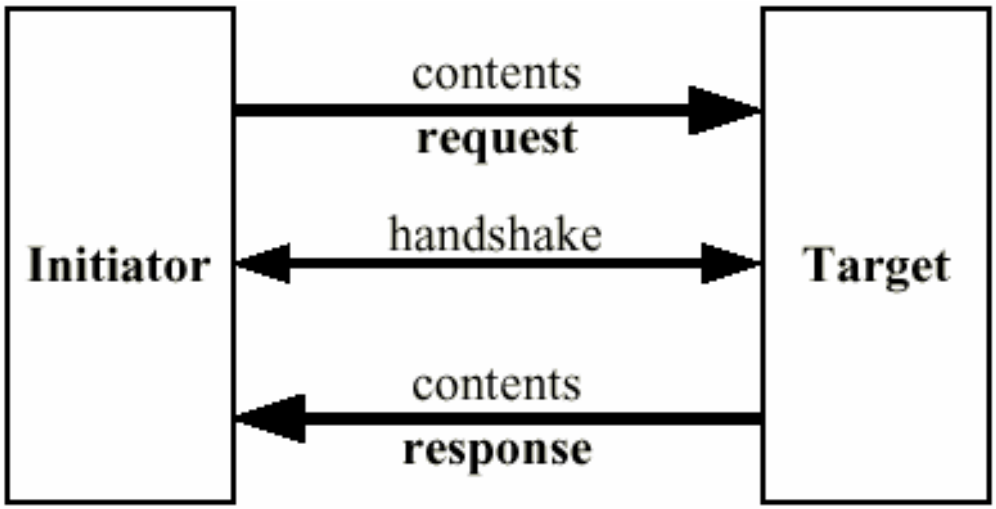
Split Protocol

- The timing of the request and the response are fully separate. The initiator can issue as many requests as needed, without waiting for the response.

BVCI	order kept
AVCI	request tagged with identifiers, allow different order
PVCI	no split protocol
	each request must be followed by a response before the initiator can issue a new request

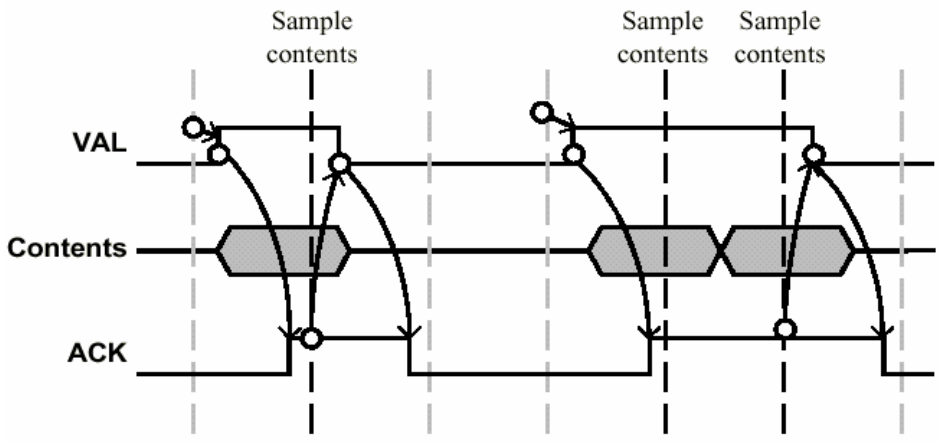
Initiator – Target Connection (PVCI)

- The request contents and the response contents are transferred under control of the protocol: 2-wire handshake Valid (VAL) and Acknowledge (ACK)

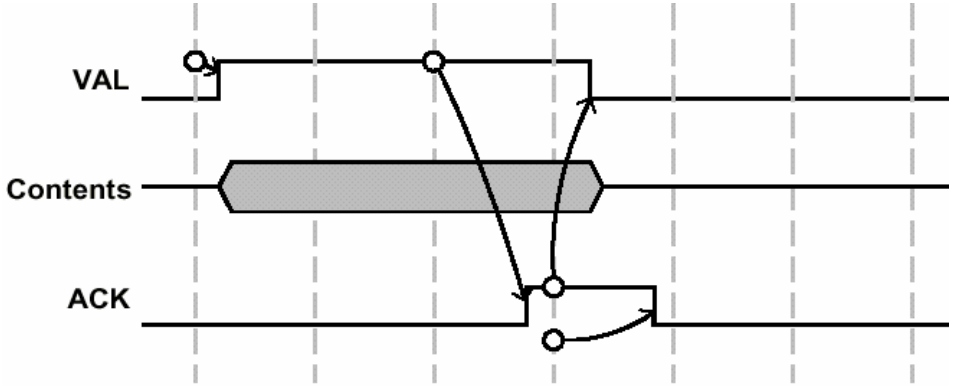


Control Handshake

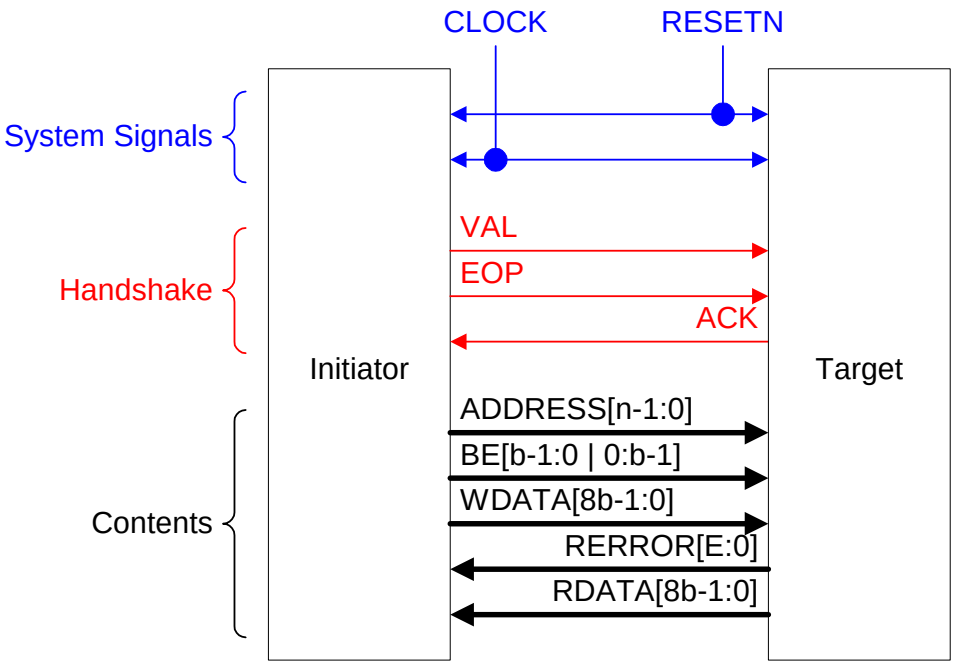
- Asynchronous



- Synchronous



Request and Response Contents



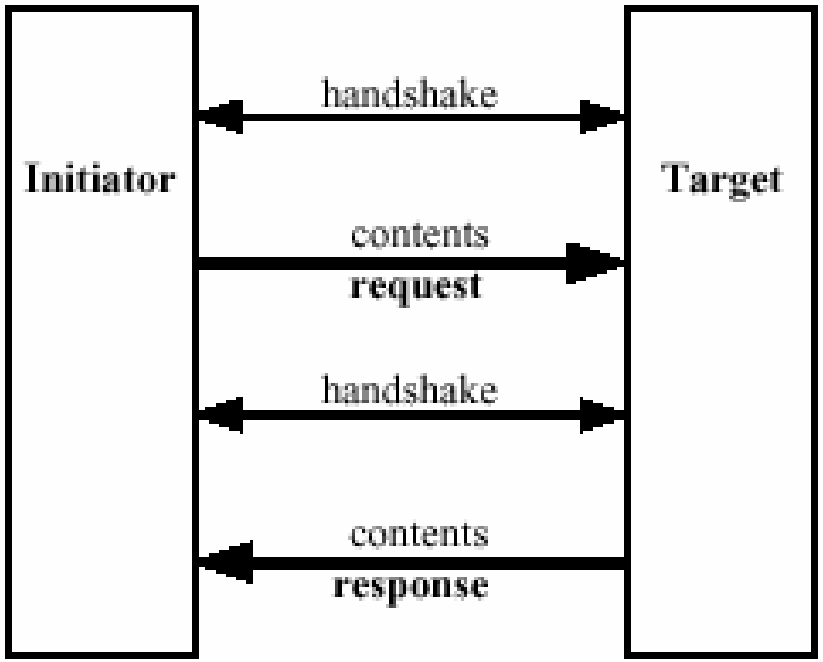
- Main PVCI features
  - Up to 32-bit Address
  - Up to 32-bit Read Data
  - Up to 32-bit Write Data
  - Synchronous
  - Allows for 8-bit, 16-bit, and 32-bit devices
  - 8-bit, 16-bit, and 32-bit Transfers
  - Simple packet, or ‘burst’ transfer

PVCI Protocol

- Transfer Request
  - Read8, Read16, Read32, Read N cells
  - Write8, Write16, Write32, Write N cells
- Transfer Response
  - Not Ready
  - Transfer Acknowledged
  - Error
- Packet Transfer
  - The packet (burst) transfer makes is to transfer a block of cells with consecutive addresses
  - While the EOP signal is de-asserted during a request, the address of the next request will be ADDRESS+cell_size

Initiator – Target Connection (BVCI)

- The request and response handshakes are independent of each other
  - Request handshake: CMDVAL and CMDACK
  - Response handshake: RSPVAL and RSPACK



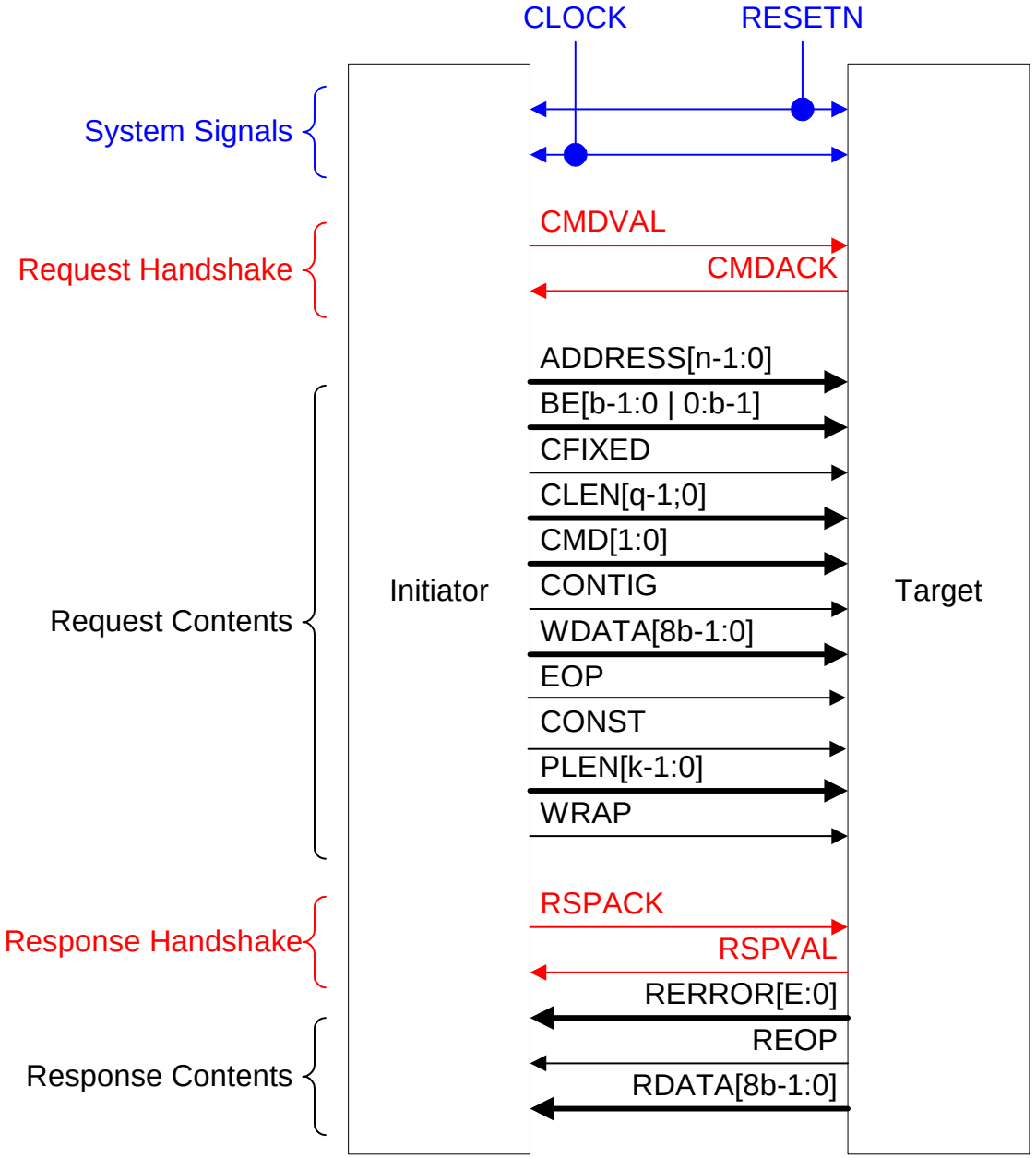
Cells, Packets, and Packet Chains

- Each handshake transfers a cell across the interface. The cell size is the width of the data passing across a VCI.
  - 1, 2, 4, 8, or 16 bytes for BVCI
  - 1, 2, 4, bytes for PVCI
- Cell transfers can be combined into packets, which may map onto a burst on a bus.
  - A VCI operation consists of a request packet and a response packet
  - Packets are atomic
  - Packets are similar in concept to “frames” in PCI
- Packets can be combined into chains, to allow longer chains of operations to go uninterrupted.

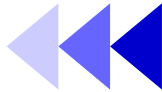
Request and Response Contents

- Request contents are partitioned into three signal groups and validated by the CMDVAL signal
  - Opcode, specify the nature of the request (read or write)
  - Packet Length and Chaining
  - Address and Data
- Response contents validated with the RSPVAL. Each request has its response.
  - Response Error
  - Read Data

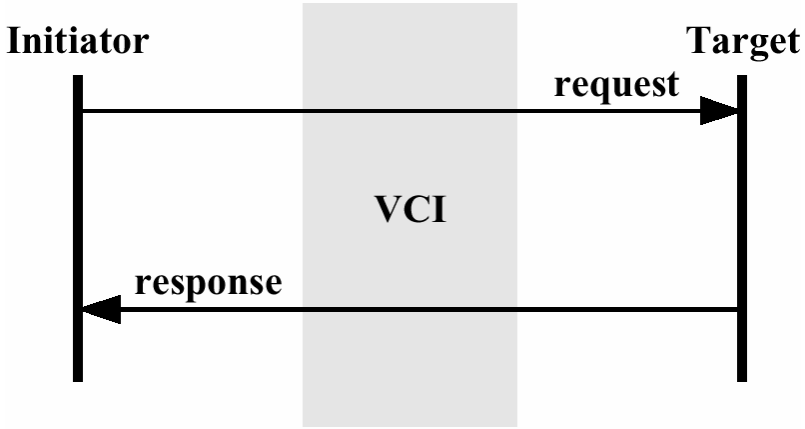
BVCI Signals



BVCI Protocol

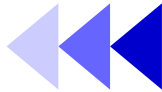


- The protocol has three stacked layers: transaction layer, packet layer, and cell layer
- Transaction layer: A pair of request and response transfers

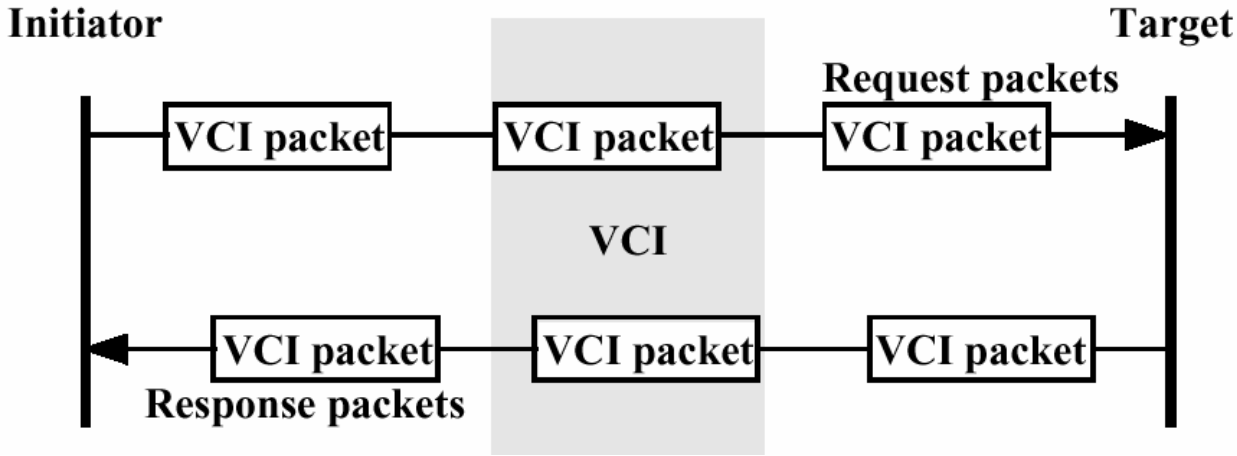


  - Above hardware implementation
  - A series of communicating objects that can be either hardware or software modules
  - The information exchanged between initiator and target nodes is in the form of a request-response pair

Packet Layer



- The packet layer adds generic hardware constraints to the system model
- In this layer, VCI is a bus-independent interface, just physically point-to-point



- A transaction is called a “VCI operation” if the information is exchanged using atomic request and response transfers. In a packet layer, a VCI transaction decomposes into one or more operations.

Packet

- Packet is the basic unit of information that can be exchanged over the VCI in an atomic manner.
- Multiple packets can be combined to form larger, non-atomic transfer units called packet chains.
- A VCI operation is a single request-response packet pair.
- Packet length is the number of bytes transferred
- The content of a packet depends on whether it is a request or response packet and the type of operation being carried out - such as read, write, etc.

Cell Layer

- The cell layer adds more hardware details such as interface width, handshake scheme, wiring constraints, and a clock to the system.
- A cell is the basic unit of information, transferred on rising CLOCK edges under the VAL-ACK handshake protocol, defined by the cell layer. Multiple cells constitute a packet.
- Both request and response packets are transferred as series of cells on the VCI. The number of cells in a packet depends on the packet length and the interface width.

BVCI Operations

- The basic transfer mechanism in VCI is packet transfer. A packet is sent as a series of cells with the EOP field in the last cell set to value 1. Each cell is individually handshaken under the VAL-ACK handshake. Either the initiator or the target can insert wait cycles between cell transfers by de-asserting VAL or ACK.

- Transfer Requests
  - Read/Write a cell
  - Read/Write a packet from random/contiguous addresses
  - Read/Write a packet from one address
  - Issue a chain of packets

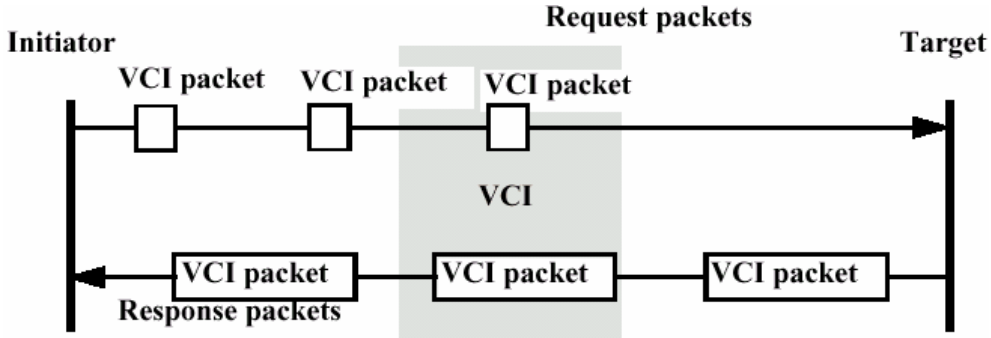
- Transfer Responses
  - Read/Write cell/packet successful
  - Read/Write packet general error
  - Read/Write bad data error
  - Read/Write Abort disconnect

Advanced VCI

- AVCI supports out-of-order transactions and an advanced packet model
- Advanced Packet Model
  - Request and response packets do not have the same size
  - Need
    - request packet: one cell, set the start address and address behavior
    - response packet: many cells, read data return
- Arbitration hiding
  - pipelines of both the request and response packets
- Source Identification
  - a unique identifier for each initiator

AVCI Protocol

- Still 3 layers similar to BVCI. No difference in the transaction layer, slightly differ in the packet and cell layers
- Packet layer



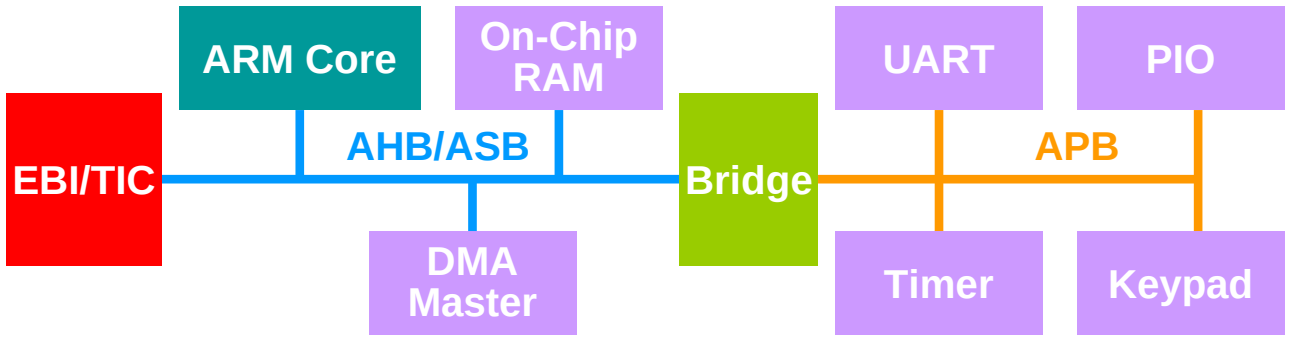
- Cell layer
  - AVCI cell layer differs from BVCI with some additional fields, with side band signals for arbitration hiding
  - Arbitration hiding signals are separately handshaken

Outline

- VCI Interface Standards
- AMBA - On Chip Buses
- AMBA 3.0 - AXI

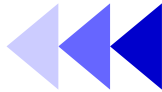
ARM OCB - AMBA

- Advanced Microcontroller Bus Architecture (AMBA)
- AMBA 2.0 specifies
  - the Advanced High-performance Bus (AHB)
  - the Advanced System Bus (ASB)
  - the Advanced Peripheral Bus (APB)
  - test methodology



A typical AMBA system

Features of AMBA



- AHB is superior to ASB in
  - performance and synthesizibility and timing verification

Advanced High-performance Bus (AHB)
High performance
Pipelined operation
Multiple bus master
Burst transfers
A single centralized decoder

Split transactions
single-cycle bus master handover
single-clock edge operation
non-tristate implementation
wider data bus configurations (8/16/32/64/128 bits)

Advanced System Bus (ASB)
High performance
Pipelined operation
Multiple bus master
Burst transfers
A single centralized decoder

Advanced Peripheral Bus (APB)
Low power
Simple interface

APB access MUST take 2 PLCK cycles

Notes on the AMBA Specification

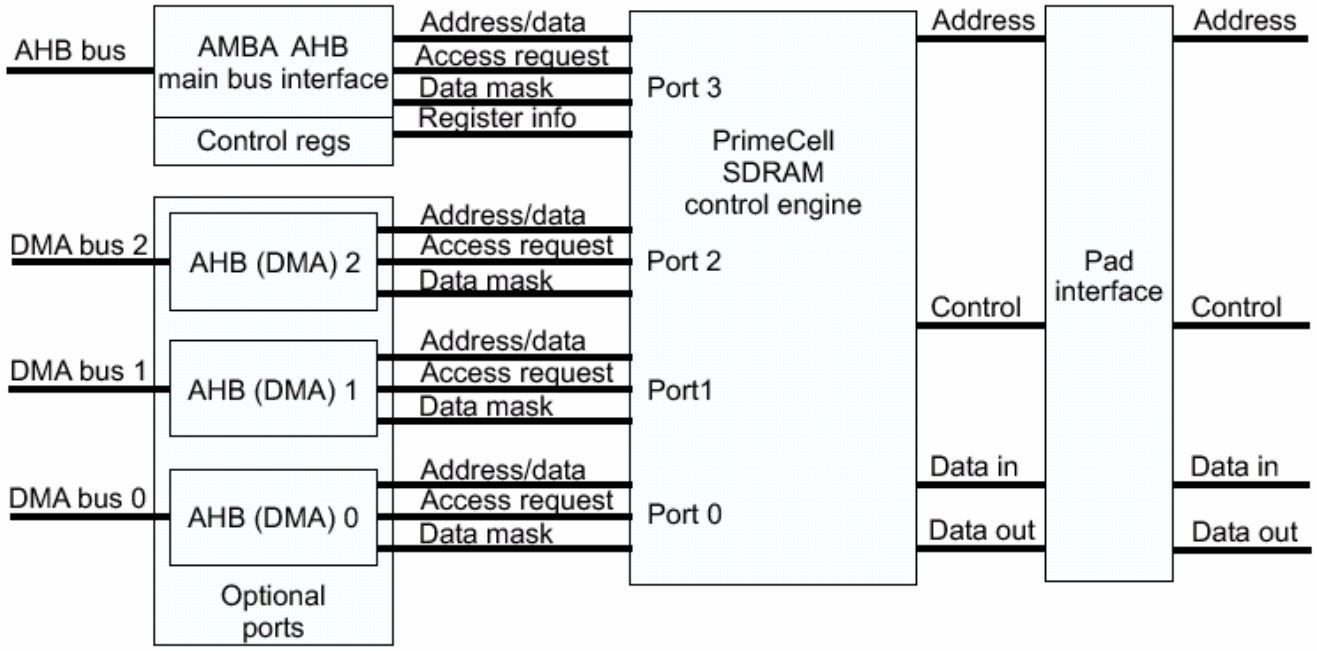
- Technology independence
  - The specification only details the bus protocol at the clock cycle level
- Electrical characteristics
  - No information regarding the electrical characteristics is supplied
- Timing specification
  - The system integrator is given maximum flexibility in allocating the signal timing budget amongst the various modules on the bus
  - More free, but may also be more danger and time-consuming

Notes on AMBA (1/3)

- Split transaction
  - NOT truly split transaction - the arbiter only masks the access of the master which gets a SPLIT transfer response
  - Master does not need extra slave interface
  - Only allows a single outstanding transaction per bus master
- NOT support Sideband signals
  - Sideband signals: reset, interrupts, control/status, generic flags, JTAG test interface, etc.
  - Require the system integrator to deal with them in an ad-hoc way for each system design.
  - Good references of sideband signals: VSIA VCI or Sonics OCP

Notes on AMBA (2/3)

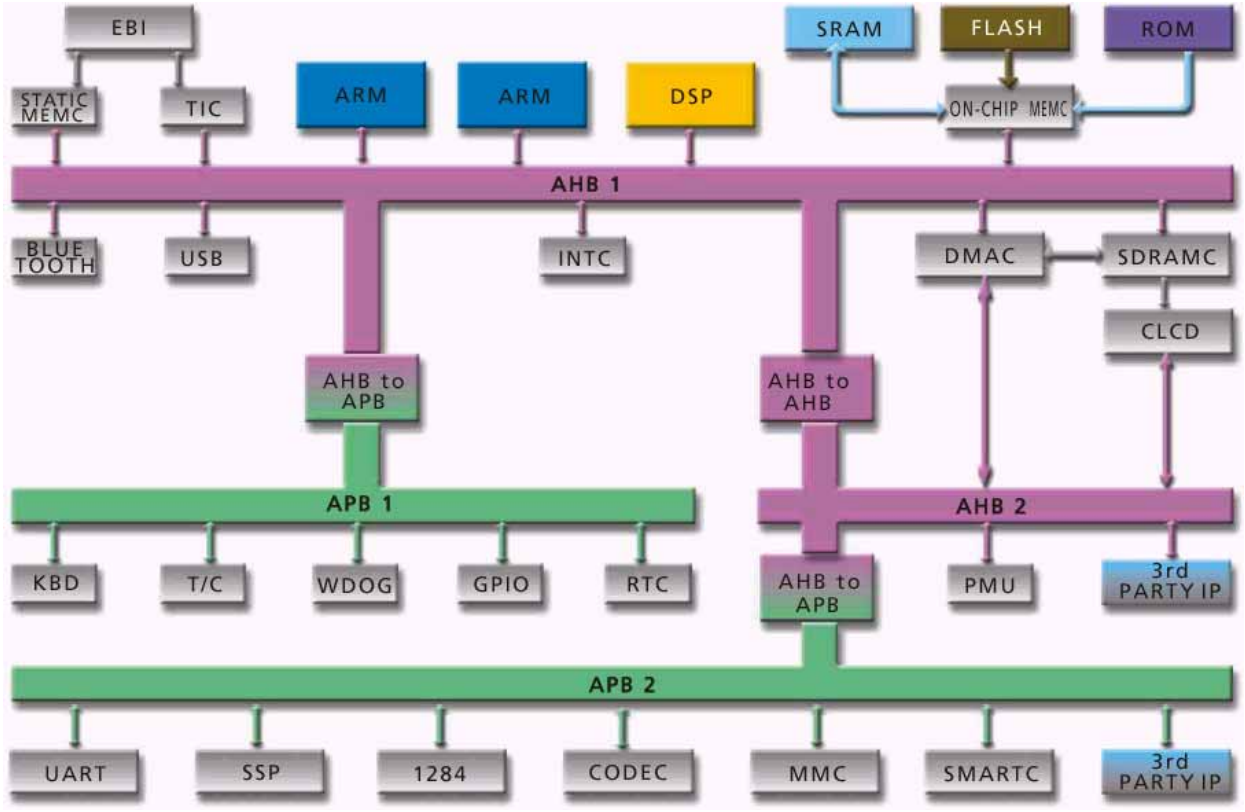
- DMA channels
  - Use AHB protocol
    - E.g. PrimeCell SDRAM Controller
    - Easy to connect to another AHB bus



  - Adopt user defined protocol
    - Lower the complexity of the DMA interface

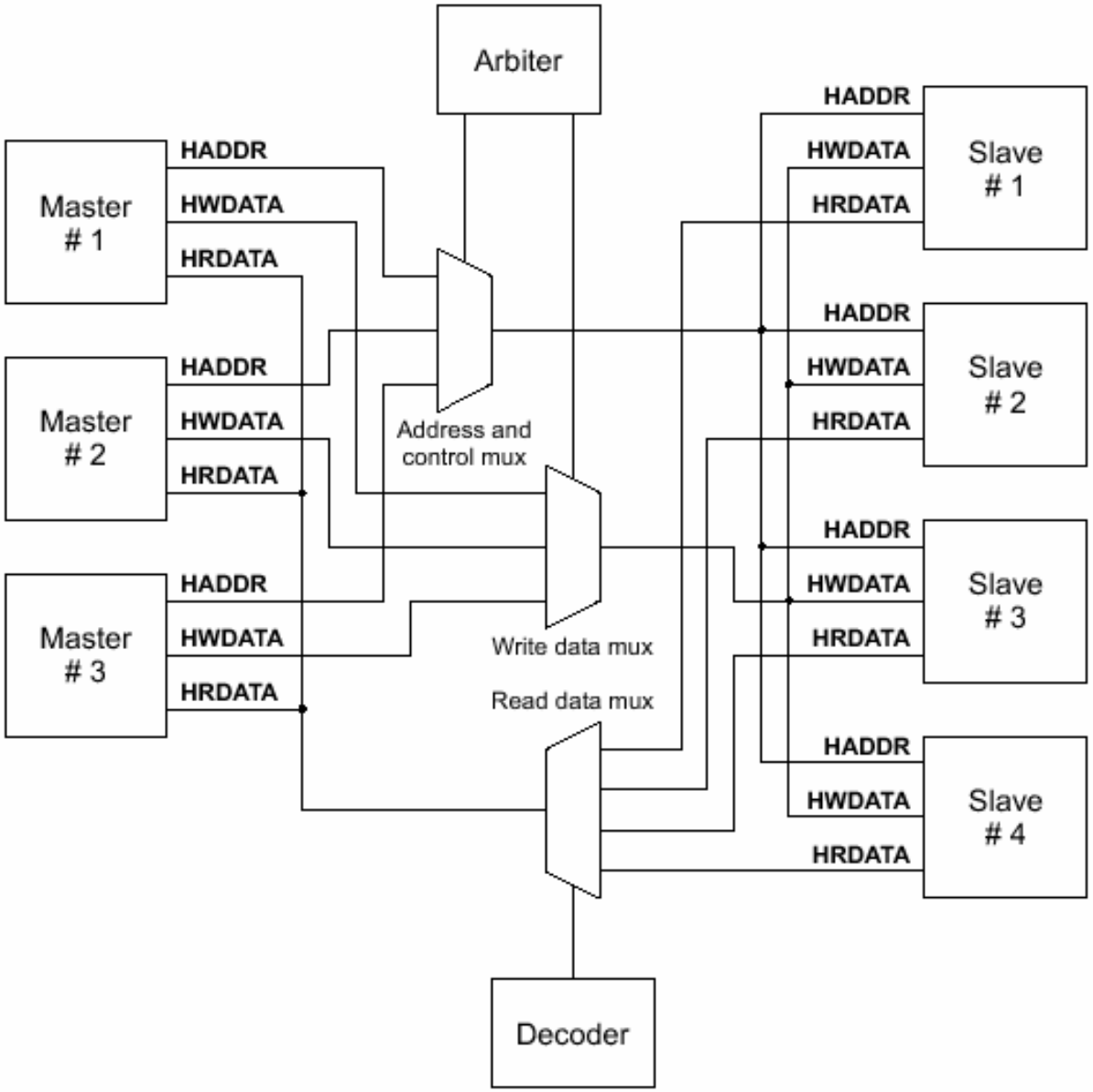
Notes on AMBA (3/3)

- APB does not support WAIT transaction
  - Access status register first, then access data register
  - Alternative: designed as AHB slaves
  - Multiple AHB/APB to reduce loading



Wipro's SOC-RaPtor™ Architecture

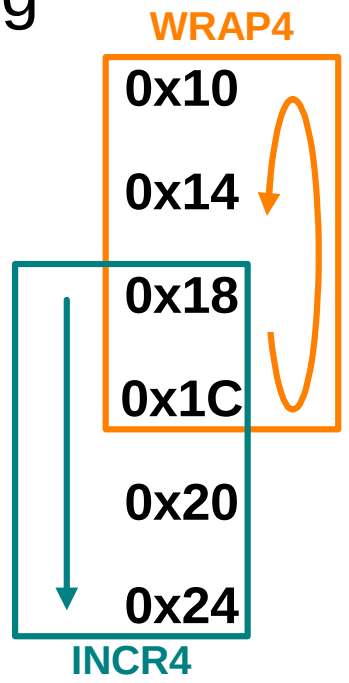
AHB Interconnect



- Bus master drives the address and control
- Arbiter selects one of the master

AHB Operation (1/2)

- Master asserts a request signal to the arbiter. Arbiter then gives the grant to the master.
- A granted bus master starts an AHB transfer by driving address and control signals:
  - address
  - direction
  - width
  - burst forms
    - Incrementing burst: not wrap at address boundaries
    - Wrapping burst: wrap at particular address boundaries
- Write data bus: move data from the master to a slave
- Read data bus: move data from a slave to the master



Address wrap in 4-word boundary

AHB Operation (2/2)

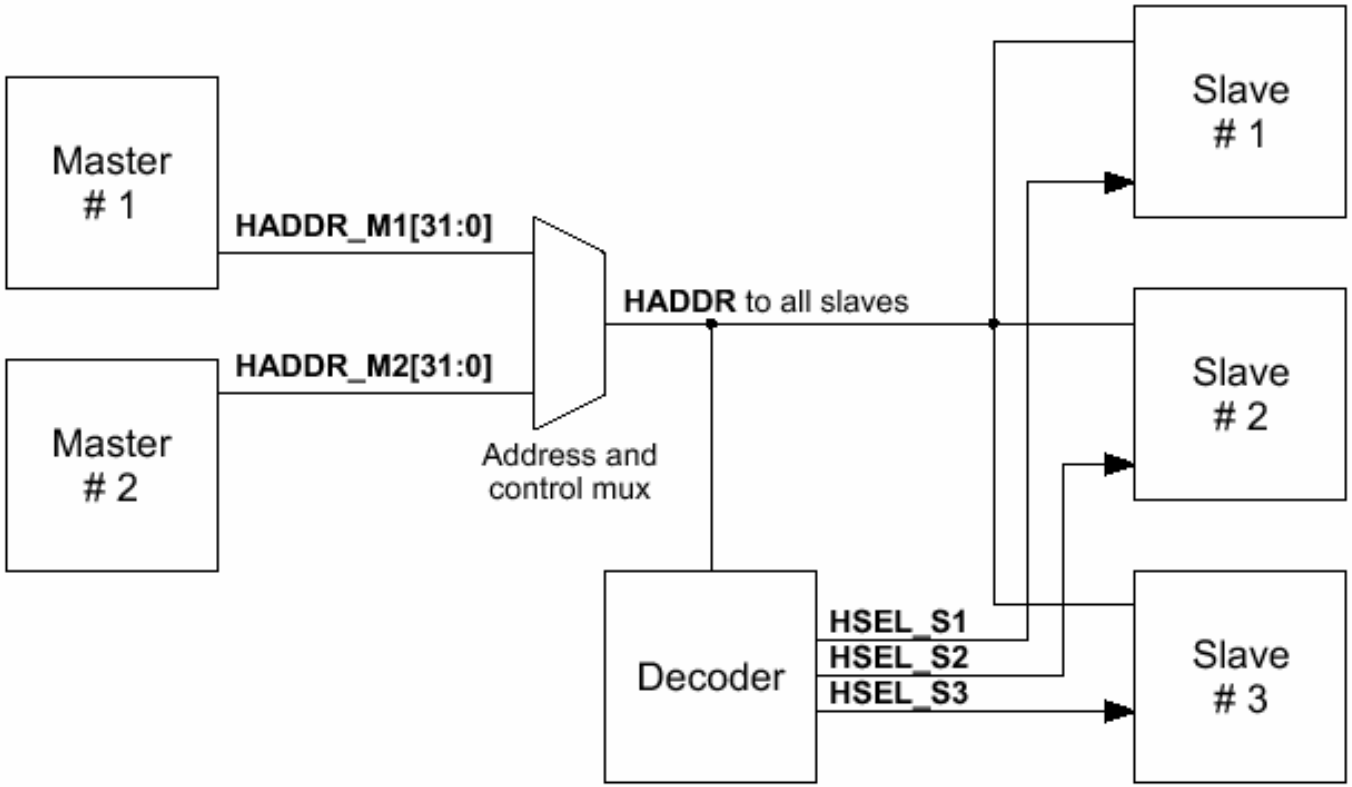
- All slaves sample the address

  Data can be extended using the HREADY signal, when LOW, wait states be inserted and allow extra time for the slave to provide or sample data
- During a transfer the slave shows the status using the response signals HRESP[1:0]
  - OKAY: transfer progressing normally

    when HREADY is HIGH, transfer has completed successfully
  - ERROR: transfer error
  - RETRY and SPLIT: transfer can't complete immediately, but the bus master should continue to attempt the transfer
- As burst transfer, the arbiter may break up a burst and in such cases the master must re-request for the bus.

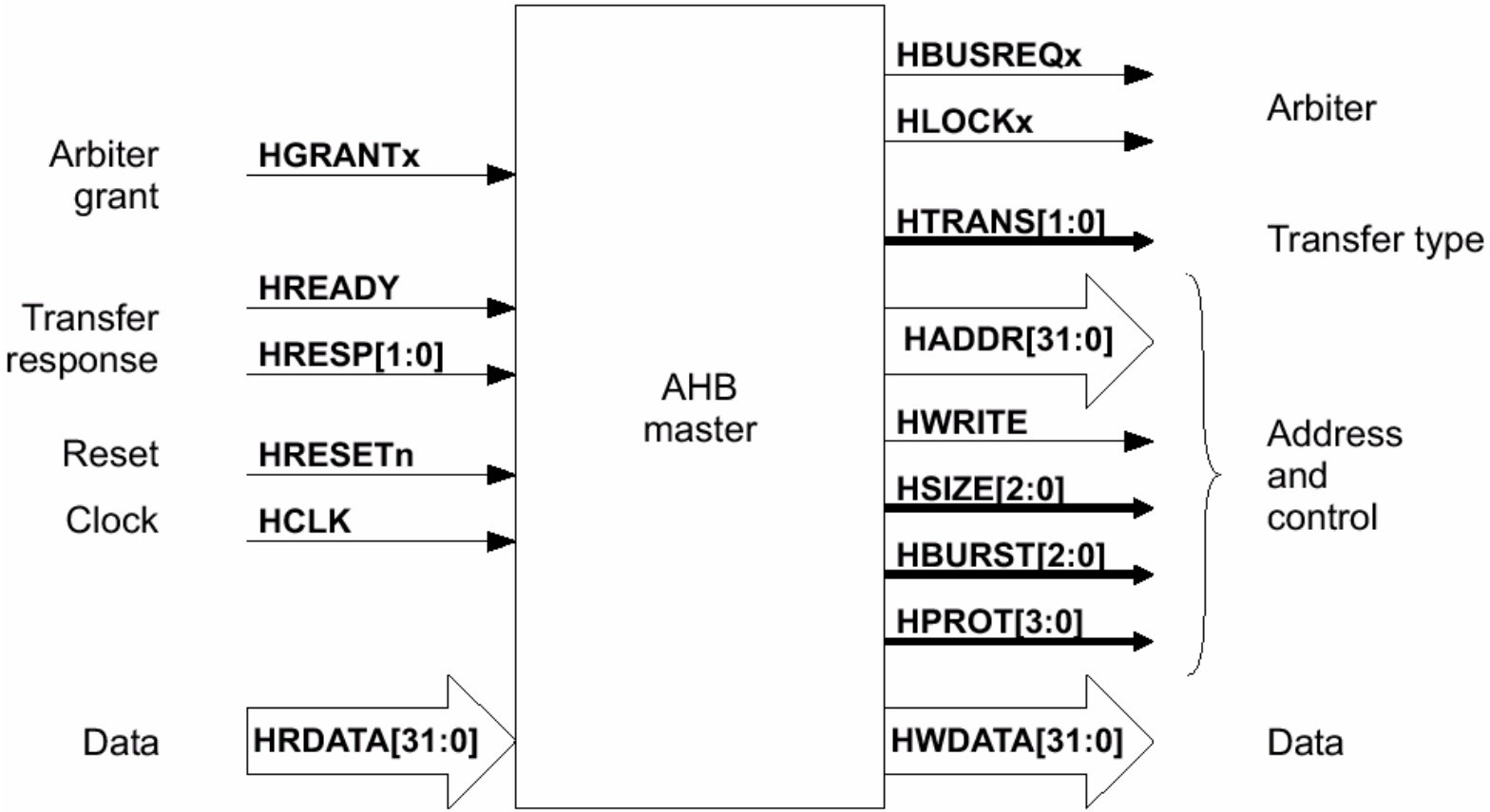
Address Decoding

- A central address decoder provides HSELx for each slave
- Minimum address space that can be allocated to a single slave is 1K Byte
  - No incrementing transfers can over a 1K Byte boundary



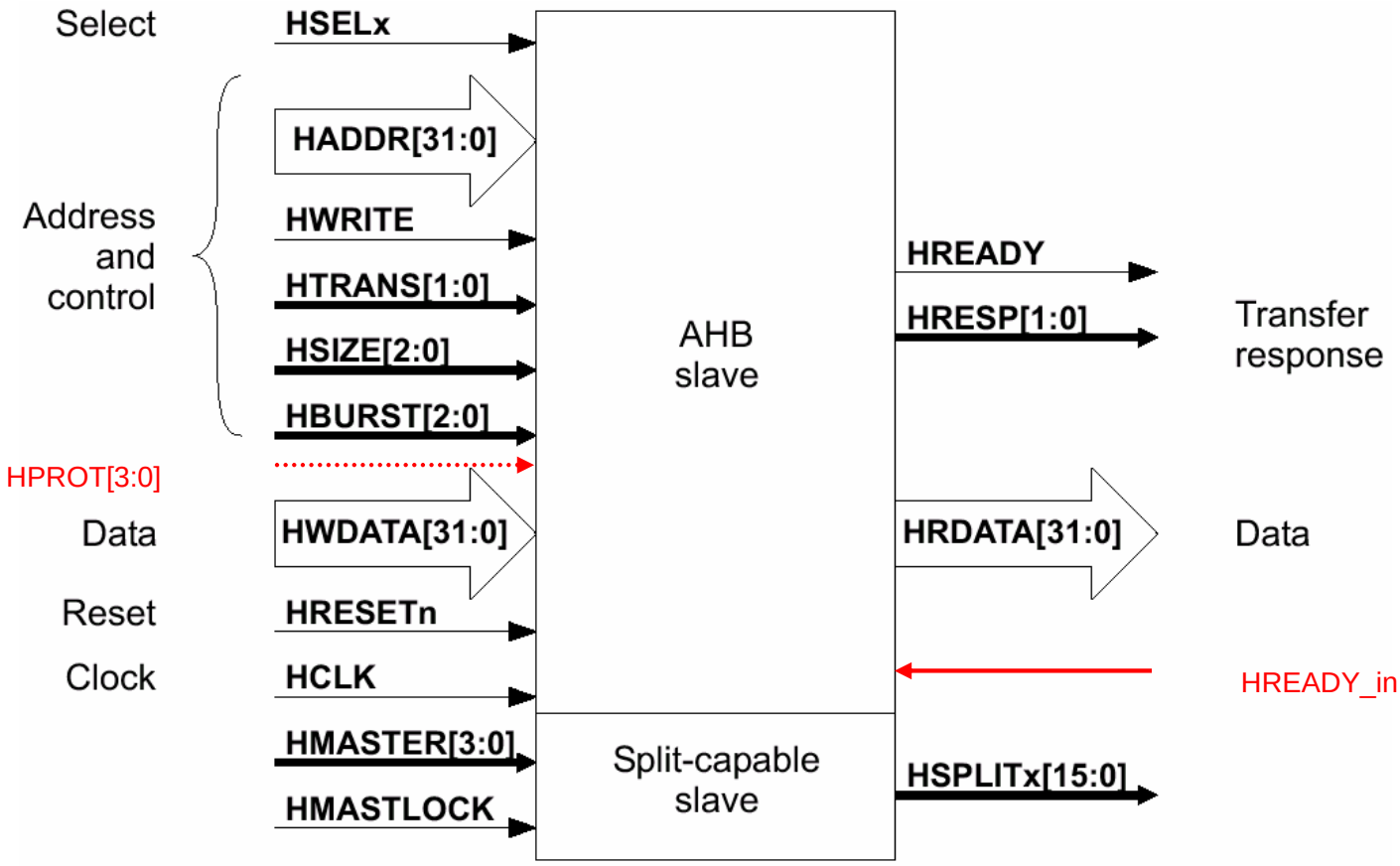
AHB Master

- Initiate read and write by providing an address and control interface
- Processor, DMA, DSP test interface

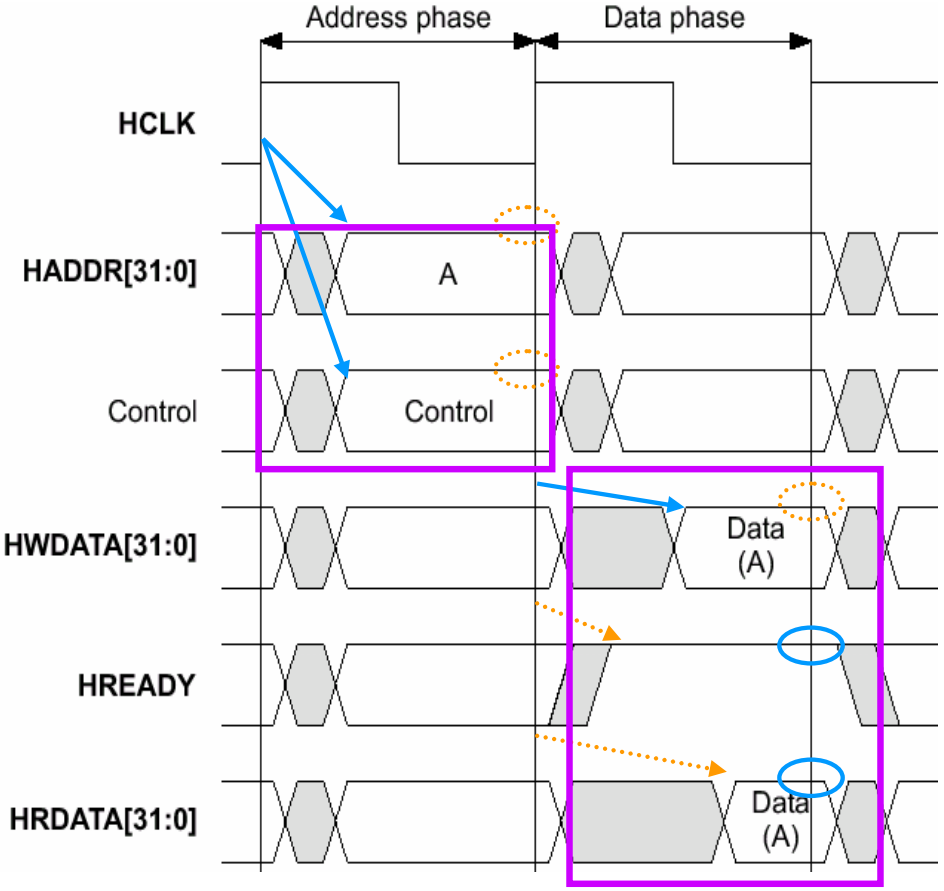


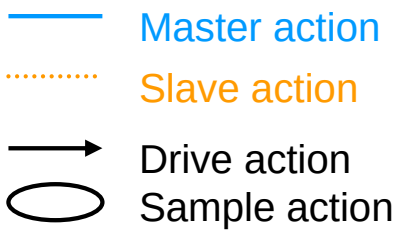
AHB Slave

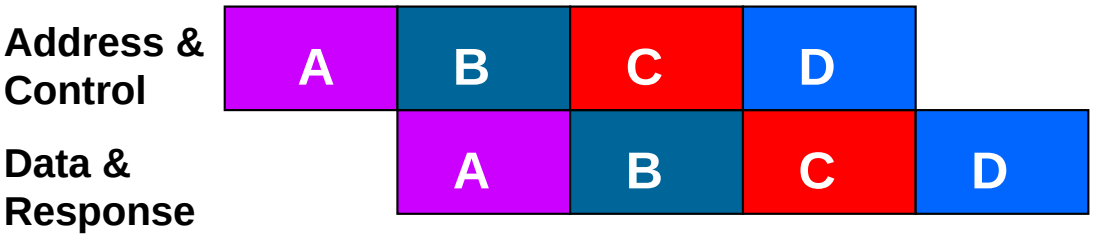
- Respond to a read or write operation within a given address-space range
- Back to the master the success, failure or waiting



Basic Transfer



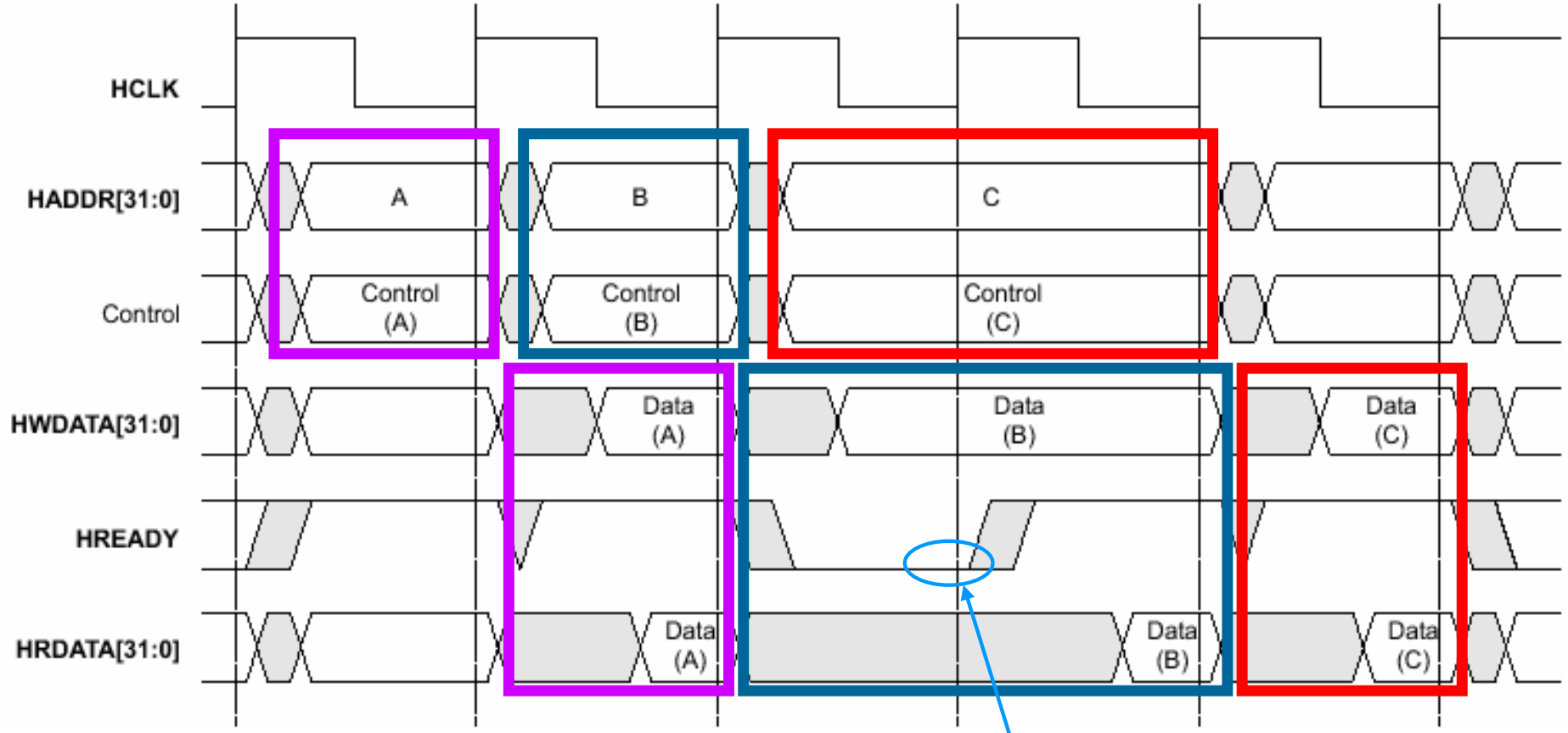


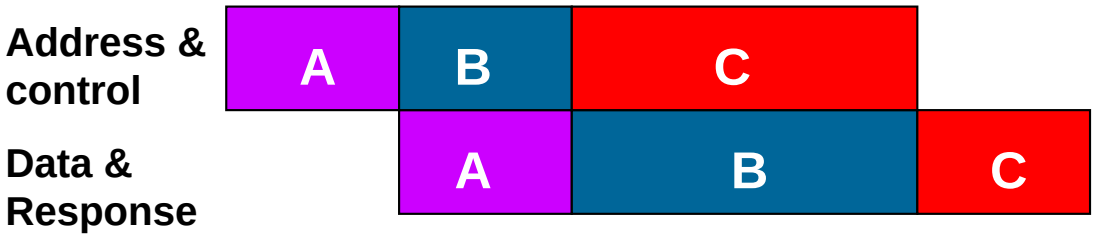


- Address phase : one cycle
  Data phase : one or several cycles
- 1st clock : master drives address and control
- 2nd clock : slave samples address and control
- 3rd clock : bus master sample the slave's response

Multiple Transfers

- Three transfers to run related address A, B, and C



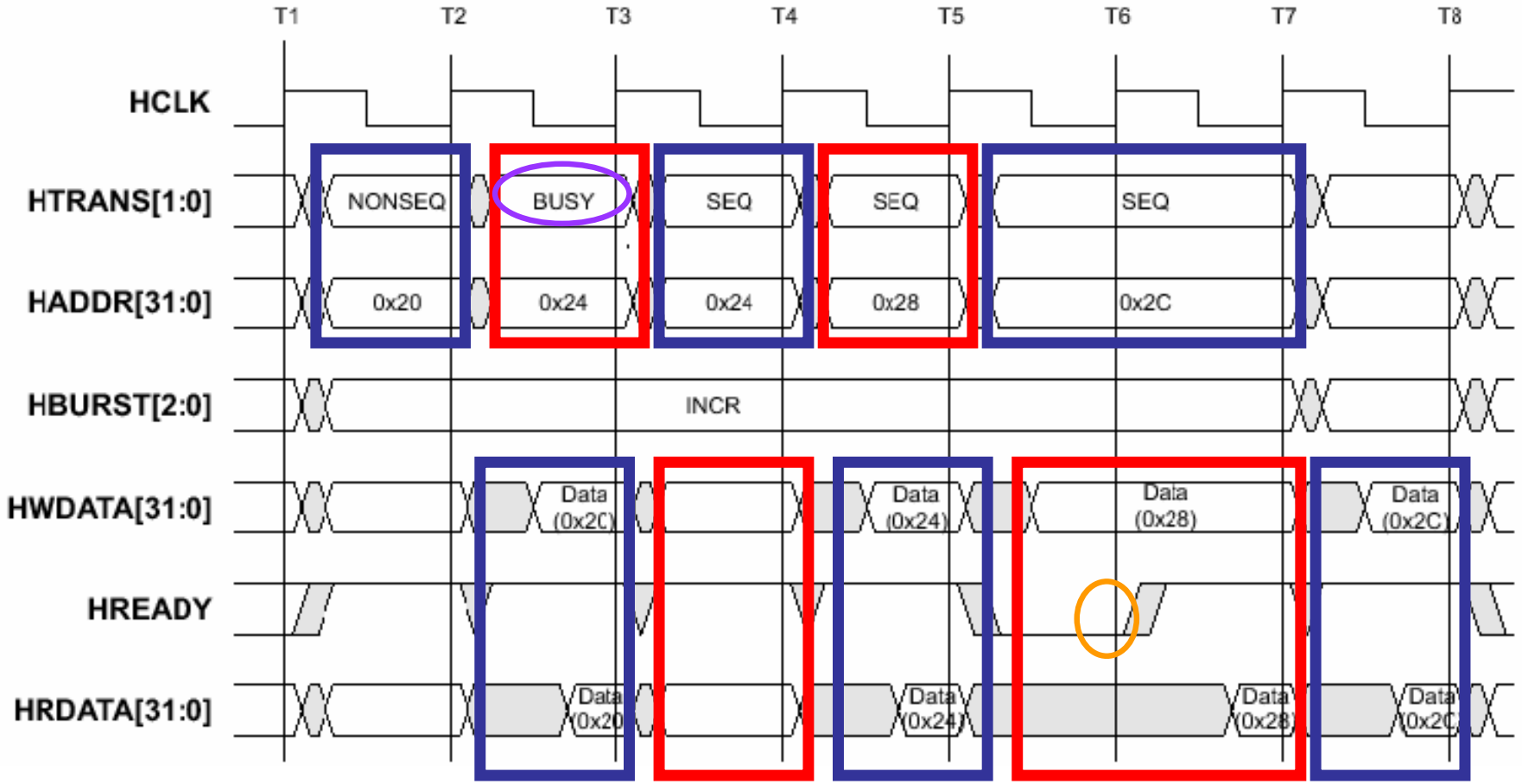


1 wait cycle for address B

Transfer Type (1/2)

HTRANS[1:0]	Type	Descripton
00	IDLE	Slaves must always provide a zero wait state OKAY response to IDLE transfers and the transfer should be ignored by the slave
01	BUSY	Masters cannot take next trnsfer place immediately during a burst transfer. Slaves take actions as they take for IDLE.
10	NONSEQ	Indicates the first transfer of a burst or a single transfer
11	SEQ	The remaining transfers in a burst are SEQUENTIAL. The control information is identical to the previous transfer.

Transfer Type (2/2)

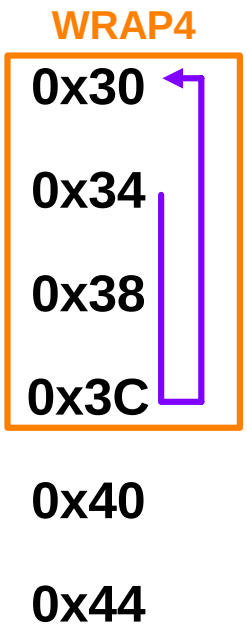


- During T2-T3, **master** is unable to perform the second transfer of burst immediately and therefore the master uses **BUSY** transfer to delay the start of the next transfer.
- During T5-T6, **slave** is unable to complete access immediately, and uses **HREADY** to insert a single wait state.

Burst Operation

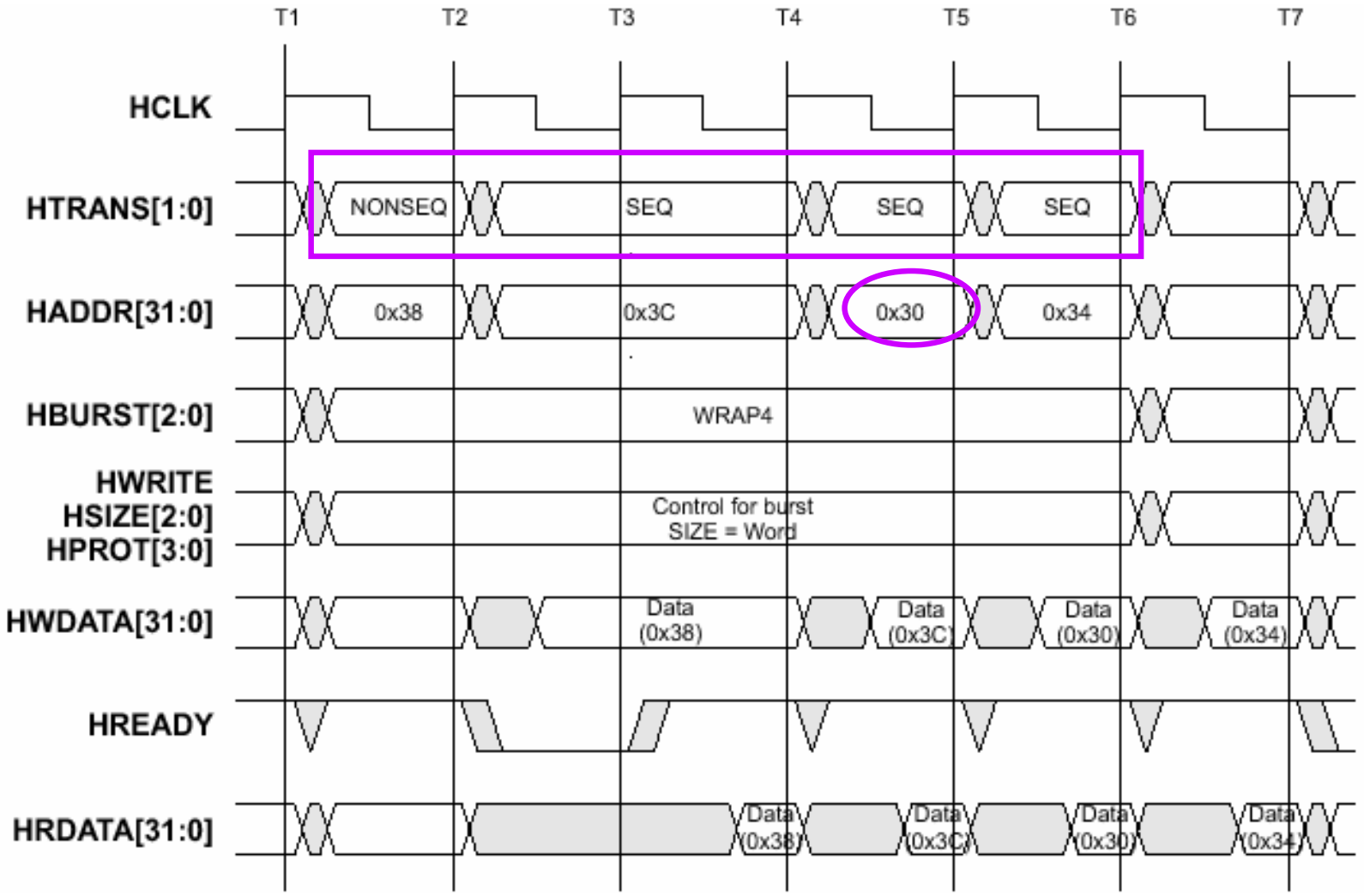
- 4-, 8-, 16-beat
- e.g., 4-beat, start address 0x34, wrapping burst
  ➔four transfers: 0x34, 0x38, 0x3C, 0x30
- Burst length

HBURST[1:0]	Type	Descripton
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



- Limitation: bursts must not cross a 1k Byte address boundary

Four-beat Wrapping Burst

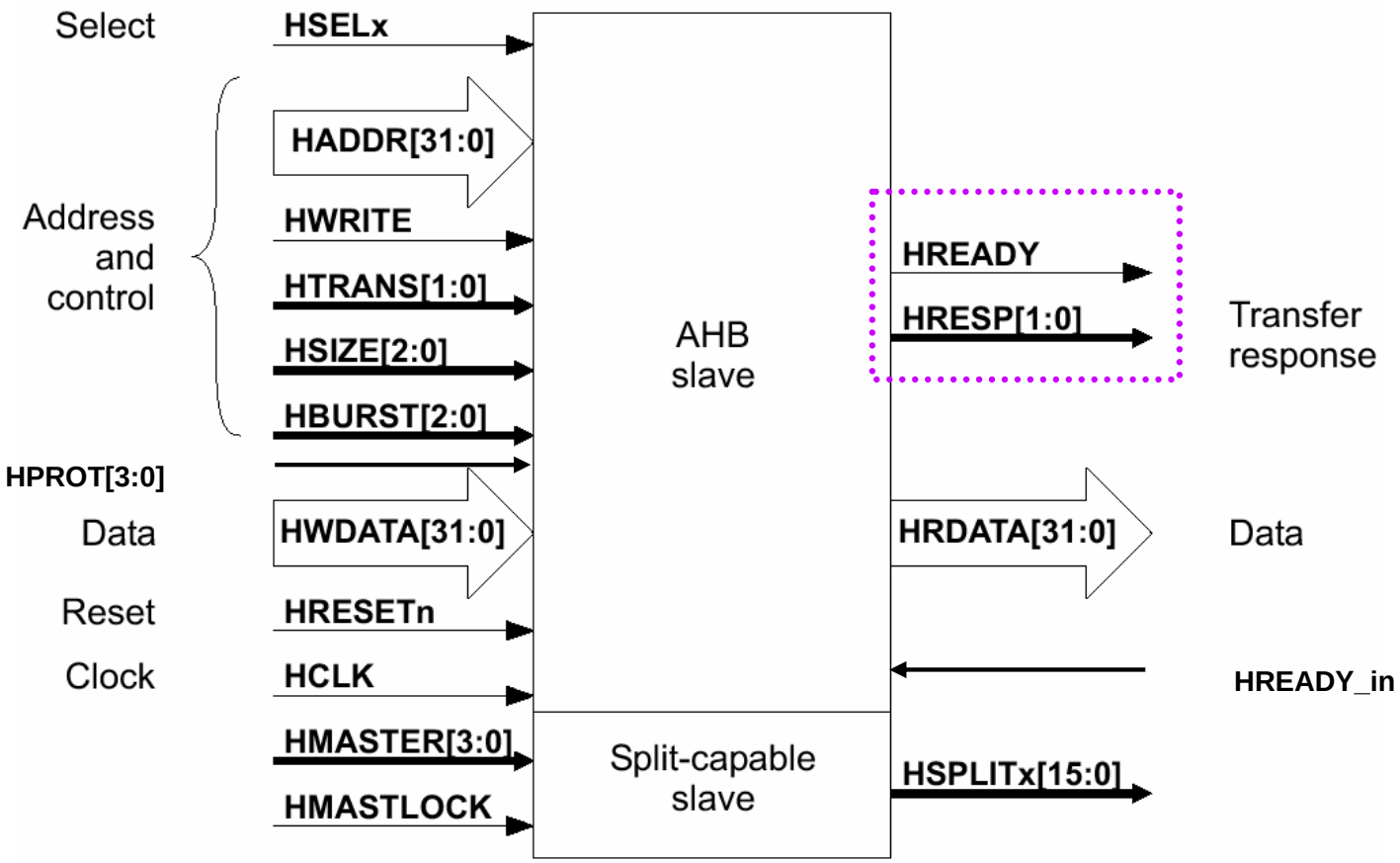


Control Signals

- Have exactly the same timing as the address bus
- Must remain constant throughout a burst of transfers
- Types
  - HWRITE : Transfer direction
  - HSIZE[2:0] : Transfer size
  - HPROT[3:0]) : Protection control

  indicate if the transfer is:
    - An opcode fetch or data access
    - A privileged mode access or user mode access
    - Access is cacheable or bufferable (for bus masters with a memory management unit)

Transfer Responses (from slave)



Transfer Responses

- HREADY
- HRESP[1:0] Response

  00 OKAY

  01 ERROR

  10 RETRY

  11 SPLIT

- Two-cycle response
  - ERROR & RETRY & SPLIT
  - To complete current transfer, master can take following action

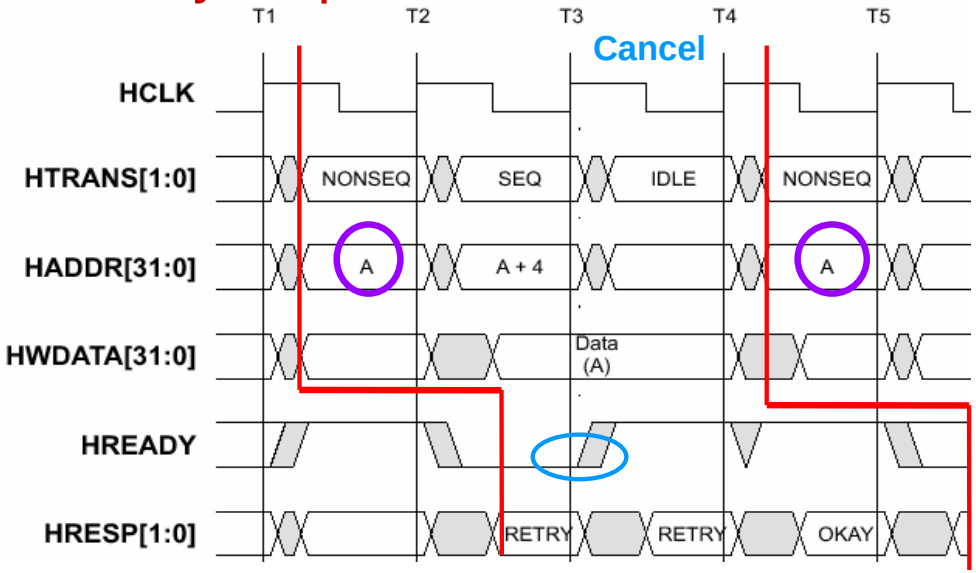
    Cancel for RETRY

    Cancel for SPLIT

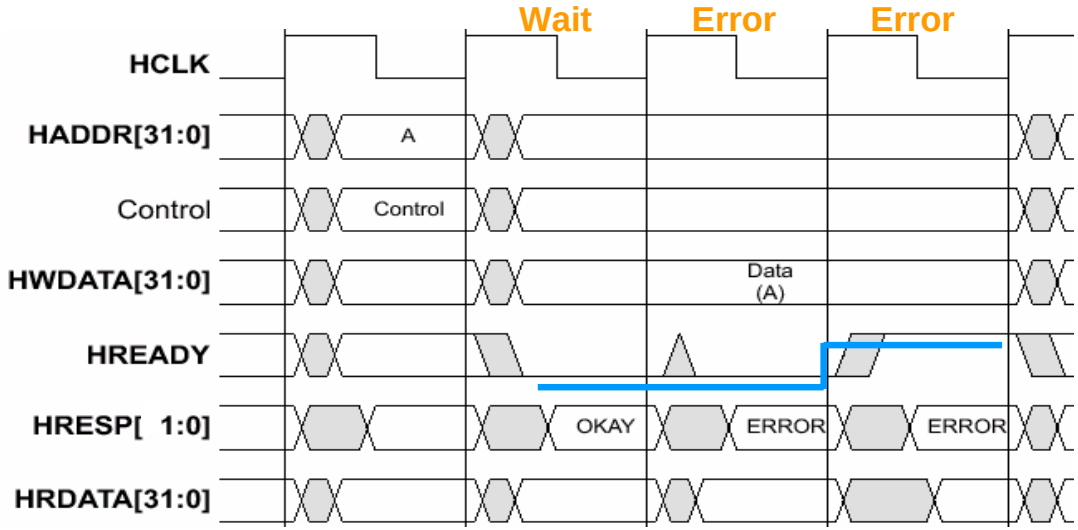
    Either cancel or continue for ERROR

Examples of Two-cycle Response

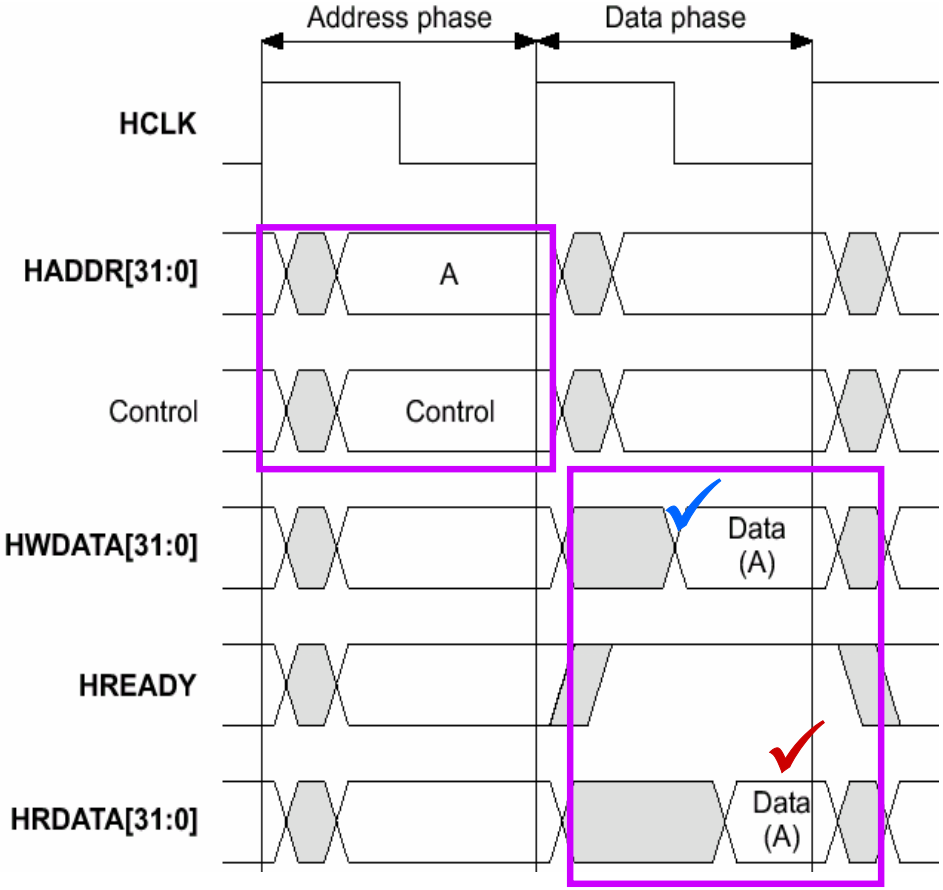
- Retry response



- Error response

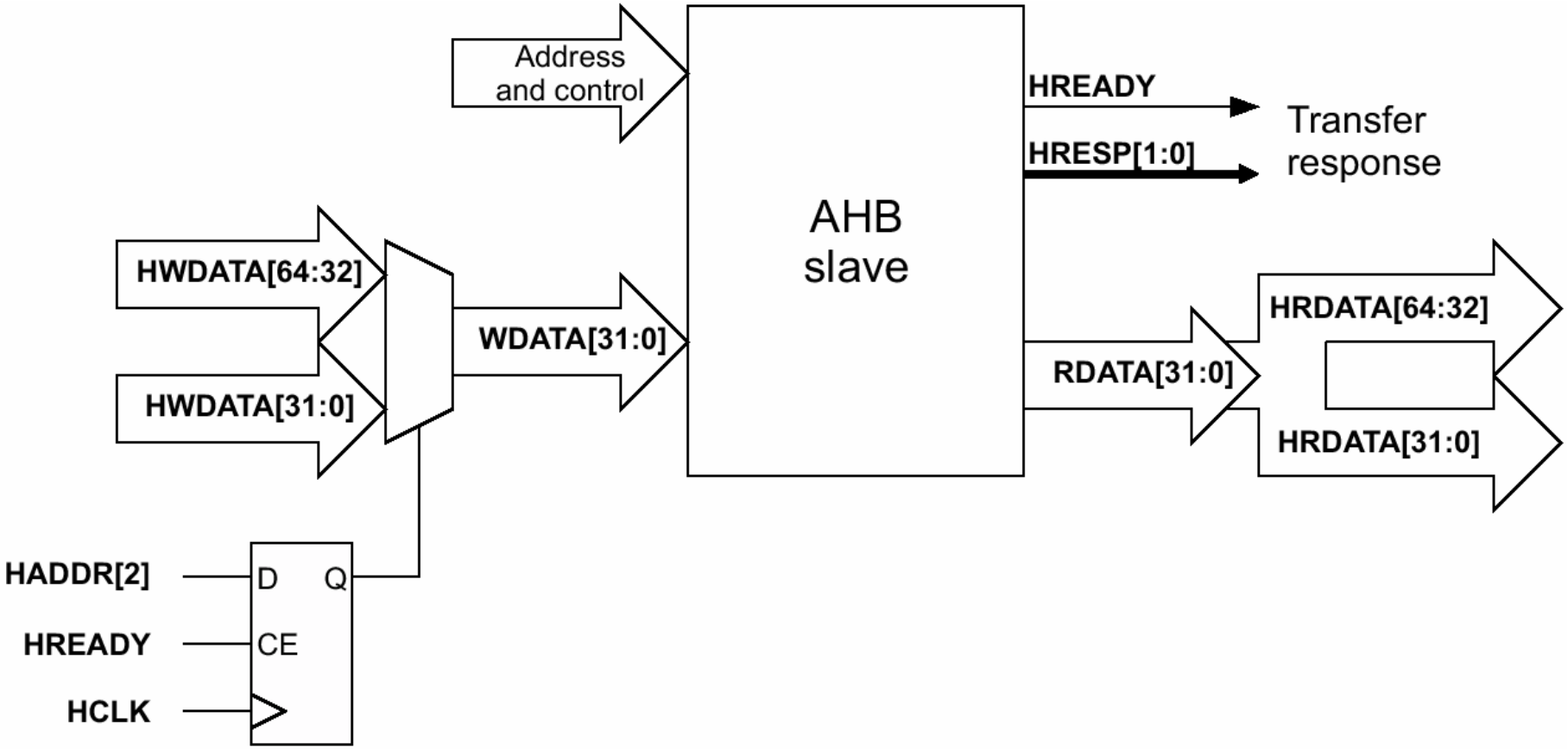


Data Buses

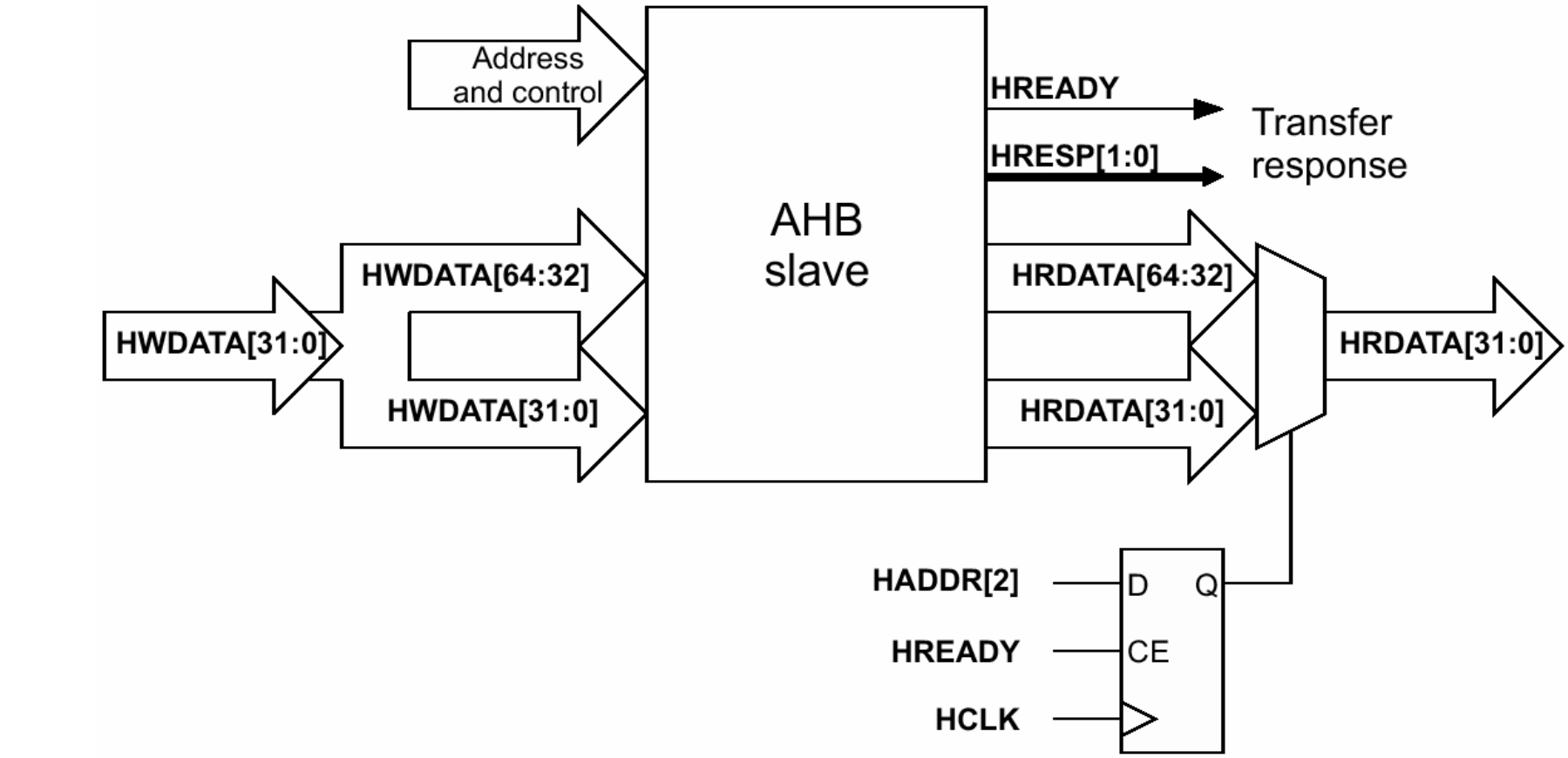


- HWDAA : 32 bits
- RDATA : 32 bits
- Endianness : fixed, lower power; higher performance

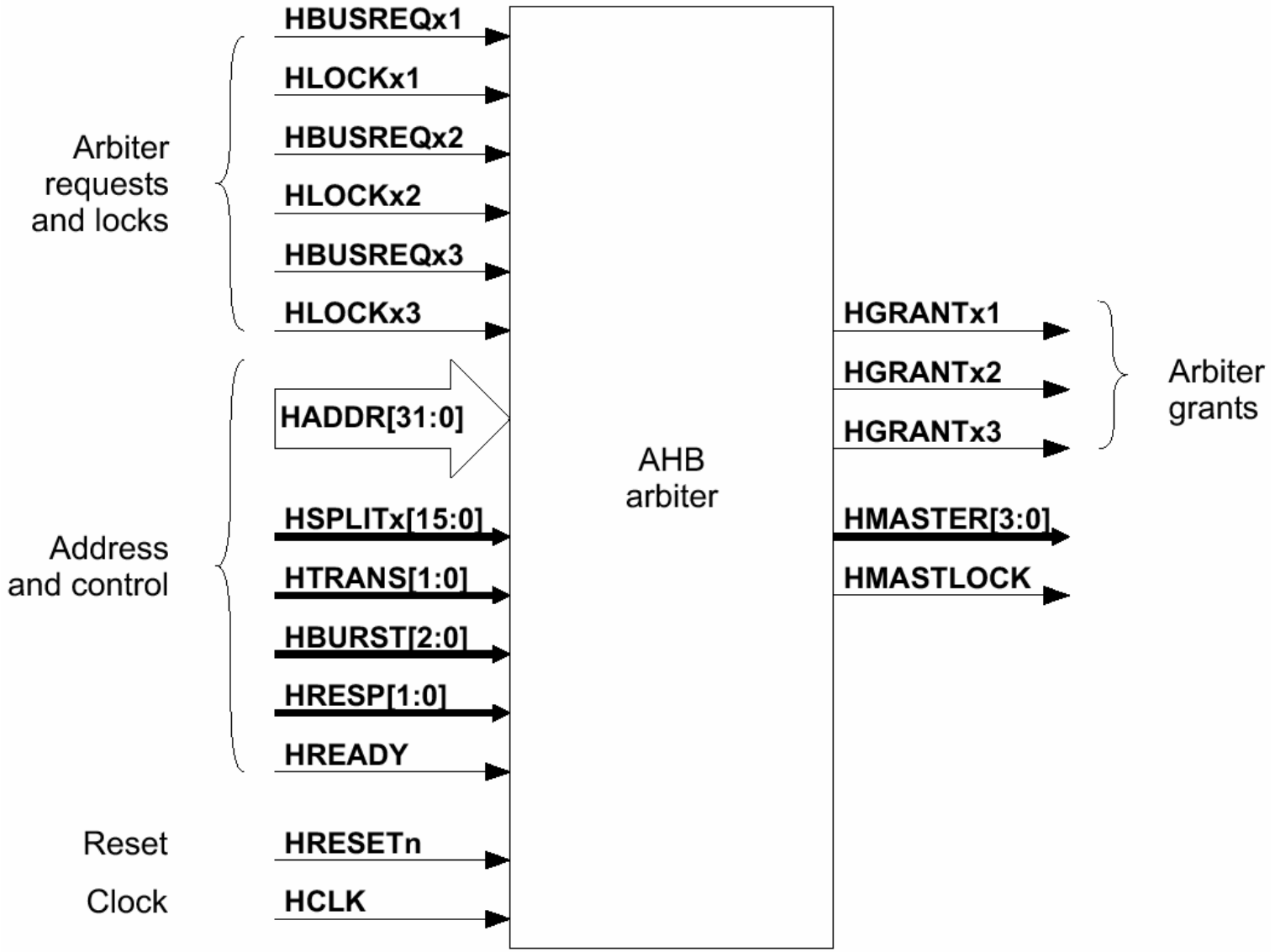
Narrow Slave on A Wide Bus



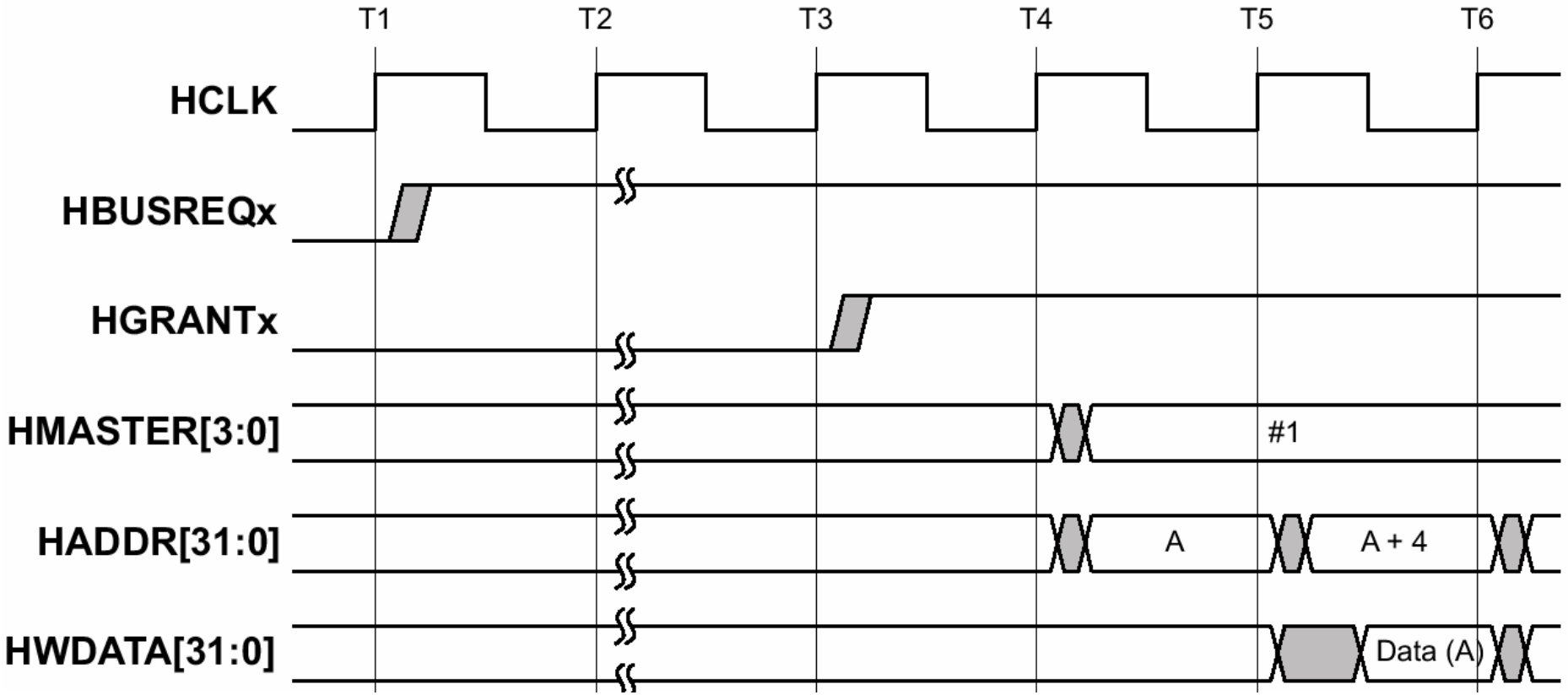
Wide Slave on A Narrow Bus



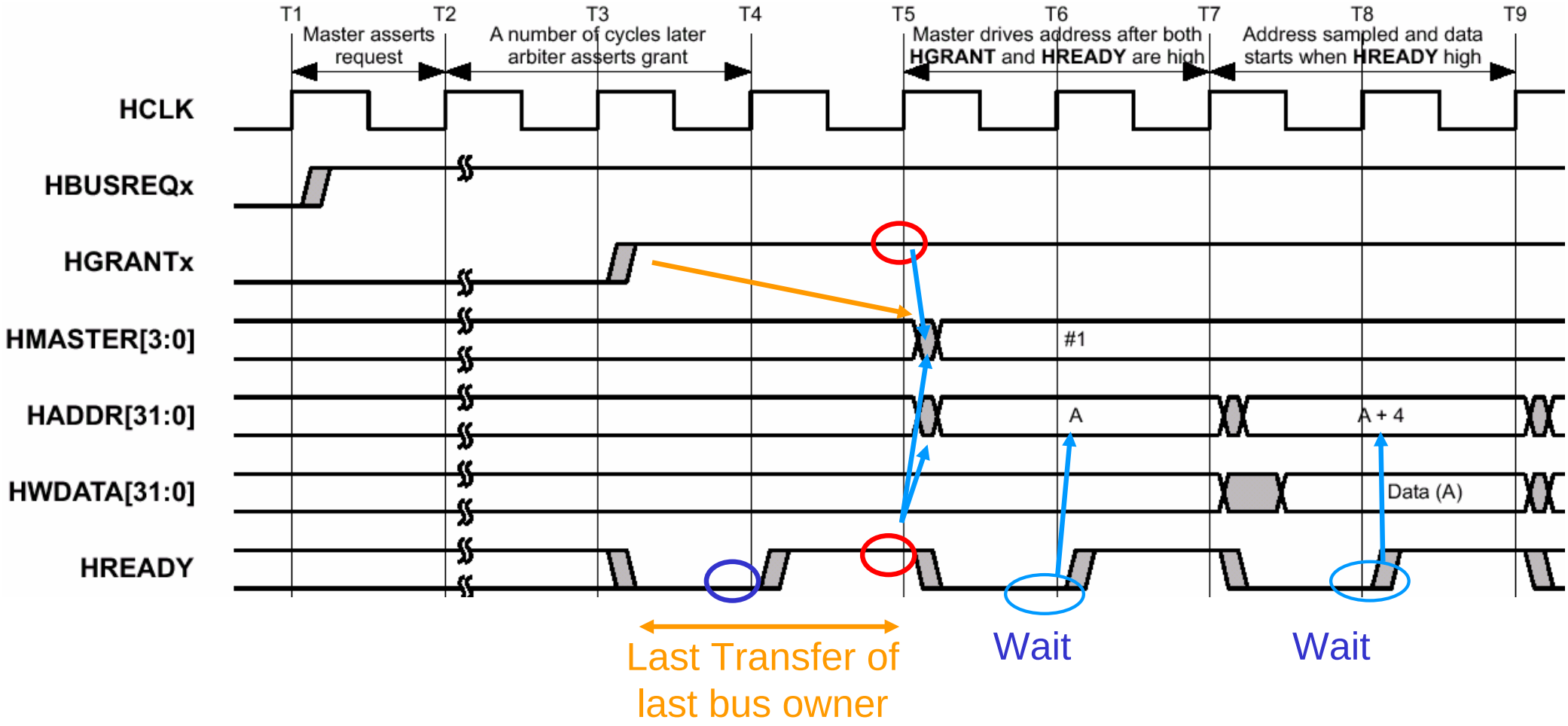
Arbitration



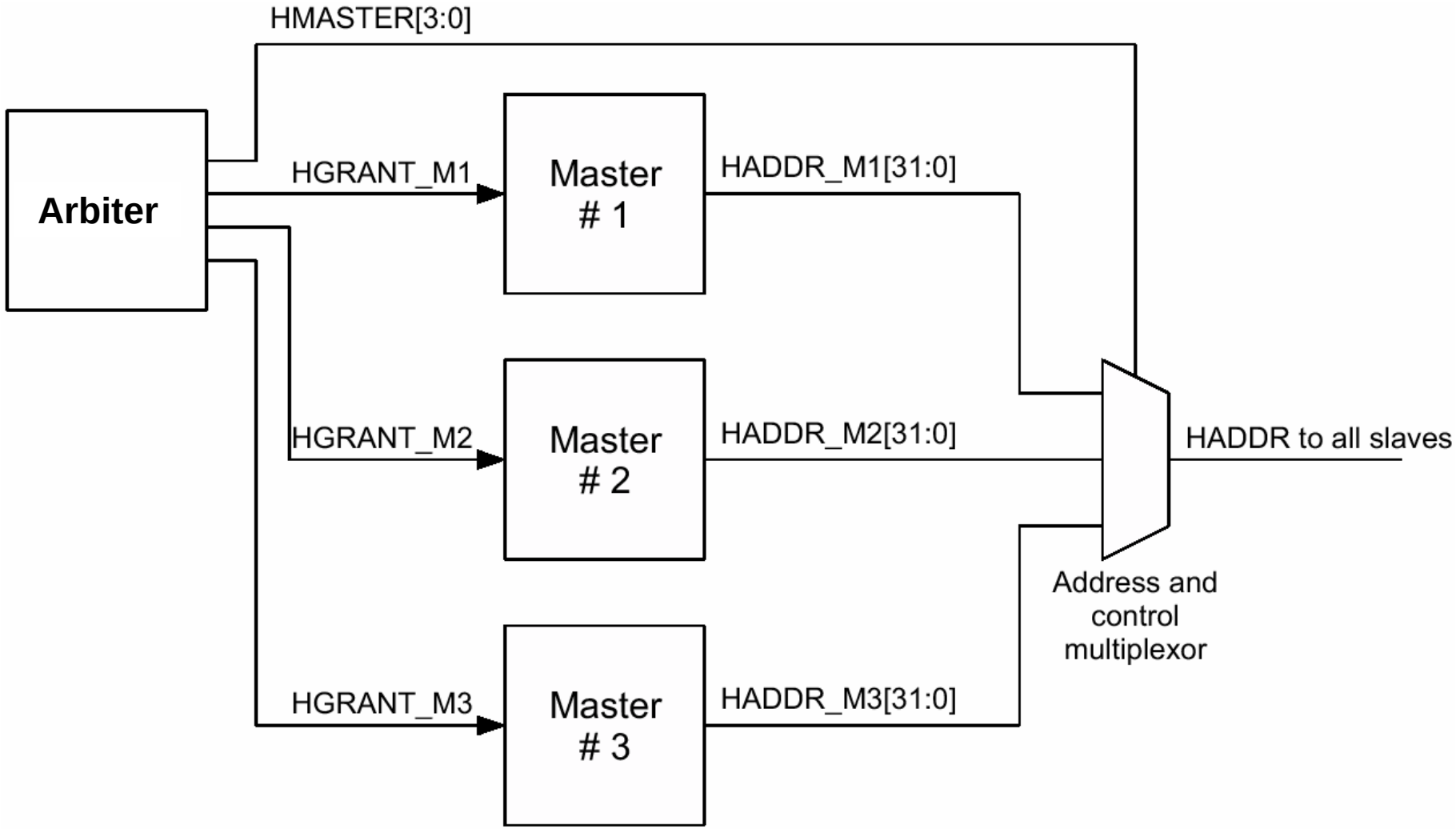
Granting Bus Access With No Wait States



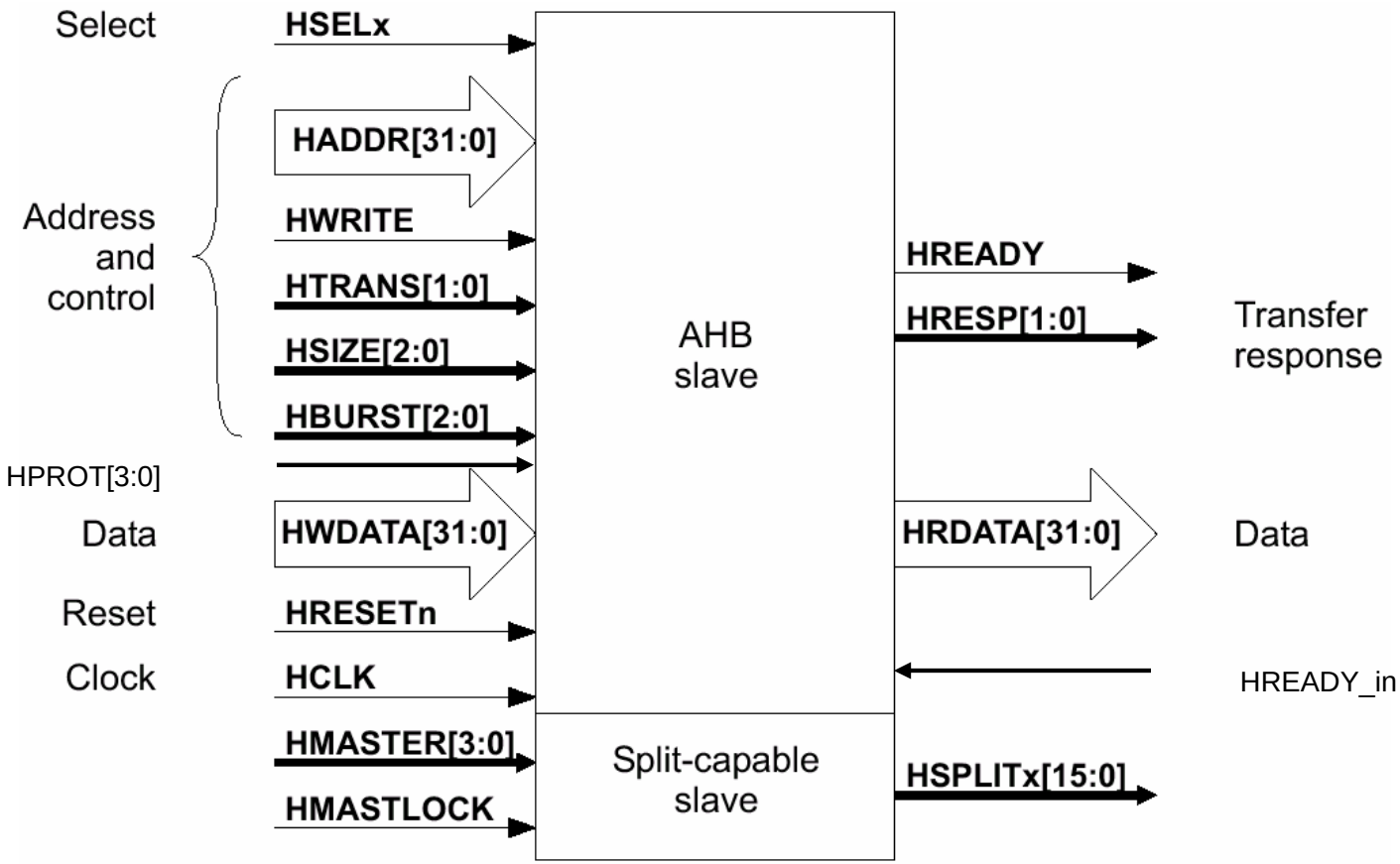
Granting Bus Access



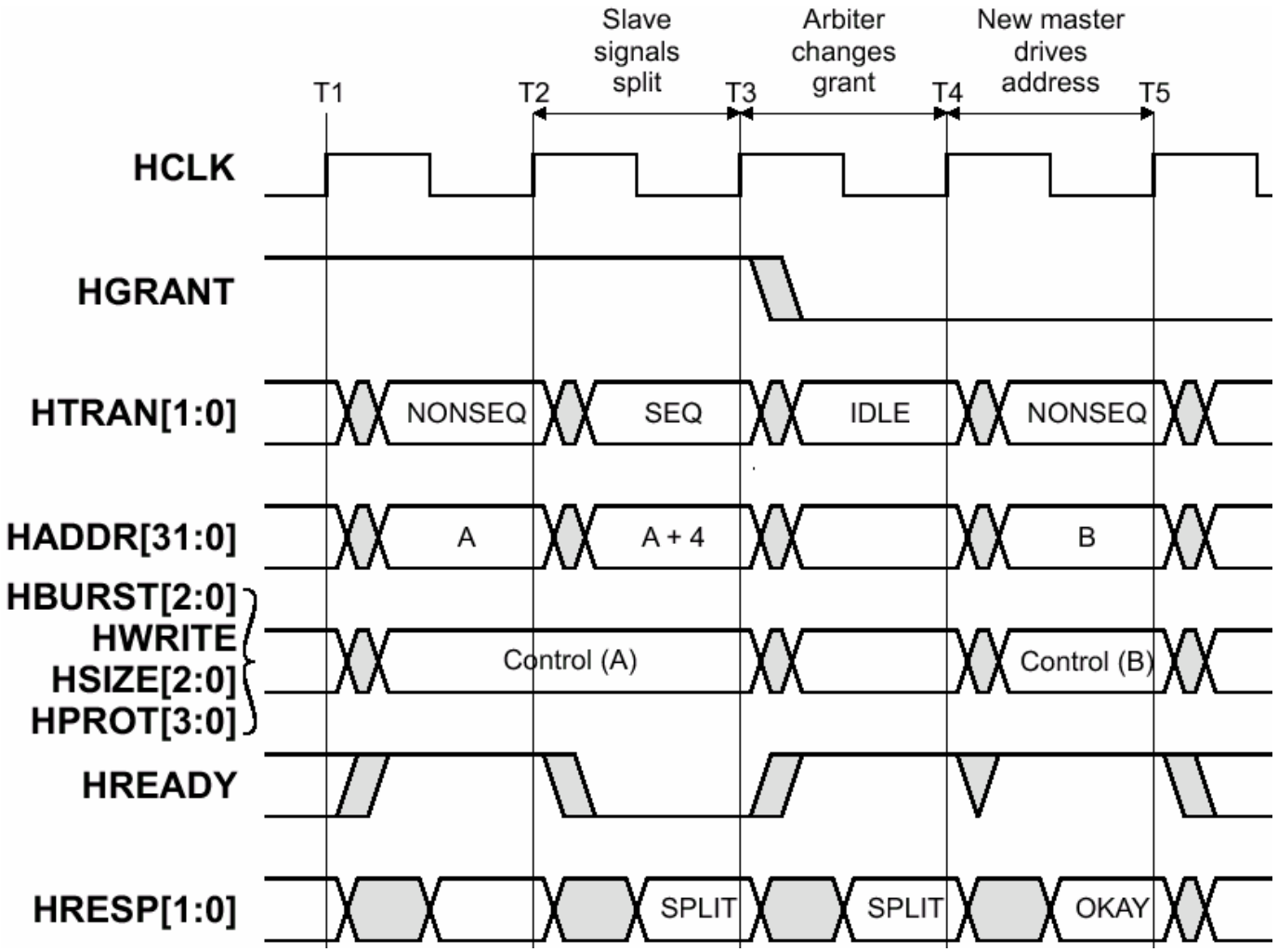
Bus Master Grant Signals



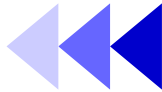
Split Transfer



Split Transfer

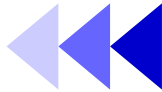


AHB-Lite



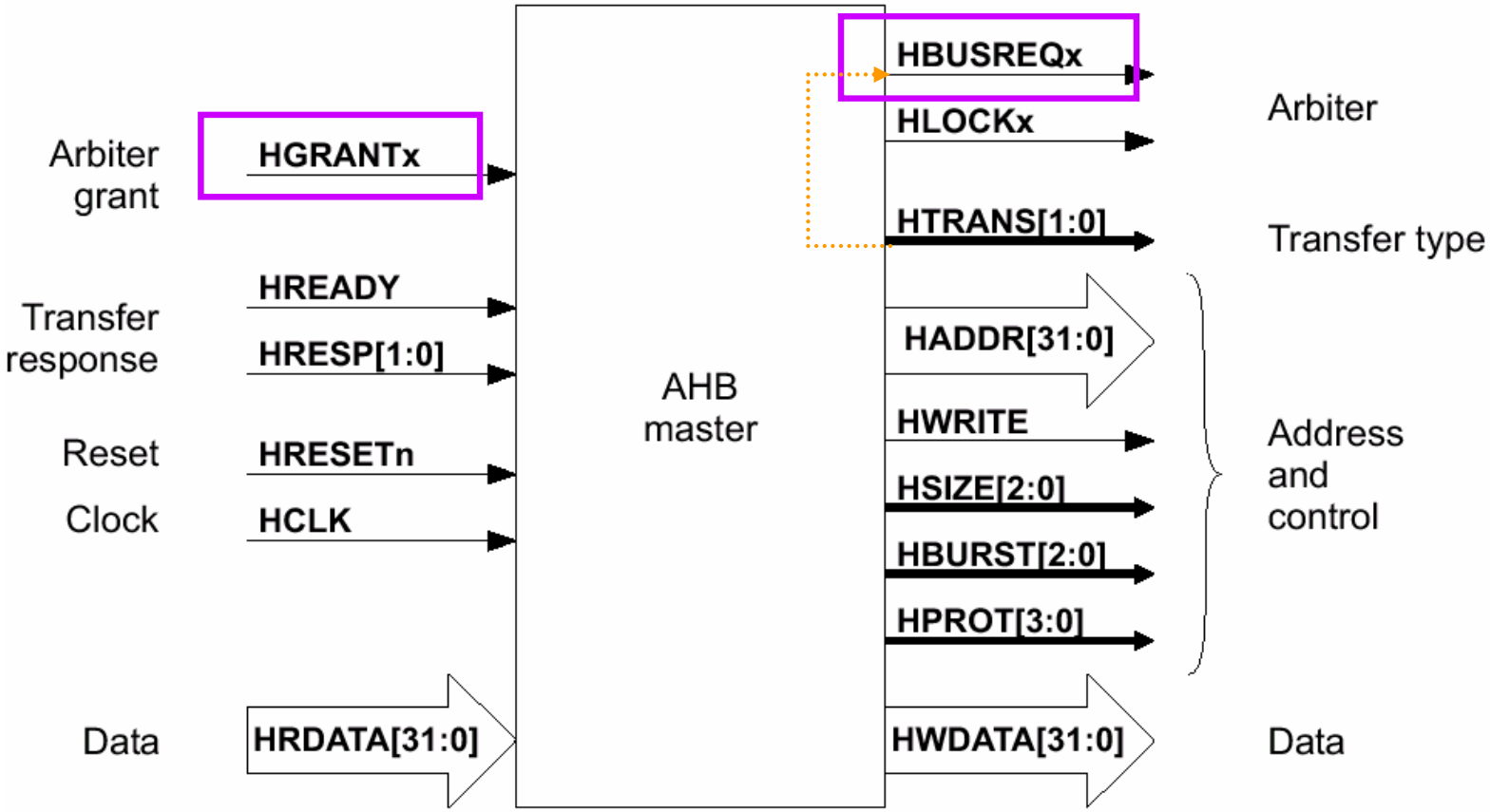
- Requirement
  - Only one master
  - Slave must not issue Split or Retry response
- Subset of AHB Functionality
  - Master: no arbitration or Split/Retry handling
  - Slave: no Split or Retry responses
- Standard AHB masters can be used with AHB-Lite
- Advantage
  - Master does not have to support: the following cases:
    - Losing bus ownership
    - Early bus termination
    - Split and Retry response
  - No arbiter
  - No Master-to-slave mux
  - Allows easier module design/debug

AHB-Lite Interchangeability

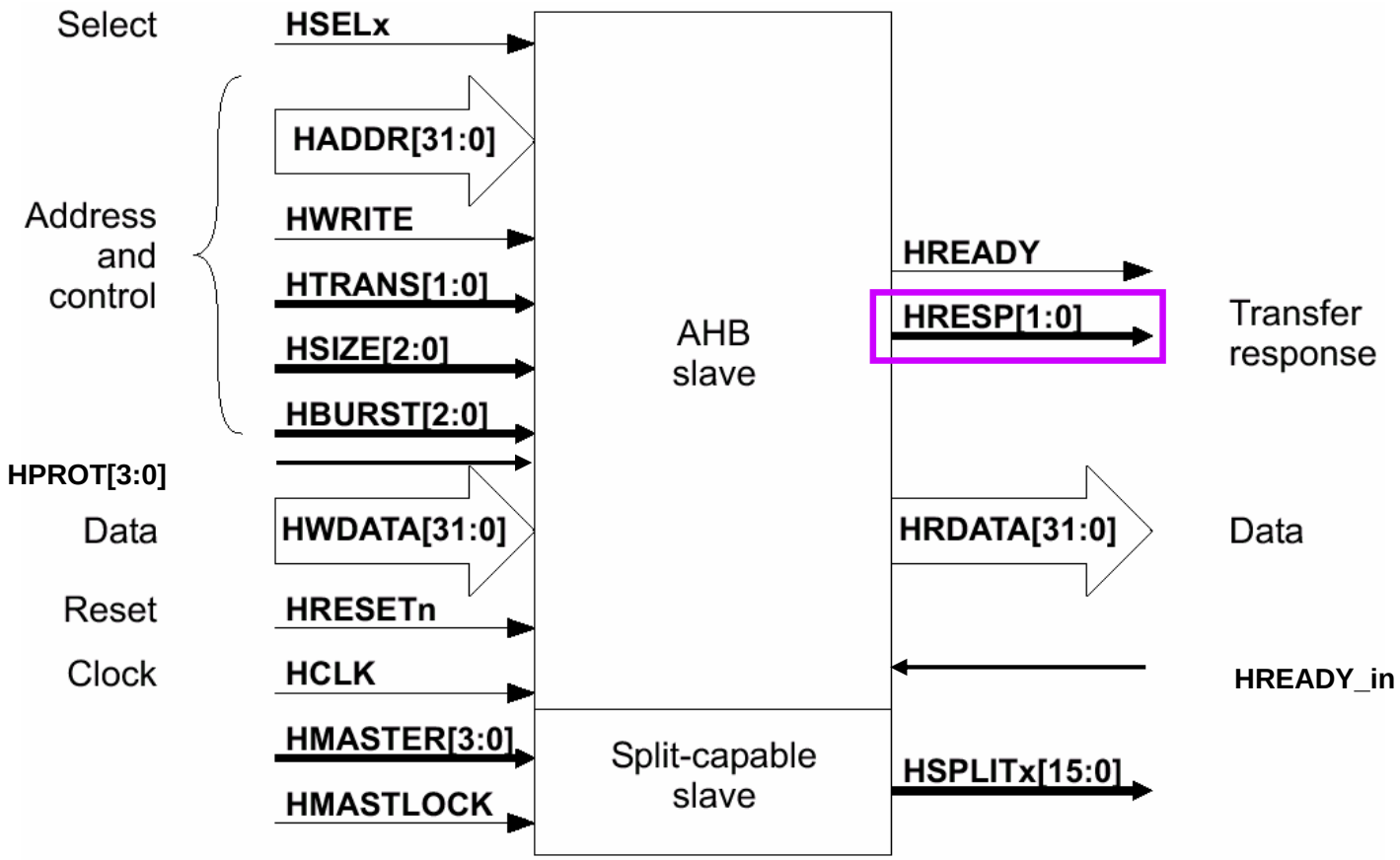


Component	Full AHB system	AHB-Lite system
Full AHB master	✔	✔
AHB-Lite master	Use standard AHB master wrapper	✔
AHB slave (no Split/Retry)	✔	✔
AHB slave with Split/Retry	✔	Use standard AHB slave wrapper

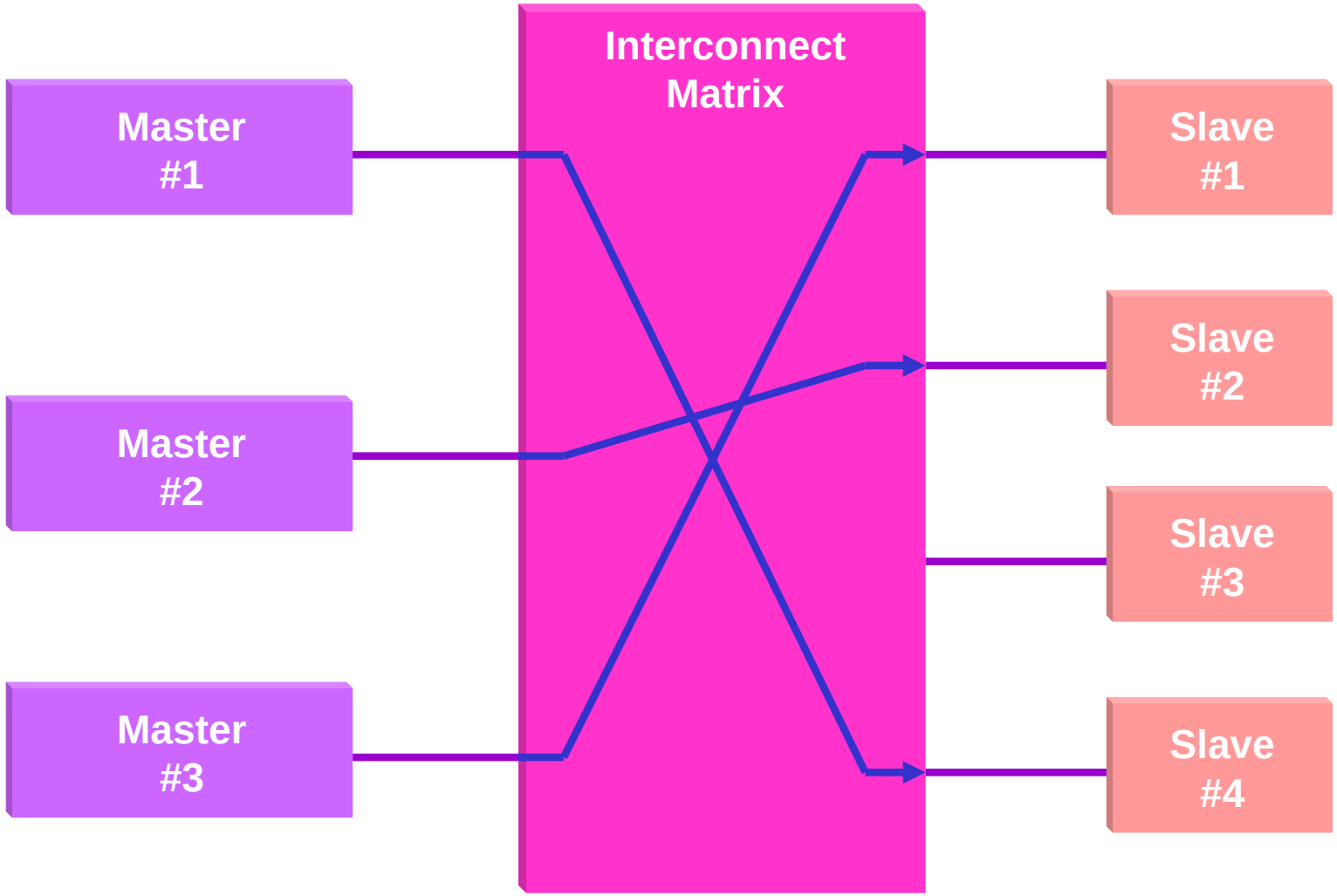
AHB-Lite Master



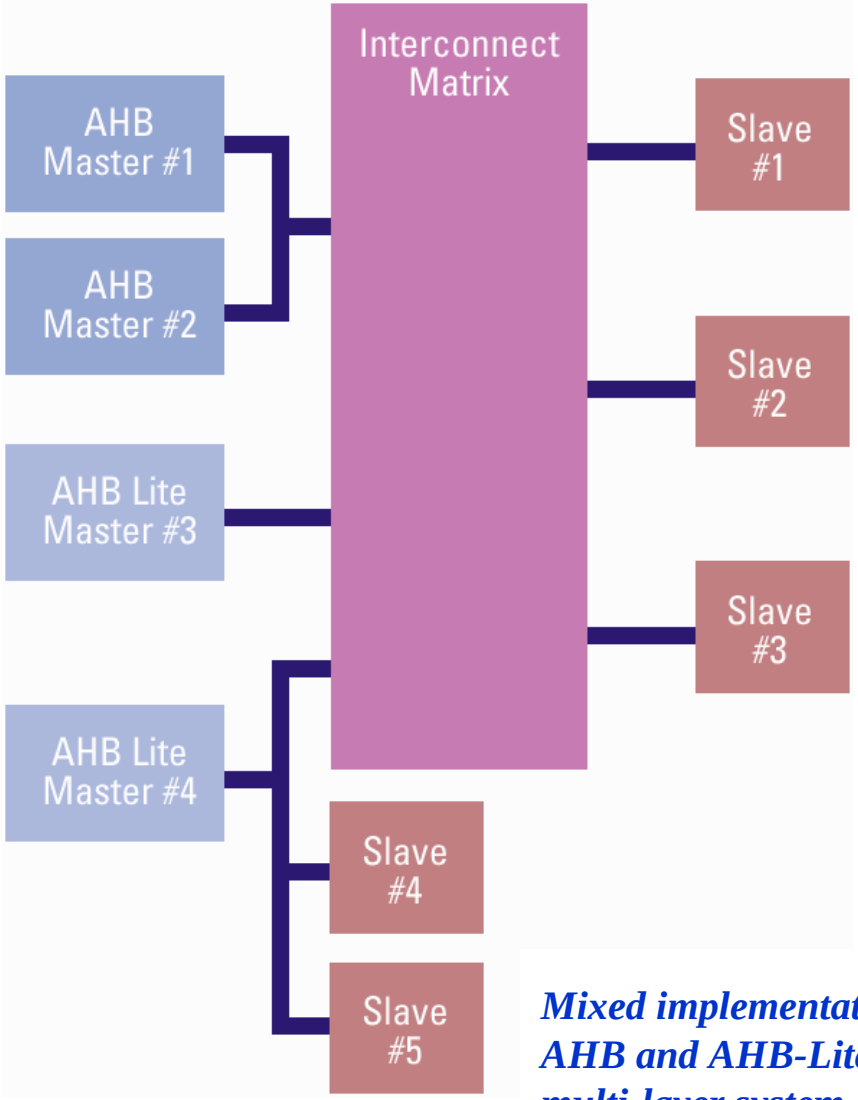
AHB-Lite Slave



Multi-layer AHB (1/2)



Multi-layer AHB (2/2)



Mixed implementation of AHB and AHB-Lite in a multi-layer system.

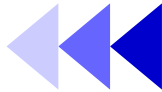
- Local slaves
- Multiple slaves on one slave port
- Multiple masters on one layer

Comparison among AMBA and other OCBs

	OPB	PLB	APB	ASB	AHB	PIbus	PIbus2	Mbus	PalmBus	FISPbus
Width (bits)	8, 16, 32	32 2.9 GB/s, 183 MHz 128 bit	up to 32 bit	8, 16, 32	2^n, n=3~10	8, 16, 32	8, 16, 32,64	8, 16, 32,64	8, 16, 32	
Peak Bandwidth (size/per cycle)	1 Data Tranfer	2 Data Tranfer	4 bytes 0.5 bus width	4 bytes 1 bus width	1 bus width	1 Data Tranfer	1 Data Tranfer	Data Bus Width	Data Bus Width	0.5/1 Data Tranfer (v1/v2)
Timing Guidelines	%	%	Symbolic term	Symbolic term	Symbolic term	early, middle. late	early, middle. late			
Synchronous			rising clock edge	falling clock edge*	rising clock edge	rising clock edge	rising clock edge			
Data Bus Implementation	Distributed And-Or/ Multiple xor	Multiple xor			Multiple xor	Tristate	Tristate			

Source - Black : OCB 1 1.0
- Other colors : Update

ARM Cores and Their Bus Interfaces

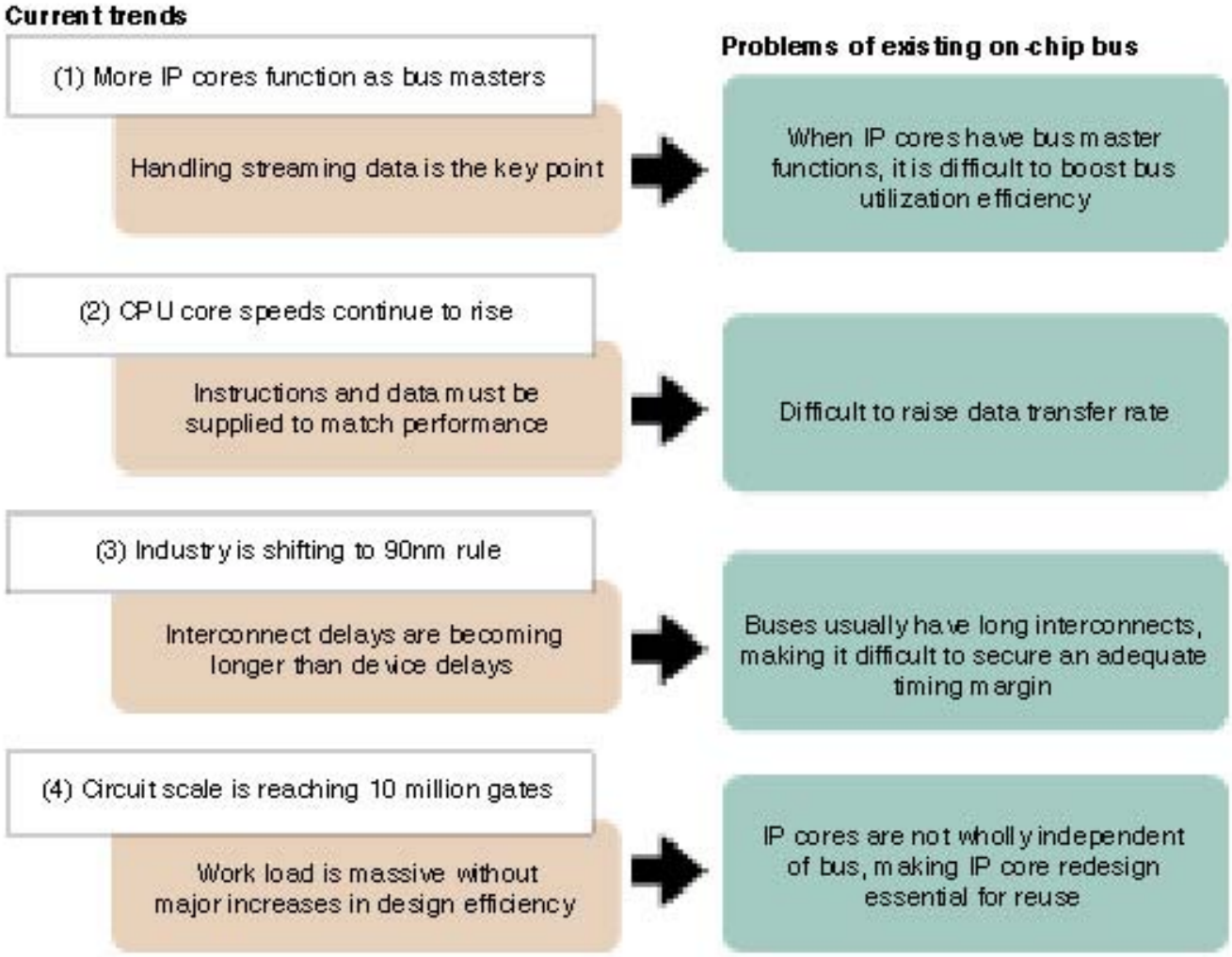


ARM	ARM7TDMI	ARM8	ARM9	ARM1020E
Transistors	74,209	124,554	111,000	7,000,000
Process Technology	0.35u	0.5u	0.25u	0.18u
Clock Rate	66MHz	72MHz	200MHz	400MHz
Vdd	3.3V	-	2.5V	1.5
MIPS	60	-	220	500
Data Bus	32bits	32bits	32bits	32-bit A 64bit W 64-bit R External memory bus interface is AMBA AHB compliant

Outline

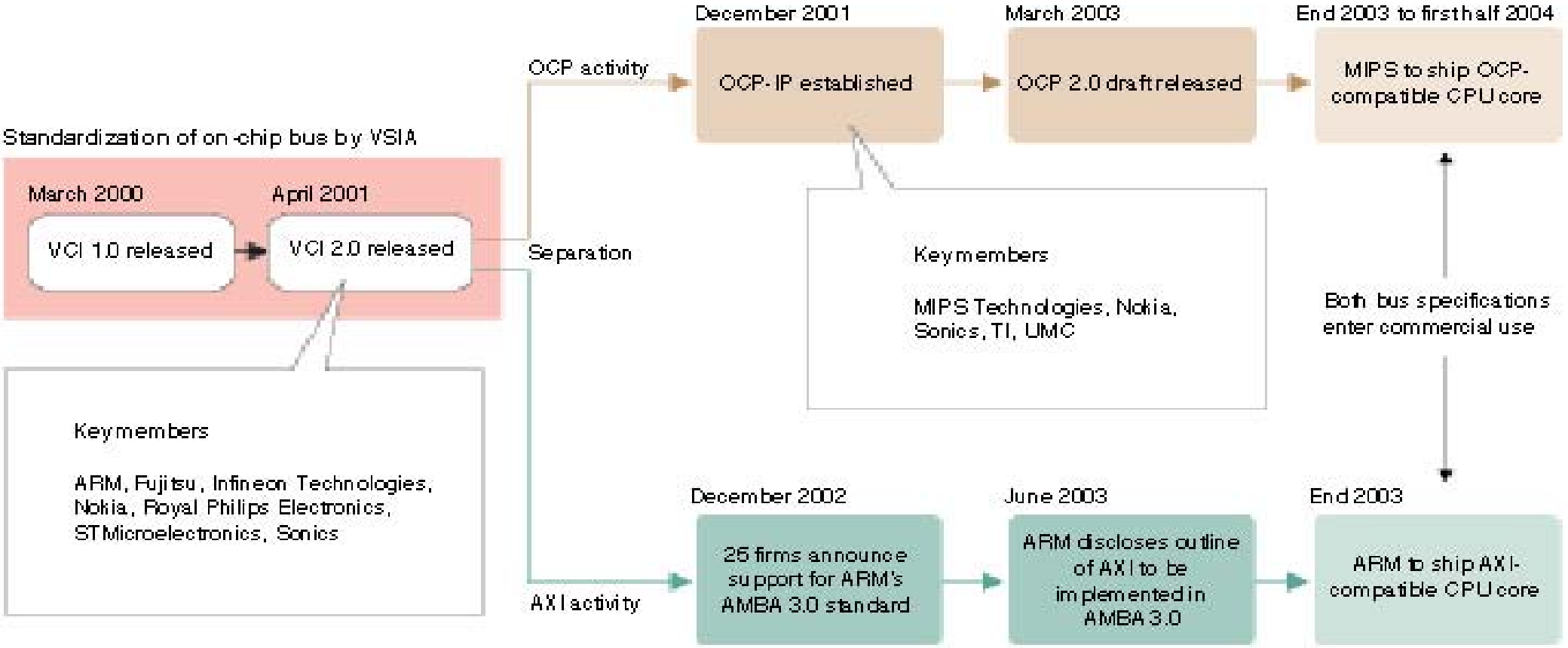
- VCI Interface Standards
- AMBA - On Chip Buses
- AMBA 3.0 - AXI

Design Trends



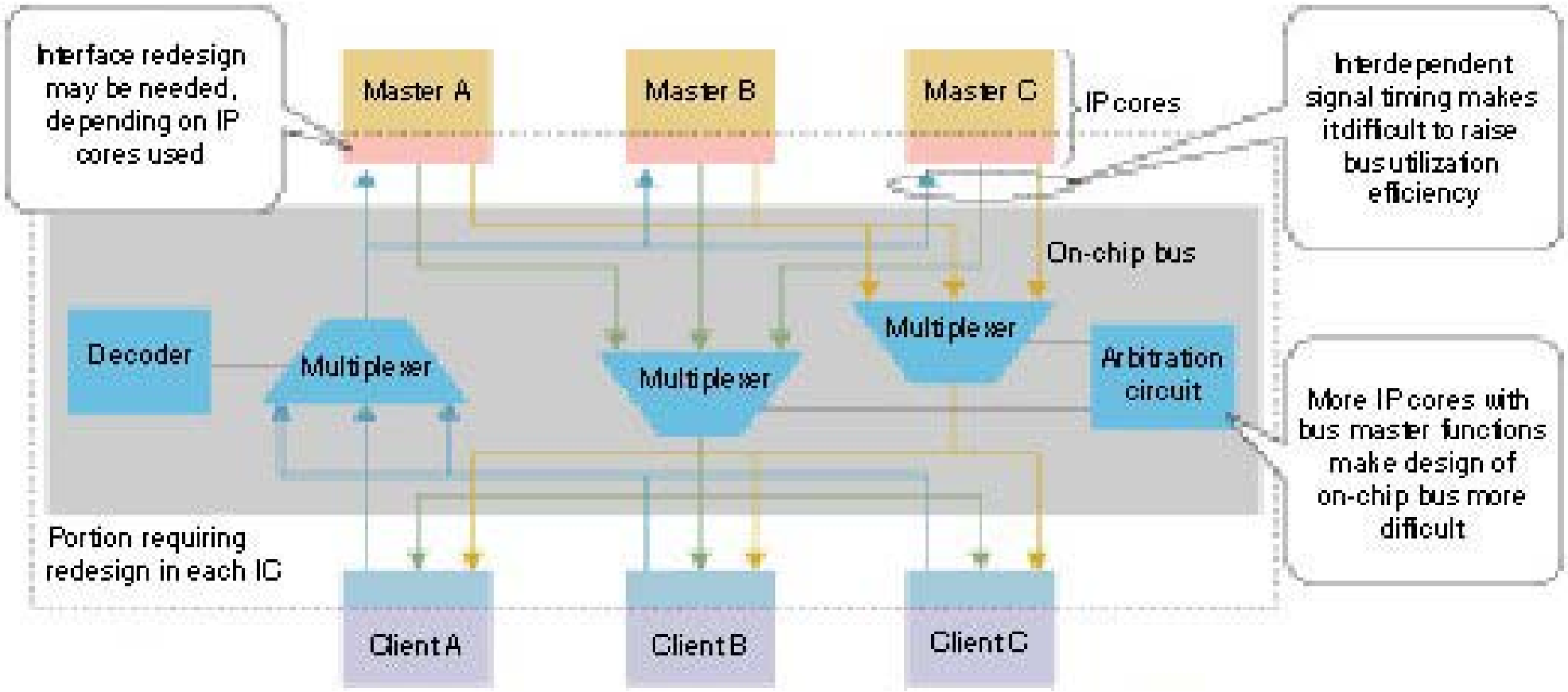
Source: November 2003 Issue, Nikkei Electronics Asia

Evolution of On-chip Bus



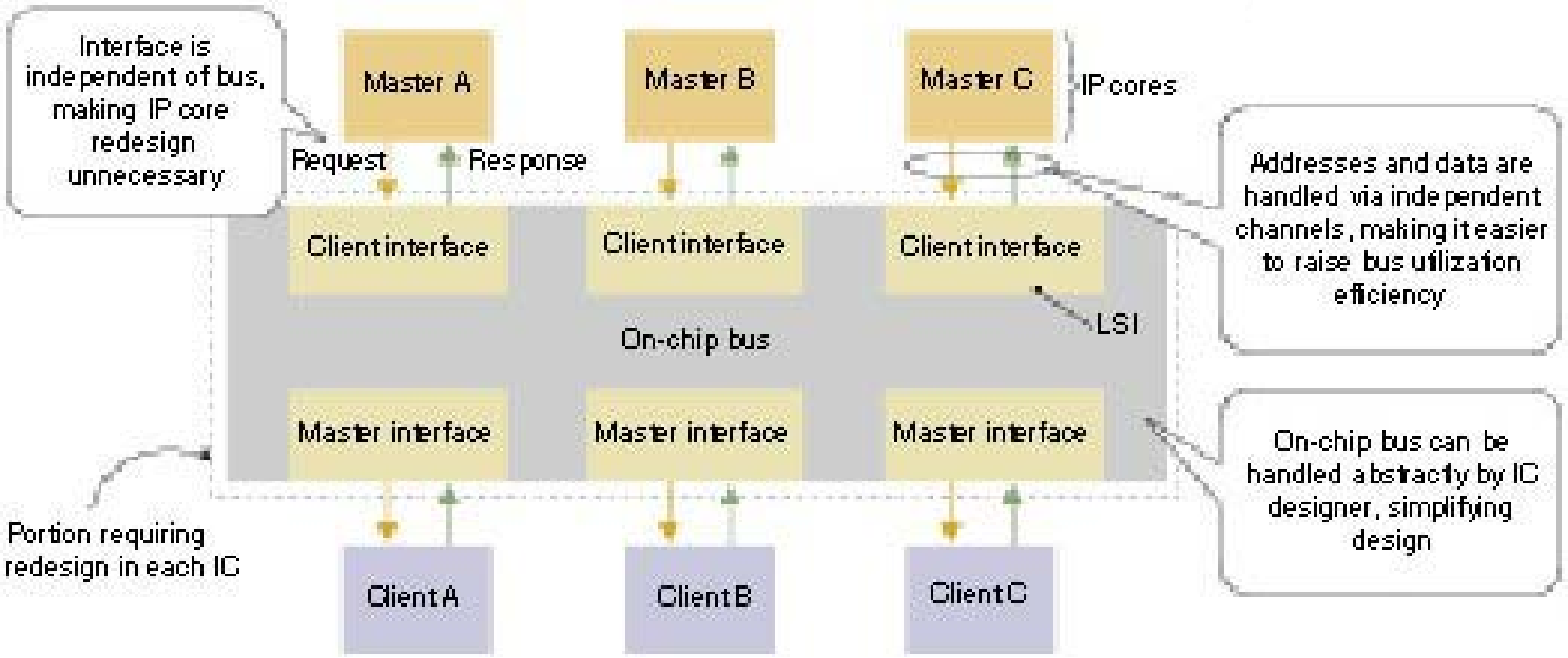
Source: November 2003 Issue, Nikkei Electronics Asia

Problems of Existing OCBs



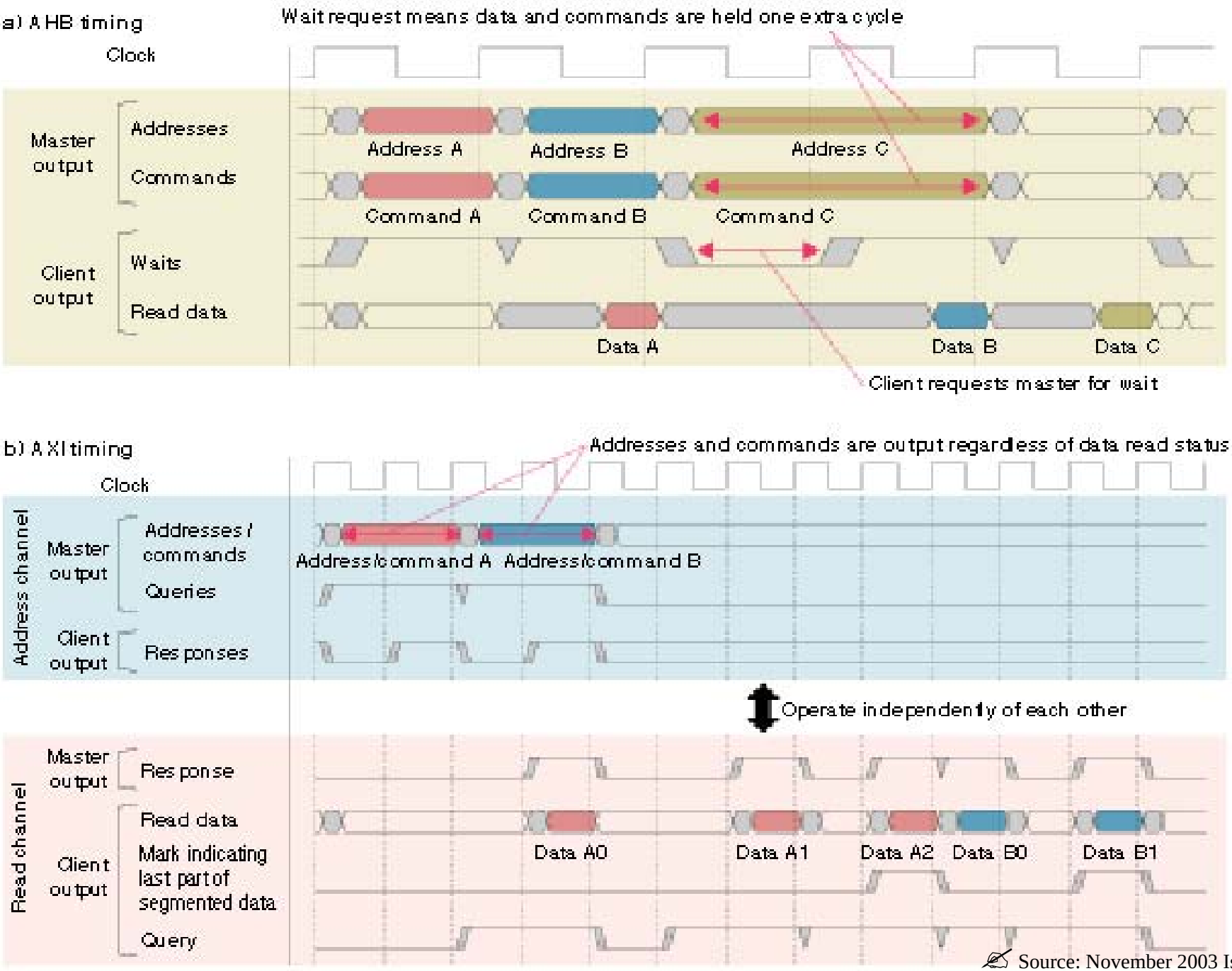
Source: November 2003 Issue, Nikkei Electronics Asia

New Generations of OCBs



Source: November 2003 Issue, Nikkei Electronics Asia

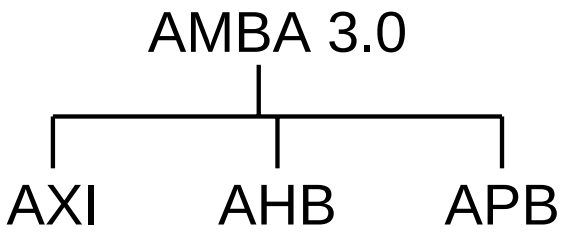
AHB/AXI Timing



Source: November 2003 Issue, Nikkei Electronics Asia

Introduction to AXI

- Brief history of AMBA
  - ASB in 1995
  - AHB in 1999
  - AXI in 2003
- AMBA 3.0 development
  - Over 30 participants

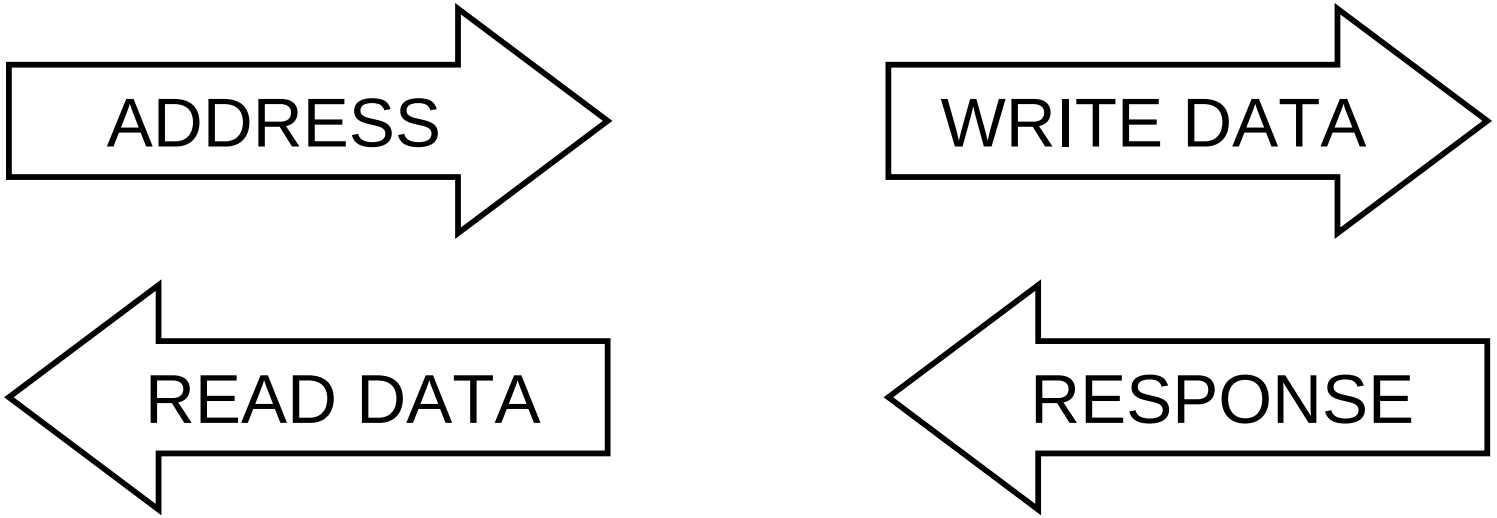


Agere Systems	Fujitsu	NEC Electronics (Europe)
Agilent	Hewlett-Packard Company	OKI Electric Industry
ARM	Infineon	Philips Semiconductors
Atmel	LSI Logic	QUALCOMM
Cadence	Mentor Graphics	Samsung
Conexant Systems	Matsushita	STMicroelectronics
CoWare Inc.	Micronas	Synopsys
Epson	Motorola	Toshiba Corporation
Ericsson Mobile Platforms	NEC Electronics Corporation	Verisity

Source: First details of AXI, ARM

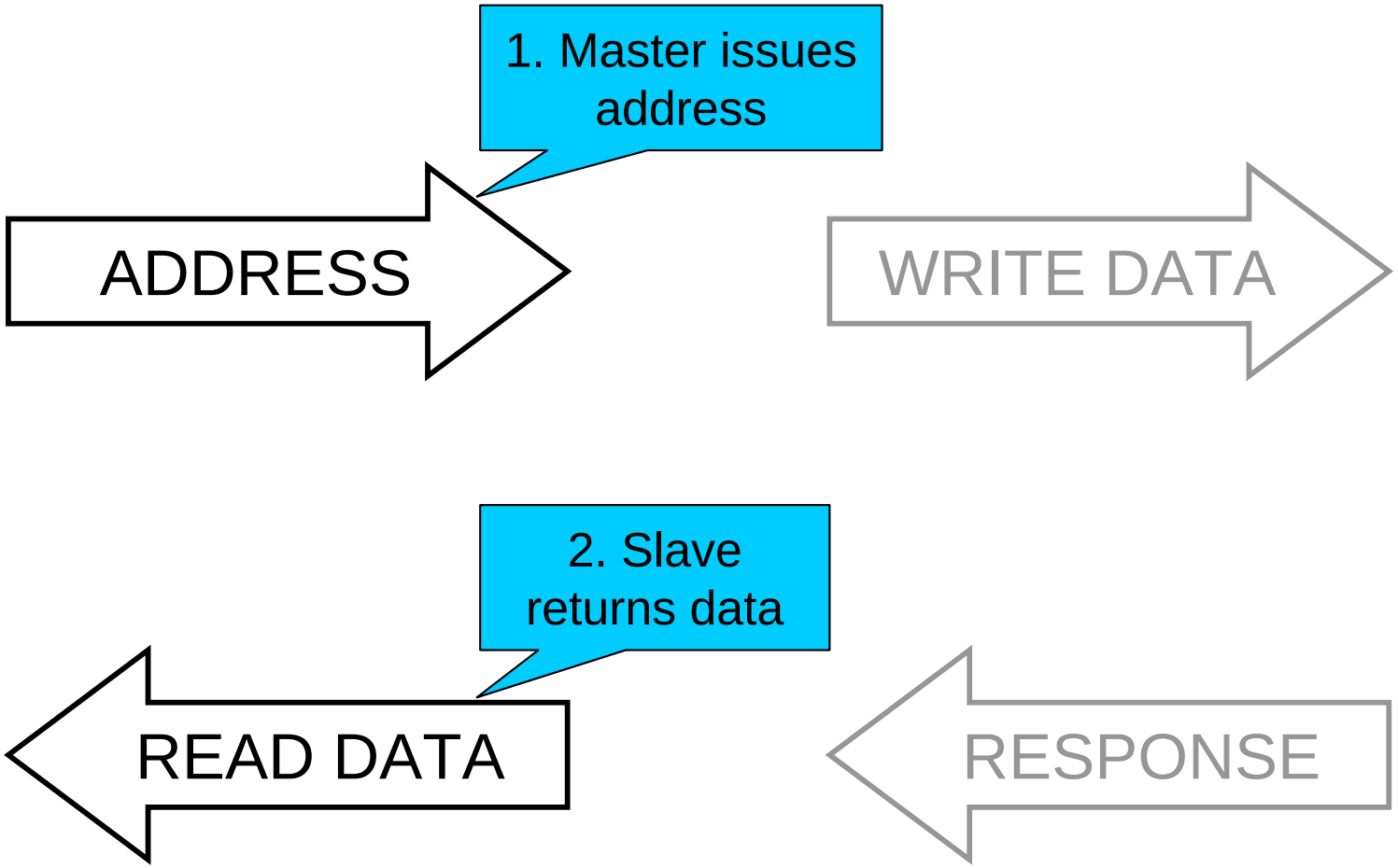
Channel Architecture

- Four groups of signals
  - Address — “A” signal name prefix
  - Read — “R” signal name prefix
  - Write — “W” signal name prefix
  - Write Response — “B” signal name prefix



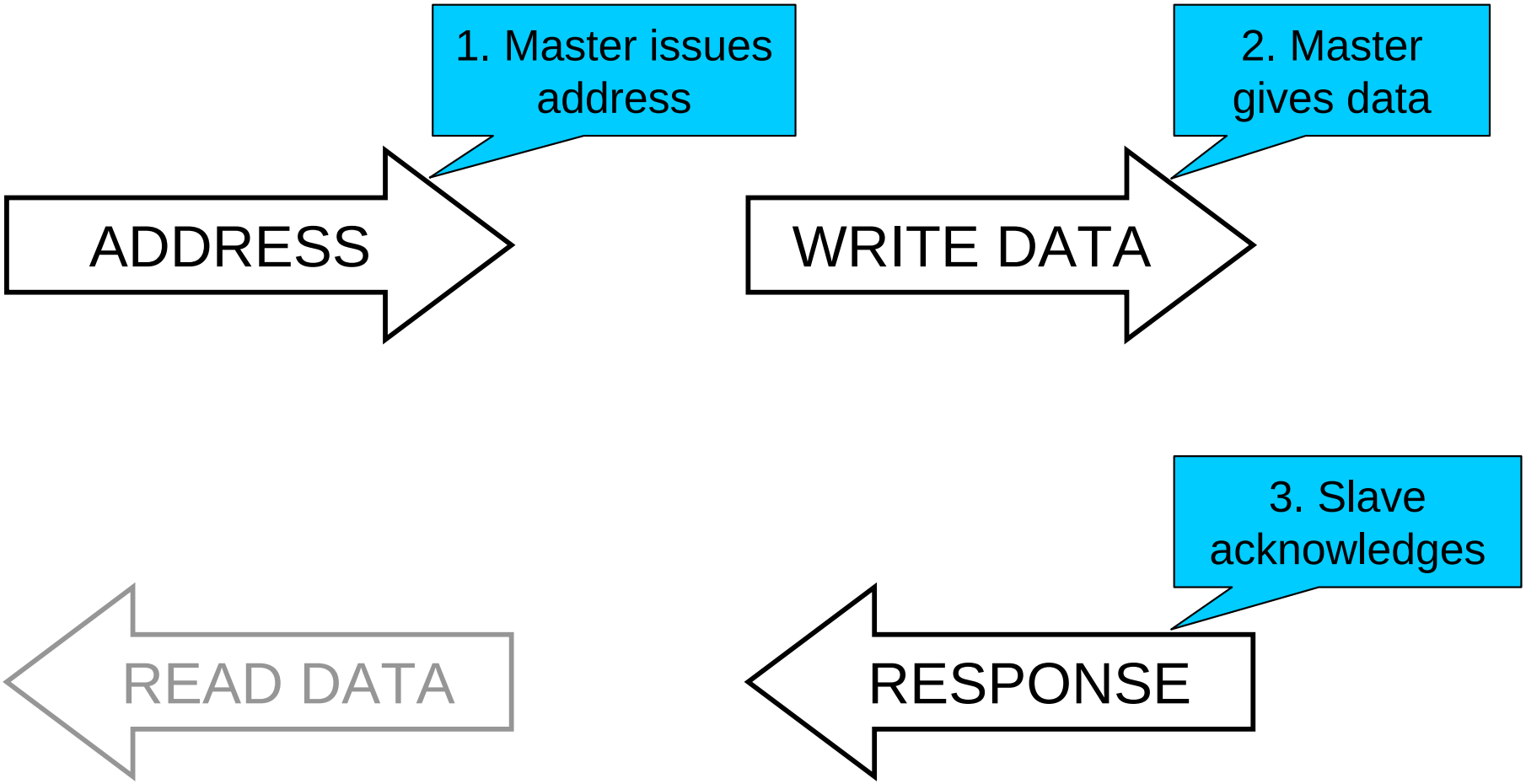
Source: First details of AXI, ARM

To read ...



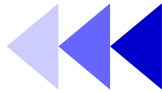
Source: First details of AXI, ARM

To write ...



Source: First details of AXI, ARM

Channels - One way flow



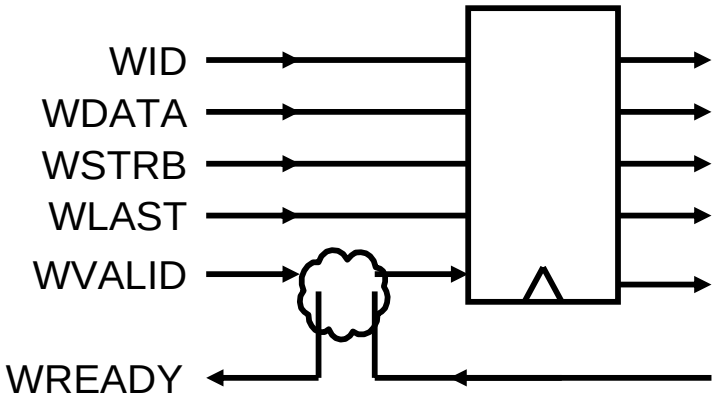
→ AVALID	→ WVALID	← RVALID	← BVALID
→ ADDR	→ WLAST	← RLAST	← BRESP
→ AWRITE	→ WDATA	← RDATA	← BID
→ ALEN	→ WSTRB	← RRESP	→ BREADY
→ ASIZE	→ WID	← RID	
→ ABURST	← WREADY	→ RREADY	
→ ALOCK			
→ ACACHE			
→ APROT			
→ AID			
← AREADY			

- Each channel has information flowing in one direction only
- READY is the only return signal

Source: First details of AXI, ARM

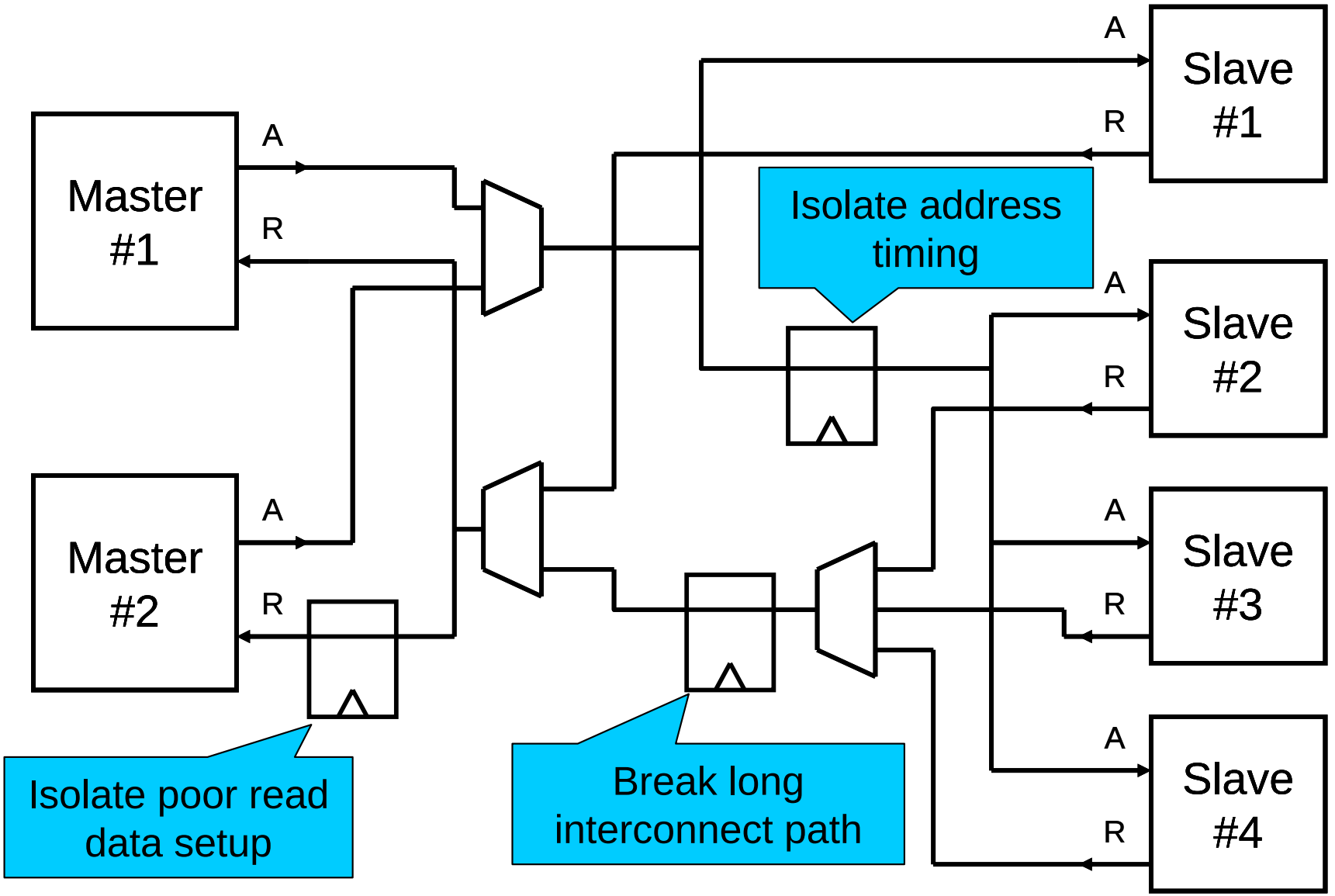
Register slices for max frequency

- Register slices can be applied across any channel
- Allows maximum frequency of operation by matching channel latency to channel delay
- Allows system topology to be matched to performance requirements



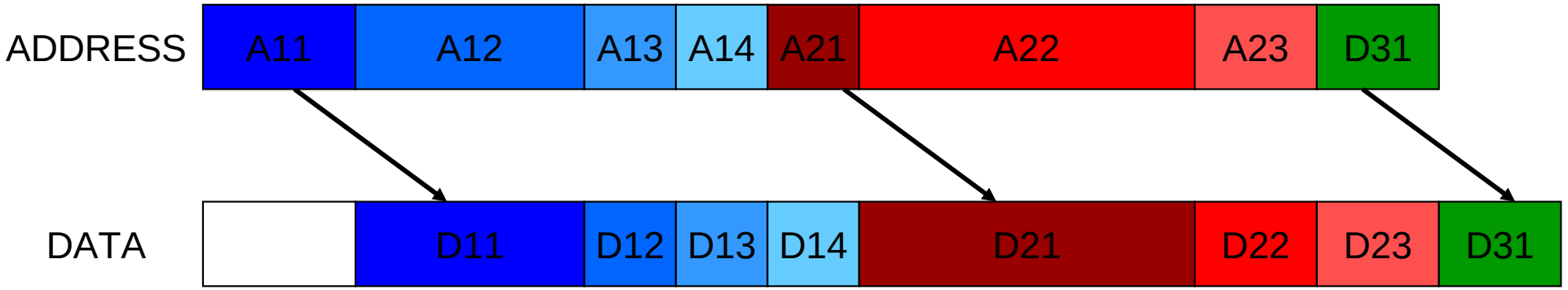
Source: First details of AXI, ARM

Example Register Slices



Source: First details of AXI, ARM

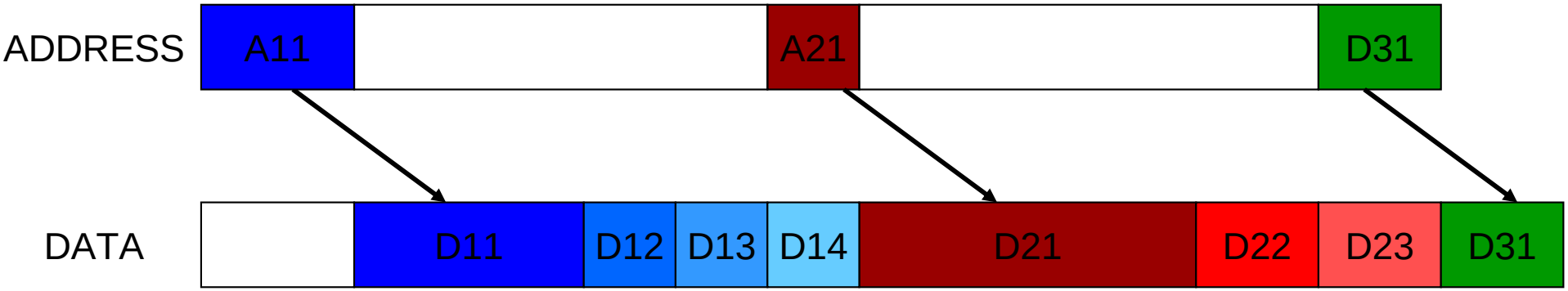
AMBA 2.0 AHB Burst



- AHB Burst
  - Address and Data are locked together
  - Single pipeline stage
  - HREADY controls intervals of address and data

Source: First details of AXI, ARM

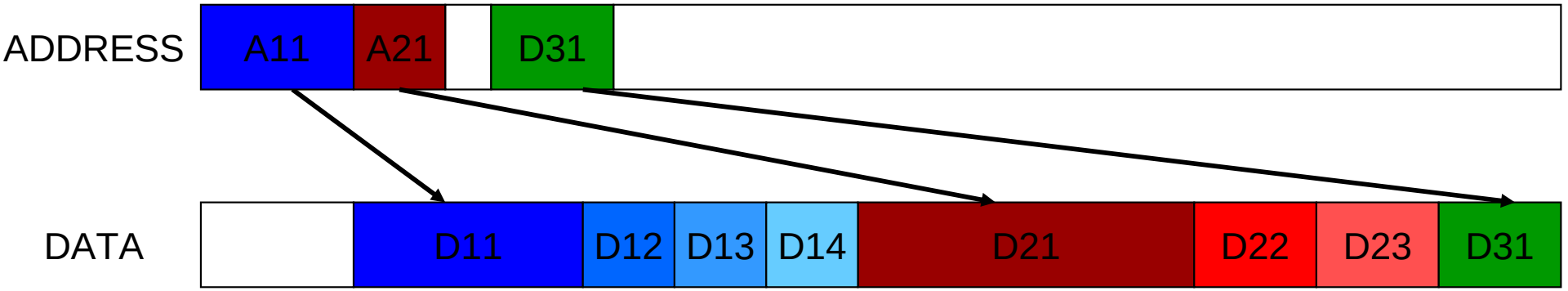
AXI - One Address for Burst



- AXI Burst
  - One Address for entire burst

Source: First details of AXI, ARM

AXI - Outstanding Transactions



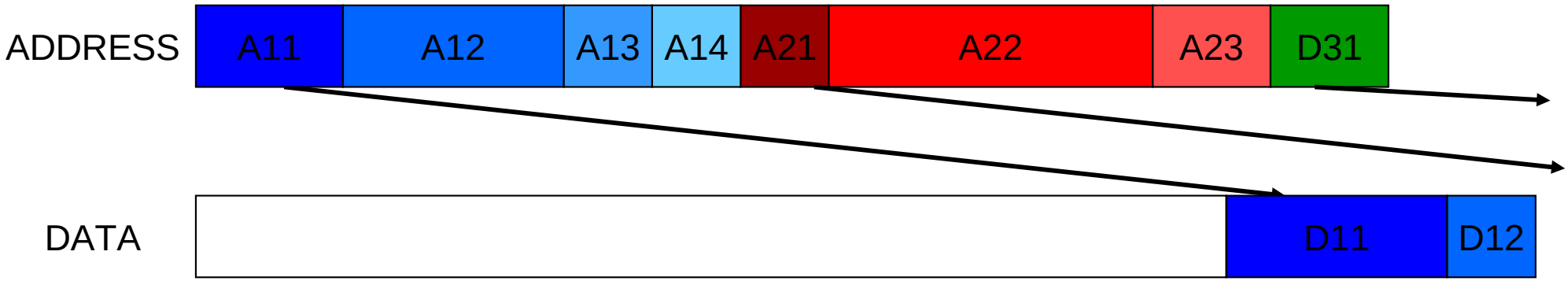
- AXI Burst
  - One Address for entire burst
  - Allows multiple outstanding addresses

Source: First details of AXI, ARM

Out of Order Interface

- Each transaction has an ID attached
  - Channels have ID signals - AID, RID, etc.
- Transactions with the same ID must be ordered
- Requires bus-level monitoring to ensure correct ordering on each ID
  - Masters can issue multiple ordered addresses

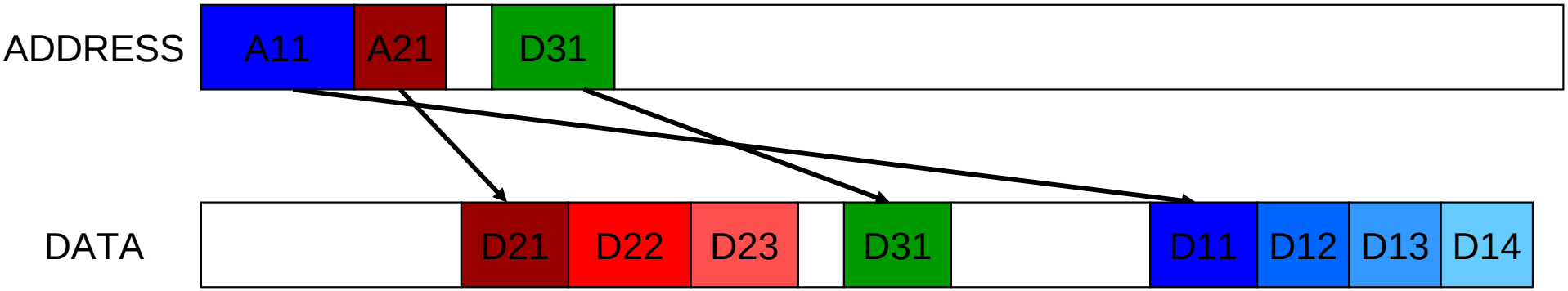
AMBA 2.0 AHB Burst - Slow slave



- With AHB
  - If one slave is very slow, all data is held up.

Source: First details of AXI, ARM

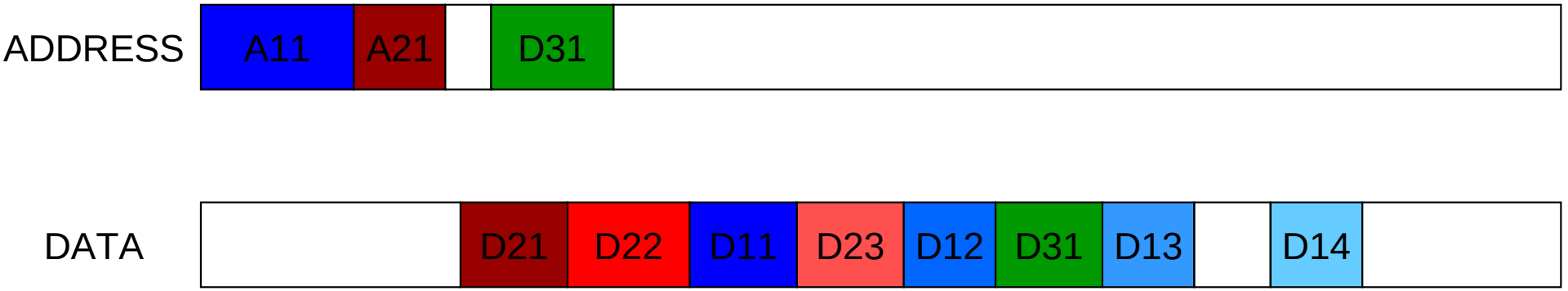
AXI - Out of Order Completion



- Out of order completion allowed
- Fast slaves may return data ahead of slow slaves
- Complex slaves may return data out of order

Source: First details of AXI, ARM

AXI - Data Interleaving

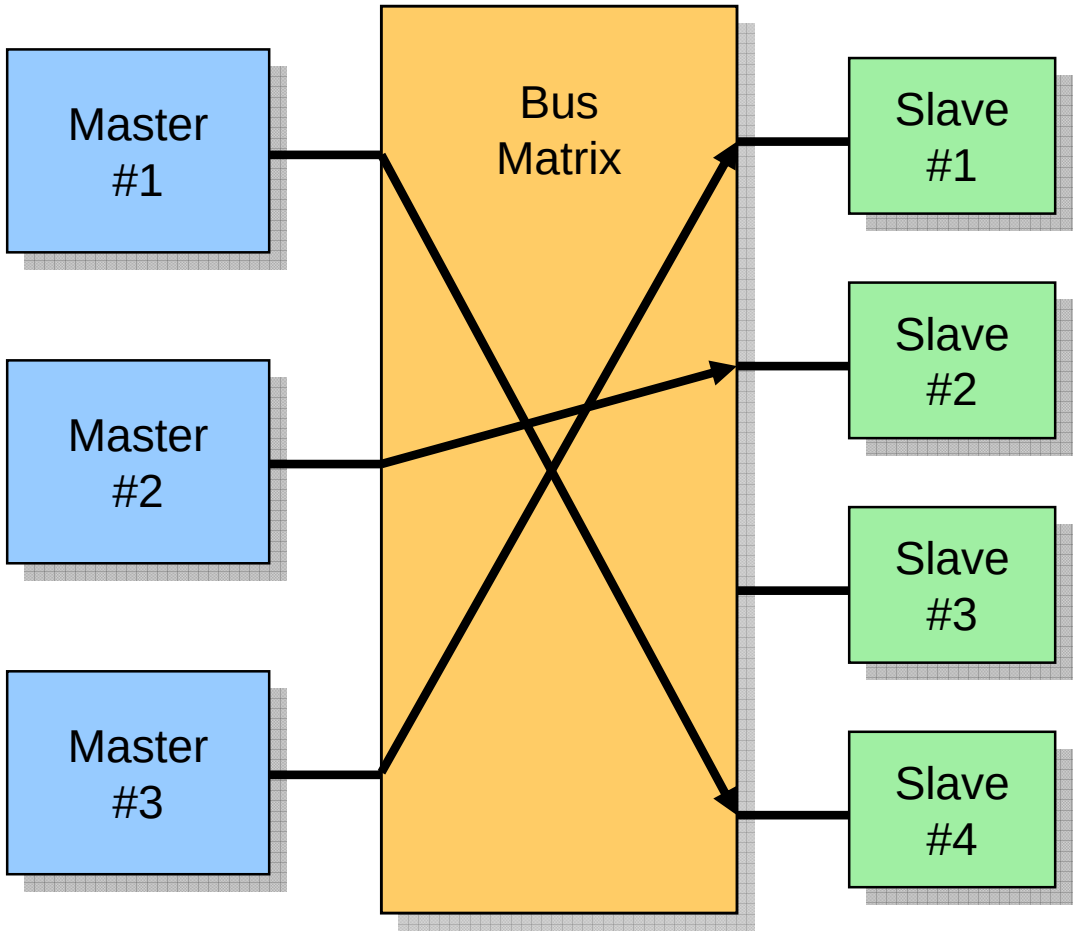


- Returned data can even be interleaved
- Gives maximum use of data bus
- Note - Data within a burst is always in order

Source: First details of AXI, ARM

AXI Multi-layer

- Parallel paths between masters and slaves
- Key Advantages
  - Increased bandwidth
  - Design flexibility
- Uses the same interface protocol



Source: First details of AXI, ARM

Summary

- AXI is the next generation AMBA bus
  - Channel architecture
  - Registers Slices
  - Burst addressing
  - Multiple outstanding bursts
  - Out of order completion
- Interconnect Options
  - Shared bus, multi-layer and mixed