SoC Bus Interconnect Structures

COE838/EE8221: Systems-on-Chip Design
http://www.ecb.torontomu.ca/~courses/coe838/

Dr. Gul N. Khan
http://www.ecb.torontomu.ca/~gnkhan
Electrical, Computer & Biomedical Engineering
Toronto Metropolitan University

Overview

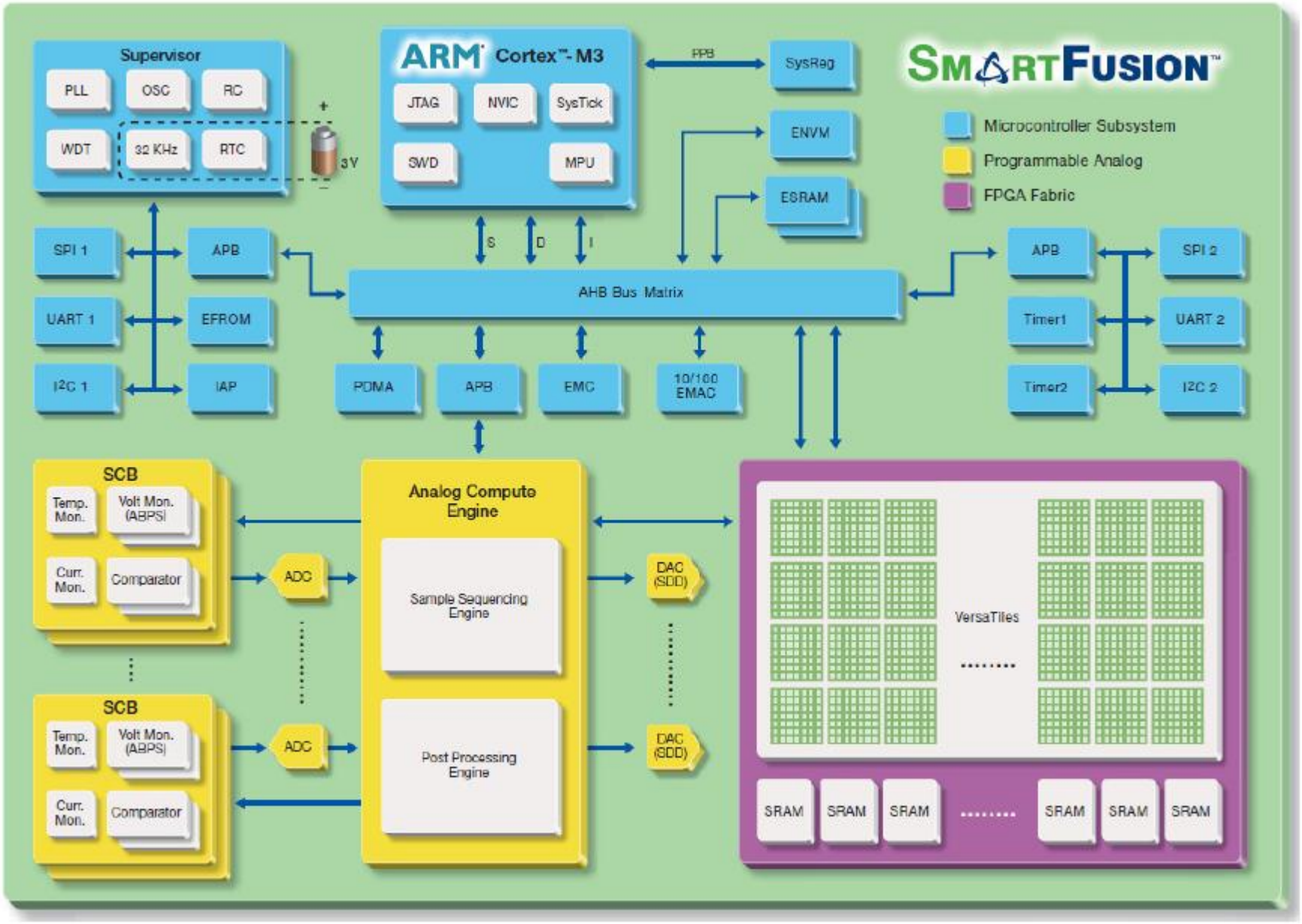
- Basics of a Bus and SoC/On-chip Busses
- AMBA 2.0 and 3.0 - AHB, APB and AXI Protocols
- IBM Core Connect Bus – PLB and OPB
- Avalon Bus
- StBus (STMicroelectronics)

Chapter 5: Computer System Design – System on Chip by M.J. Flynn and W. Luk
Chapter 3: On-Chip Communication Architectures – SoC Interconnect by S. Pasricha & N. Dutt

SoC Integration and Interconnect Architectures

- **SoC Integration is the most important part of SoC design.**
  - Integration of IP cores.
  - The method connect the IP cores.
  - Maximize the reuse of design to lower cost.
- **SoC Interconnect Architectures**
  - Bus-based Interconnection.
  - NoC: Network on Chip that hides the physical interconnects from the designer.

Actel SmartFusion system/bus



SoC Bus Architectures

Technology	AMBA	AXI (AMBA 3)	CoreConnect
Company	ARM	ARM	IBM
Core type	Soft/hard	Soft/hard	Soft
Architecture	Bus	Unidirectional channels	Bus
Bus width	8–1024	8–1024	32/64/128
Frequency	200 MHz	400 MHz*	100–400 MHz
Maximum BW (GB/s)	3	6.4*	2.5–24
Minimum latency (ns)	5	2.5*	15

*As implemented in the ARM PL330 high-speed controller.

BW, bandwidth.

HW Area for a Slave

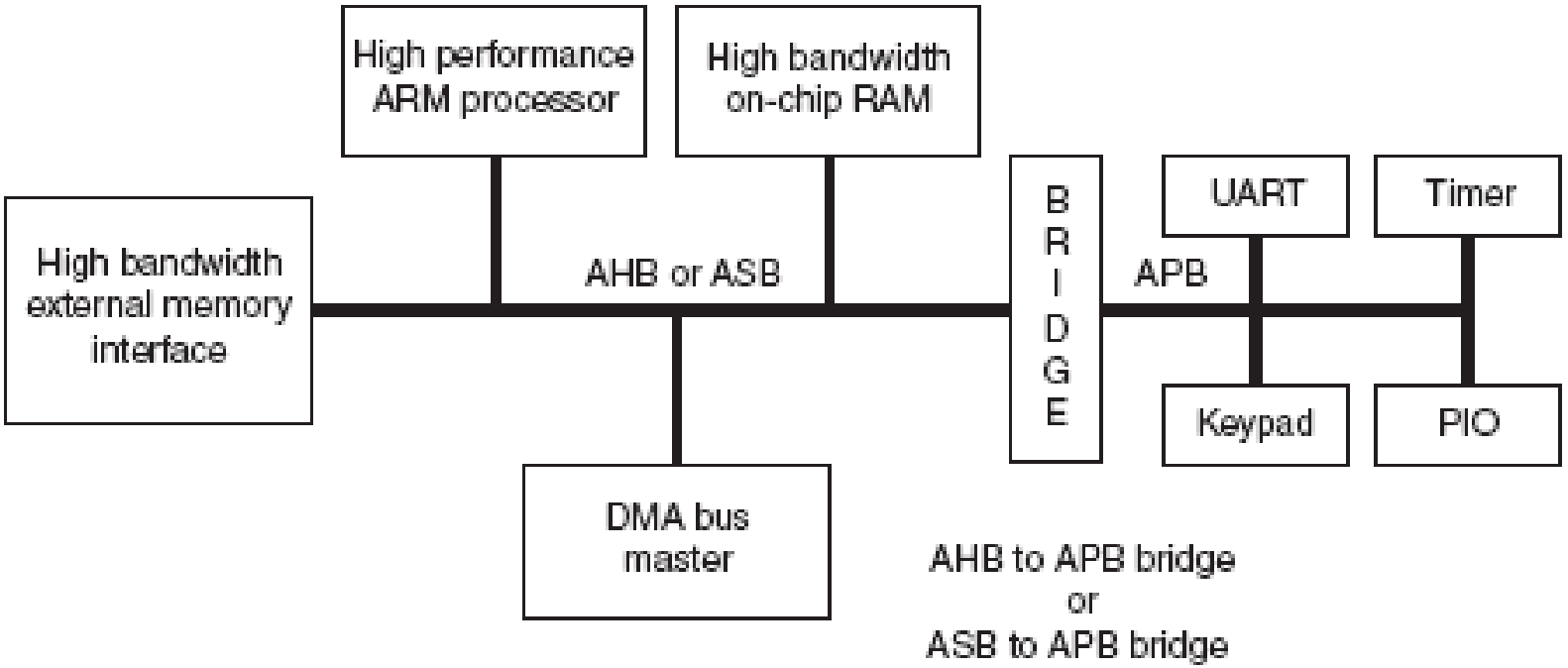
Standard	Speed (MHz)	Area (rbe*)
AMBA(implementation dependent)	166–400	175,000
CoreConnect	66/133/183	160,000

*rbe = register bit equivalent; estimates are approximate and vary by implementation.

On-Chip Busses

- AMBA 2.0, 3.0 (ARM)
- CoreConnect (IBM)
- STBus (STMicroelectronics)
- Sonics Smart Interconnect (Sonics)
- Wishbone (Opencore)
- Avalon (Altera)
- PI Bus (OMI)
- MARBLE (Univ. of Manchester)
- CoreFrame (PalmChip)

AMBA 2.0



AMBA AHB

* High performance
* Pipelined operation
* Multiple bus masters
* Burst transfers
* Split transactions

AMBA ASB

* High performance
* Pipelined operation
* Multiple bus masters

AMBA APB

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

AMBA Busses

Advanced Microcontroller Bus Architecture

Actually three standards: APB, AHB, and AXI

AHB – Advanced High-Performance Bus

- Pipelining of Address / Data
- Split Transactions
- Multiple Masters

APB – Advanced Peripheral Bus

- Low Power / Bandwidth Peripheral Bus

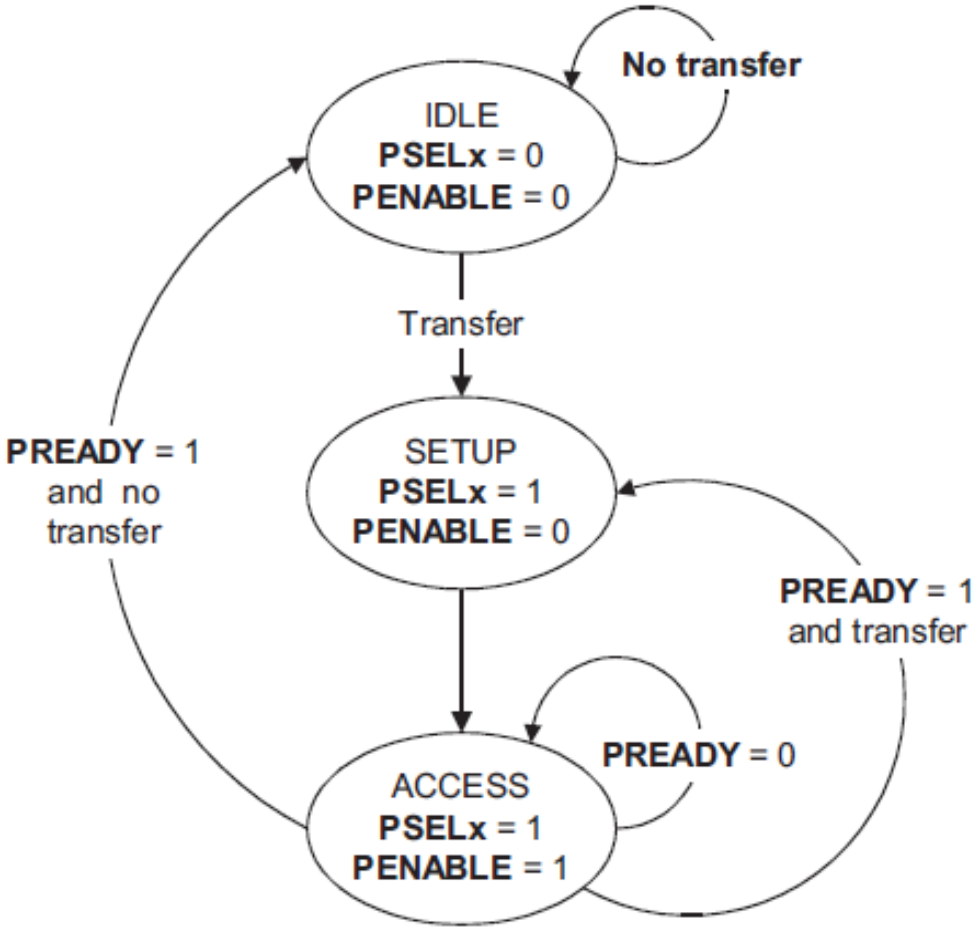
Very commonly used for commercial IP cores

APB Bus

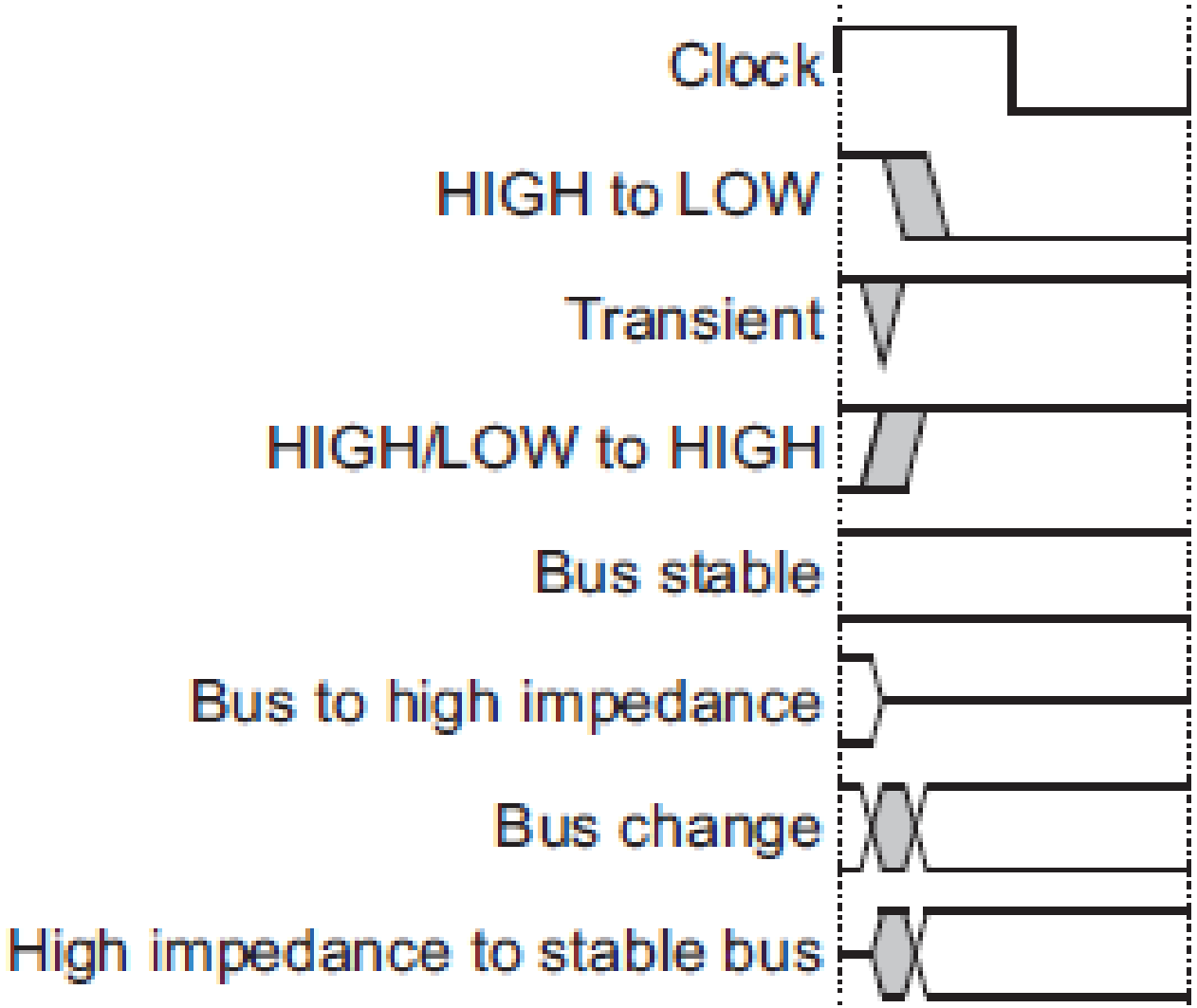
- A simple bus that is easy to work with
- Low-cost
- Low-power
- Low-complexity
- Low-bandwidth
- Non-pipelined
- Ideal for peripherals

APB bus state machine

- **IDLE**
  - Default APB state
- **SETUP**
  - When transfer required
  - **PSELx** is asserted
  - Only one cycle
- **ACCESS**
  - **PENABLE** is asserted
  - Addr, write, select, and write data remain stable
  - Stay if **PREADY** = L
  - Goto IDLE if **PREADY** = H and no more data
  - Goto SETUP is **PREADY** = H and more data pending

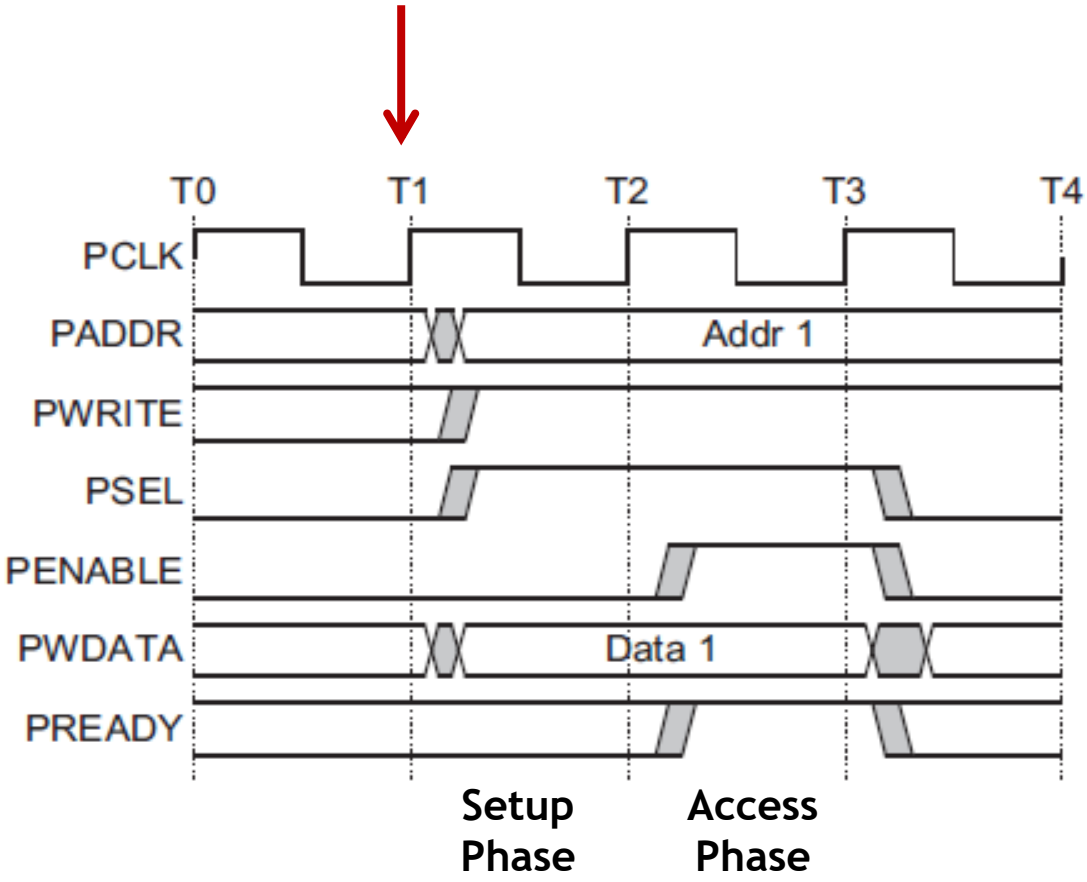


Notations


Clock
HIGH to LOW
Transient
HIGH/LOW to HIGH
Bus stable
Bus to high impedance
Bus change
High impedance to stable bus

APB bus States

- IDLE
  - Default APB state
- SETUP
  - When transfer required
  - PSELx is asserted
  - Only one cycle
- ACCESS
  - PENABLE is asserted
  - Addr, write, select, and write data remain stable
  - Stay if PREADY = L
  - Goto IDLE if PREADY=H and no more data
  - Goto SETUP is PREADY is H and more data pending

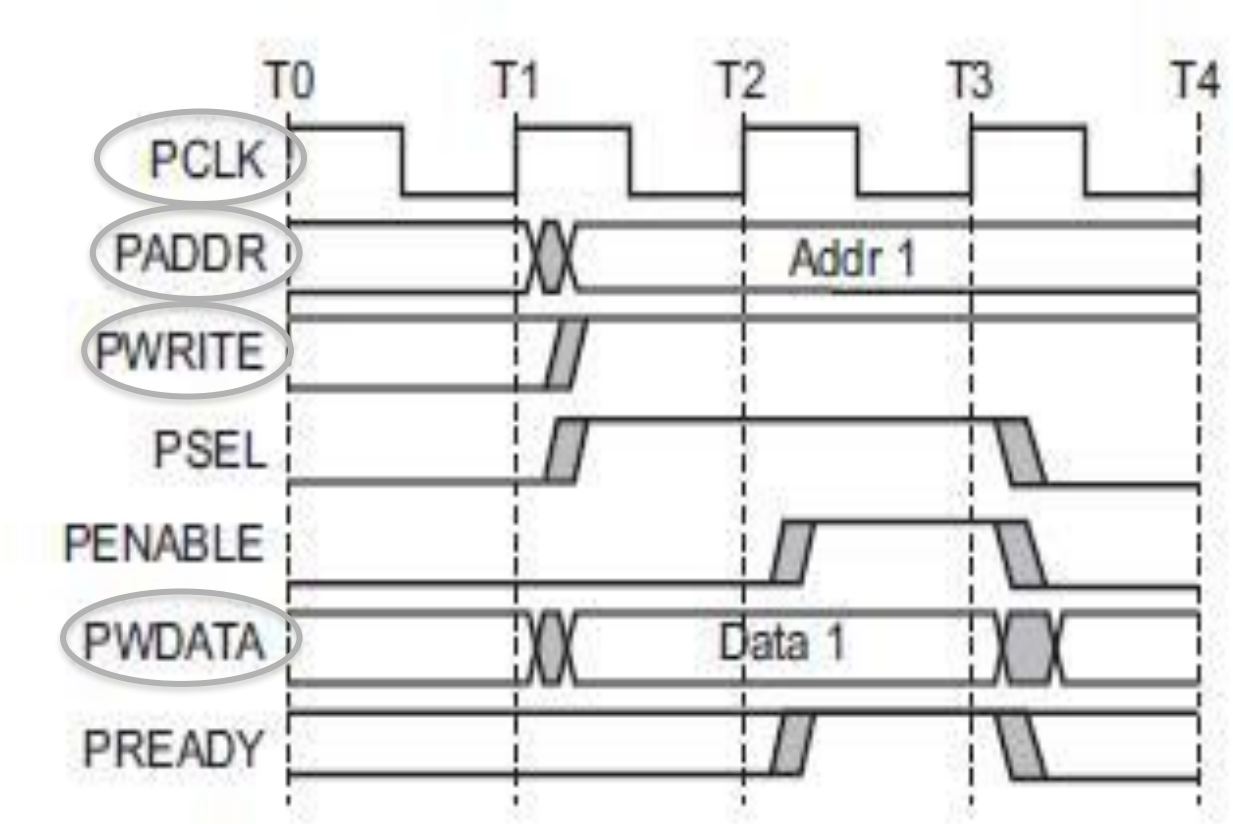


APB Signals

- PCLK: bus clock source (rising-edge triggered)
- PRESETn: bus (and typically system) reset signal (active low)
- PADDR: APB address bus (can be up to 32-bits wide)
- PSELx: select line for each slave device
- PENABLE: indicates the 2nd and subsequent cycles of an APB xfer
- PWRITE: indicates transfer direction (Write=H, Read=L)
- PWDATA: write data bus (can be up to 32-bits wide)
- PREADY: used to extend a transfer
- PRDATA: read data bus (can be up to 32-bits wide)
- PSLVERR: indicates a transfer error (OKAY=L, ERROR=H)

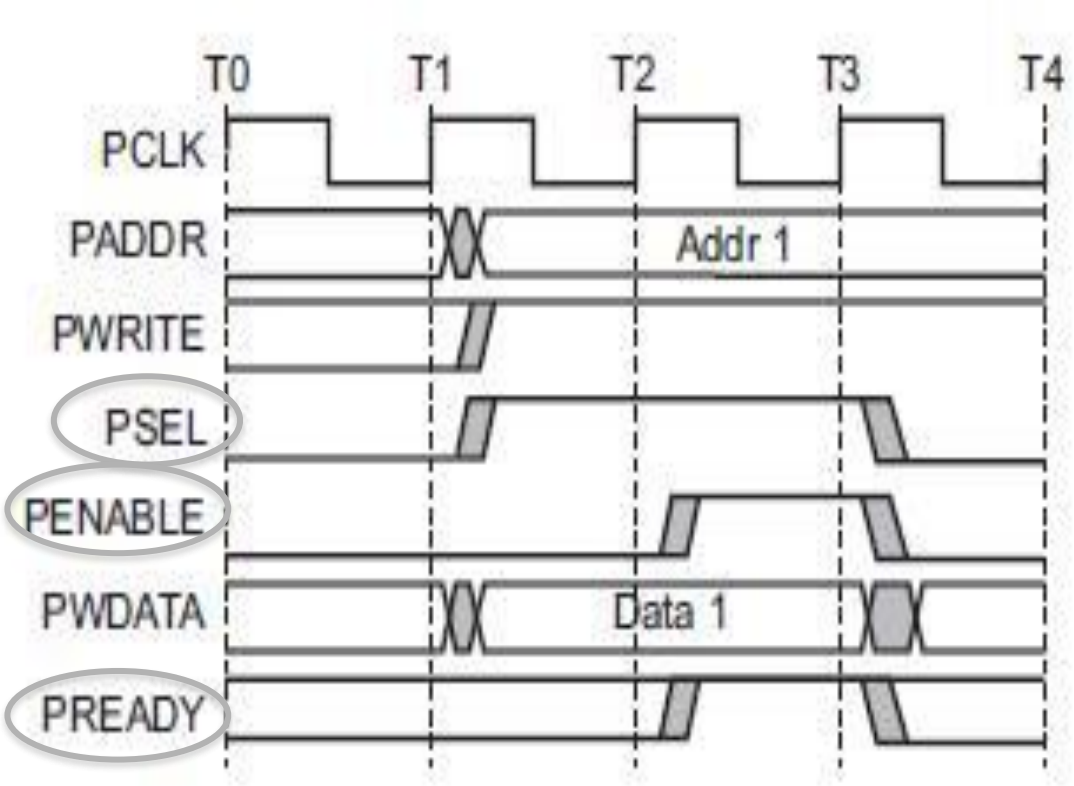
APB bus Signals

- PCLK
  Clock
- PADDR
  Address on bus
- PWRITE
  1=Write,
  0=Read
- PWDATA
  * Data written to the I/O device.
  * Supplied by the bus master/processor.



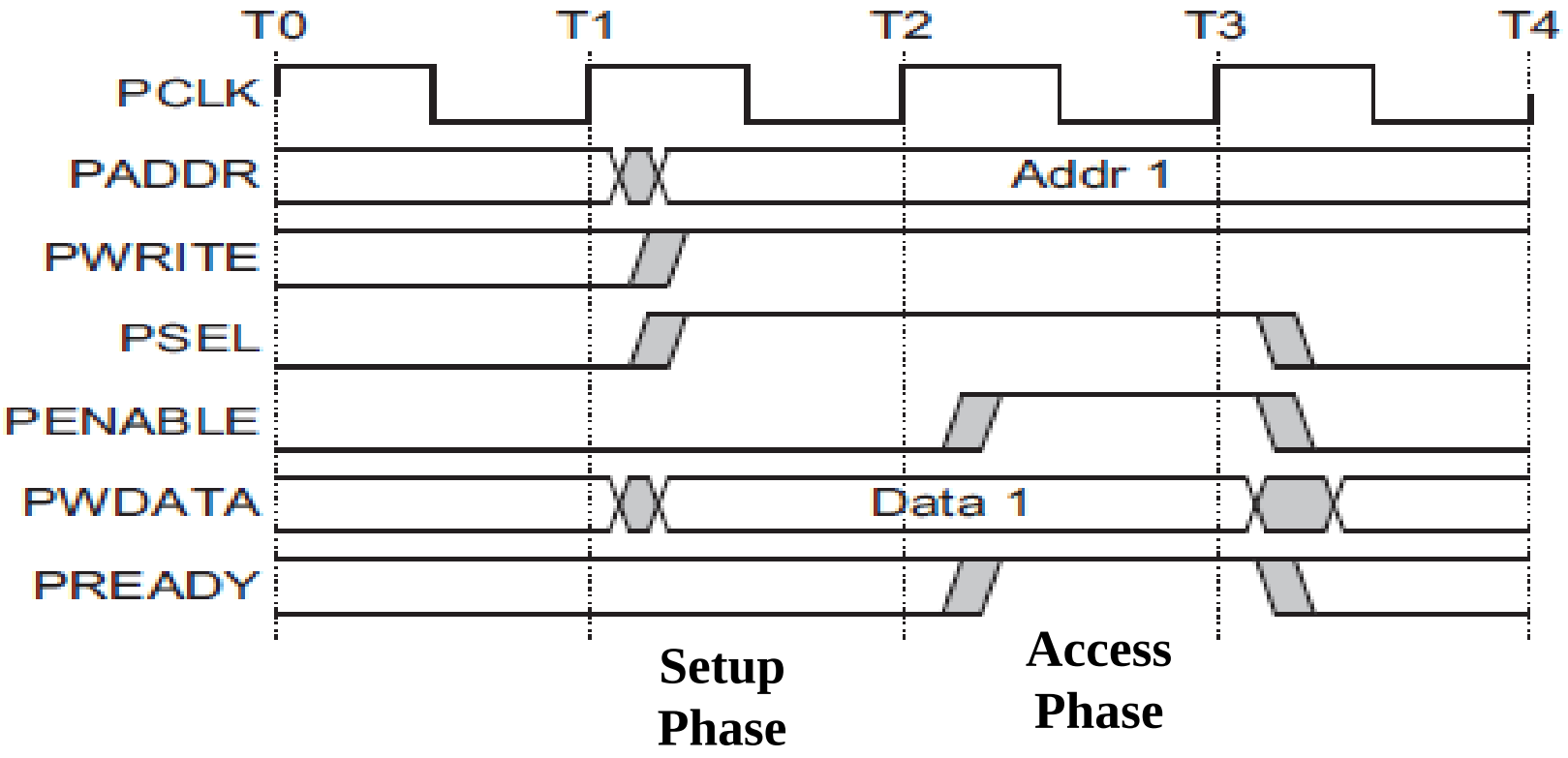
APB bus signals

- PSEL
  - Asserted if the current bus transaction is targeted to ***this*** device
- PENABLE
  - High during entire transaction *other than* the first cycle.
- PREADY
  - Driven by target. Similar to our #ACK. Indicates if the target is *ready* to do transaction.



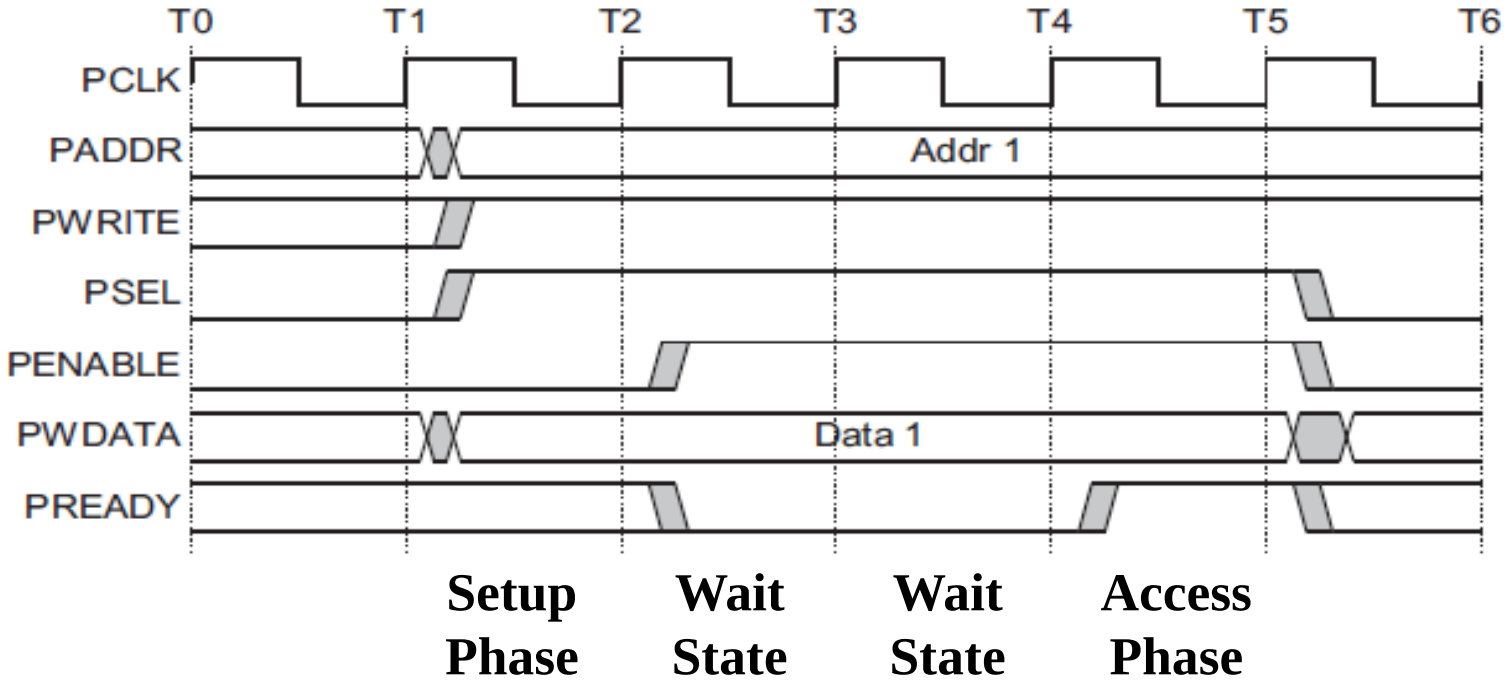
A Write Transfer - No Wait States

Setup phase begins with this rising edge

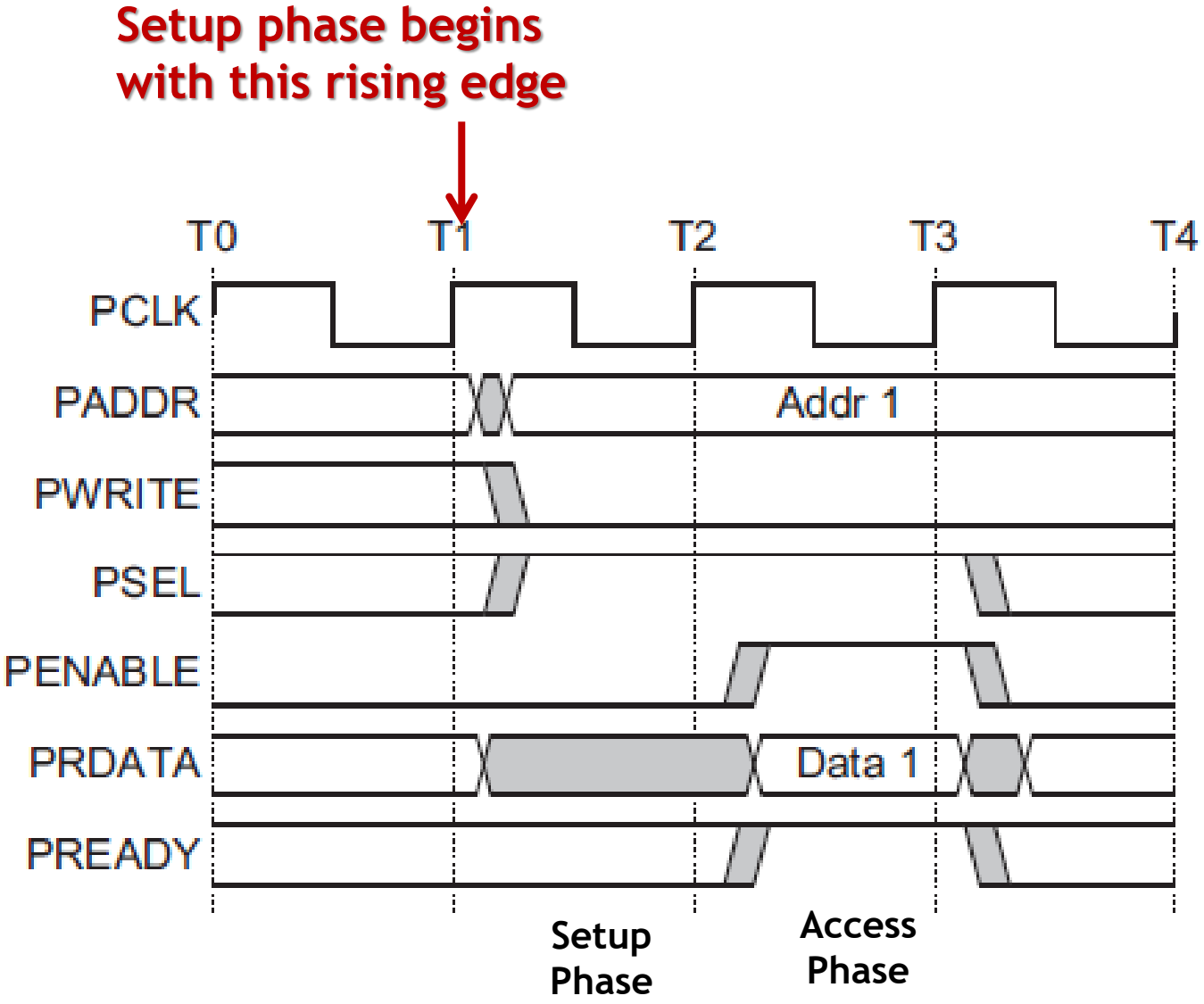


A Write Transfer with Wait States

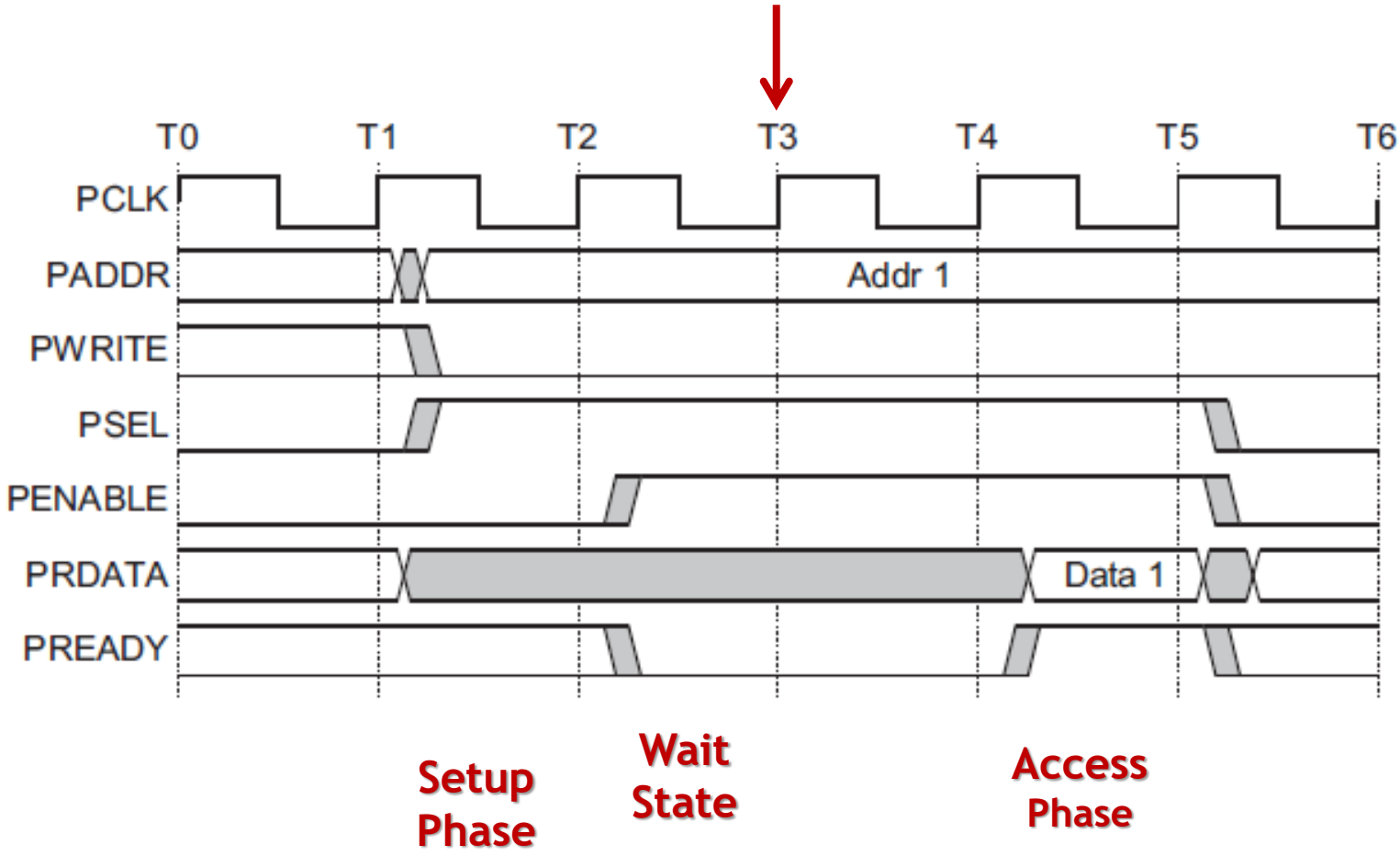
Wait as Peripheral is not ready



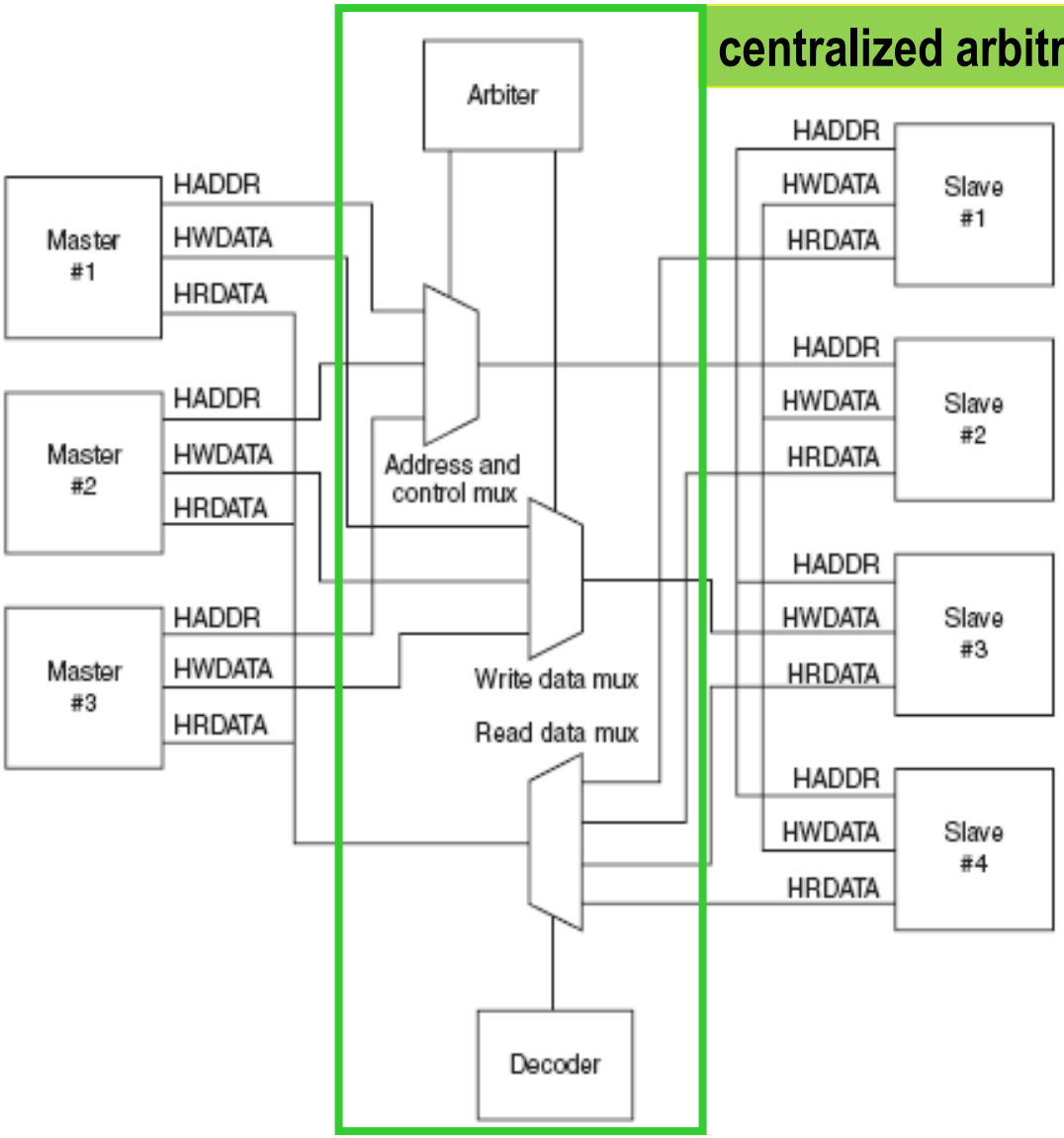
A Read Transfer - No Wait States



A Read Transfer with Wait States



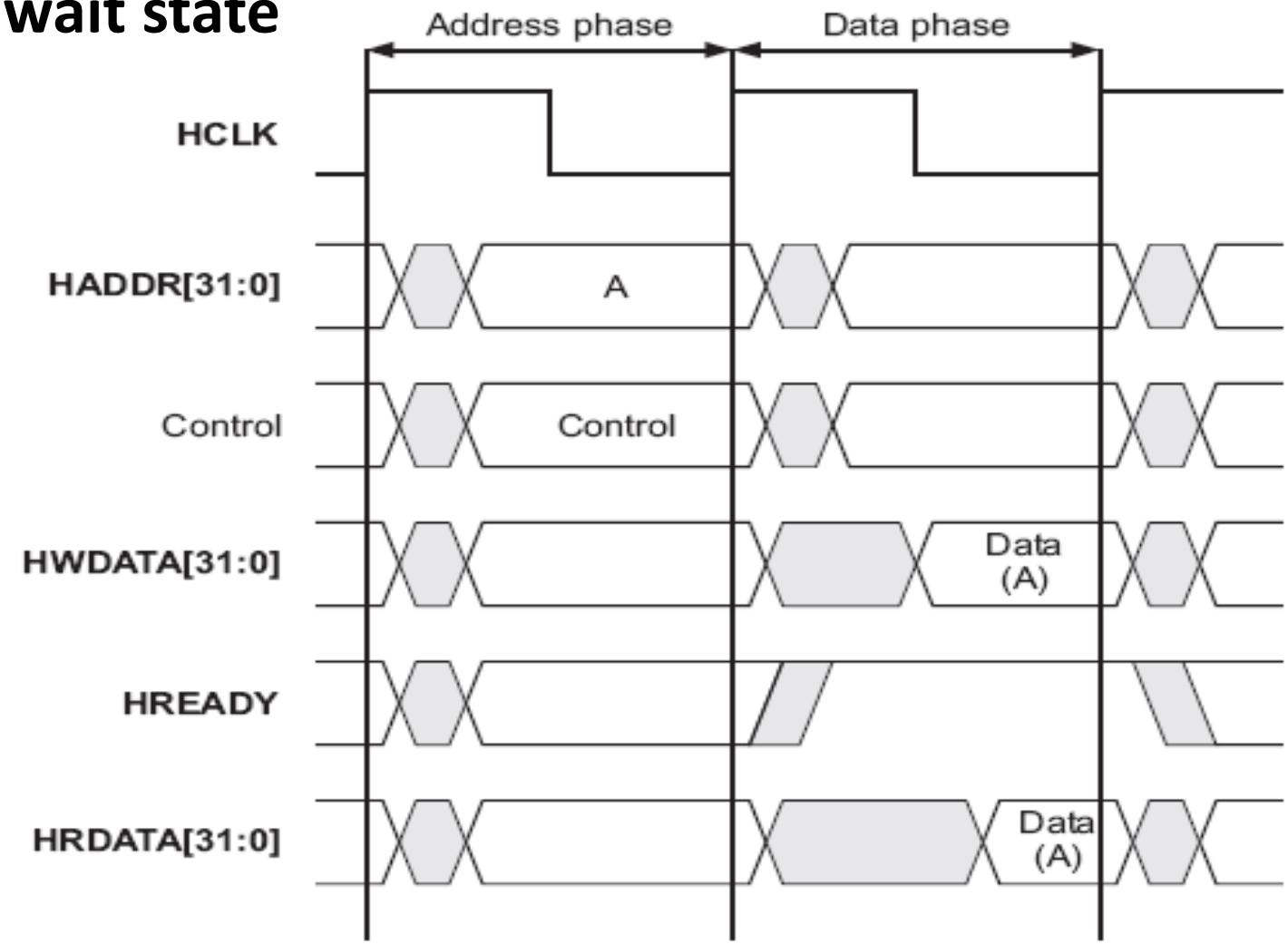
AHB Bus



- one unidirectional address bus (HADDR)
- two unidirectional data buses (HWDATA, HRDATA)
- At any time only one active data bus

Simple AHB Transfer

no wait state

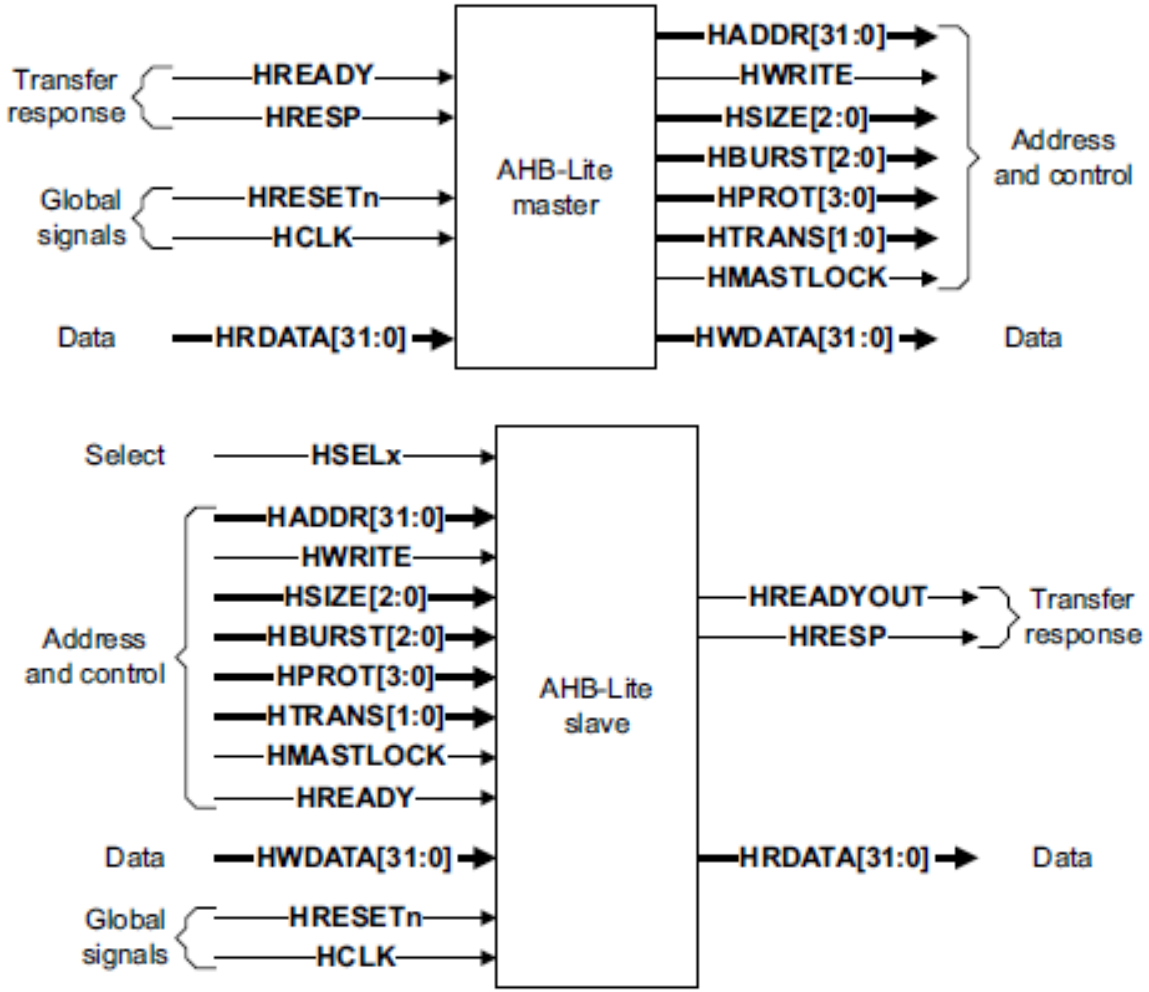


AHB-Lite Bus Master/Slave Interface
AHB-Lite: Single Master

Global signals
- **HCLK**
- **HRESETn**

Master out/Slave in
- **HADDR (address)**
- **HWDATA (write data) Control**
  - HWRITE
  - HSIZE
  - HBURST
  - HPROT
  - HTRANS
  - HMASTLOCK

Slave out/Master in
- **HRDATA (read data)**
- **HREADY**
- **HRESP**



AHB-LITE Signals

Global Signals

- HCLK: the bus clock source (rising-edge triggered)
- HRESETn: the bus (and system) reset signal (active low)

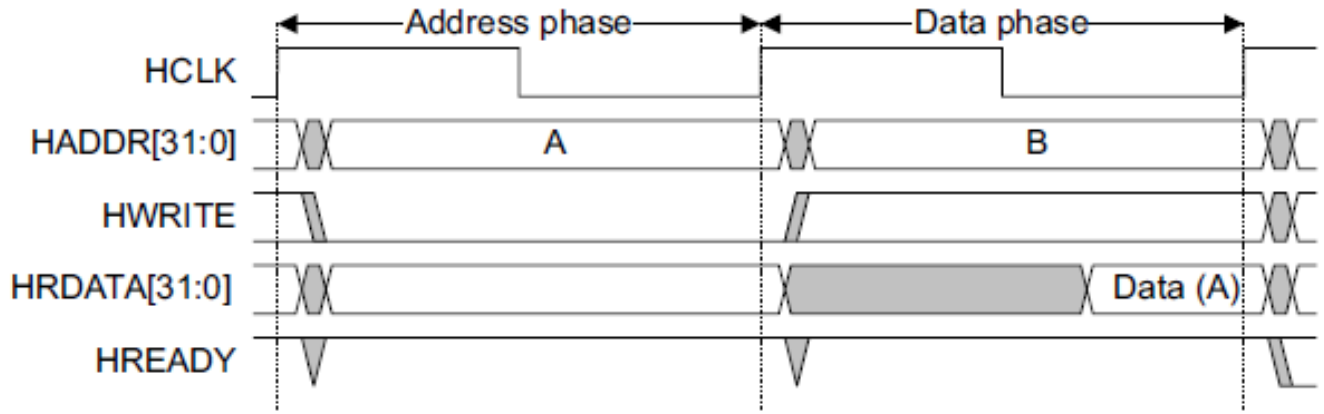
Master out/slave in

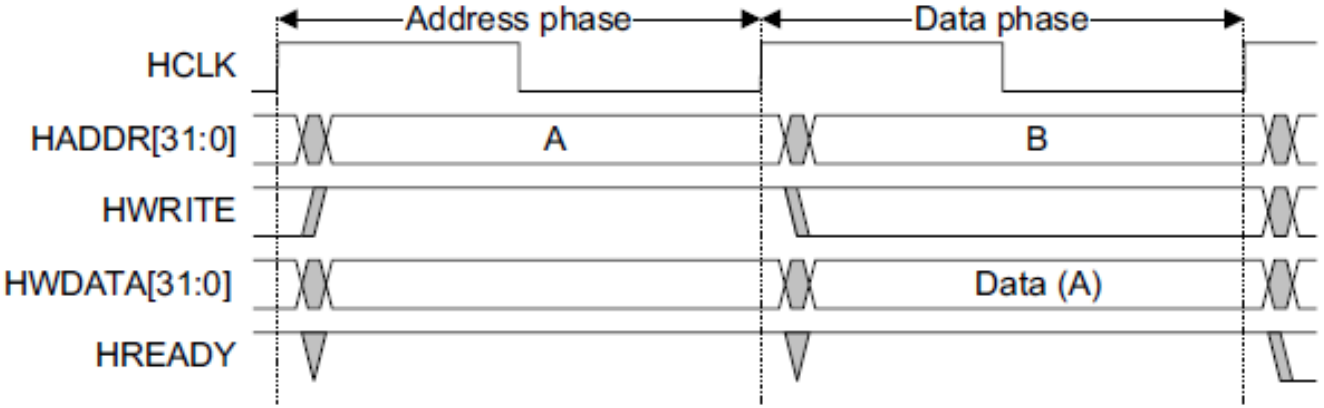
- HADDR[31:0]: the 32-bit system address bus
- HWDATA[31:0]: the system write data bus
- Control
  - HWRITE: indicates transfer direction (Write=1, Read=0)
  - HSIZE[2:0]: indicates size of transfer (byte, halfword, or word)
  - HBURST[2:0]: indicates single or burst transfer (1, 4, 8, 16 beats)
  - HPROT[3:0]: provides protection information (e.g. I or D; user or handler)
  - HTRANS: indicates current transfer type (e.g. idle, busy, nonseq, seq)
  - HMASTLOCK: indicates a locked (atomic) transfer sequence

Slave out/master in

- HRDATA[31:0]: the slave read data bus
- HREADY: indicates previous transfer is complete
- HRESP: the transfer response (OKAY=0, ERROR=1)

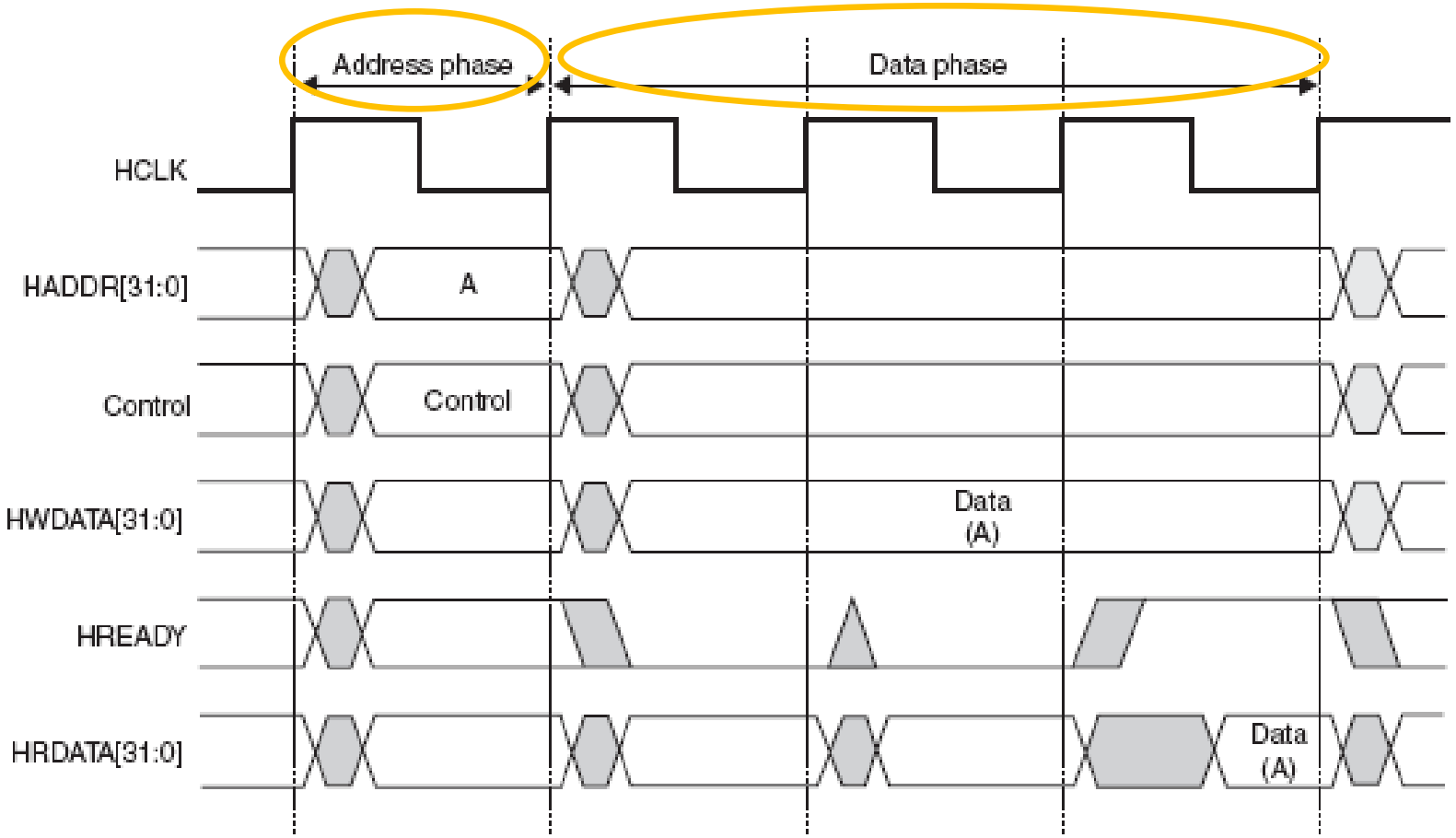
Basic Read and Write – No Wait States





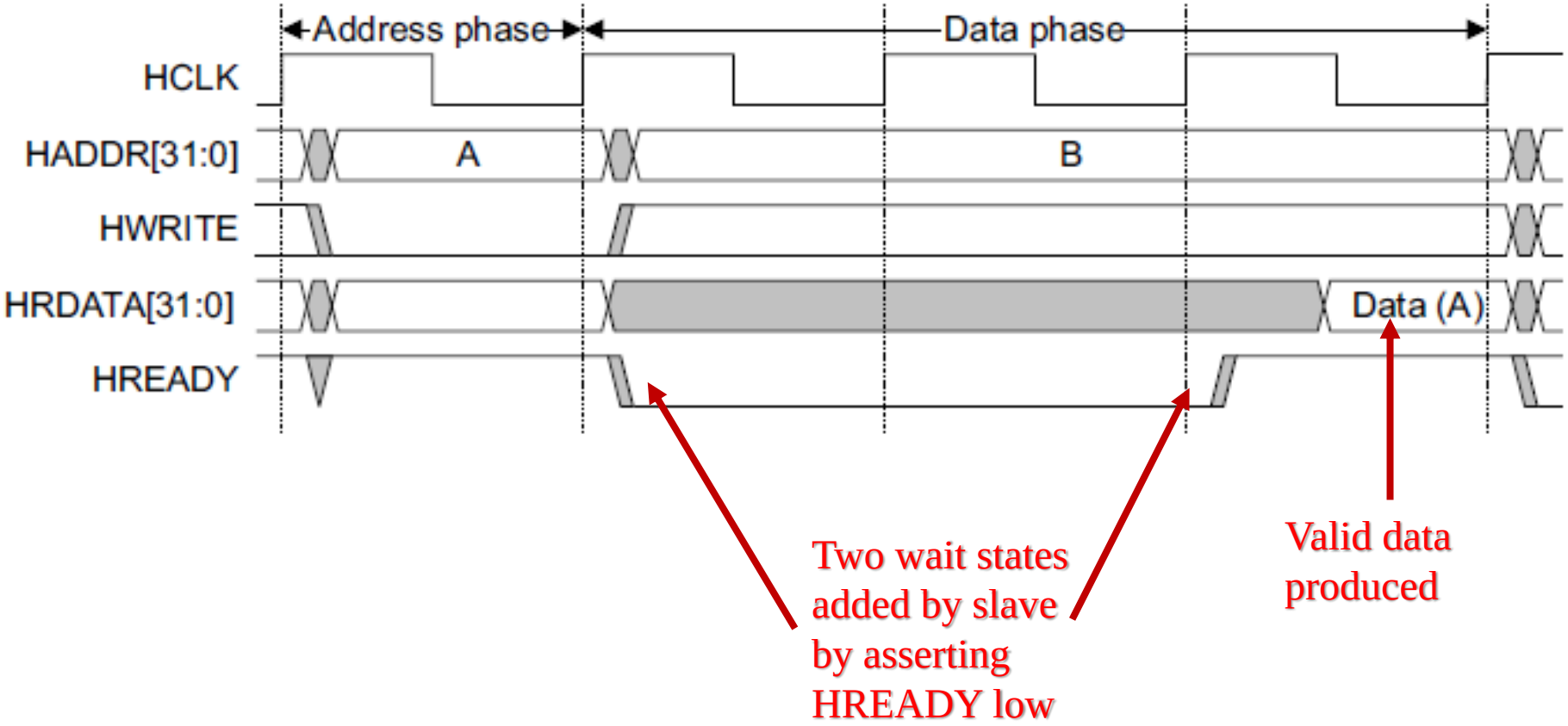
Pipelined Address & Data Transfer

Simple AHB Transfer

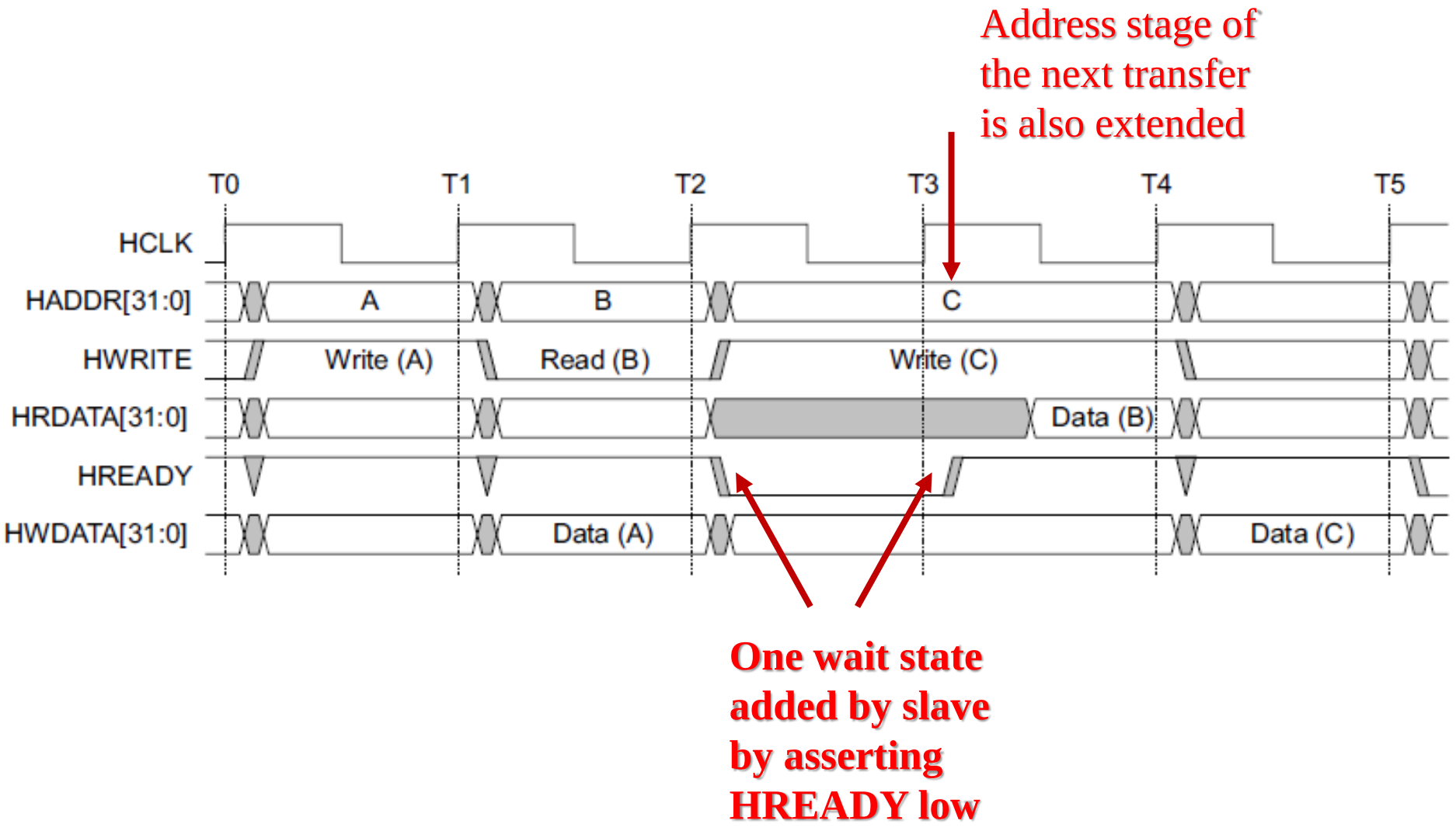


Data transfer with slave wait states

Read – Two Wait States



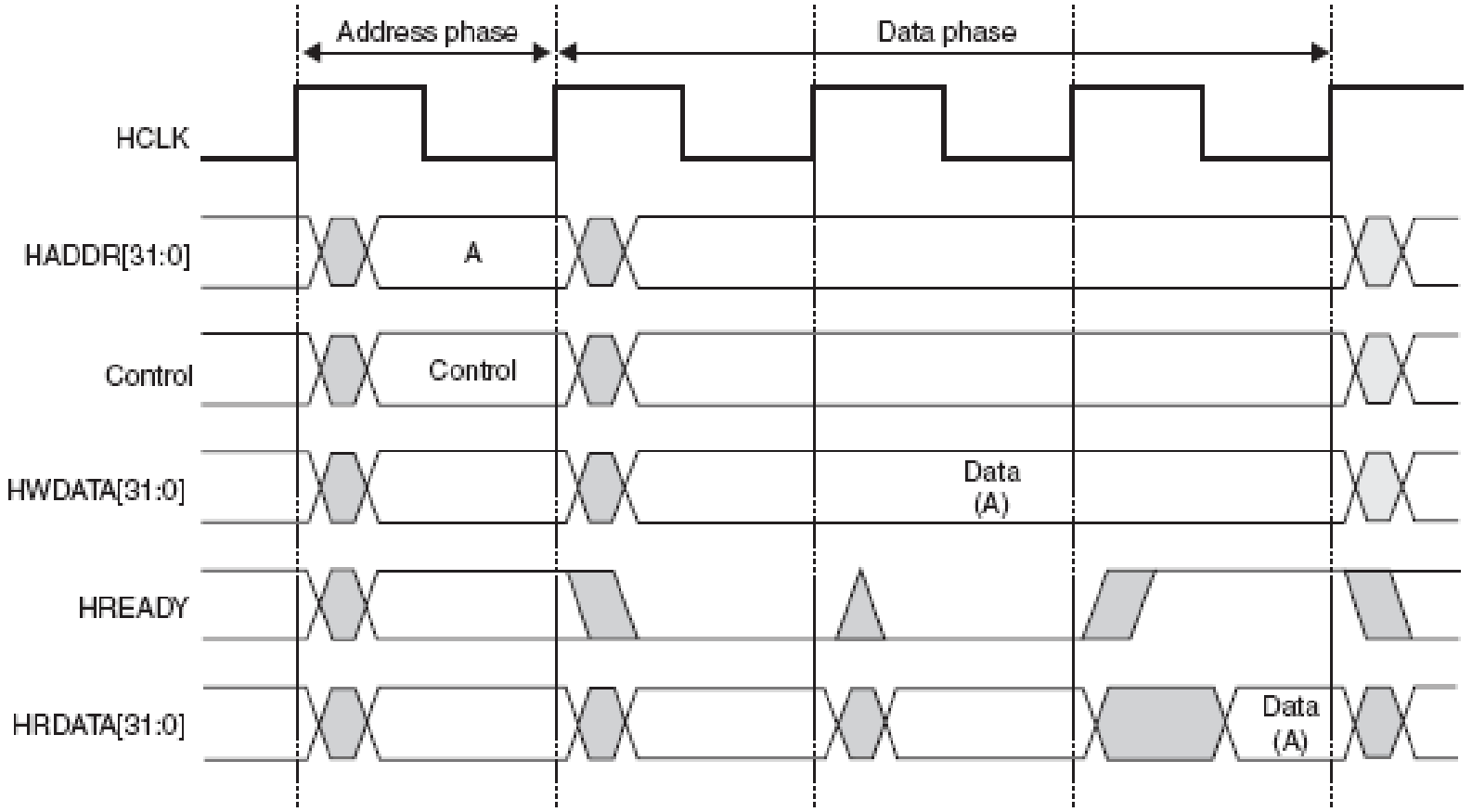
Wait States Extend the Address Phase of Next Transfer



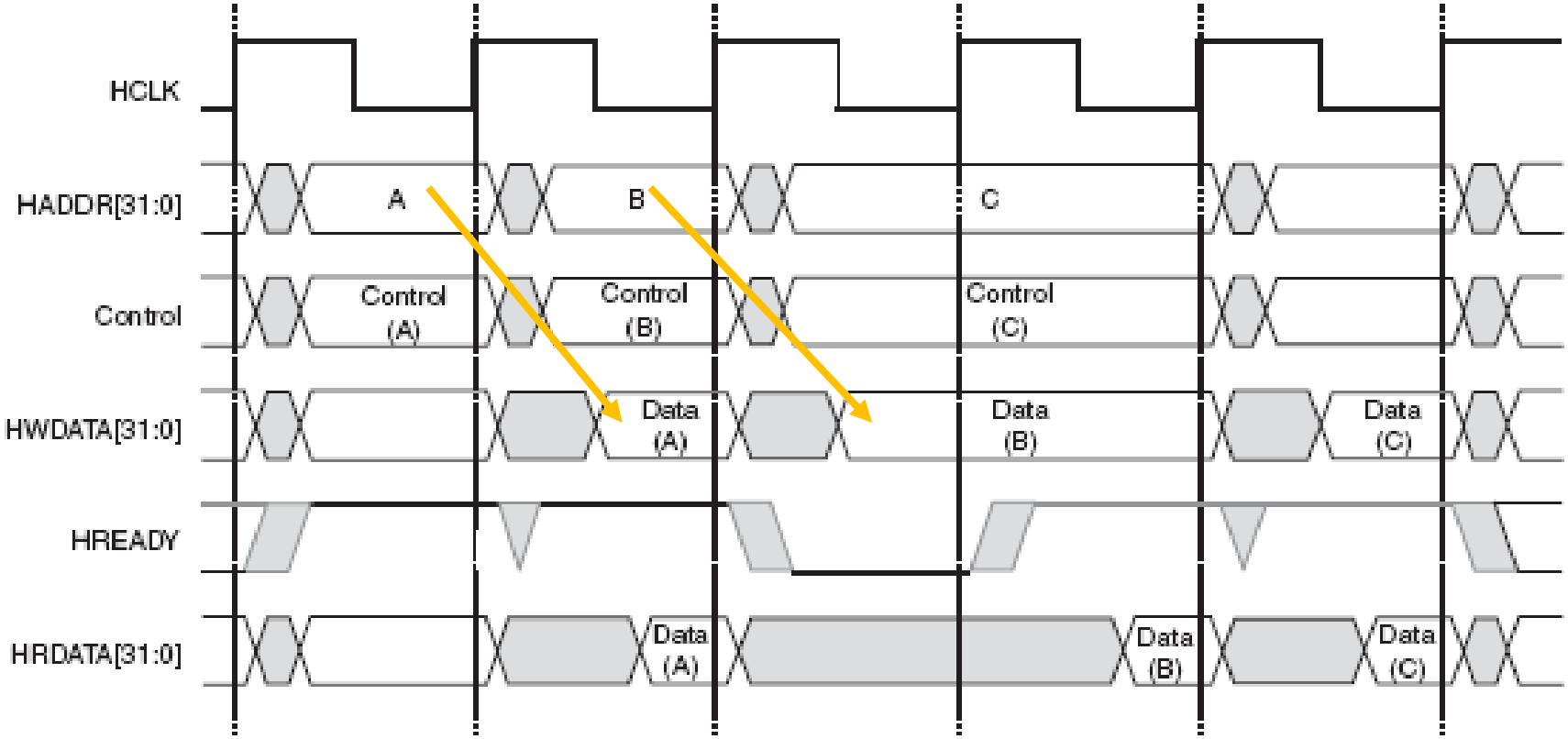
AHB Wait Cycles

Slave may not be ready to service the request

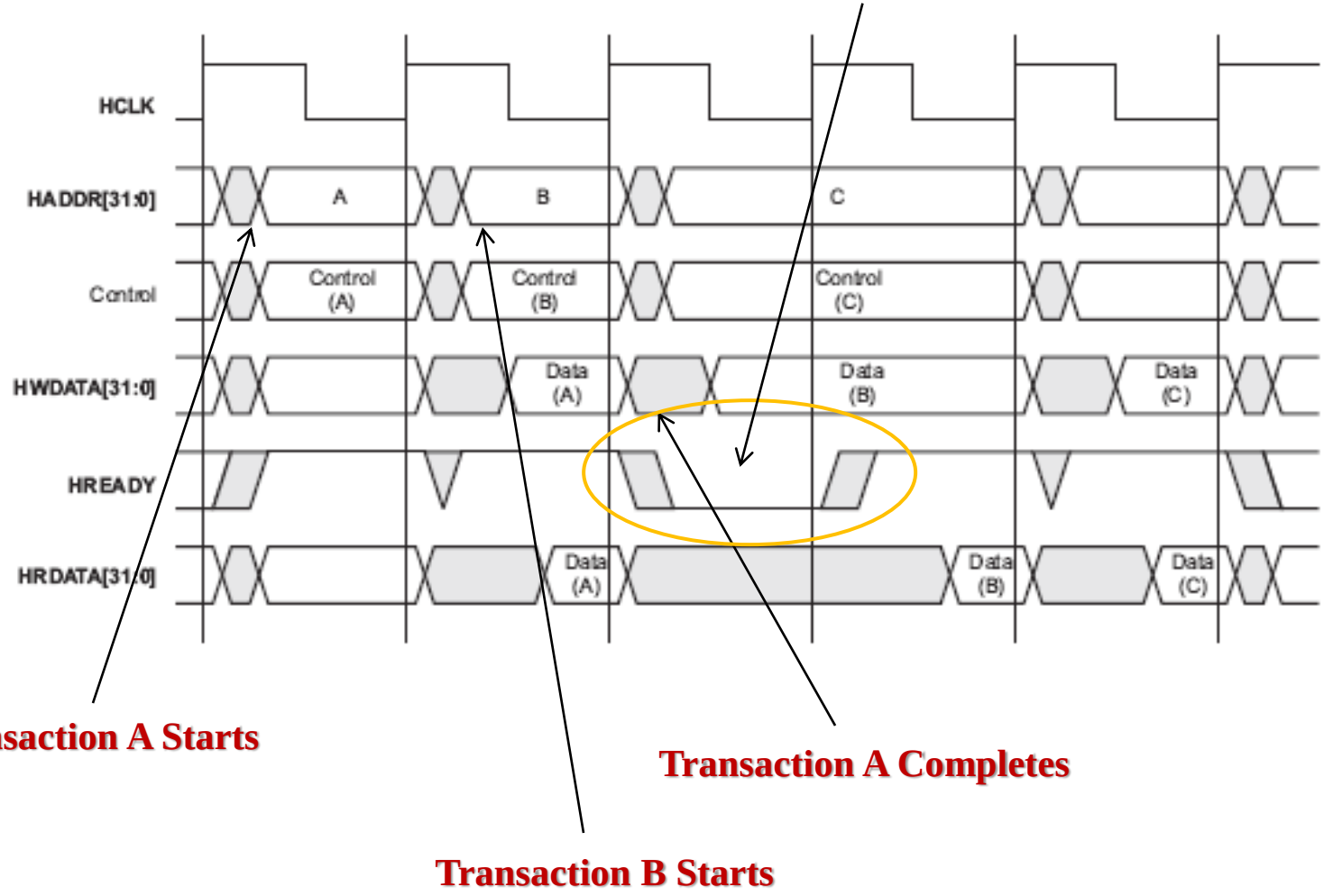
It inserts Wait cycle(s) by using HREADY



AHB Pipelining

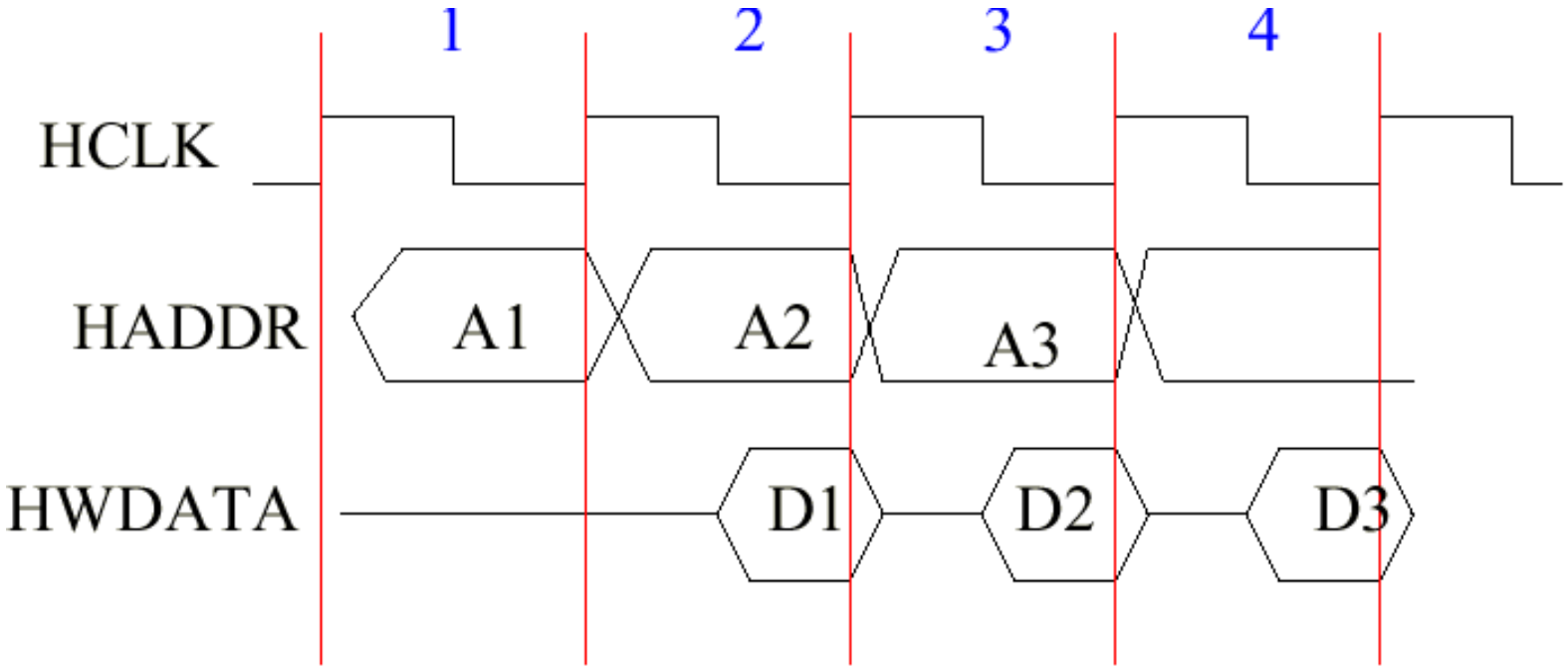


AHB Pipelined Transactions



AHB Pipelining with Burst

Address and data of consecutive transfers are transmitted in the same clock cycle



Transfer Types

Four types (HTRANS[1:0])

IDLE (00)

- No data transfer is required
- Slave must OKAY w/o waiting
- Slave must ignore IDLE

BUSY (01)

- Insert idle cycles in a burst
- Burst will continue afterward
- Address/control reflects next transfer in burst
- Slave must OKAY w/o waiting
- Slave must ignore BUSY

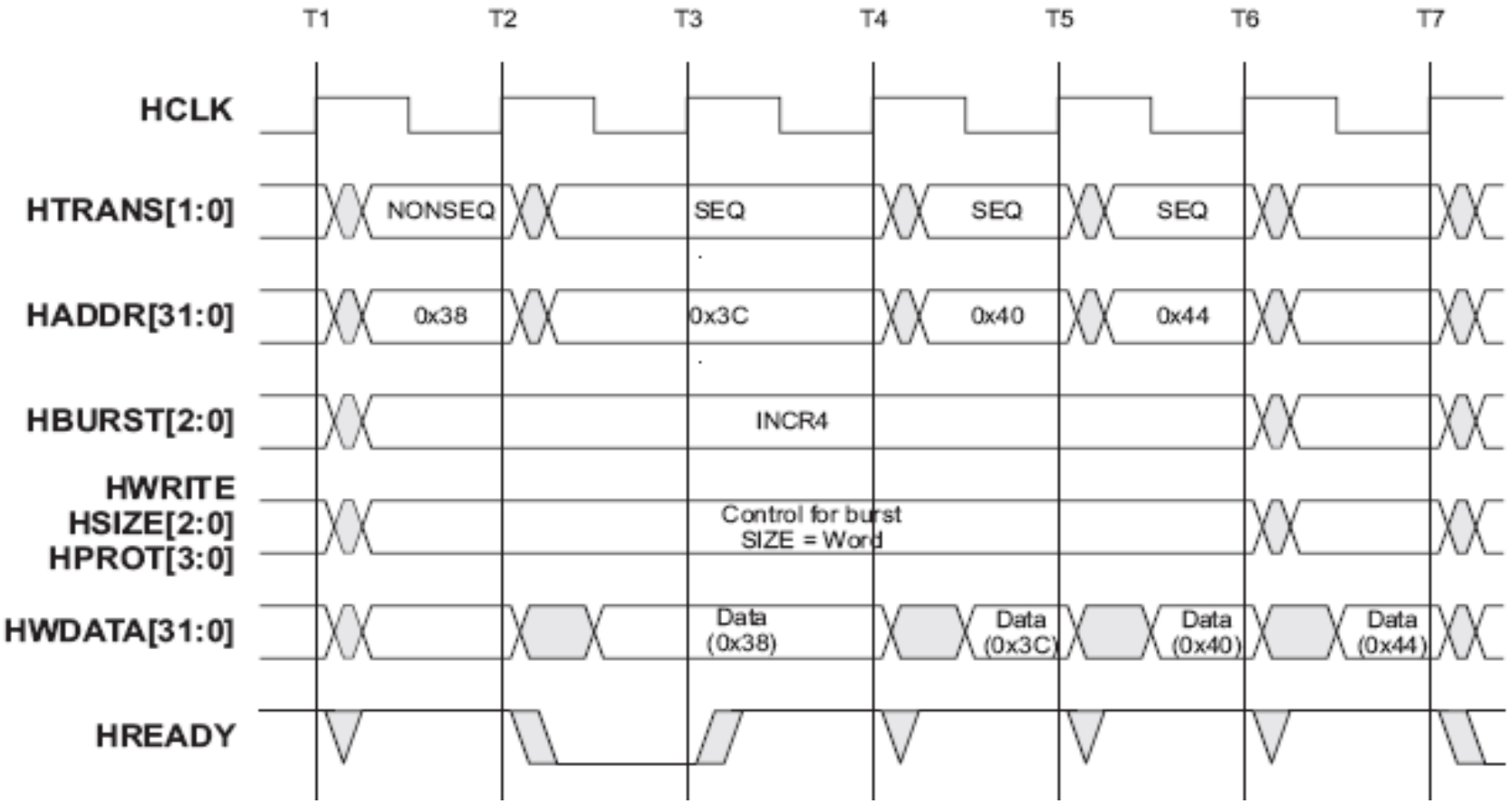
NONSEQ (10)

- Indicates single transfer or first transfer of a burst
- Address/control unrelated to prior transfers

SEQ (11)

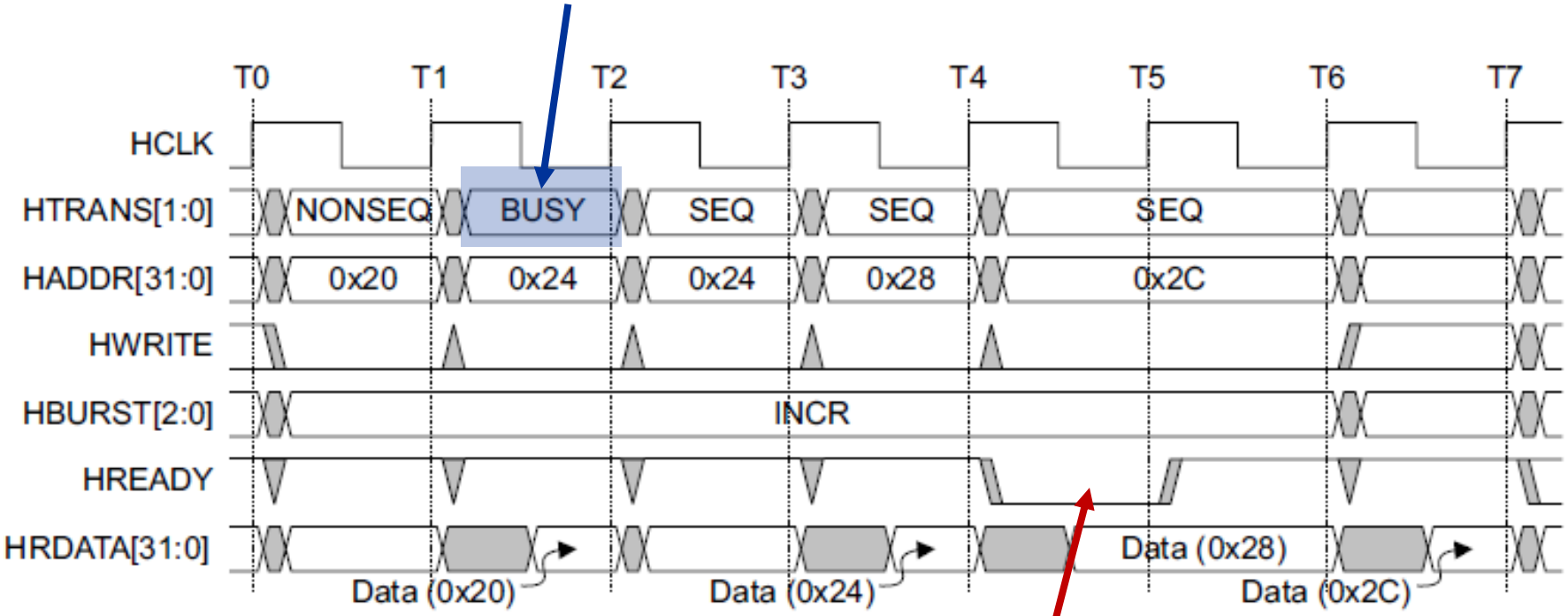
- Remaining transfers in a burst
- Addr = prior addr + transfer size

AHB Pipelined Burst Transfers



Bursts cut down arbitration, handshaking time, improve performance

4-beat Burst – Master Busy, Slave Wait



One wait state added by slave by asserting HREADY low

AHB Burst Types

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

- **Burst of 1, 4, 8, 16 and undef. INCR bursts access sequential locations.**
  e.g. 0x64, 0x68, 0x6C, 0x70 for INCR4, transferring 4 byte data

Wrapping bursts: "wrap around" address if starting address is not aligned to total no. of bytes in transfer. e.g. 0x64, 0x68, 0x6C, 0x60 for WRAP4 that transfer 4 byte data. Another example, 0x34, 0x38, 0x3C, 0x30, …..

- **Burst must not cross 1KB address boundaries.**

AHB Control Signals

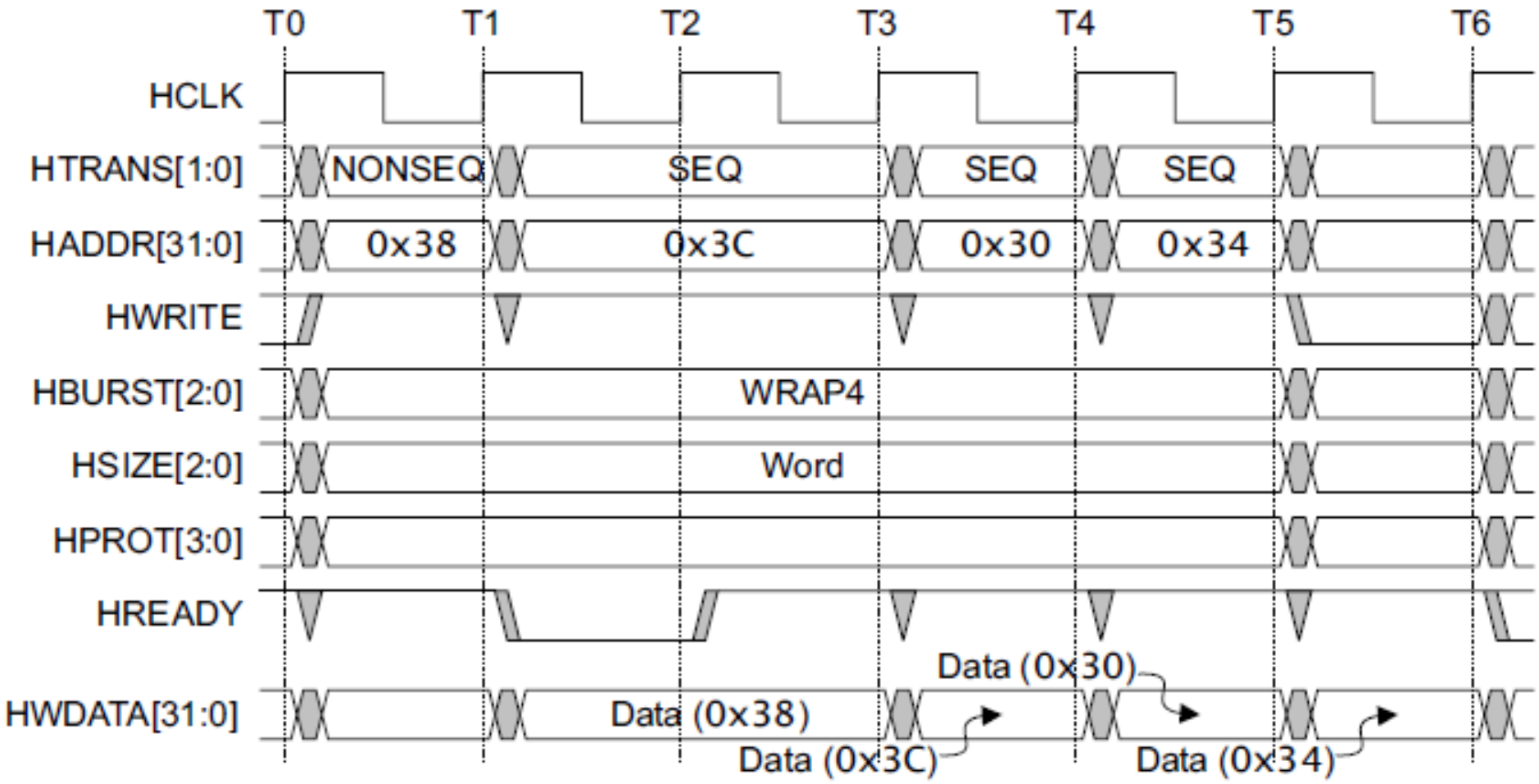
Transfer Direction: HWRITE – write transfer when high, read transfer when low

Transfer Size: HSIZE[2:0] indicates the size of the transfer

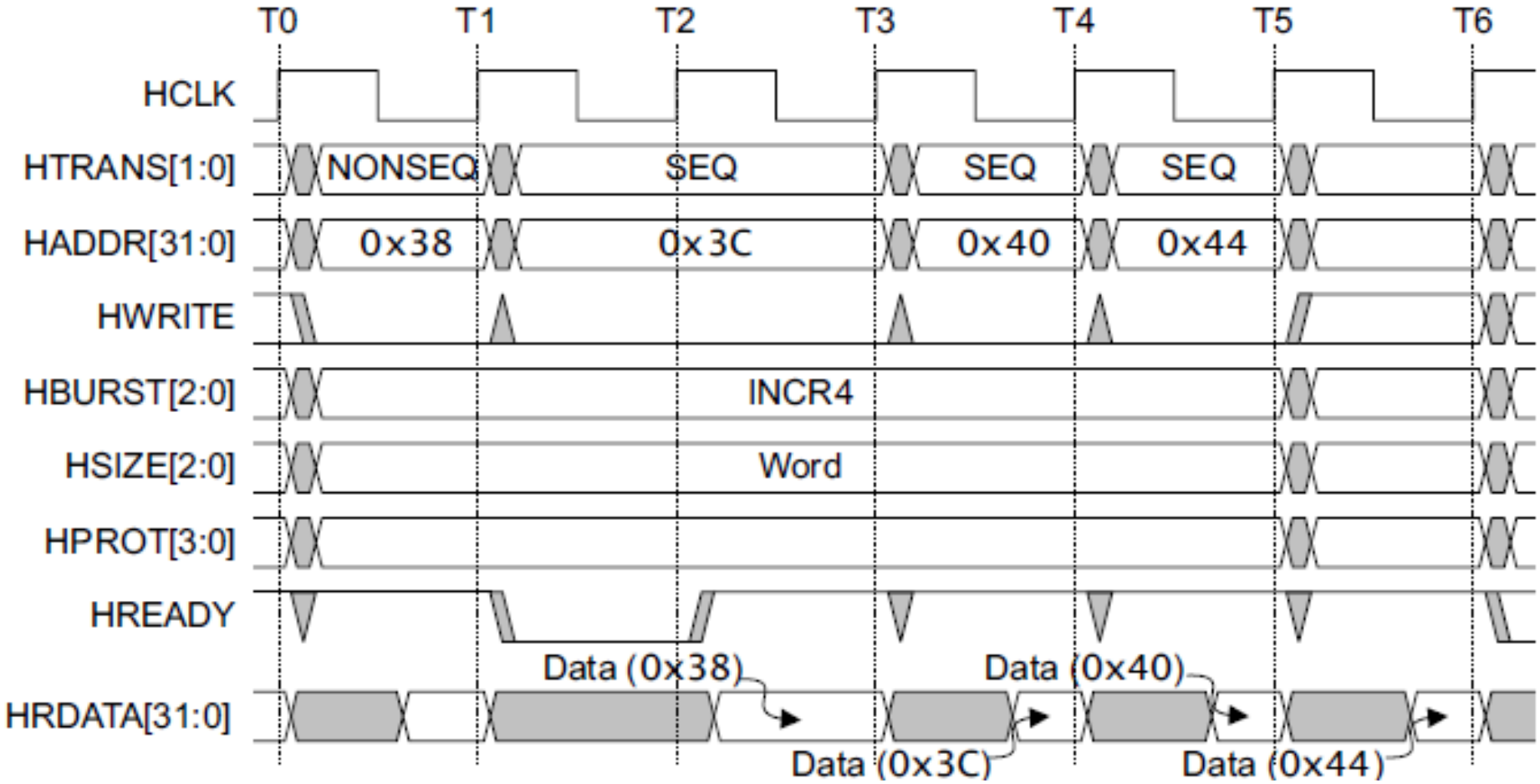
HSIZE + HBURST determine wrapping boundary for WRAP burst.

HSIZE[2]	HSIZE[1]	HSIZE[0]	Size	Description
0	0	0	8 bits	Byte
0	0	1	16 bits	Halfword
0	1	0	32 bits	Word
0	1	1	64 bits	-
1	0	0	128 bits	4-word line
1	0	1	256 bits	8-word line
1	1	0	512 bits	-
1	1	1	1024 bits	-

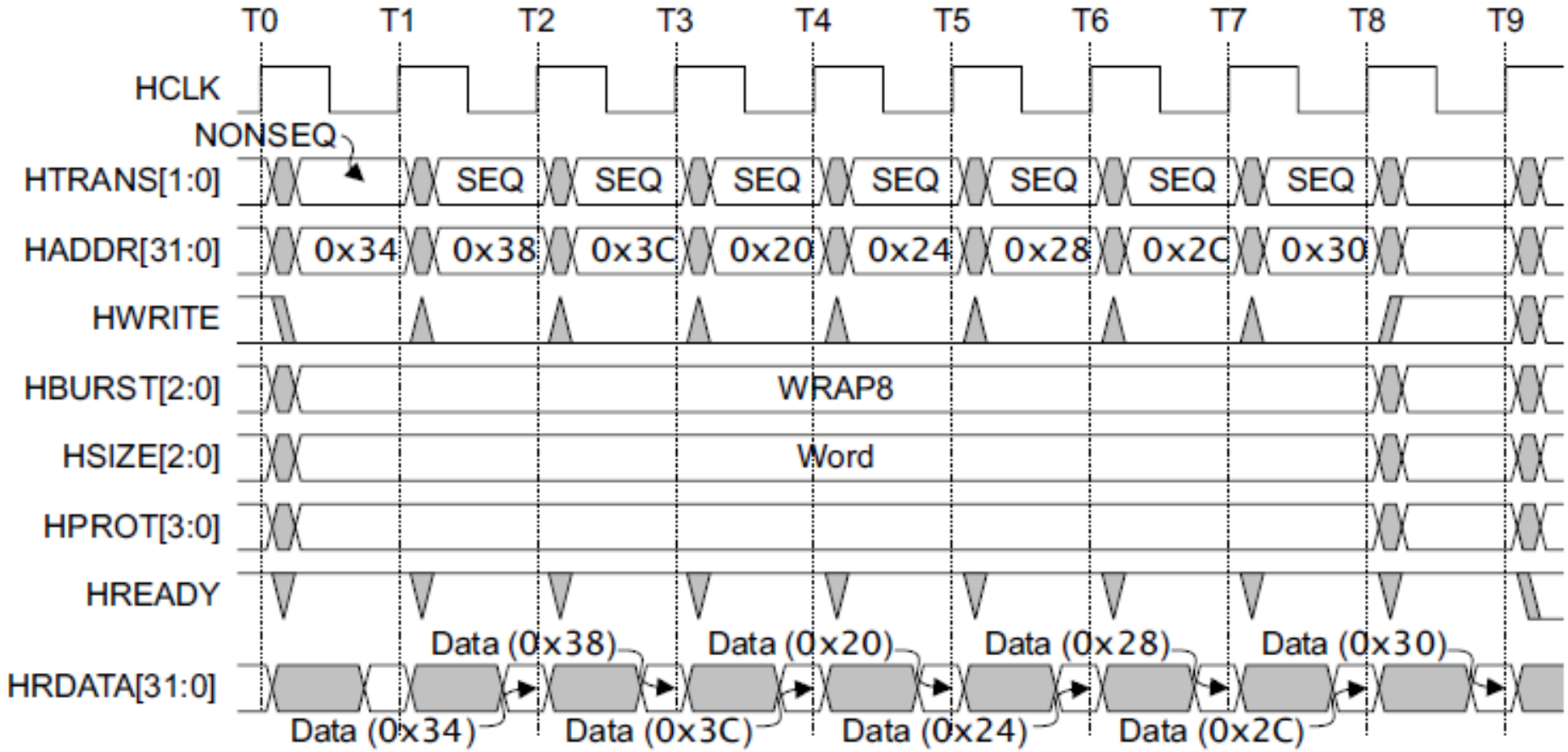
WRAP4: 4 Beat Wrapping Burst



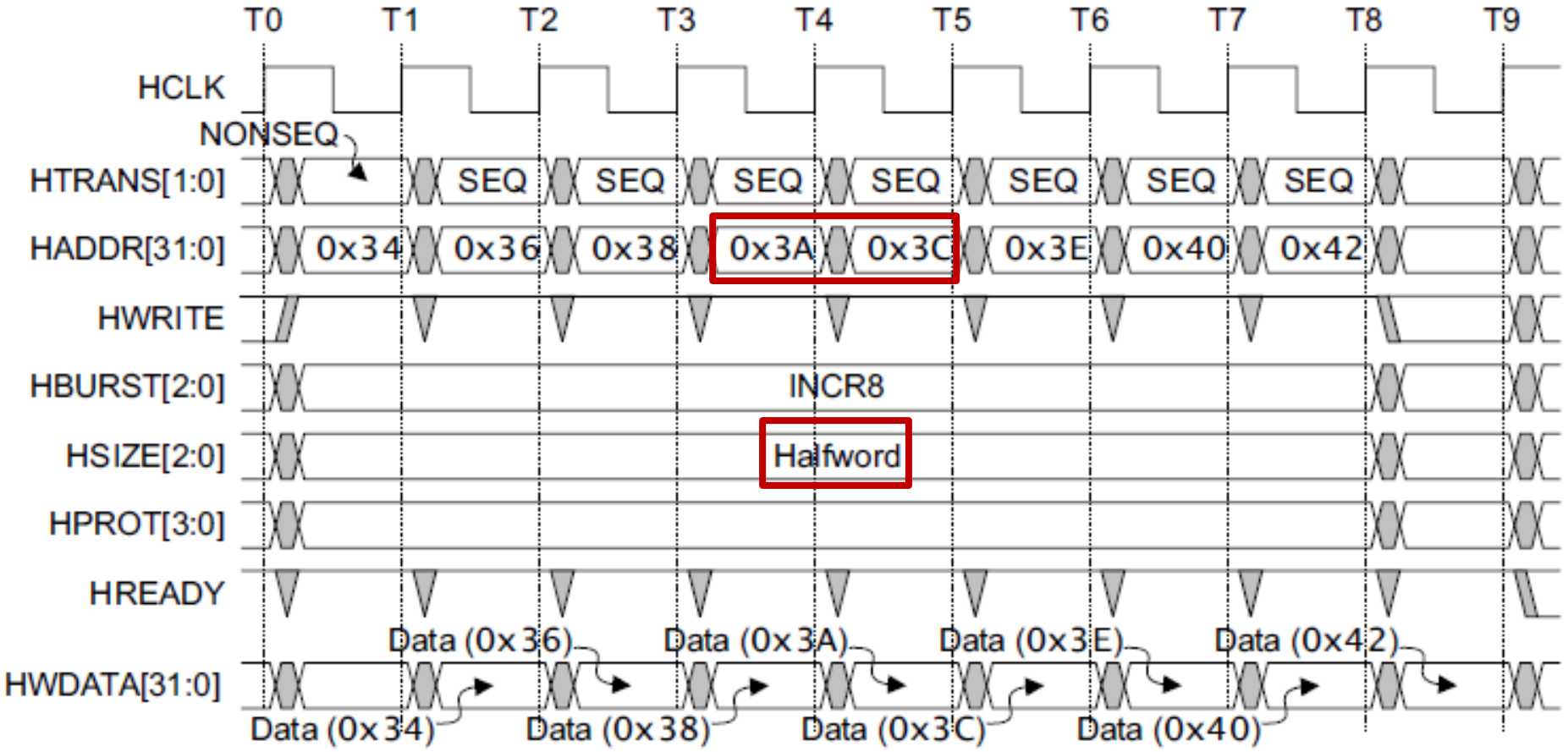
INCR4: 4 Beat Incrementing Burst



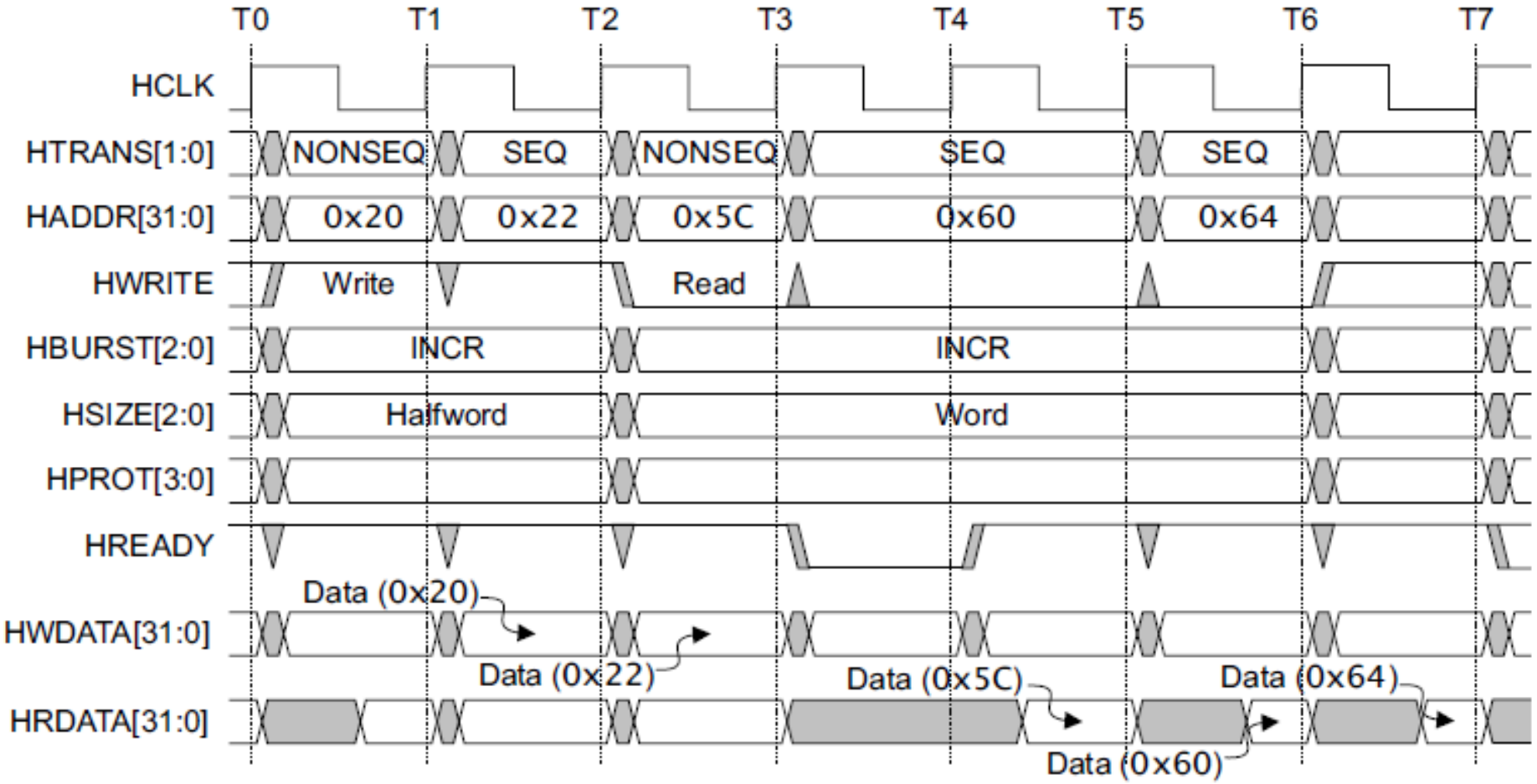
WRAP8: 8-Beat Wrapping Burst



INCR8: 8-Beat Incrementing Burst for Half-word Transfers



INCR: Undefined Incrementing Burst



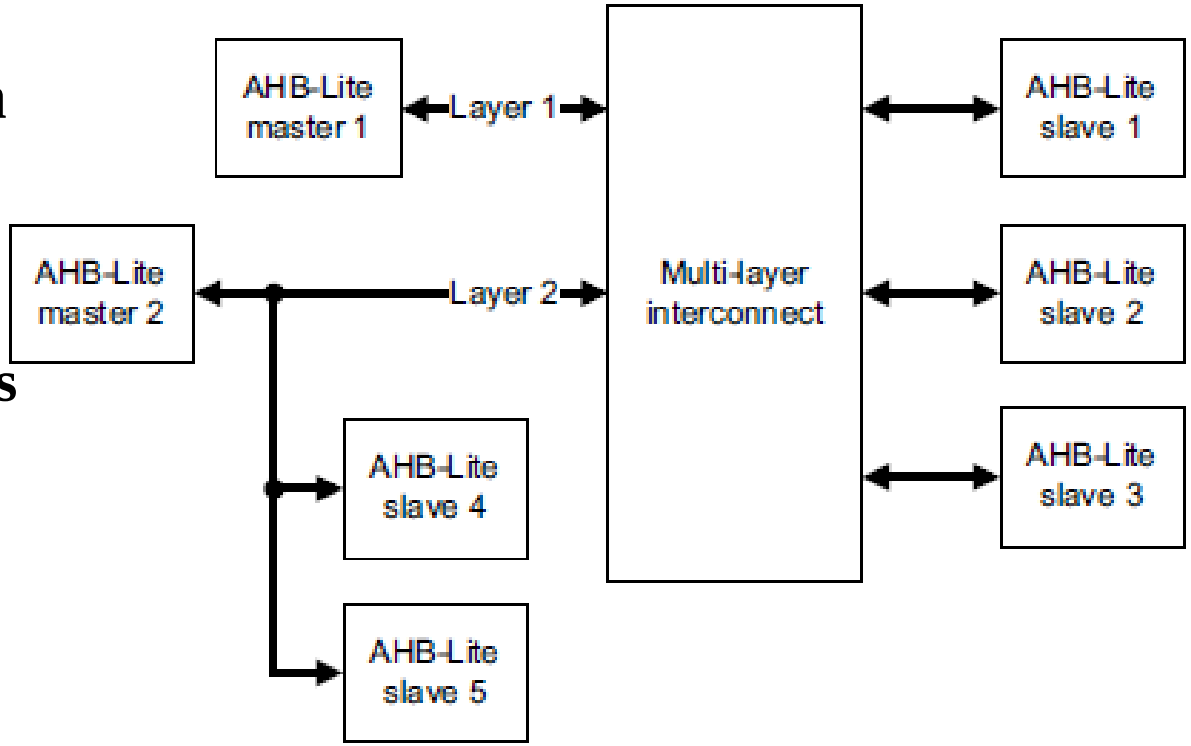
Multi-master AHB Requires a Multi-layer Interconnect

Multi-master operation

- **Must isolate masters**
- **Each master assigned to layer**
- **Interconnect arbitrates slave accesses**

Full crossbar switch often not needed

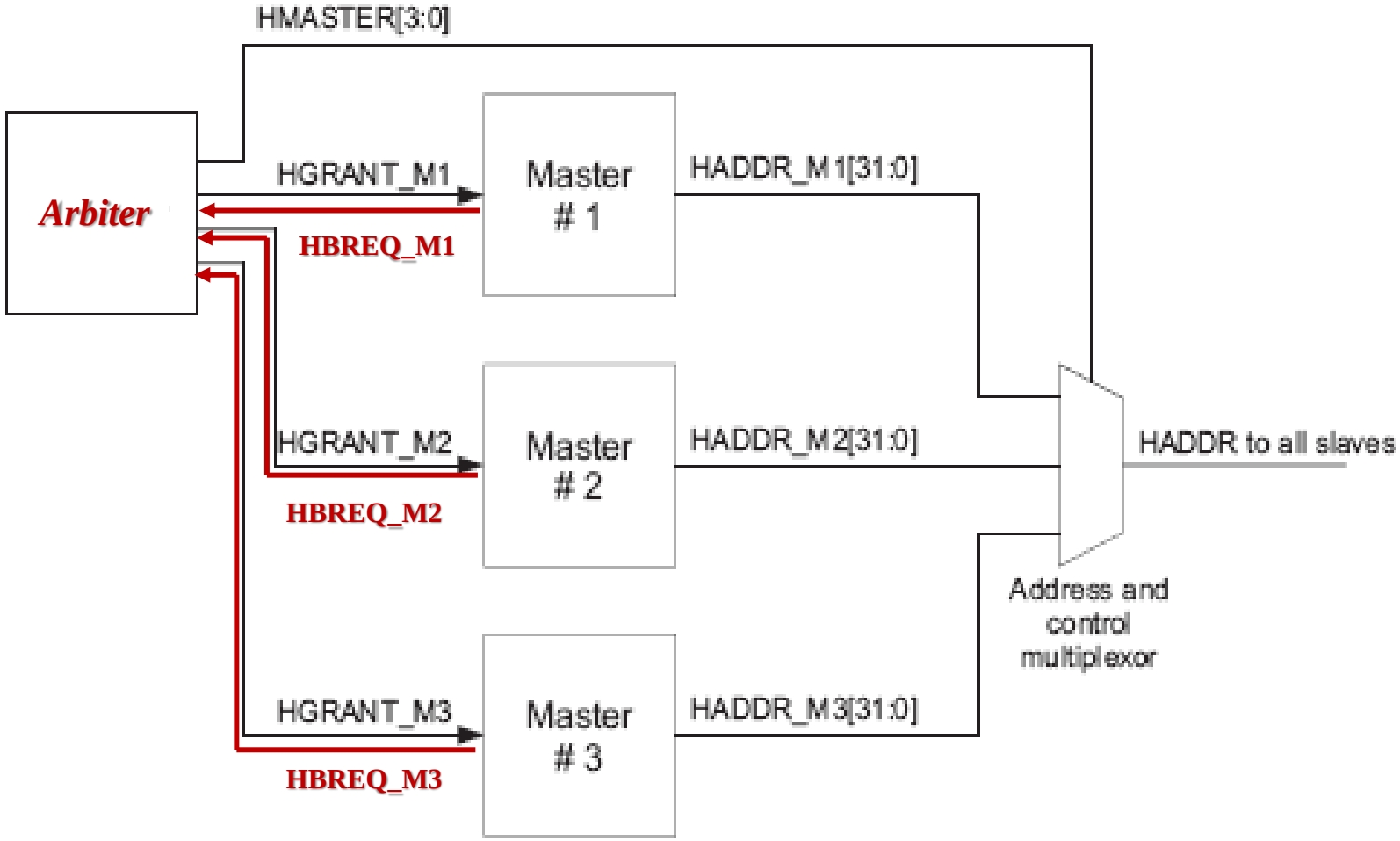
- **Slaves 1, 2, 3 are shared**
- **Slaves 4, 5 are local to Master 2**



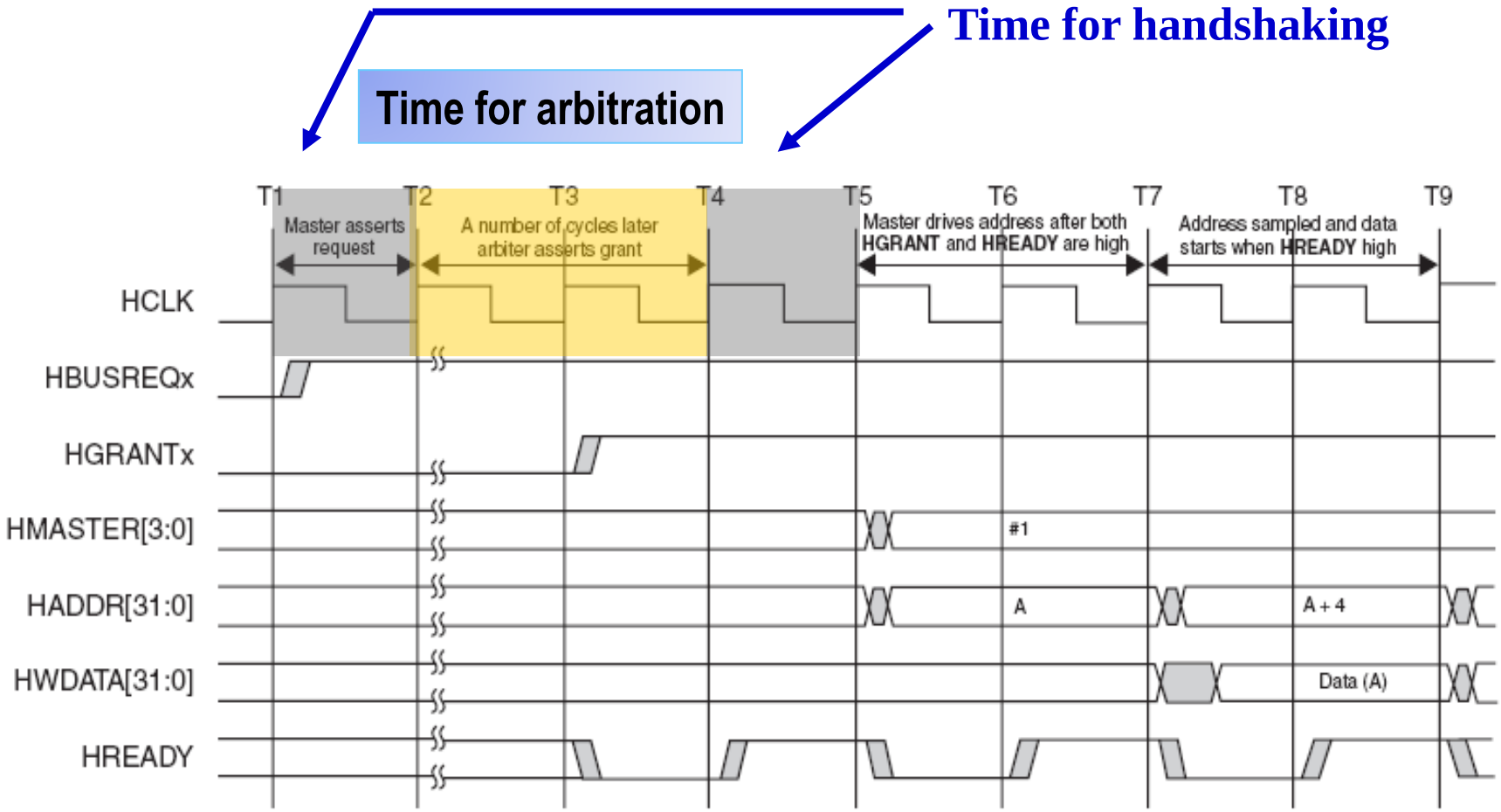
AMBA Bus Arbitration

- Several masters and slaves are connected to AHB.
- An arbiter decides which master will transfer data.
- Data is transferred from a master to a slave in bursts.
- Any burst involves read/write of a sequence of addresses.
- The slave to service a burst is chosen depending on the addresses (decided by a decoder).
- AHB is connected to APB via a bus bridge.

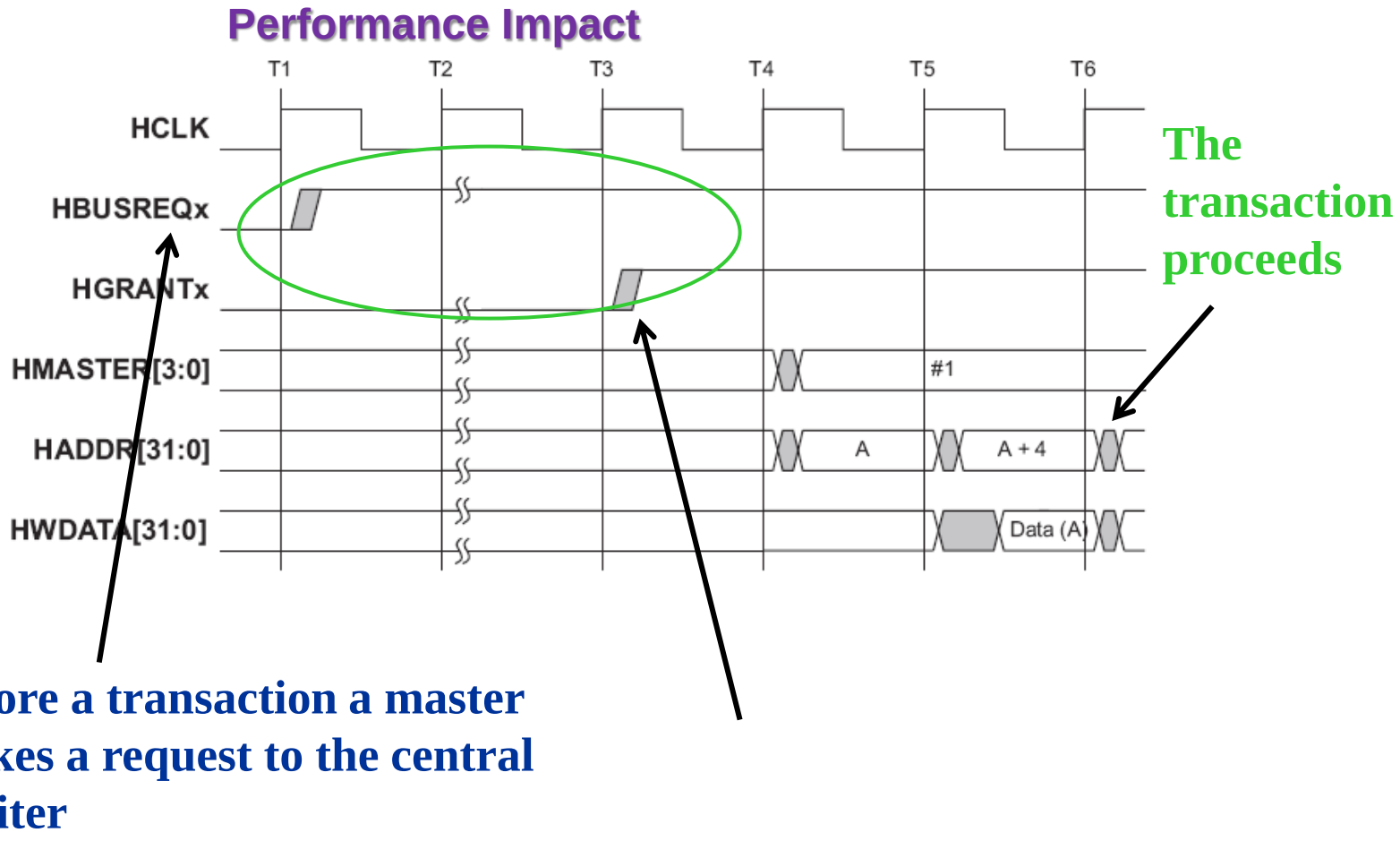
AHB Arbitration



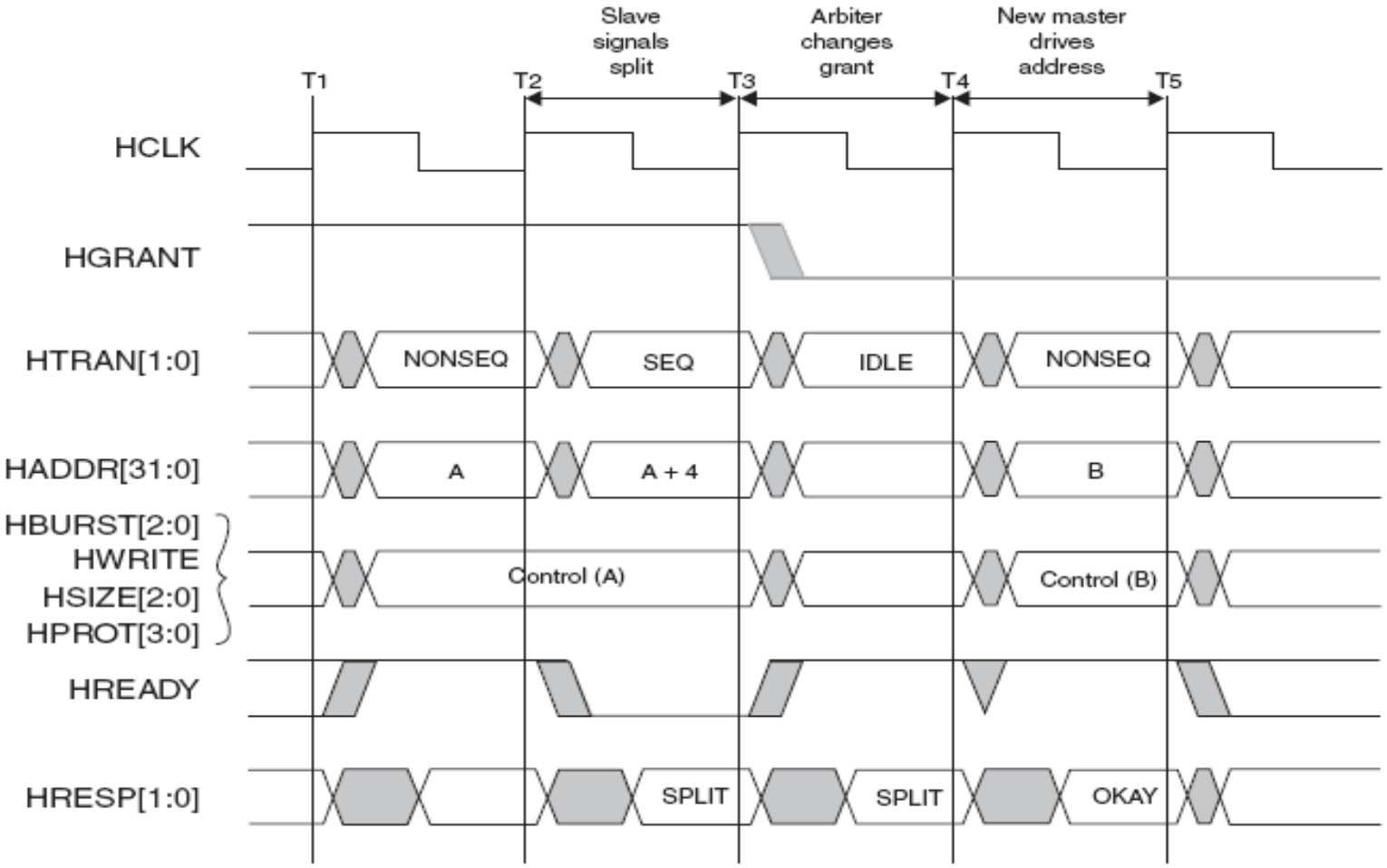
Arbitration Cost



Request Grant Protocol



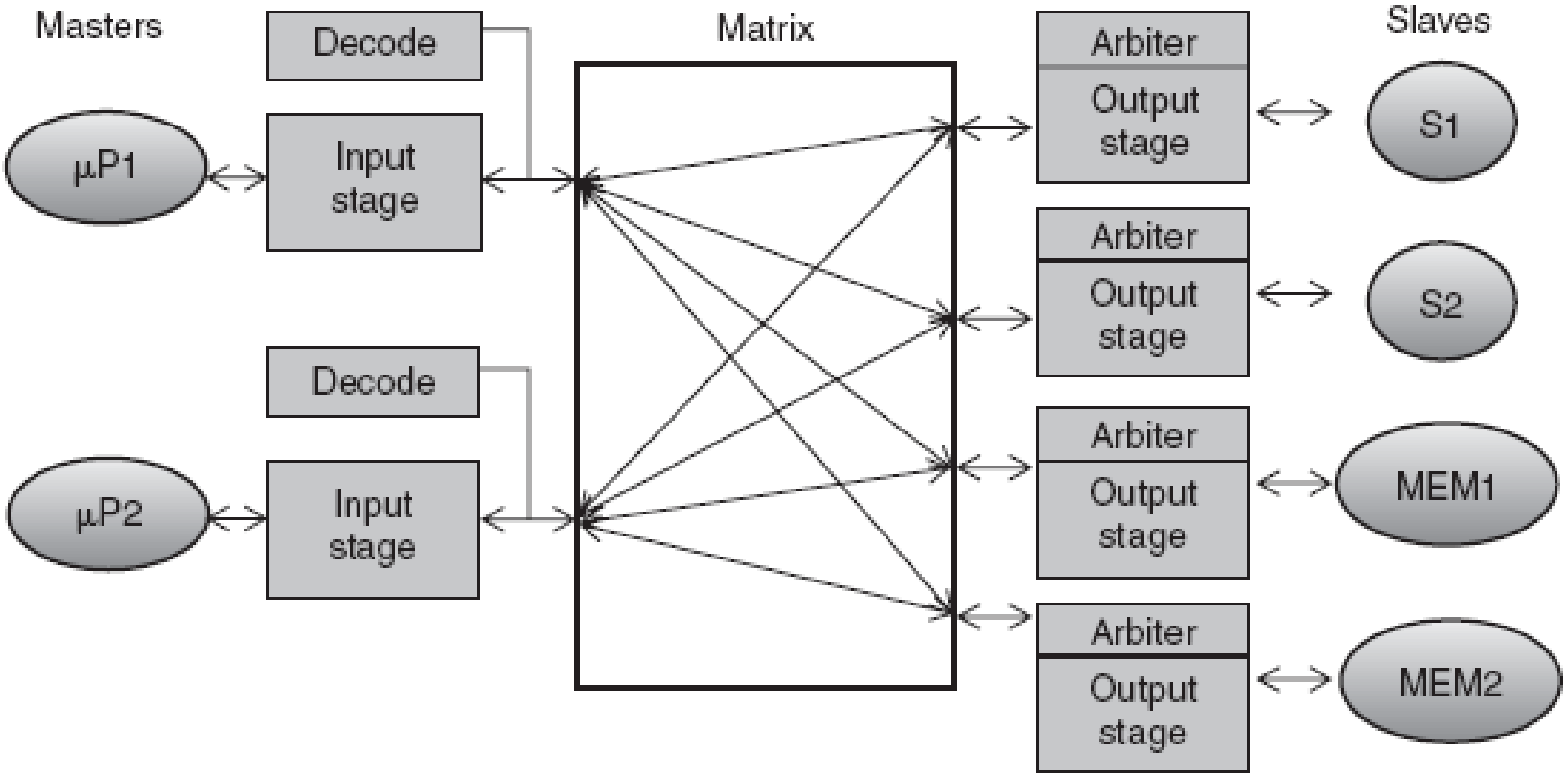
AHB Split Transfers



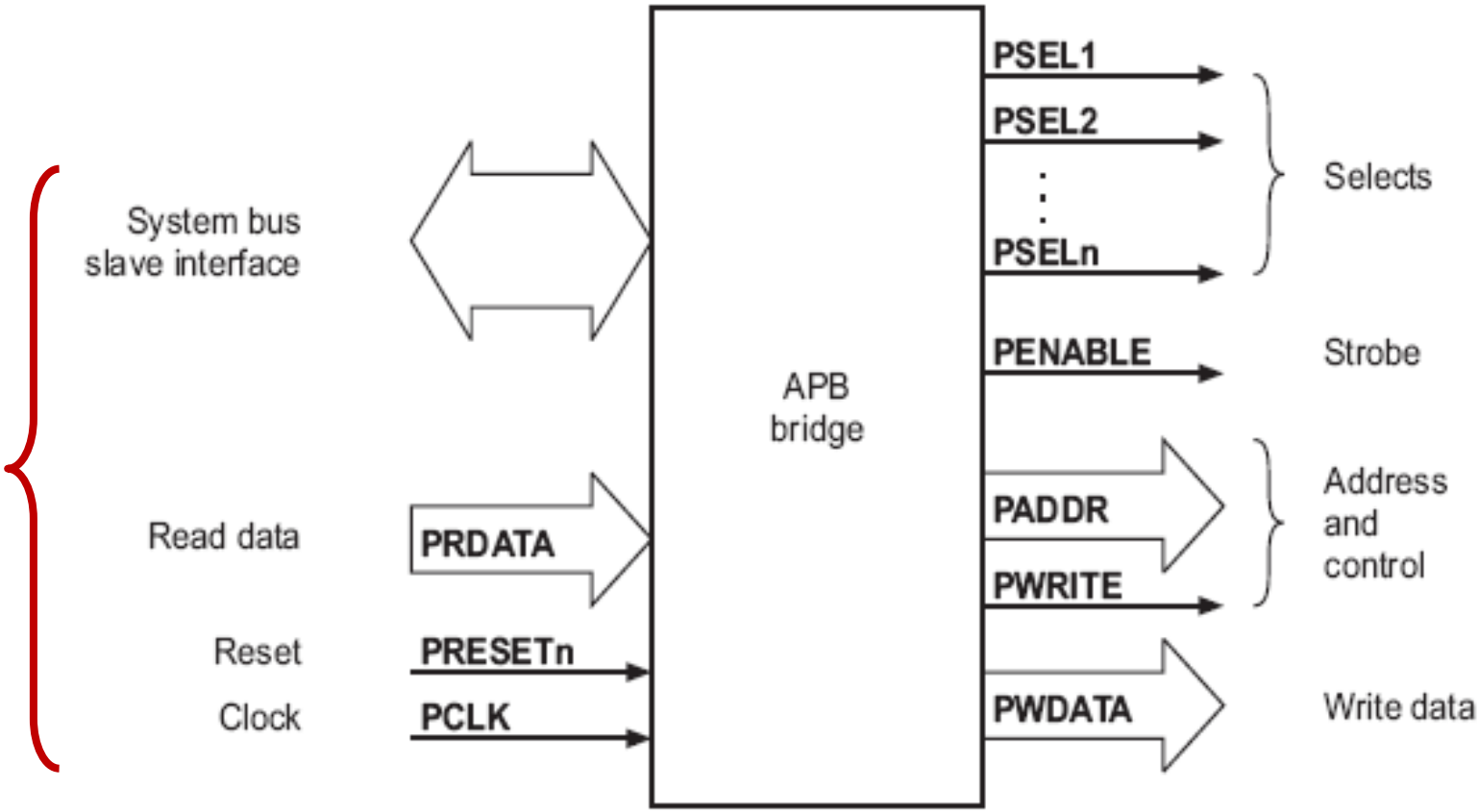
- Improves bus utilization

AHB Bus Matrix

AHB can be employed and implemented as a bus matrix.



AHB-APB Bridge



High performance

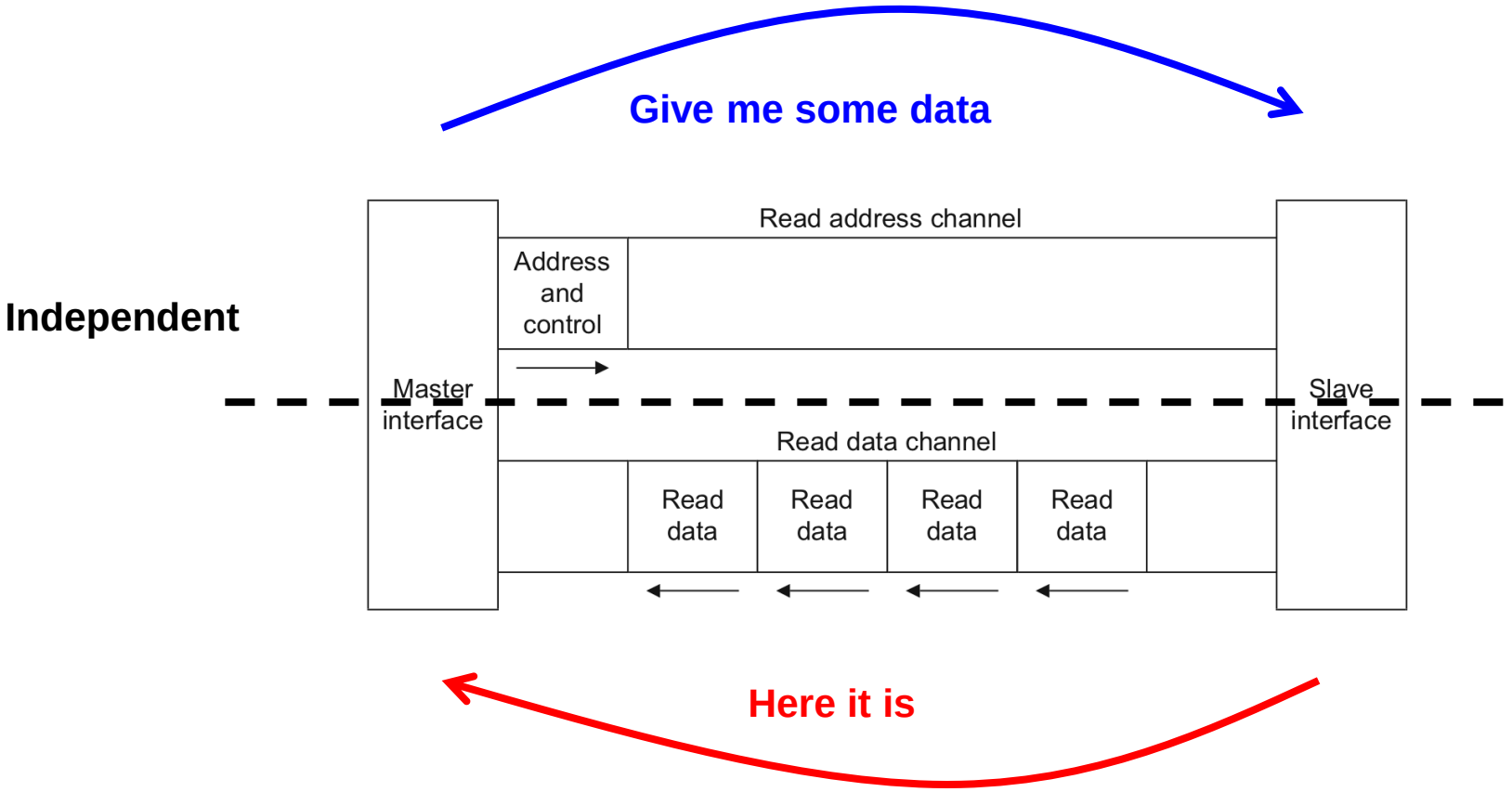
Low power (& performance)

AMBA 3.0

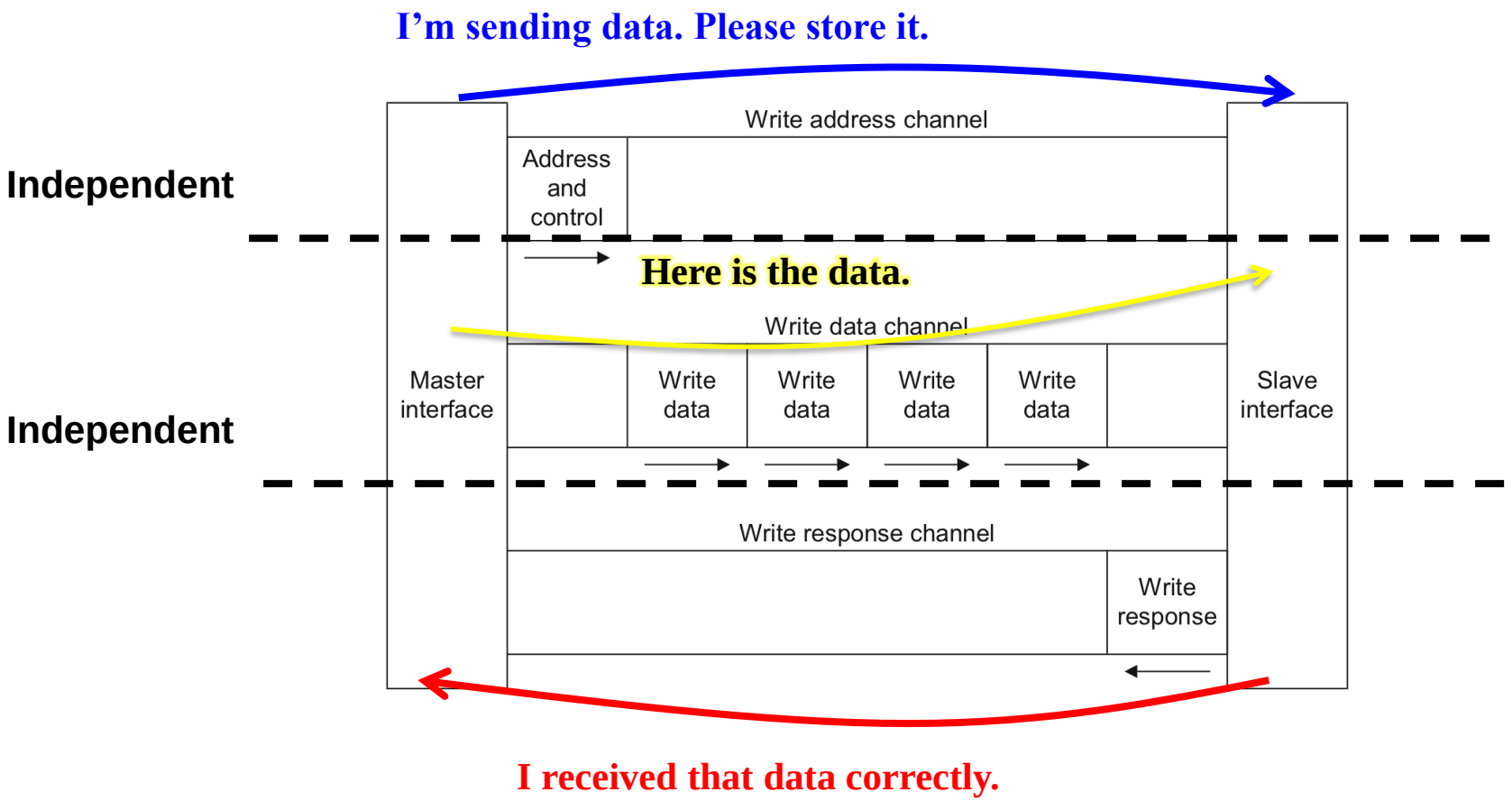
Introduces AXI

- Support for separate read address, write address, read data, write data, write response channels
- Out of order transaction completion
- Fixed mode burst support
  - Useful for I/O peripherals
- Advanced system cache support
  - Specify if transaction is cacheable and buffer-able
  - Specify attributes such as write-back/write-through
- Enhanced protection support
  - Secure/non-secure transaction specification
- Exclusive access (for semaphore operations)
- Register slice support for high frequency operation

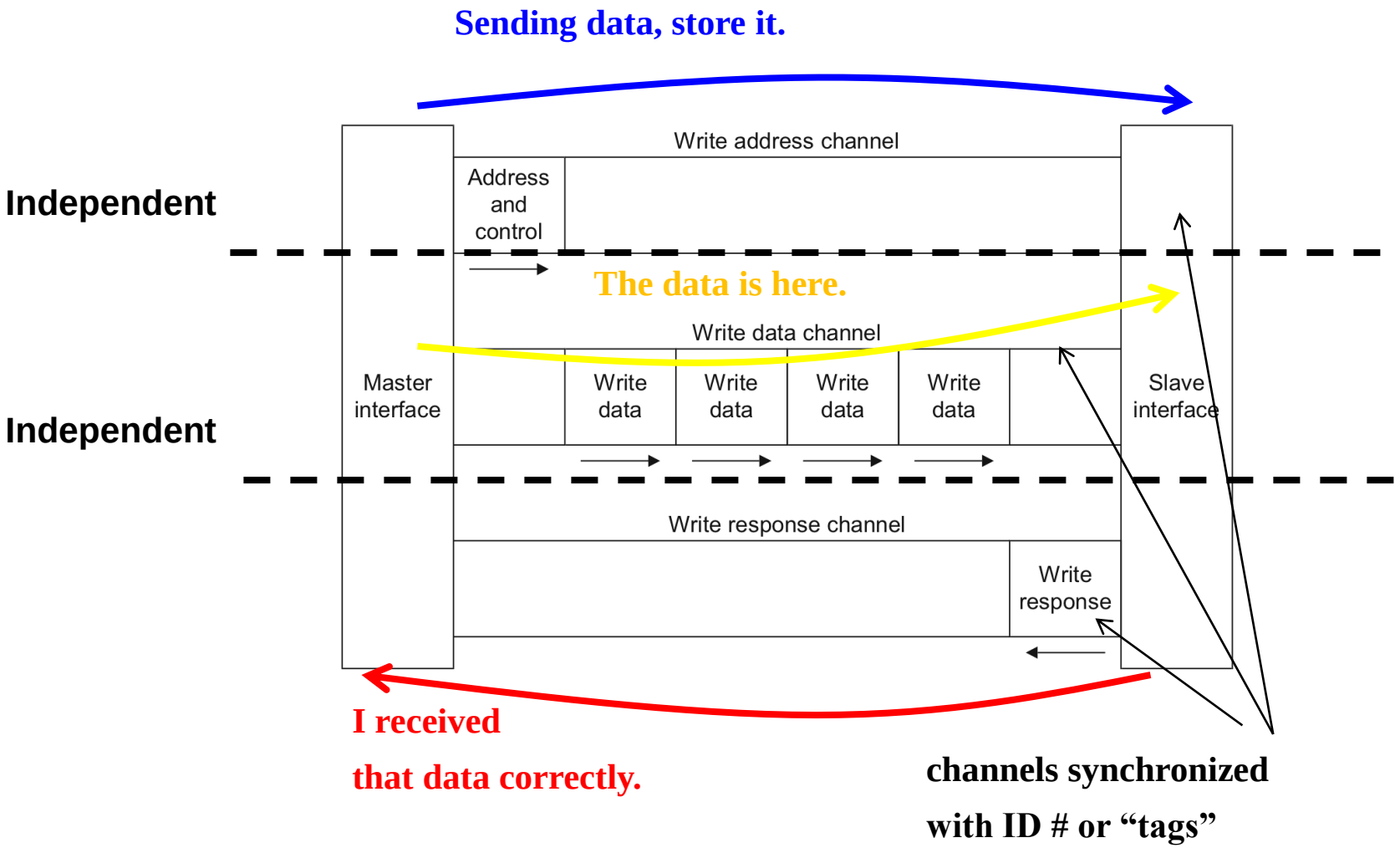
AMBA AXI Read Channels



AMBA AXI Write Channels

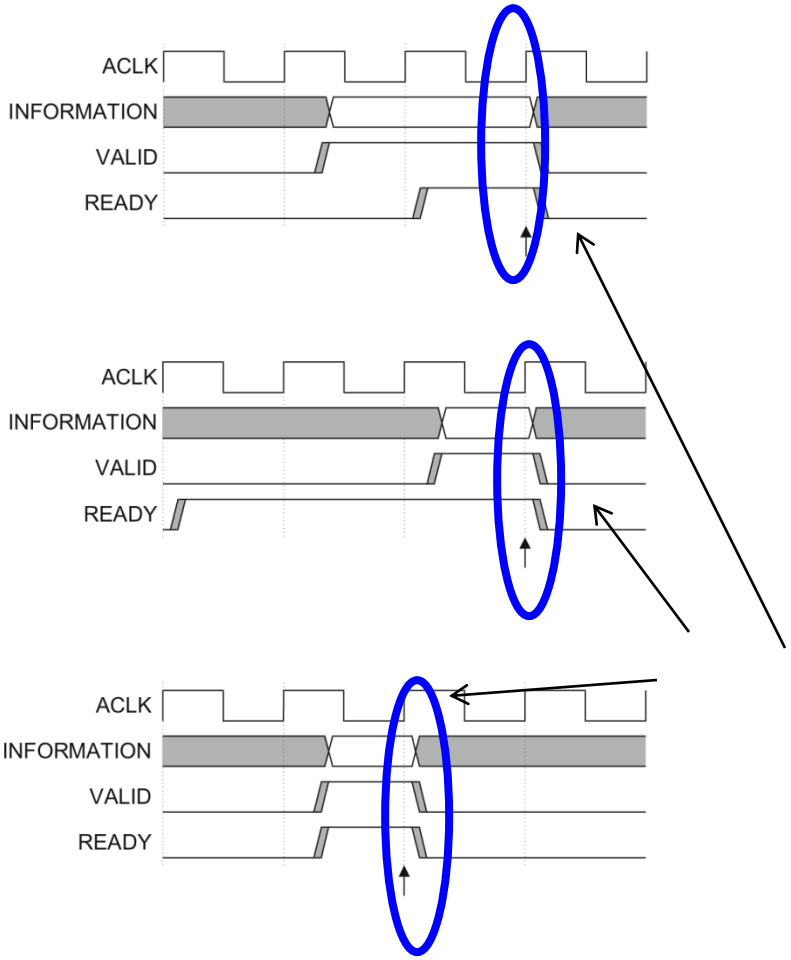


AMBA AXI Write Channels

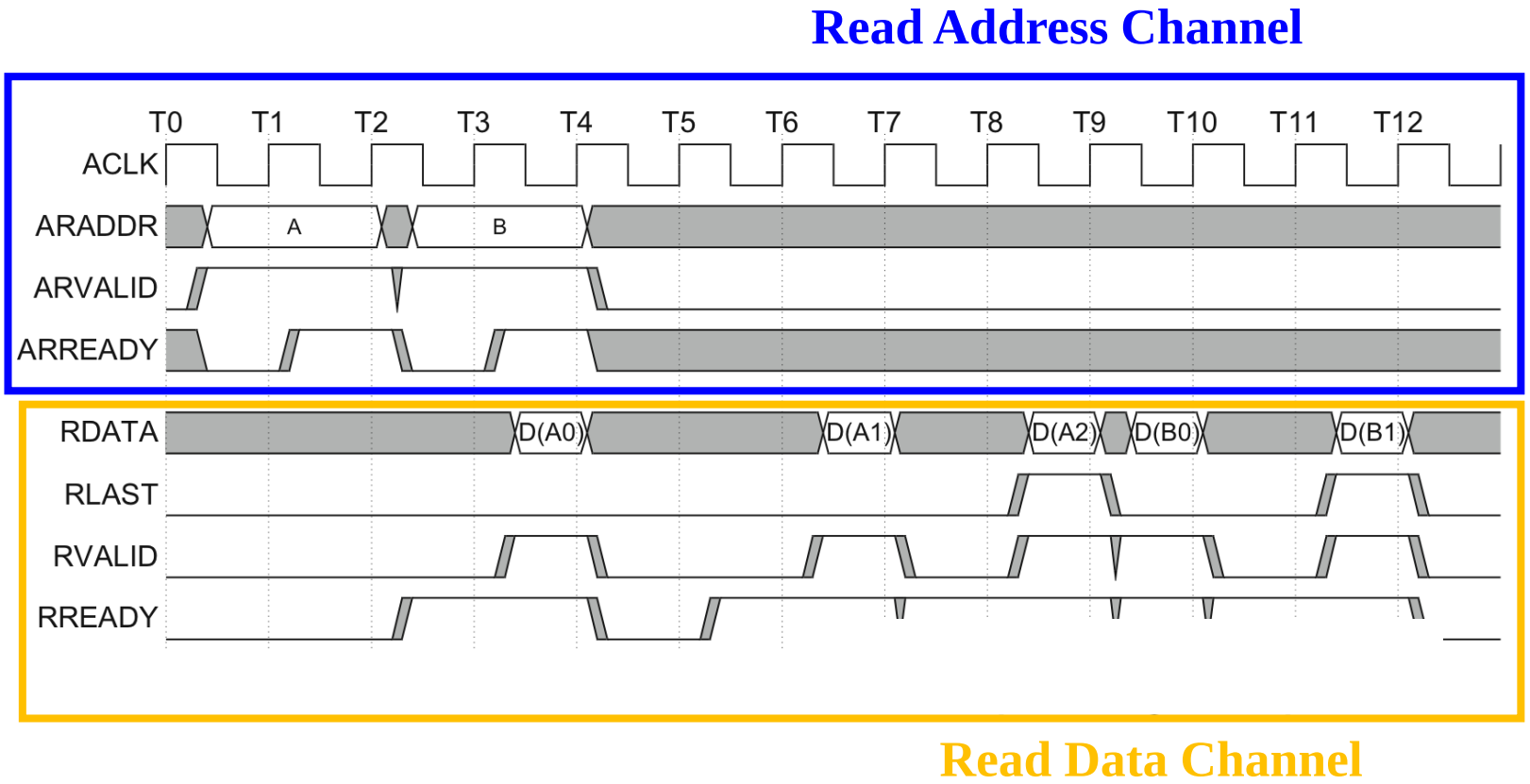


AMBA AXI Flow-Control

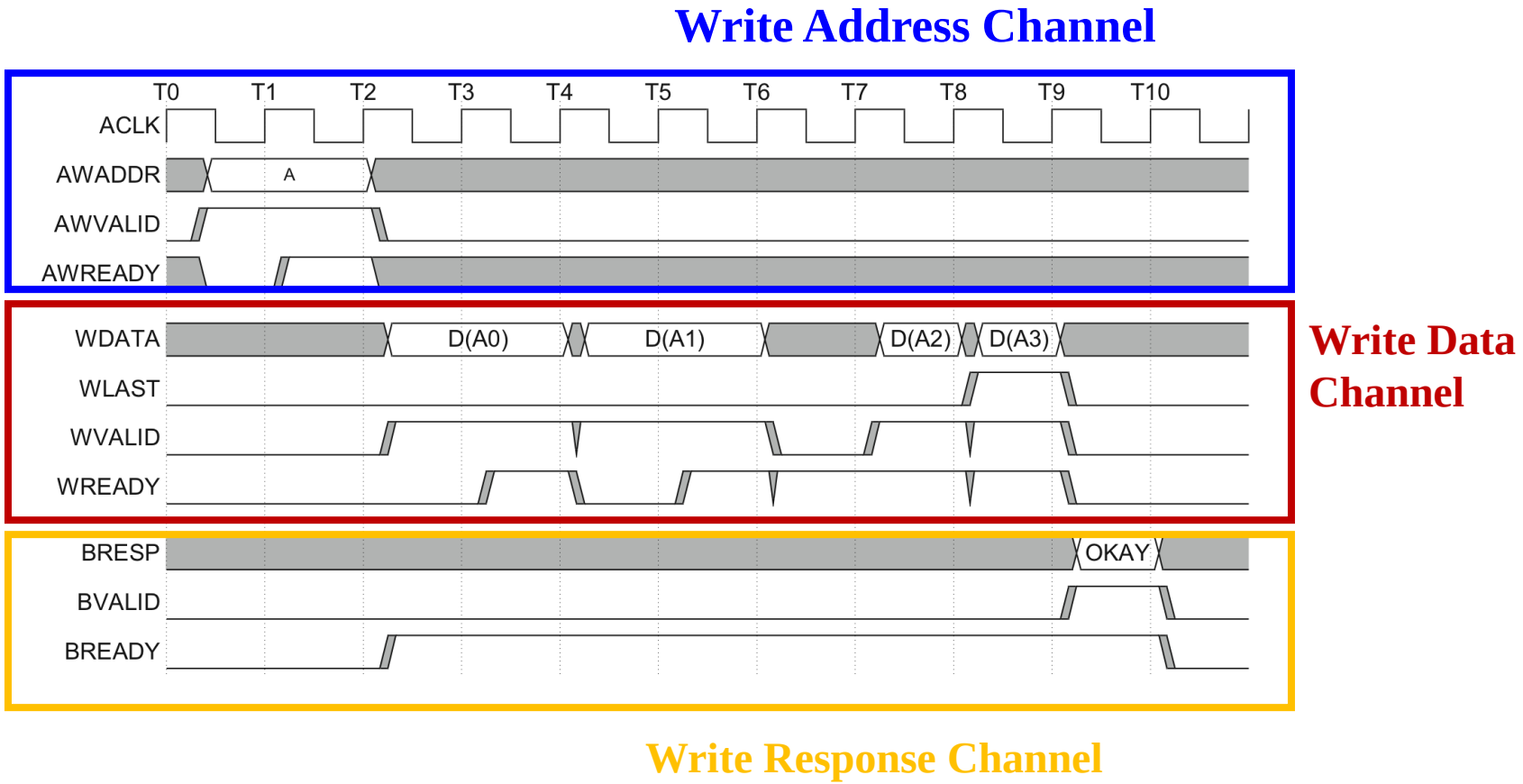
- Information moves only when:
  - Source is Valid, and
  - Destination is Ready
- On each channel the master or slave can limit the flow
- Very flexible



AMBA AXI Read



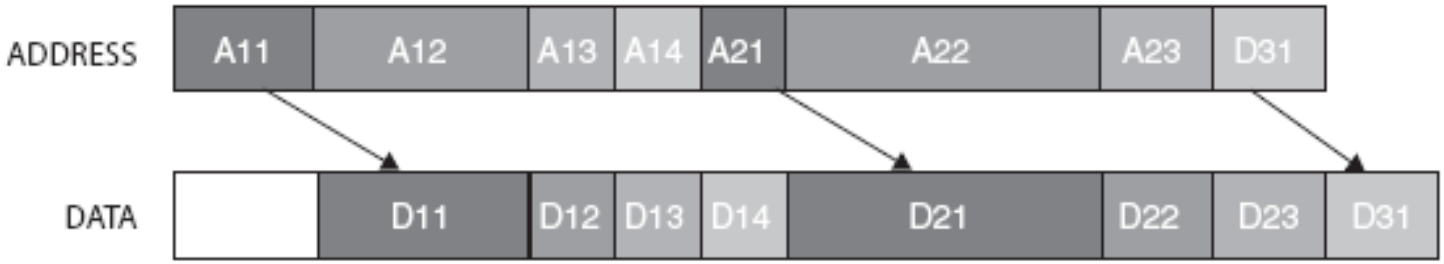
AMBA AXI Write



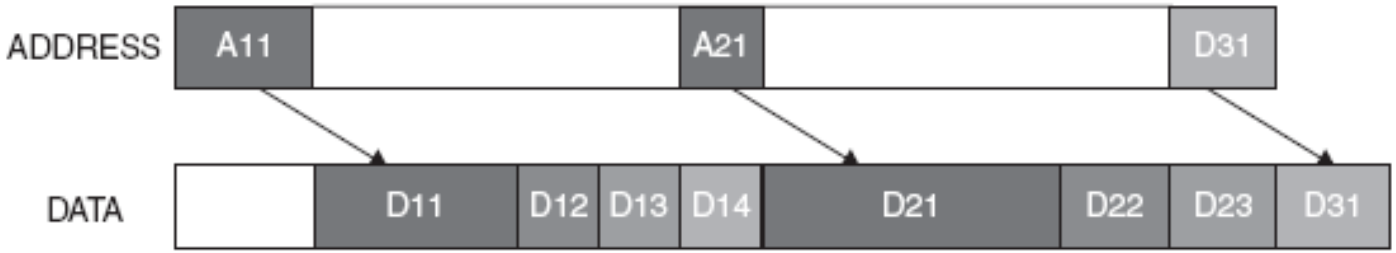
AHB vs. AXI Burst

AHB Burst

- Address and Data are locked together (a single pipeline stage).
- HREADY controls intervals of address and data.



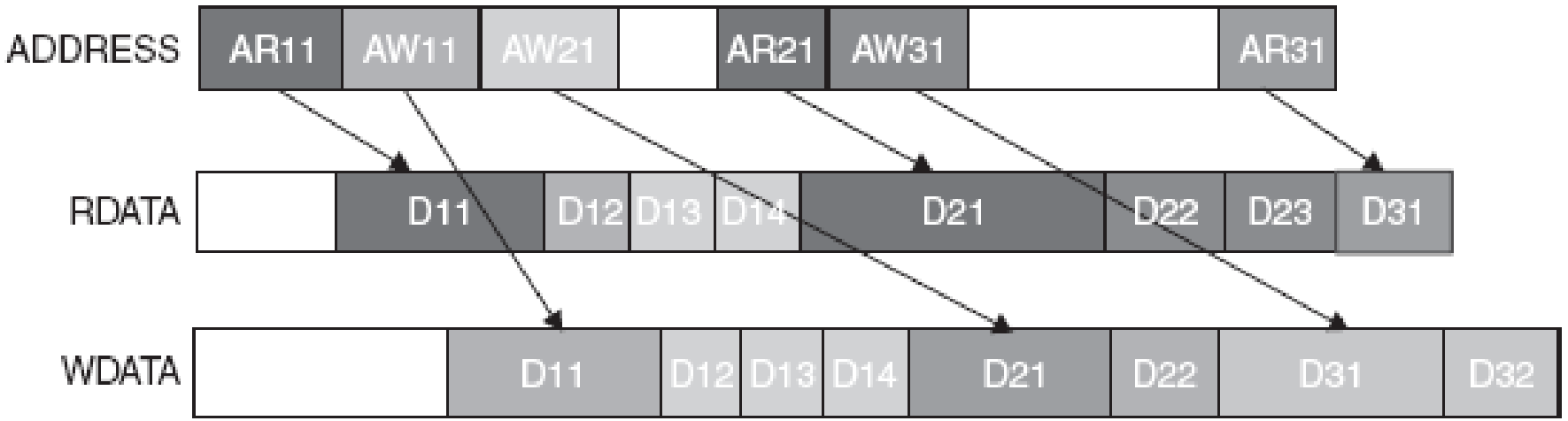
AXI Burst: One Address for entire burst



AHB *vs.* AXI Burst

AXI Burst

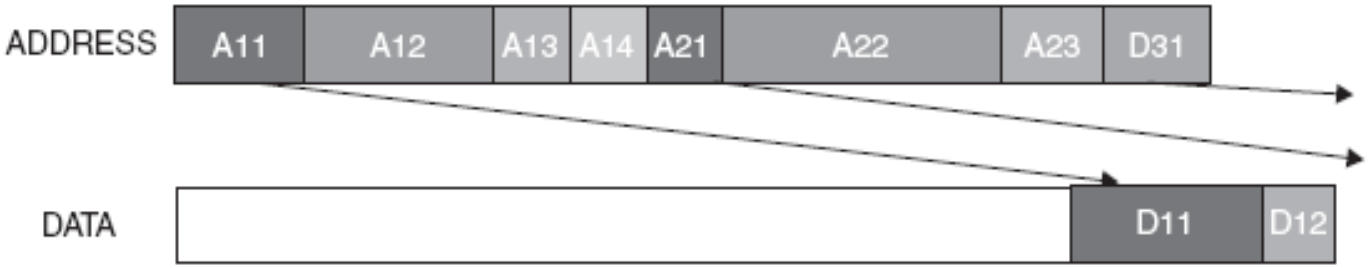
- Simultaneous read, write transactions
- Better bus utilization



AXI Out of Order Completion

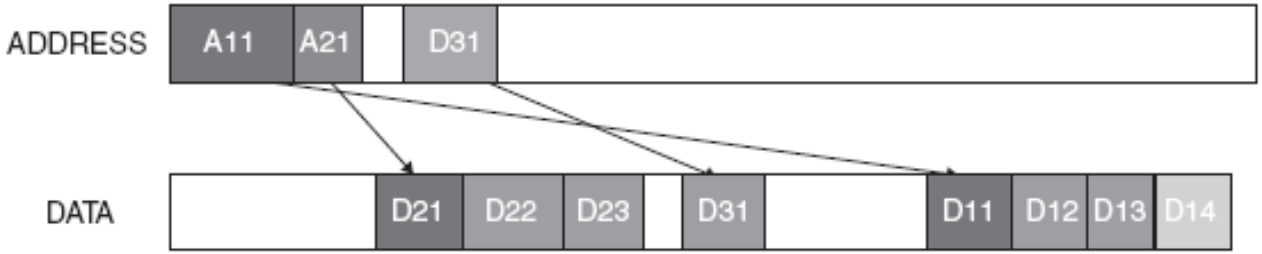
With AHB

- If one slave is very slow, all data is held up
- SPLIT transactions provide very limited improvement



With AXI Burst

- Multiple outstanding addresses, out of order (OO) completion allowed
- Fast slaves may return data ahead of slow slaves



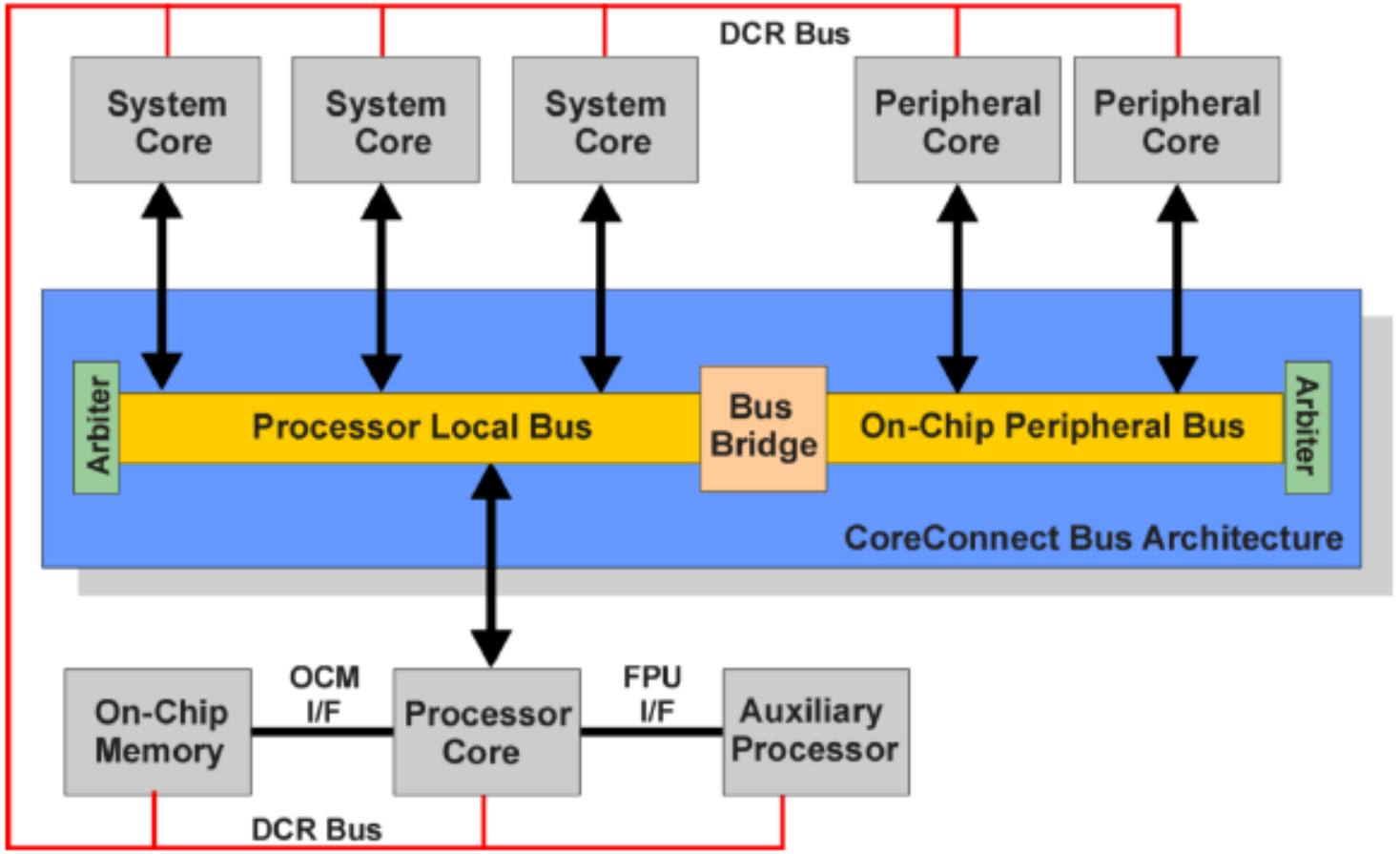
AHB vs. AXI -Summary

AMBA 3.0 AXI	AMBA 2.0 AHB
Channel-based specification, with five separate channels for read address, read data, write address, write data, and write response enabling flexibility in implementation.	Explicit bus-based specification, with single shared address bus and separate read and write data buses.
Burst mode requires transmitting address of only first data item on the bus.	Requires transmitting address of every data item transmitted on the bus.
OO transaction completion provides native support for multiple, outstanding transactions.	Simpler SPLIT transaction scheme provides limited and rudimentary outstanding transaction completion.
Fixed burst mode for memory mapped I/O peripherals.	No fixed burst mode.
Exclusive data access (semaphore operation) support.	No exclusive access support.
Advanced security and cache hint support.	Simple protection and cache hint support.
Register slice support for timing isolation.	No inherent support for timing isolation.
Native low-power clock control interface.	No low-power interface.
Default bus matrix topology support.	Default hierarchical bus topology support.

IBM CoreConnect On-Chip Bus

CoreConnect is an SoC Bus proposed by IBM having:

- **PLB: Processor Local Bus, PLB Arbiter, PLB to OPB Bridge**
- **OPB: On-Chip Peripheral Bus, OPB Arbiter**
- **DCR: Device Control Register Bus and a Bridge**



CoreConnect Advance Features

IBM CoreConnect Bus to support a variety of applications

PLB: Fully synchronous, supports up to 8 masters

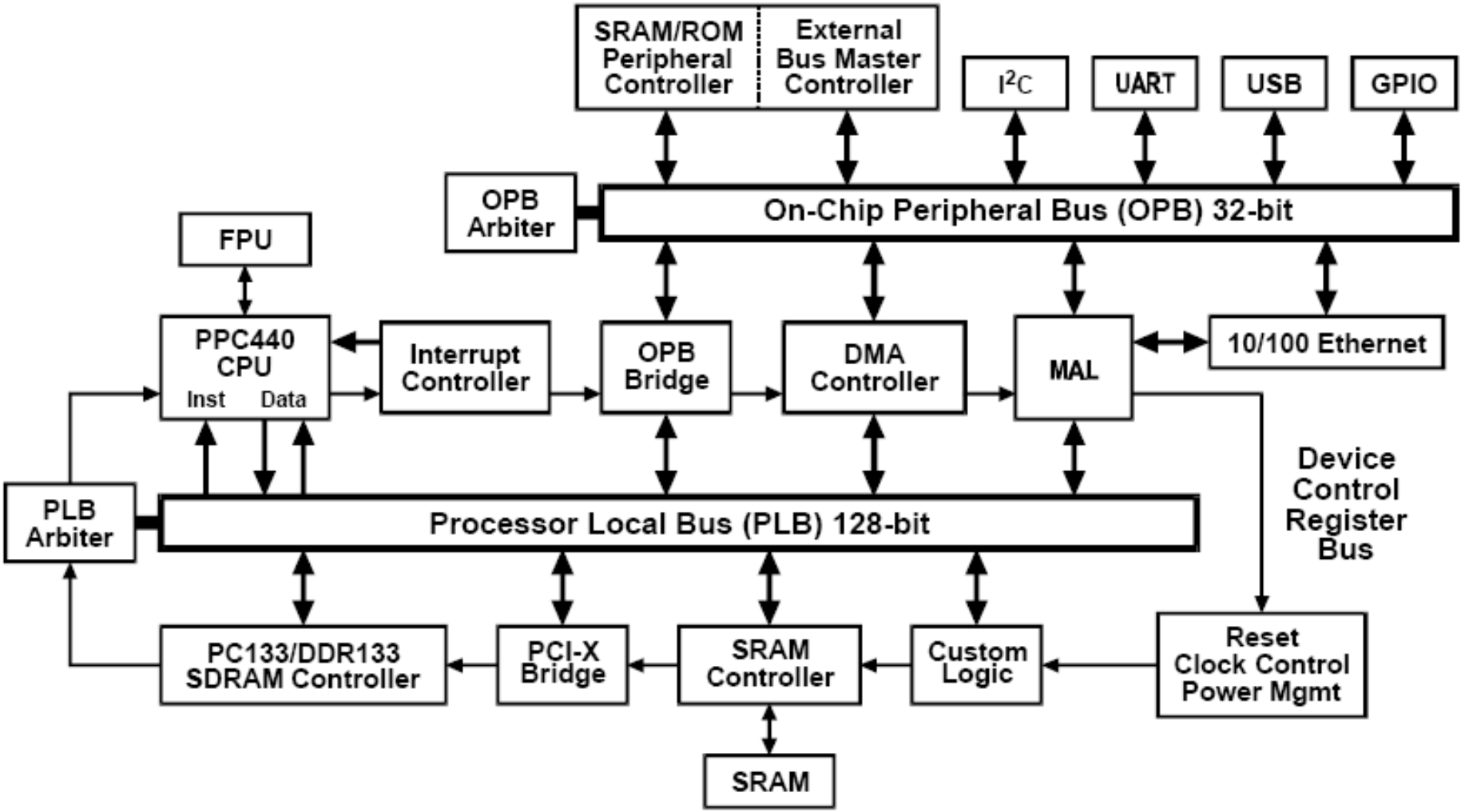
- Separate read/write data buses
- Burst transfers, variable and fixed-length, Pipelining
- DMA transfers and No on-chip tri-states required
- Overlapped arbitration, programmable priority fairness

OPB: Fully synchronous, 32-bit address and data buses

- Support 1-cycle data transfers between master and slaves
- Arbitration for up to 4 OPB master peripherals
- Bridge function can be master on PLB or OPB

DCR: Provides fully synchronous movement of GPR data between CPU and slave logic

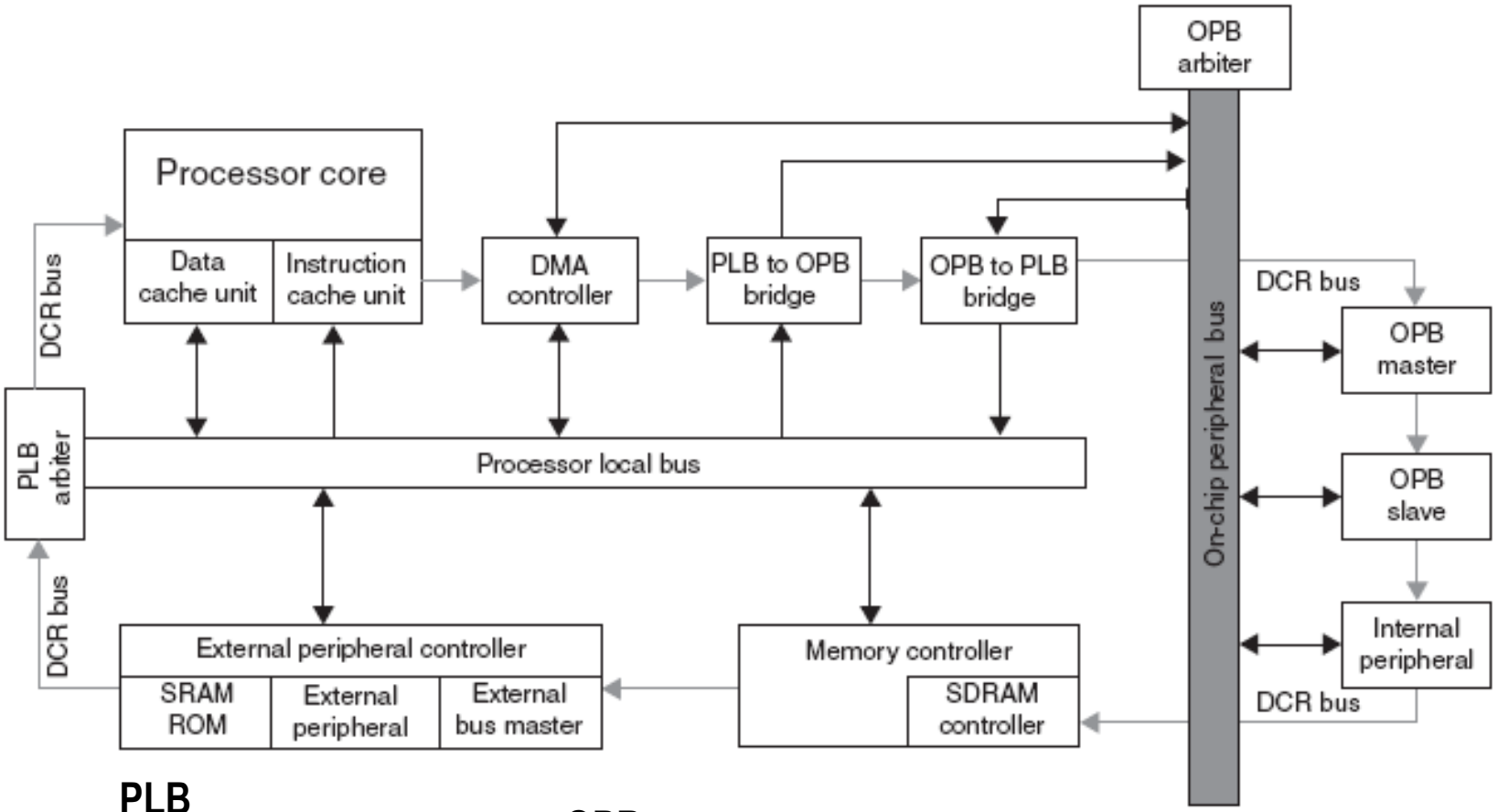
CoreConnect Bus based SoC



AMBA and CoreConnect SoC Buses

	IBM CoreConnect Processor Local Bus	ARM AMBA 2.0 AMBA High-performance Bus
Bus Architecture	32-, 64-, and 128-bits Extendable to 256-bits	32-, 64-, and 128-bits
Data Buses	Separate Read and Write	Separate Read and Write
Key Capabilities	Multiple Bus Masters 4 Deep Read Pipelining 2 Deep Write Pipelining Split Transactions Burst Transfers Line Transfers	Multiple Bus Masters Pipelining Split Transactions Burst Transfers Line Transfers
	On-Chip Peripheral Bus	**AMBA Advanced Peripheral Bus**
Masters Supported	Supports Multiple Masters	Single Master: The APB Bridge
Bridge Function	Master on PLB or OPB	APB Master Only
Data Buses	Separate Read and Write	Separate or 3-state

IBM CoreConnect



PLB

- Pipelined
- Burst modes
- Split transactions
- Multiple masters

OPB

- Low bandwidth
- Burst mode
- Multiple Masters

DCR

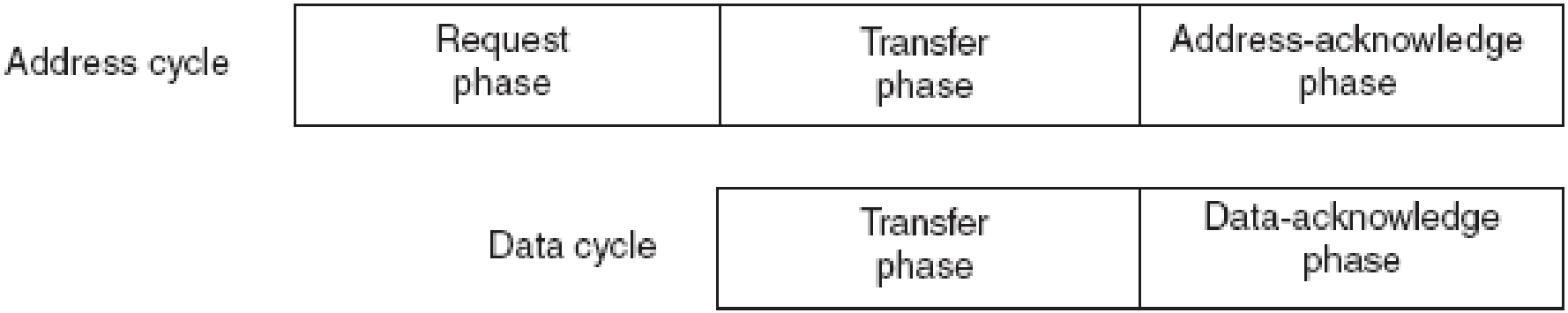
- Low throughput
- 1 r/w = 2 cycles
- Ring type data bus

Processor Local Bus (PLB)

High performance synchronous bus

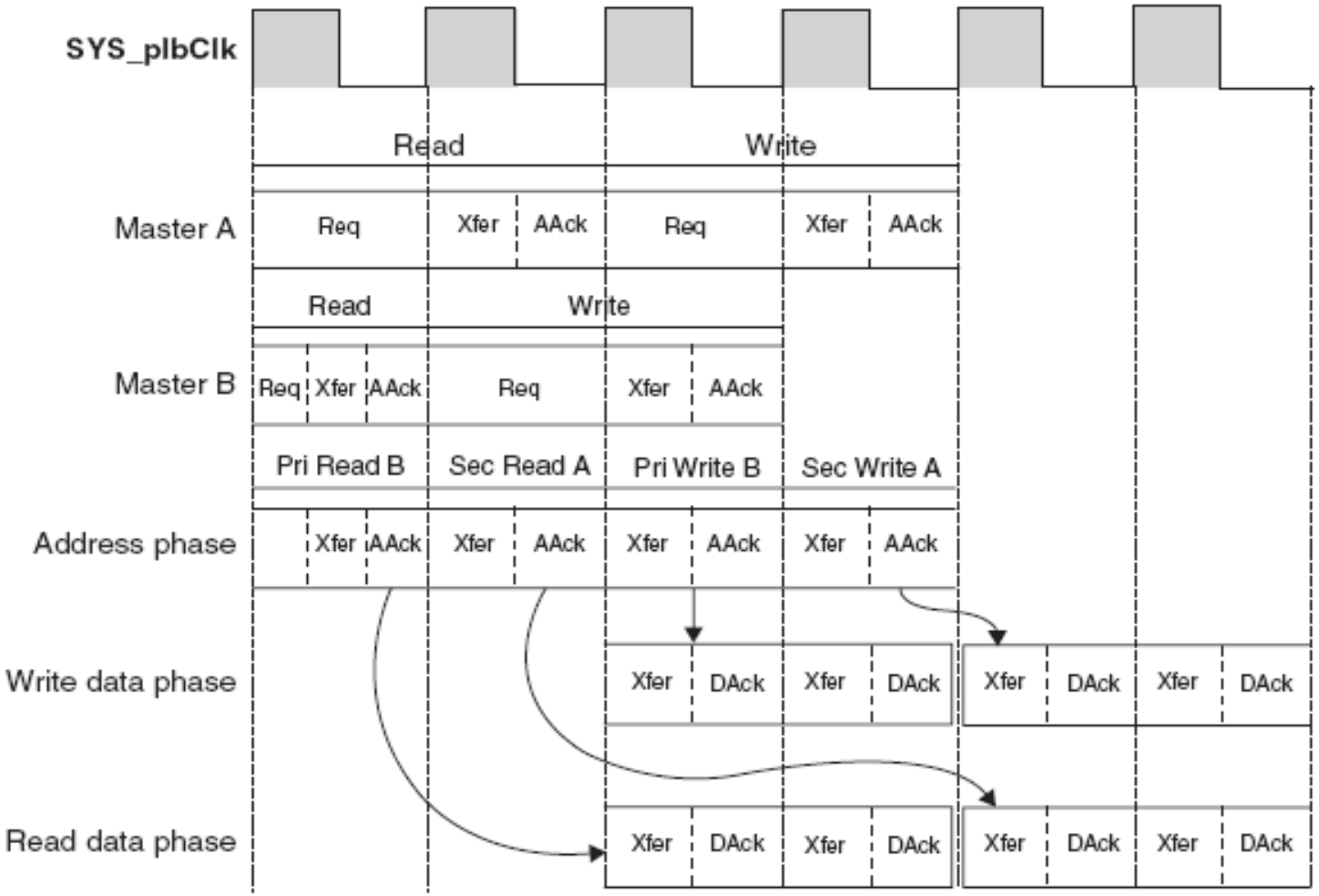
- Shared address, separate read and write data buses
- Support for 32-bit address, 16, 32, 64, & 128-bit data bus widths
- Dynamic bus sizing-byte, half-word, word, double-word transfers
- Up to 16 masters and any number of slaves
- AND-OR implementation structure
- Variable or fixed length (16-64 byte) burst transfers
- Pipelined transfers
- SPLIT transfer support
- Overlapped read and write transfers (up to 2 transfers per cycle)
- Centralized arbiter
- Locked transfer support for atomic accesses

PLB Transfer Phases



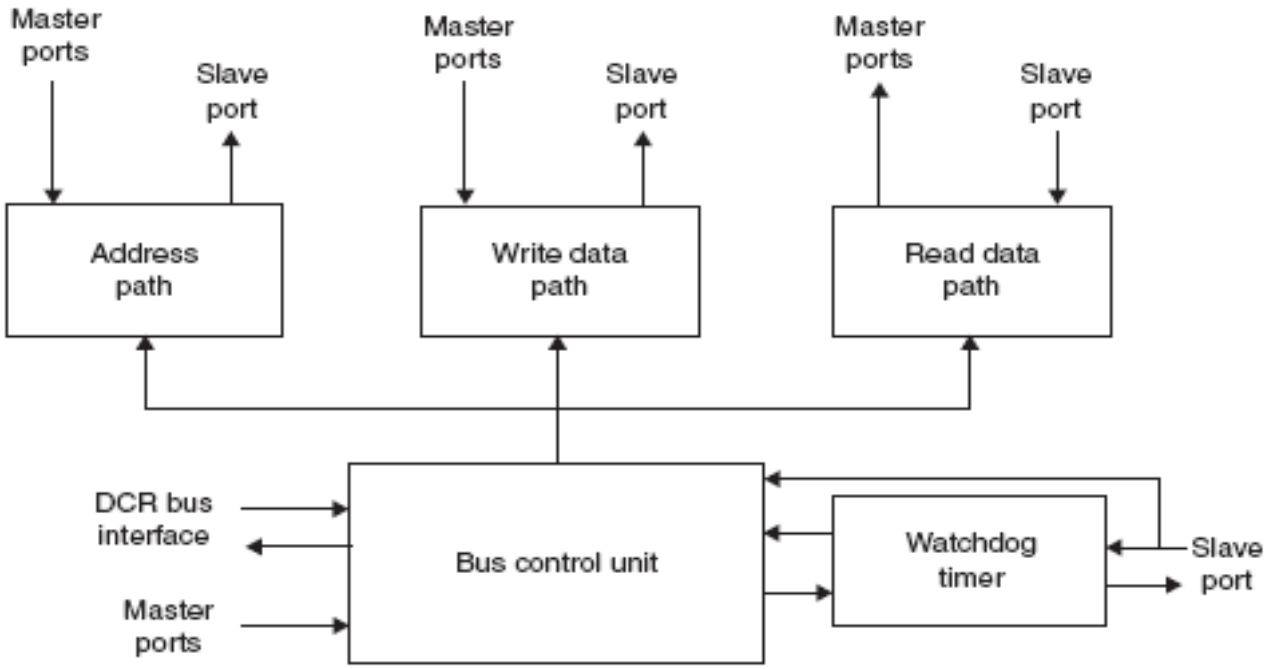
Address and data phases are decoupled

Overlapped PLB Transfers



PLB allows address and data buses to have different masters at the same time

PLB Arbiter



Bus Control Unit

- each master drives a 2-bit signal that encodes 4 priority levels
- in case of a tie, arbiter uses static or RR scheme

Timer

- pre-empts long burst masters
- ensures high priority requests served with low latency

On-chip Peripheral Bus (OPB)

Synchronous bus to connect low performance peripherals and reduce capacitive loading on PLB.

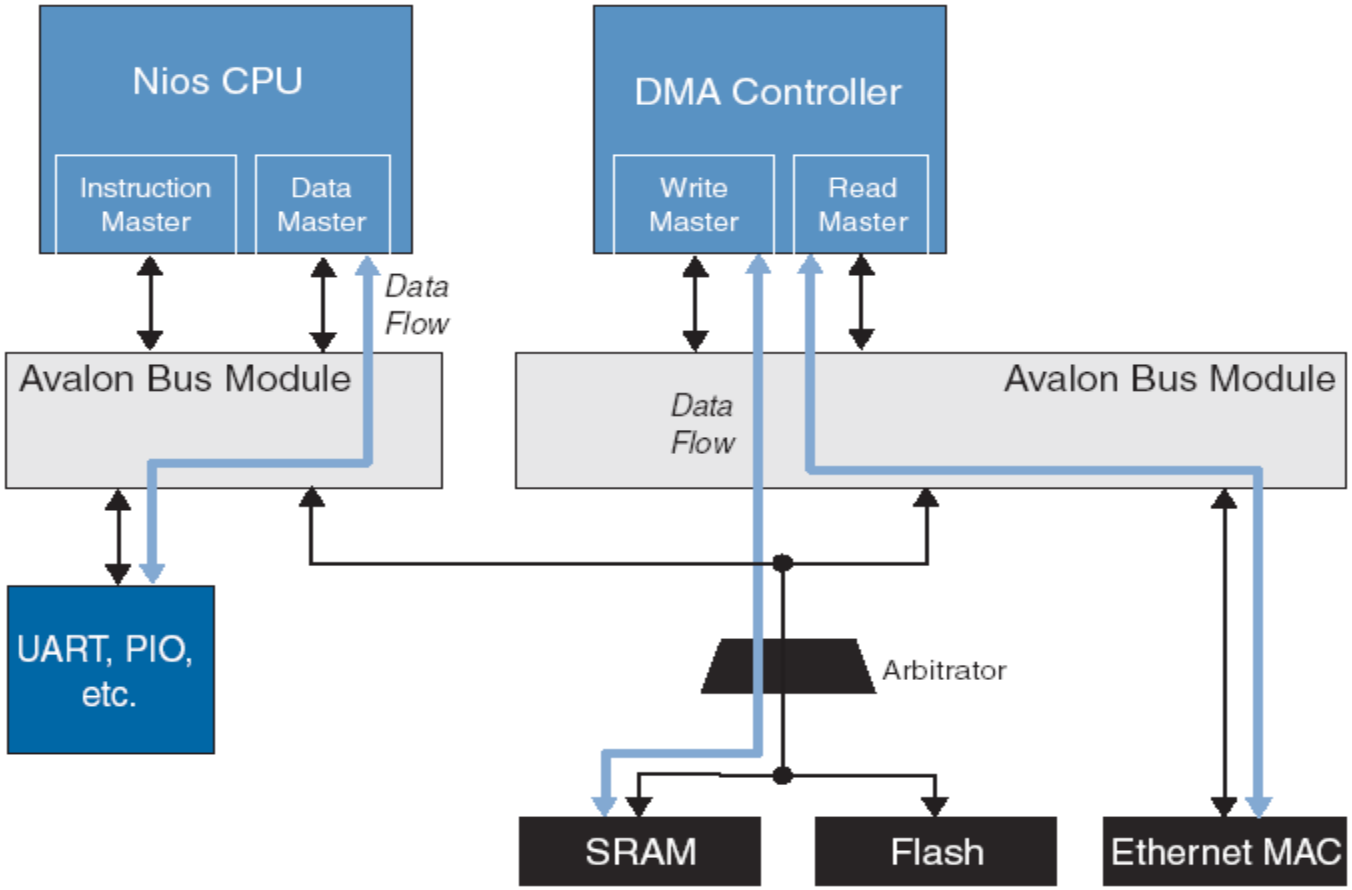
- Shared address bus, multiple data buses.
- Up to a 64-bit address bus width.
- 32- or 64-bit read, write data bus width support.
- Support for multiple masters.
- Bus parking (or locking) for reduced transfer latency.
- Sequential address transfers (burst mode).
- Dynamic bus sizing—byte, half-word, word, double-word transfers.
- MUX-based (or AND–OR) structural implementation.
- Single cycle data transfer between OPB masters and slaves.
- Timeout capability for low-latency for important xfers.

Device Control Register (DCR) Bus

Low speed synchronous bus, used for on-chip device configuration purposes

- meant to off-load the PLB from lower performance status and control read and write transfers
- 10-bit, up to 32-bit address bus
- 32-bit read and write data buses
- 4-cycle minimum read or write transfers
- Slave bus timeout inhibit capability
- Multi-master arbitration
- Privileged and non-privileged transfers
- Daisy-chain (serial) or distributed-OR (parallel) bus topologies

Nios-II CPU & Avalon Bus based System



Avalon Bus

- **Avalon bus is an active, on-chip bus architecture that accommodate the SOPC environment.**
- **The interface to peripherals is synchronous with the Avalon clock. Therefore, no complex, asynchronous handshaking and acknowledge schemes are necessary.**
- **Multiplexers (not tri-state buffers) inside the bus determine which signals drive which peripheral. Peripherals are never required to tri-state their outputs.**
  Even when the peripheral is deselected
- **The address, data and control signals use separate, dedicated ports. It simplifies the design of peripherals as they don't need to decode address and data bus cycles as well as disable its outputs when it is not selected.**

Avalon Bus Module Features

Data-Path Multiplexing - Multiplexers transfer data from the selected slave peripheral to the appropriate master peripheral.

Address Decoding - Produces chip-select signals for each peripheral.

Wait-State Generation

Dynamic Bus Sizing

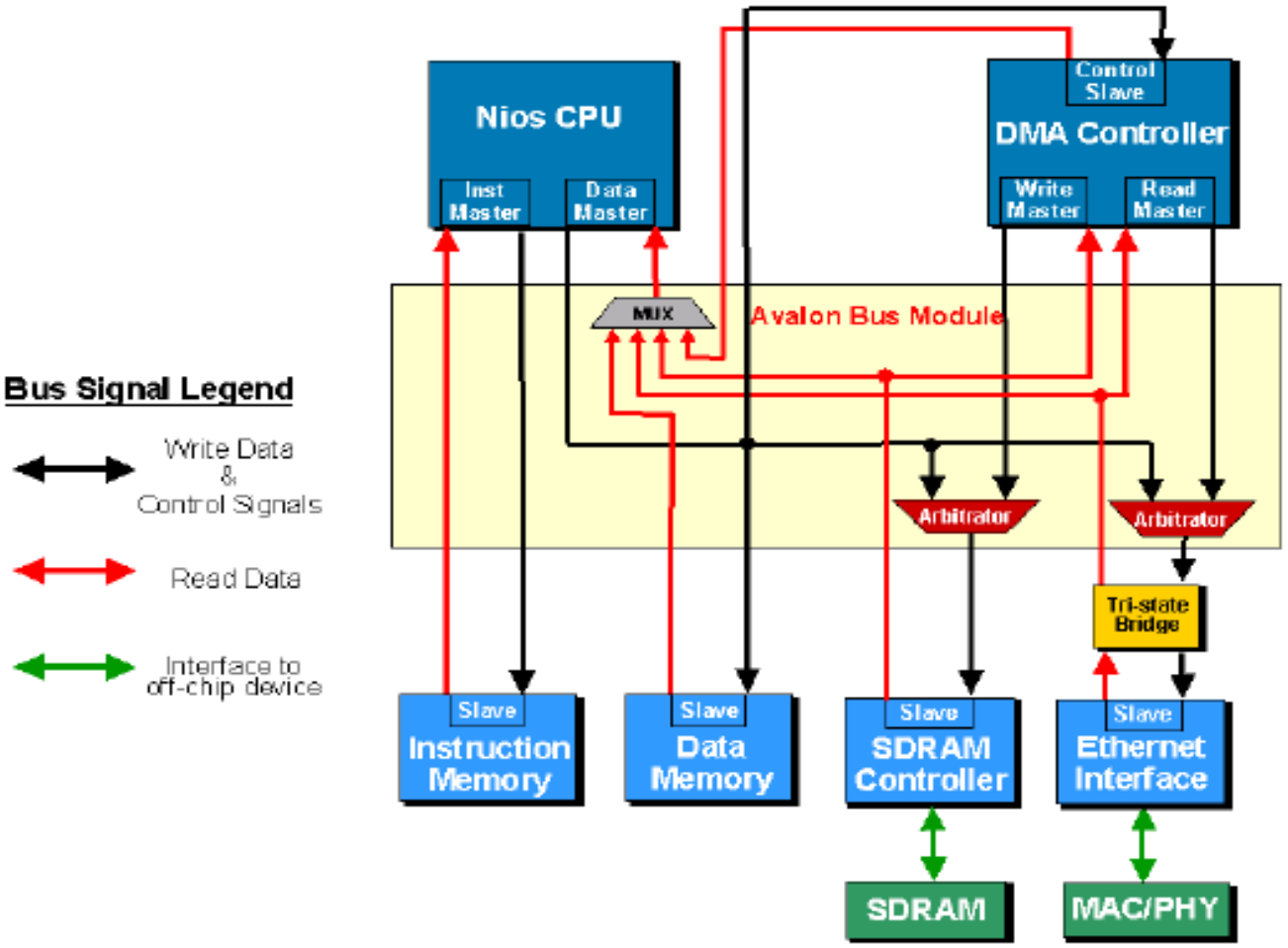
Interrupt-Priority Assignment - When one or more slave peripherals generate interrupts.

Latent Transfer Capabilities

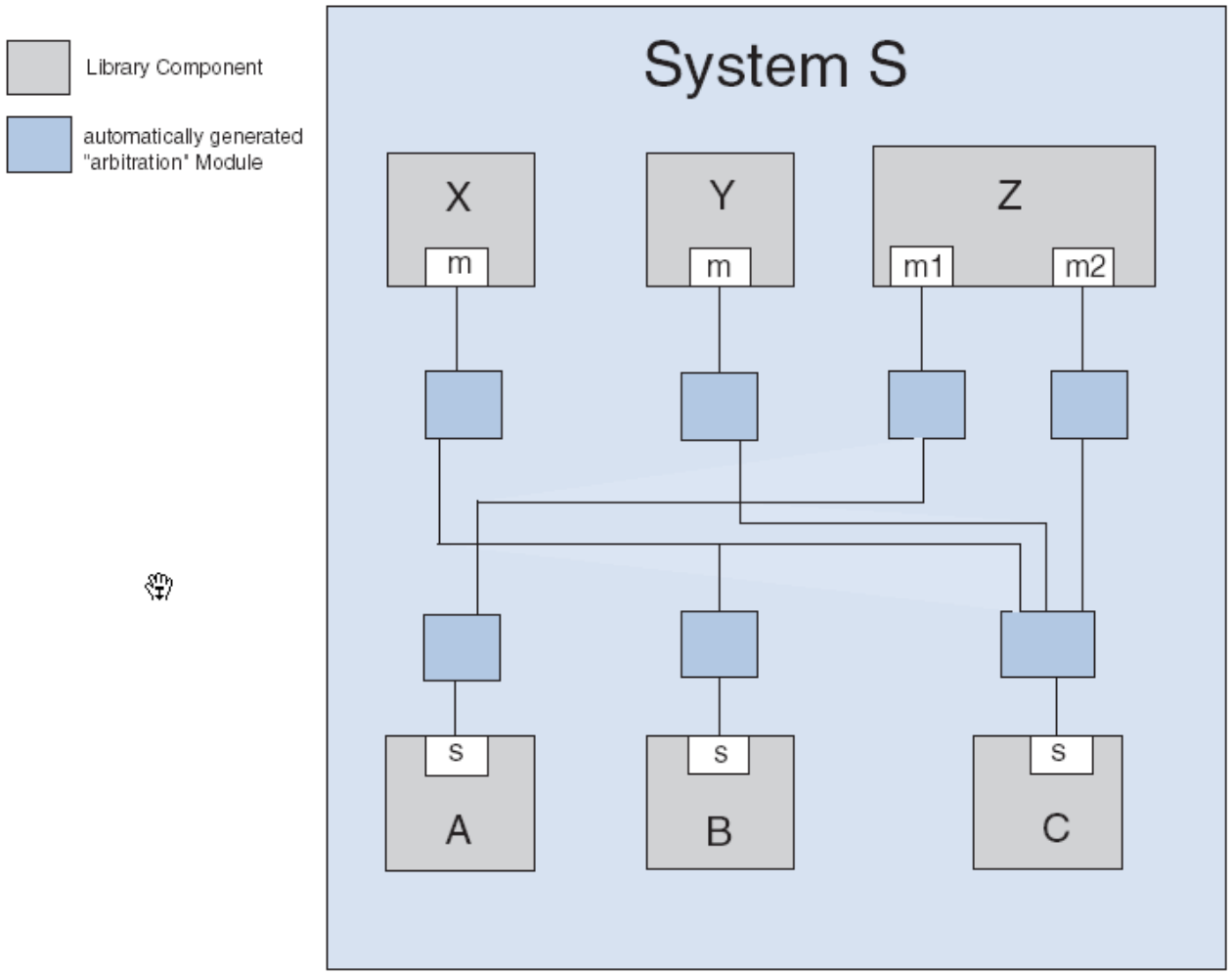
Streaming Read and Write Capabilities - The logic required to allow streaming transfers between master-slave pairs is contained inside the Avalon bus module.

Avalon Bus Module

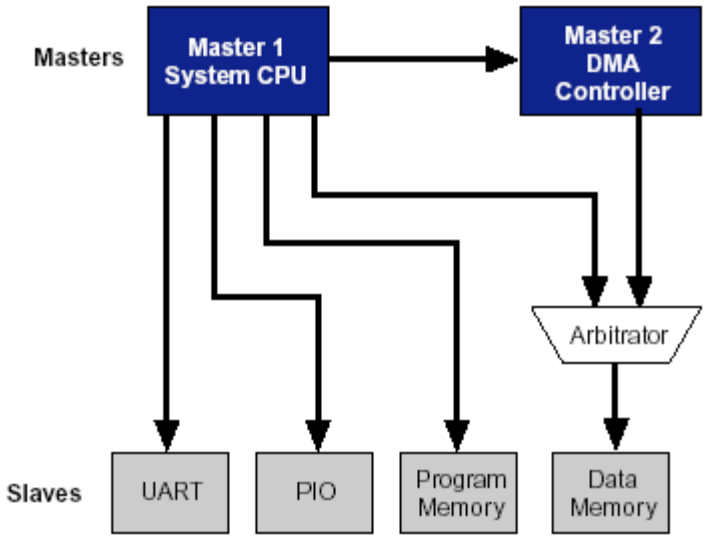
The Avalon bus module (an Avalon bus) is a unit of active logic that takes the place of passive, metal bus lines on a physical PCB.



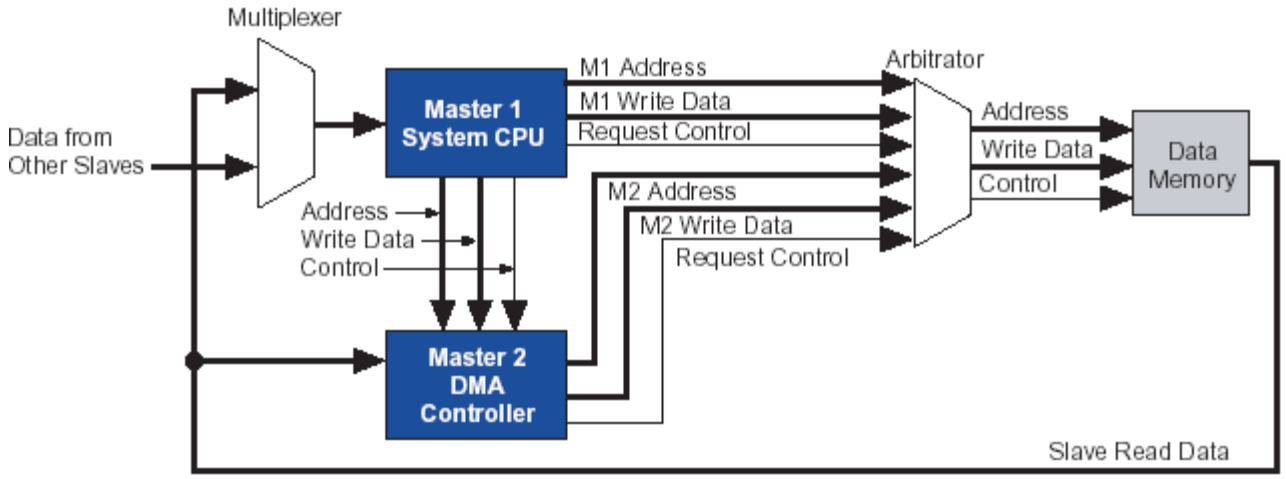
BUS System with Master Modules



Multi-Master: Avalon Bus Arbitration



Slave (data memory) is shared by two masters (Nios CPU and DMA)

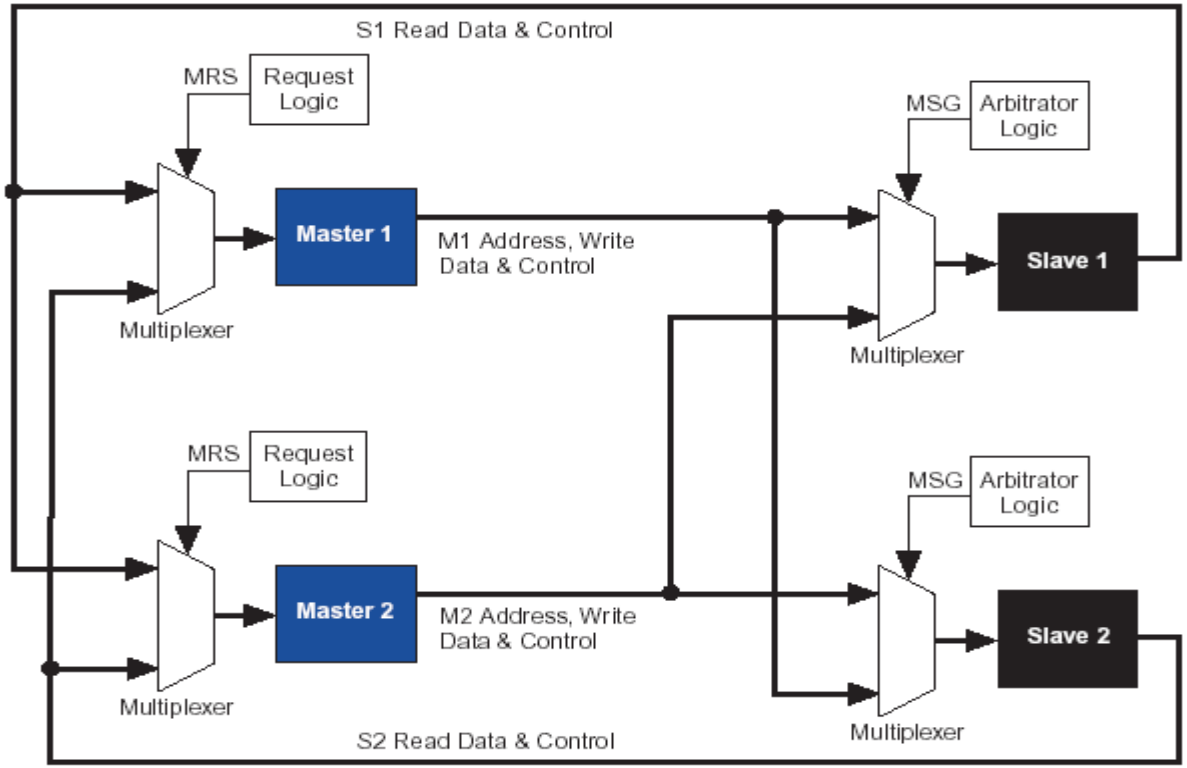


Slave Arbitrator

Avalon bus module contains one slave arbitrator for each shared slave port. Slave arbitrator performs the following.

- Defines control, address, and data paths from multiple master ports to the slave port and specifies the arbitration mechanism to use when multiple masters contend for a slave at the same time.
- At any given time, selects which master port has access to the slave port and forces all other contending masters (if any) to wait, based on the arbitration assignments.
- Controls the slave port, based on the address, data, and control signals presented by the currently selected master port.

Multi-Masters and Slaves



Request and arbitrator logic

Simultaneous multi-master system that permits bus transfers between two masters and two slaves.

Master Request Slave (MRS)	Multiplexer control that connects the wait and data signals from multiple slave ports to a single master port.
Master Select Granted (MSG)	Multiplexer control that connects the data and control signals from multiple master ports to a single slave port.
Wait	Input to each master port that indicates that the bus transfer should be held when the desired slave port cannot be accessed immediately.

Standard Bus Architectures

- AMBA 2.0, 3.0 (ARM)
- CoreConnect (IBM)
- Avalon (Altera)
- STBus (STMicroelectronics)
- Sonics Smart Interconnect (Sonics)
- Wishbone (Opencore)
- PI Bus (OMI)
- MARBLE (Univ. of Manchester)
- CoreFrame (PalmChip)
- ...

STBus

Consists of 3 synchronous bus-based interconnect specifications

- **Type 1**
  - Simplest protocol meant for peripheral access
- **Type 2**
  - More complex protocol
  - Pipelined, SPLIT transactions
- **Type 3**
  - Most advanced protocol
  - OO transactions, transaction labeling/hints

Type 1 and 3

Type 1

- Simple handshake mechanism
- 32-bit address bus
- Data bus sizes of 8, 16, 32, 64 bits
- Similar to IBM CoreConnect DCR bus

Type 3

- transaction completion
- Requires only single response/ACK Supports all Type 2 functionality
- OO for multiple data transfers (burst mode)

Type 2

- Supports all Type 1 functionality
- Pipelined transfers
- SPLIT transactions
- Data bus sizes up to 256 bits
- Compound operations
  - READMODWRITE: Returns read data and locks slave till same master writes to location
  - SWAP : Exchanges data value between master and slave
  - FLUSH/PURGE: Ensure coherence between local and main memory
  - USER: Reserved for user defined operations

STBus Arbitration

- **Static priority**
  - Non-preemptive
- **Programmable priority**
- **Latency based**
  - Each master has register with max. allowed latency (clock cycles)
  - If value is 0, Each master also has counter loaded with max. latency value when master makes request
  - Master counters are decremented at every subsequent cycle
  - Arbiter grants access to master with lowest counter value
  - In case of a tie, static priority is used
    - Higher priority master must be granted bus access as soon as it requests it.

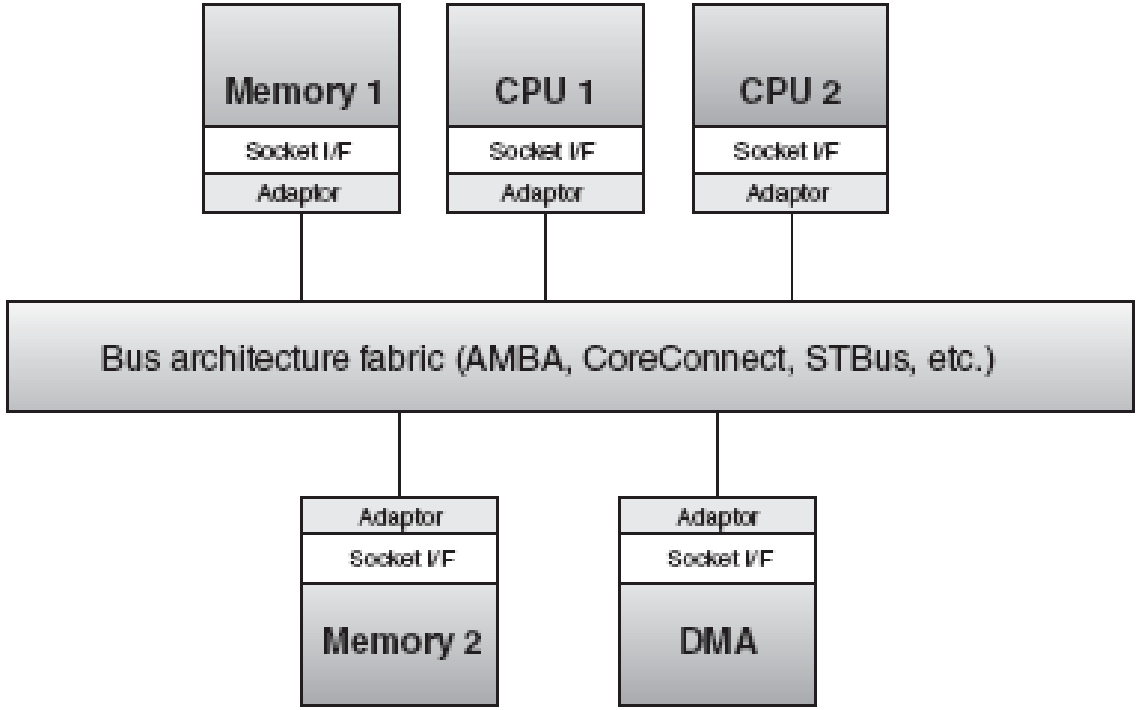
STBus Arbitration

- **Bandwidth based**
  - Similar to TDMA/RR (Round Robin) scheme
- **STB**
  - Hybrid of latency based and programmable priority schemes
  - In normal mode, programmable priority scheme is used
  - Masters have max. latency registers, counters (latency based)
  - Each master also has an additional *latency-counter-enable* bit
  - If this bit is set, and counter value is 0, master is in “panic state”
  - If one or more masters in panic state, programmable priority scheme is overridden, and panic state masters granted access
- **Message based**
  - Pre-emptive static priority scheme

Socket-based Interface Standards

Defines the interface of components

- Does not define bus architecture implementation
- Shield IP designer from knowledge of interconnection system, and enable same IP to be ported across different systems
- Requires Adaptor components to interface with implementation



Socket-based Interface Standards

- **Must be generic, comprehensive, and configurable**
  - to capture basic functionality and advanced features of a wide array of bus architecture implementations
- **Adaptor (or translational) logic component**
  - Must be created only once for each implementation (e.g. AMBA)
  - – adds area, performance penalties, more design time
  - + enhances reuse, speeds up design time across many designs
- **Commonly used socket-based interface standards**
  - Open Core Protocol (OCP) ver 2.0
    - **Most popular – used in Sonics Smart Interconnect**
  - VSIA Virtual Component Interface (VCI)
    - **Subset of OCP**

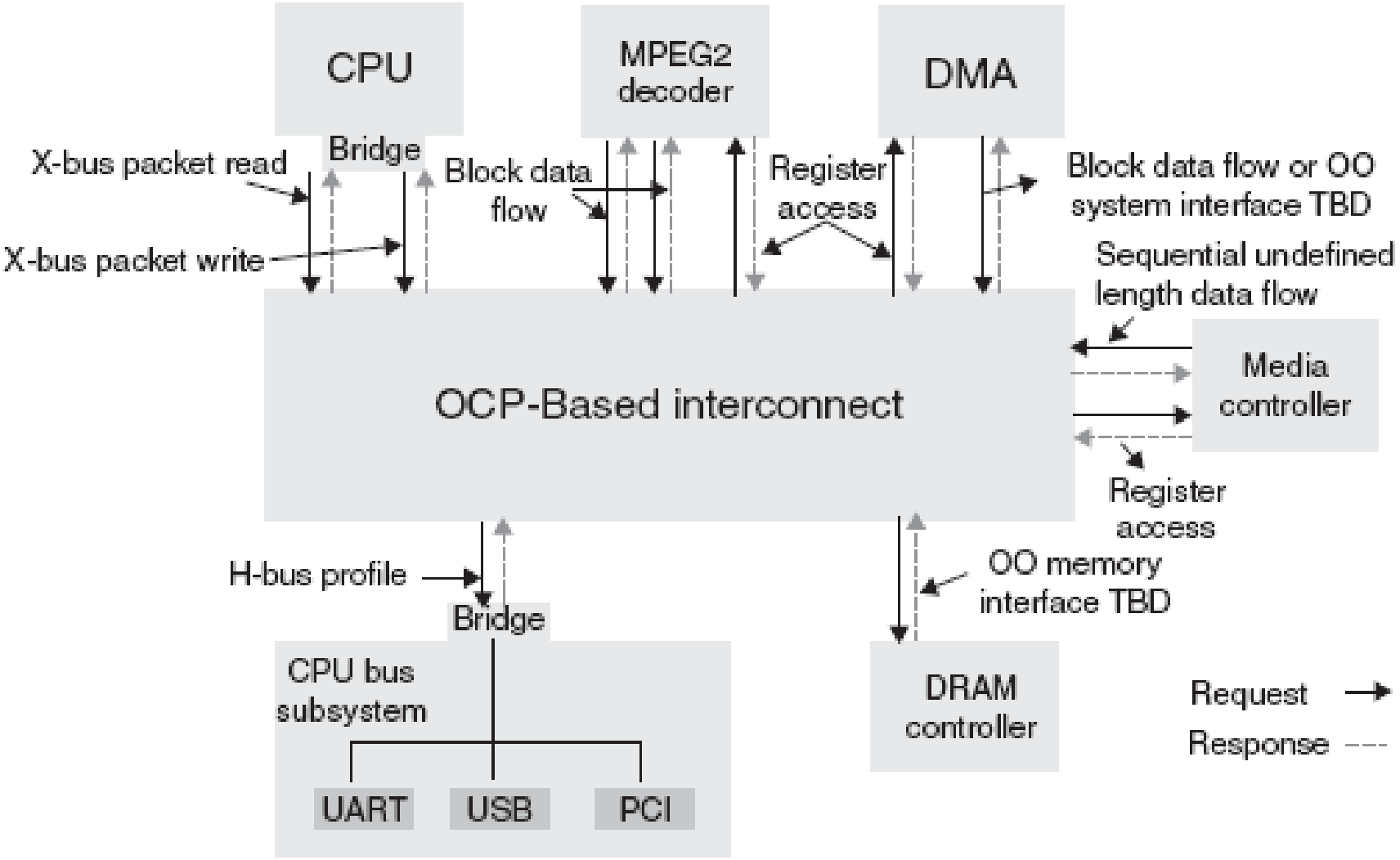
OCP 2.0/3.0
Open Core Protocol

- Point-to-point synchronous interface
- Bus architecture independent
- Configurable data flow (address, data, control) signals for area-efficient implementation
- Configurable sideband signals to support additional communication requirements
- Pipelined transfer support
- Burst transfer support
- OO (out-of-order) transaction completion support
- Multiple threads

OCP 3.0 Basic Signals

Name	Width	Driver	Function
Clk	1	varies	Clock input
EnableClk	1	varies	Enable OCP clock
MAddr	configurable	master	Transfer address
MCmd	3	master	Transfer command
MData	configurable	master	Write data
MDataValid	1	master	Write data valid
MRespAccept	1	master	Master accepts response
SCmdAccept	1	slave	Slave accepts transfer
SData	configurable	slave	Read data
SDataAccept	1	slave	Slave accepts write data
SResp	2	slave	Transfer response

Example: SoC with Mixed Profiles



Summary and Conclusions

- Standards important for seamless integration of SoC IPs
  - avoid costly integration mismatches
- Two categories of standards for SoC communication:
  - Standard bus architectures
    - define interface between IPs and bus architecture
    - define (at least some) specifics of bus architecture that implements data transfer protocol
    - e.g. AMBA 2.0/3.0, CoreConnect, Sonics Smart Interconnect, STBus
  - Socket based bus interface standards (e.g. OCP 2.0)
    - define interface between IPs and bus architecture
    - do not define bus architecture implementation specifics