# AMBA

Ref: AMBA Specification Rev. 2.0

# Outline

- Overview
- AHB
- APB
- Test methodology

# Outline

- Overview
- AHB
- APB
- Test methodology

# BUS Brief

- In a system, various subsystems must have interfaces to one another
- The bus serves as a shared communication link between subsystems
- Advantages
  - Low cost
  - Versatility
- Disadvantage
  - Performance bottleneck

# AMBA Introduction

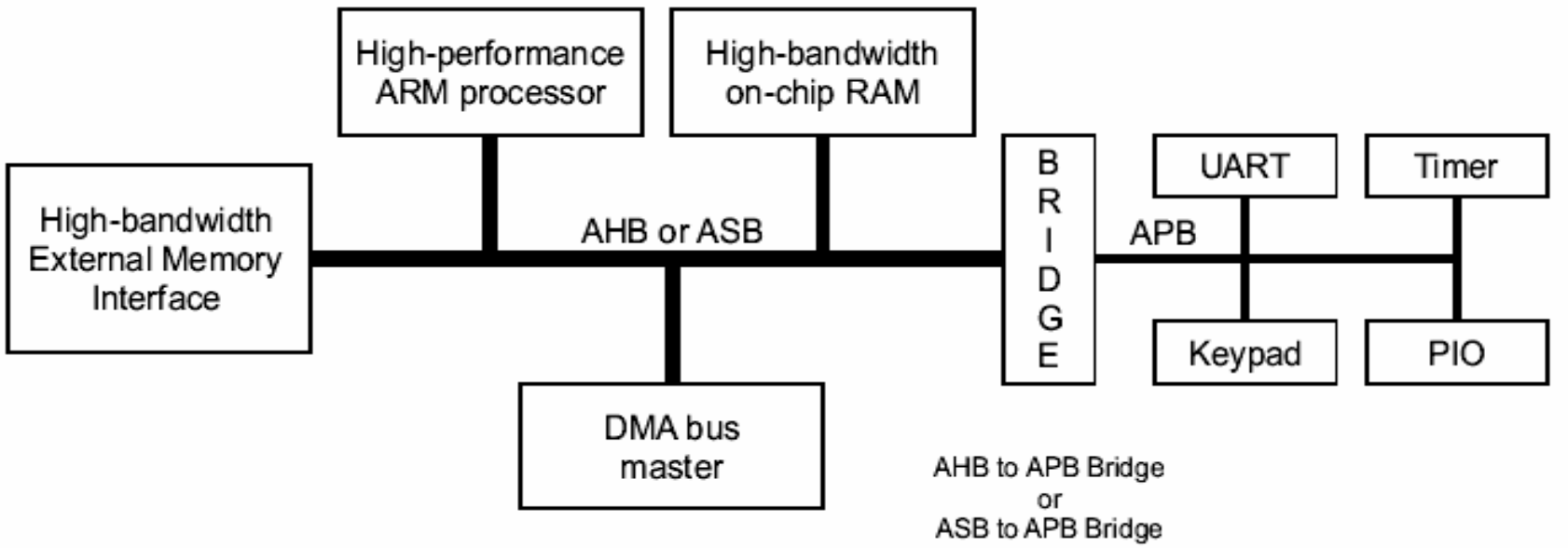
- Advanced Microcontroller Bus Architecture
  - An on-chip communication standard
- Three buses defined
  - AHB (Advanced High-performance Bus)
  - ASB (Advanced System Bus)
  - APB (Advanced Peripheral Bus)

# AMBA History

- AMBA 1.0
  - ASB and APB
  - Tri-state implementation
- AMBA 2.0
  - AHB, ASB, and APB
  - Multiplexer architecture to eliminate timing problem
- AMBA 3.0
  - AMBA Advanced eXtensible Interface (AXI)

# A Typical AMBA System

- Processors or other masters/slaves can be replaced



# AHB

- High performance
- Pipelined operation
- Multiple bus masters (up to 16)
- Burst transfers
- Split transactions
- Bus width: 8, 16, 32, 64, 128 bits
- Mux-type bus
- Single clock edge (rising edge) design
- Recommended for new designs

# ASB

- High performance
- Pipelined operation
- Multiple bus masters
- Burst transfers
- Bus width: 8, 16, 32 bits
- Tristate-type bus
- Falling edge design

# APB

- Lower power
- Latched address and control
- Simple interface
- Suitable for many peripherals
- Single clock edge (rising edge) design
- Appears as a local secondary bus that is encapsulated as a single AHB or ASB slave devices

# AHB Components

- AHB master
  - Initiate a read/write operation
  - Only one master is allowed to use the bus
  - uP, DMA, DSP, …
- AHB slave
  - Respond to a read/write operation
  - Address mapping
  - External memory I/F, APB bridge, internal memory, …
- AHB arbiter
  - Ensure that which master is active
  - Arbitration algorithm is not defined in ABMA spec.
- AHB decoder
  - Decode the address and generate select signal to slaves

# APB Components

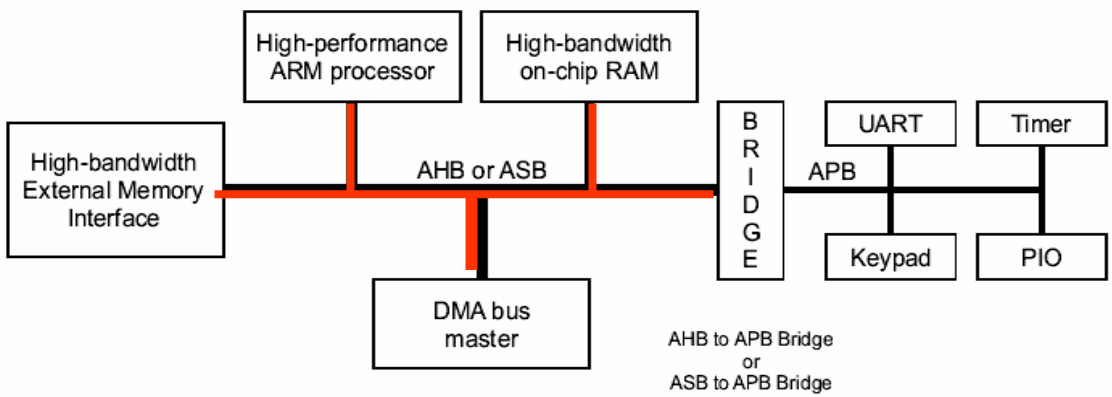
- AHB2APB Bridge
  - Provides latching of all address, data, and control signals
  - Provides a second level of decoding to generate slave select signals for the APB peripherals
- All other modules on the APB are APB slaves
  - Un-pipelined
  - Zero-power interface
  - Timing can be provided by decode with strobe timing
  - Write data valid for the whole acess

# Notes on the AMBA Specification

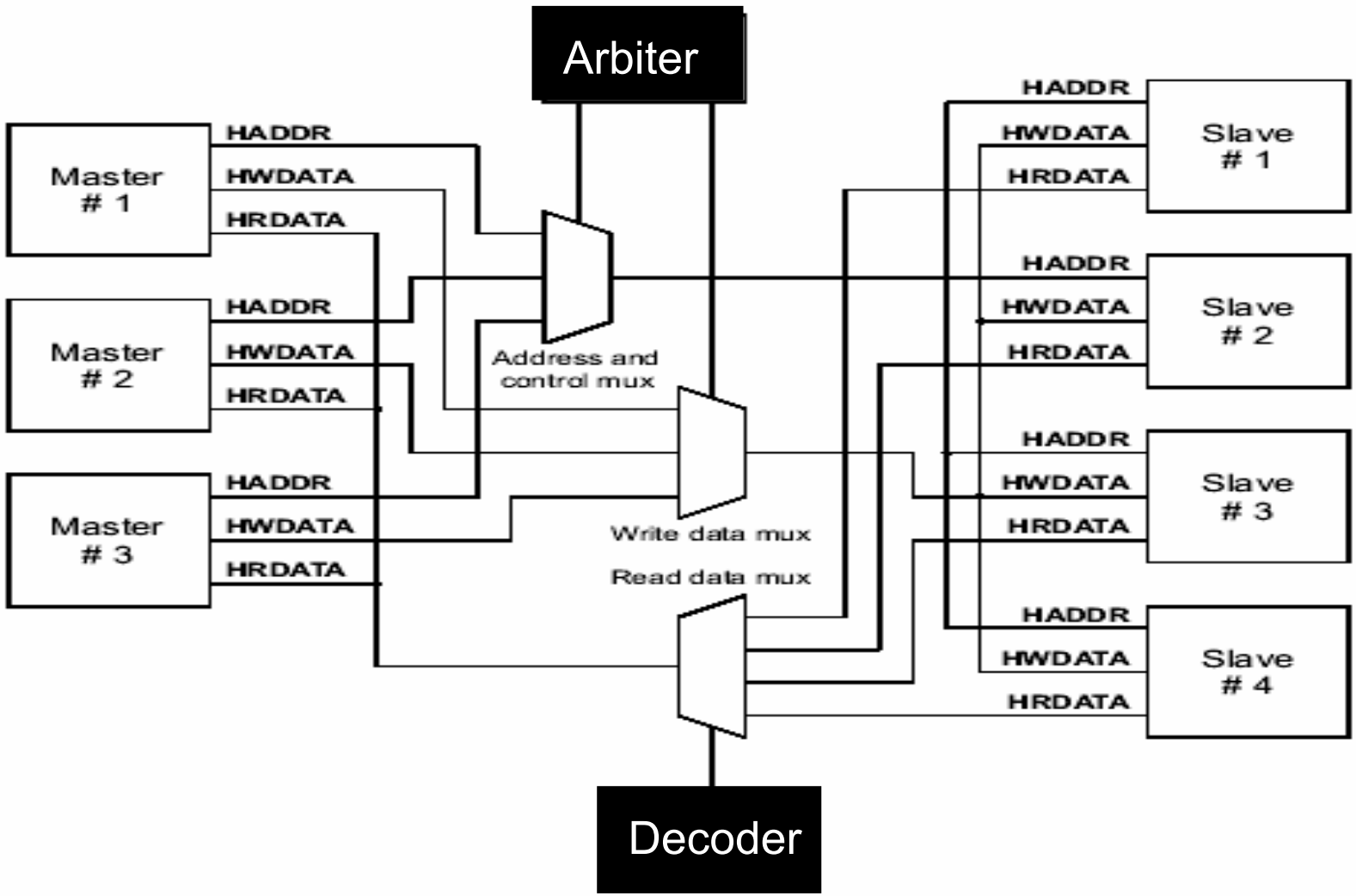
- Technology independent
- Not define electrical characteristics
- Timing specification only at the cycle level
  - Exact timing requirements will depend on the process technology used and operation frequency

# Outline

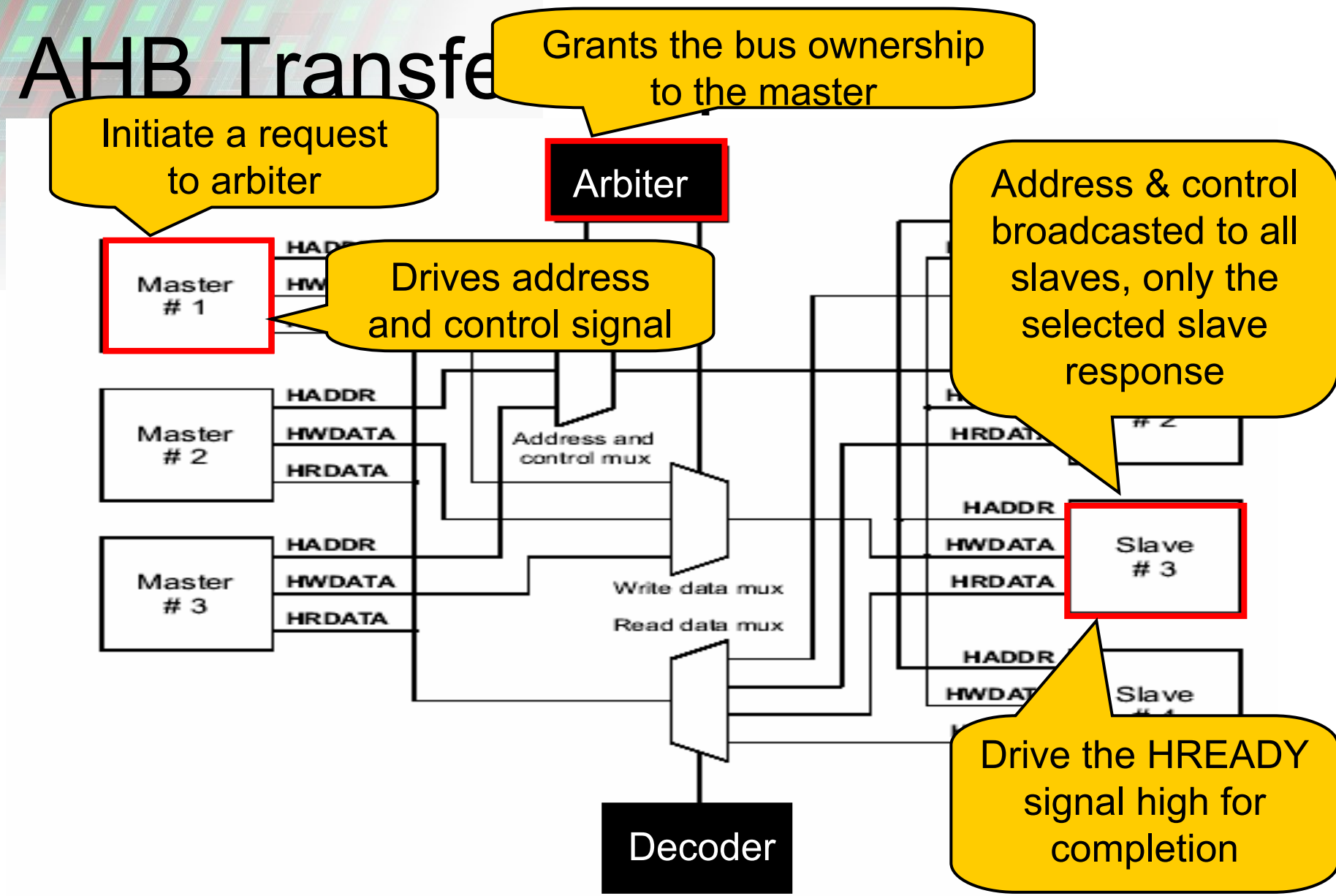
- Overview
- AHB
- APB
- Test methodology



# AHB Bus Interconnection



# AHB Transfe


Grants the bus ownership to the master
Initiate a request to arbiter
Arbiter
Master # 1
Drives address and control signal
Address & control broadcasted to all slaves, only the selected slave response
HADDR
HWDATA
HRDATA
Master # 2
Address and control mux
Master # 3
Write data mux
Read data mux
Slave # 3
Slave
Drive the HREADY signal high for completion
Decoder

# AHB Signals

| Name | Source | Description |
|---|---|---|
| HCLK | Clock source | Bus clock |
| HRESETn | Reset controller | Reset |
| HADDR[31:0] | Master | Address bus |
| HTRANS[1:0] | Master | Transfer type |
| HWRITE | Master | Transfer direction |
| HSIZE[2:0] | Master | Transfer size |
| HBURST[2:0] | Master | Burst type |
| HPROT[3:0] | Master | Protection control |
| HWDATA[31:0] | Master | Write data bus |
| HSELx | Decoder | Slave select |
| HRDATA[31:0] | Slave | Read data bus |
| HREADY | Slave | Transfer done |
| HRESP[1:0] | Slave | Transfer response (status) |

# Basic AHB Signals

- HRESETn
  - Active low
- HADDR[31:0]
  - The 32-bits system address bus
- HWDATA[31:0]
  - Write data bus, from master to slave
- HRDATA[31:0]
  - Read data bus, from slave to master

# Basic AHB Signals (cont.)

- HTRANS
  - Indicates the type of the current transfer
    - NONSEQ, SEQ, IDLE, or BUSY
- HSIZE
  - Indicate the size of the transfer
- HBURST
  - Indicate the burst type of the transfer
- HRESP
  - Shows the status of bus transfer, from slave to master
    - OKAY, ERROR, RETRY, or SPLIT

# Basic AHB Signals (cont.)

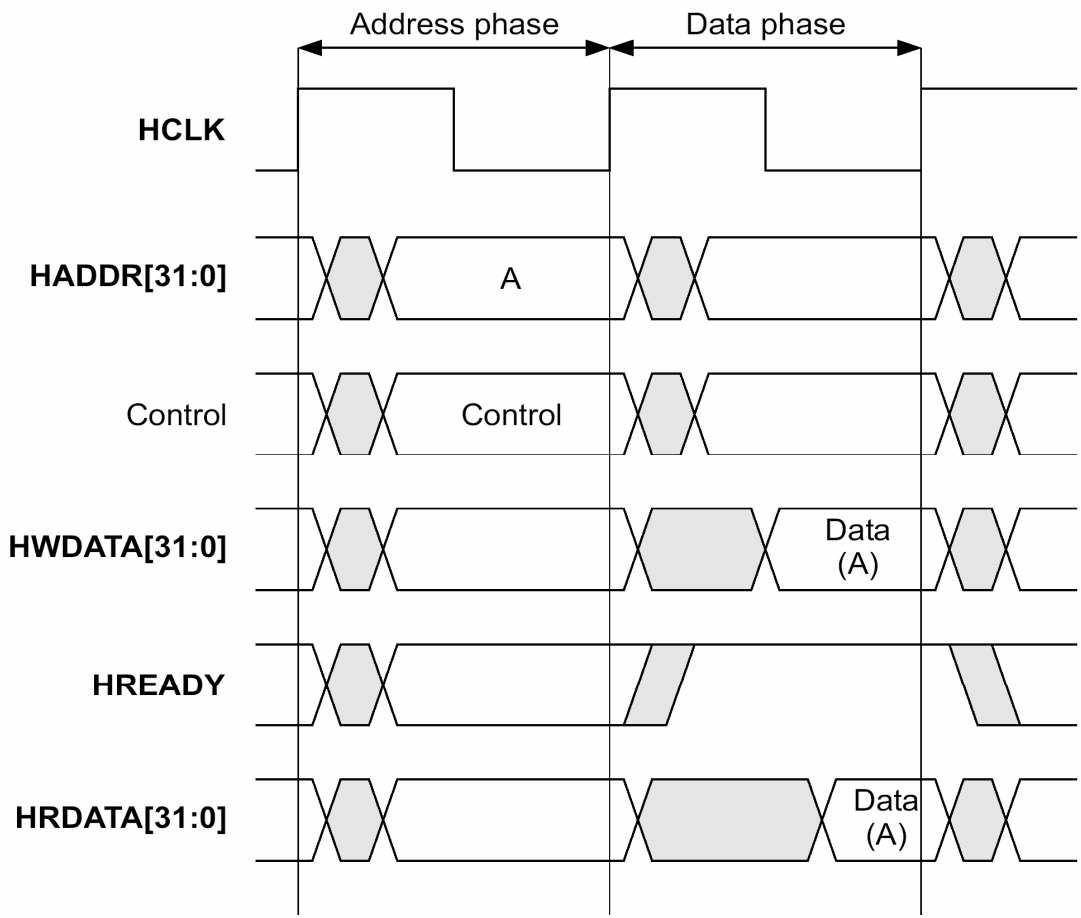
- HREADY
  - High: the slave indicate the transfer done
  - Low: the slave extend the transfer
- HREADY_IN
  - From decoder
  - Decoder tell the slave that the bus is available
  - Not explained in AMBA spec. 2.0 !!

# Basic AHB Transfers

- Two distinct sections
  - The **address phase**, only one cycle
  - The **data phase**, may required several cycles, achieved by HREADY signals
- Pipeline transaction
  - Address phase is before the data phase

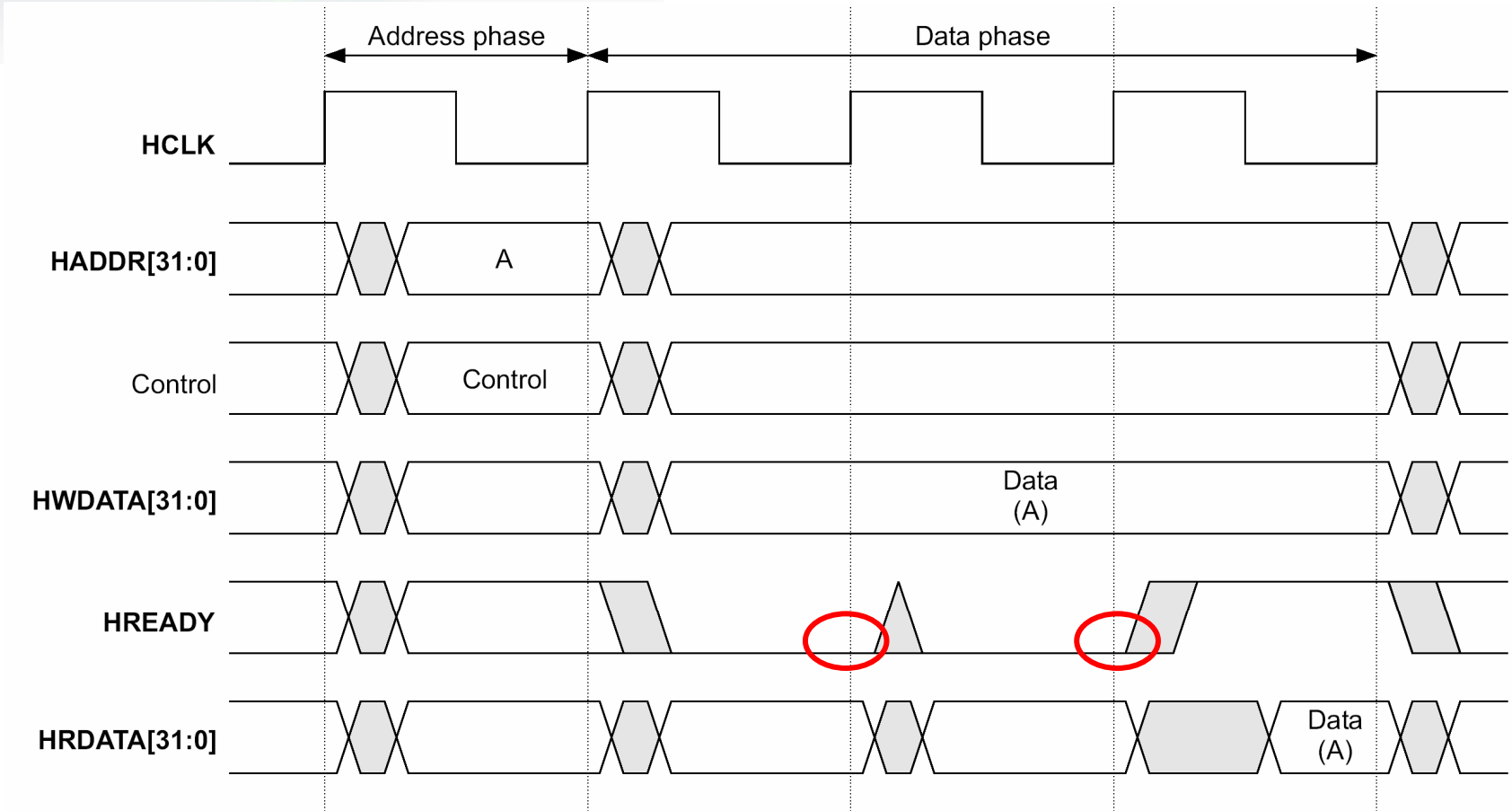
# Basic AHB Transfers (cont.)

- A simple transfer with no wait state



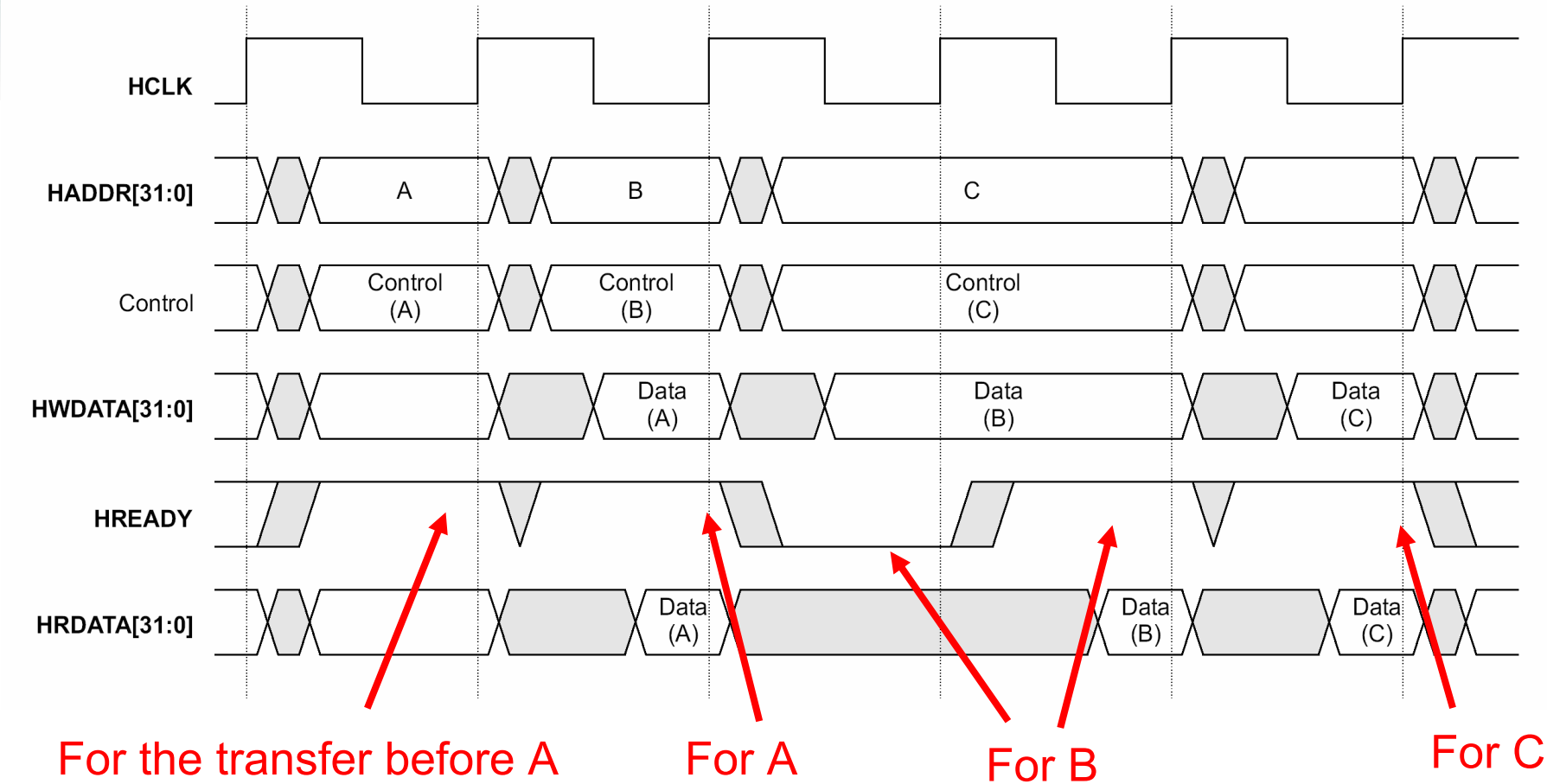
# Basic AHB Transfers (cont.)

- A simple transfer with two wait states



# Basic AHB Transfers (cont.)

- Multiple transfers



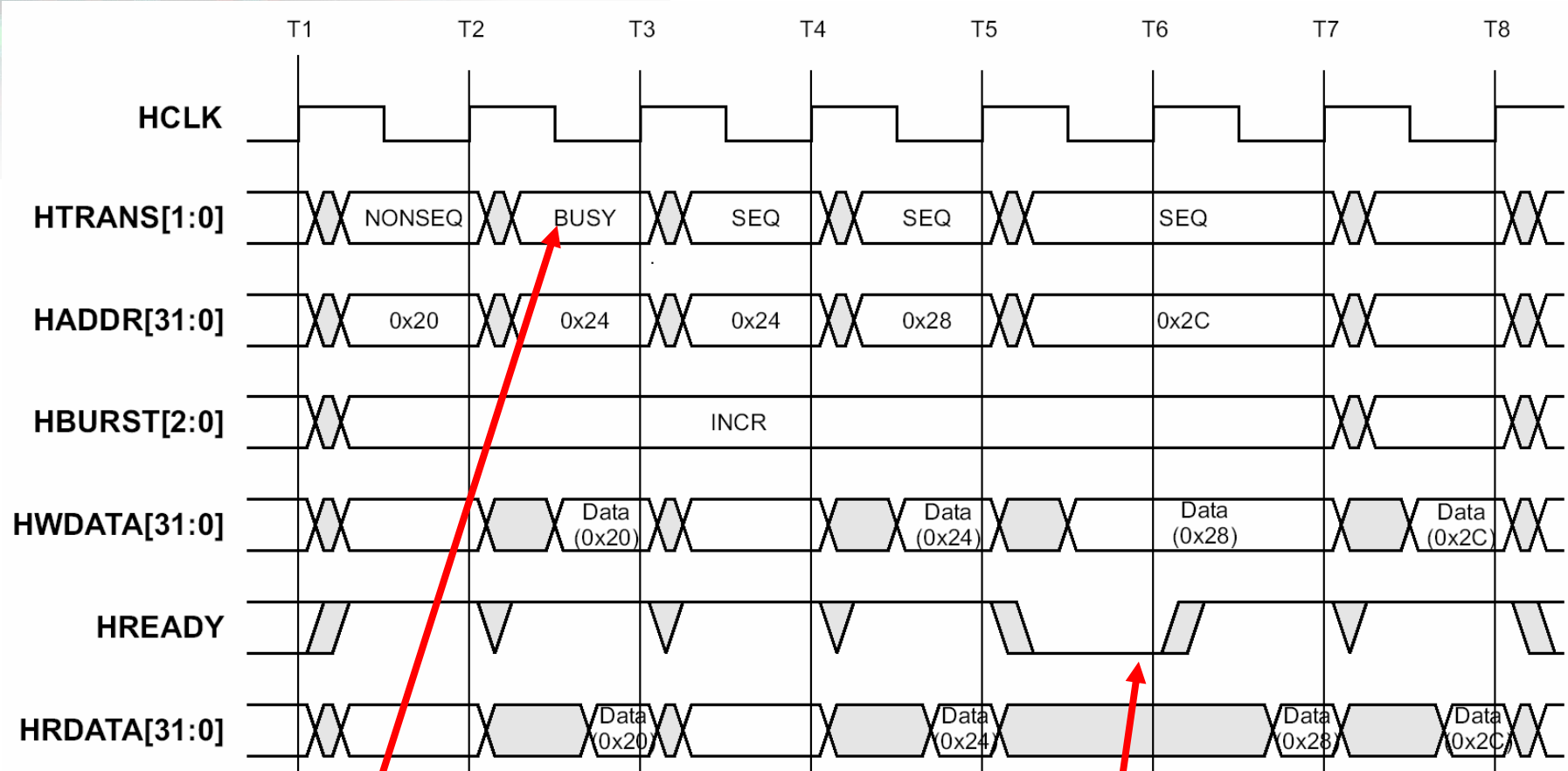
# Transfer Type

- HTRANS[1:0]: transfer type
  - Four types, IDLE, BUSY, NONSEQ,SEQ
- 00 : IDLE
  - No transfers
  - When master grant bus, but no transfer
- 01 : BUSY
  - Allow the master to insert IDLE cycle during transfers

# Transfer Type (cont.)

- 10 : NOSEQ
  - Indicate a single transfer
  - or the first transfer of a burst
  - The address & control signals are unrelated to the previous transfer
- 11 : SEQ
  - Indicate the following transfers
  - The address is related to the previous transfer

# Transfer Type Example



# AHB Control Signals

- HWRITE
  - High : write
  - Low : read

- HSIZE[2:0]
  - 000 : 8 bits
  - 001 : 16 bits
  - 010 : 32 bits
  - 011 : 64 bits
  - 100 : 128 bits
  - 101 : 256 bits
  - 110 : 512 bits
  - 111 : 1024 bits
  - The max is constrained by the bus configuration
  - 32 bits (010) is often used

# Burst Operation

- AHB burst operations
  - 4-beat, 8-beat, 16-beat, single transfer, and undefined-length transfer
  - Both incrementing & wrapping burst
- Incrementing burst
  - Sequential, the address is just the increment of the previous one
- Wrapping burst
  - If the start address is not aligned (size x beats), the address will wrap when the boundary is reached
  - Ex: 4-beat warping burst of word (4-byte): 0x34→0x38→0x3C→0x30

# Address Calculation Example

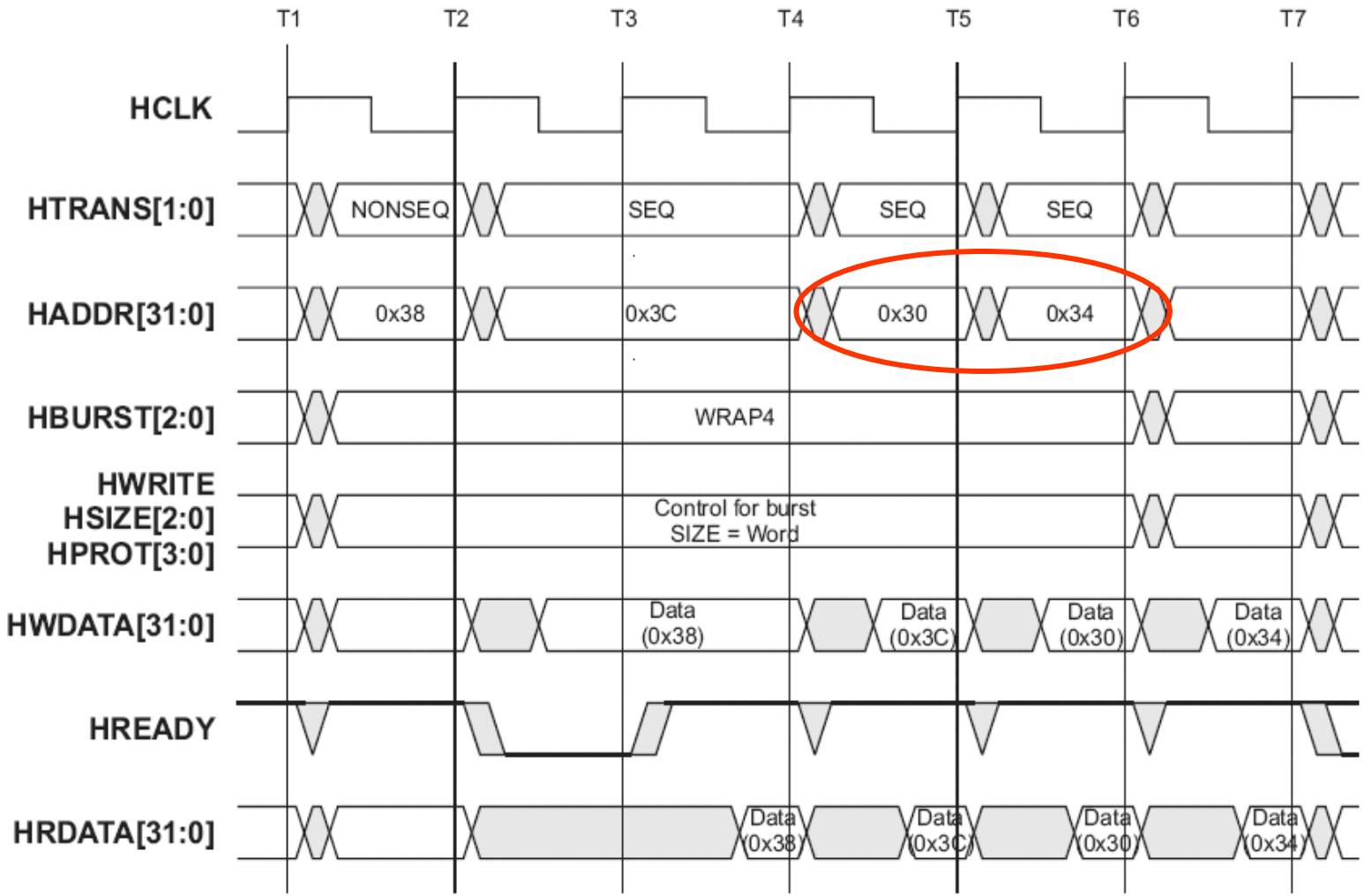
- The address calculation is according to HSIZE and HBURST
- Example: HSIZE = 010 (32 bits) with starting address = 0x48

| HBURST | Type | Address |
|---|---|---|
| 000 | SINGLE | 0x48 |
| 001 | INCR | 0x48, 0x4C, 0x50,… The most useful |
| 010 | WRAP4 | 0x48, 0x4C, 0x40, 0x44 |
| 011 | INCR4 | 0x48, 0x4C, 0x50, 0x54 |
| 100 | WRAP8 | 0x48, 0x4C, 0x50, 0x54, 0x58, 0x5c, 0x40, 0x44 |
| 101 | INCR8 | 0x48, 0x4C, 0x50, 0x54, 0x58, 0x5c, 0x60, 0x64 |
| 110 | WRAP16 | 0x48, 0x4C,…, 0x7c, 0x40, 0x44 |
| 111 | INCR16 | 0x48, 0x4C,…, 0x7c, 0x80, 0x84 |

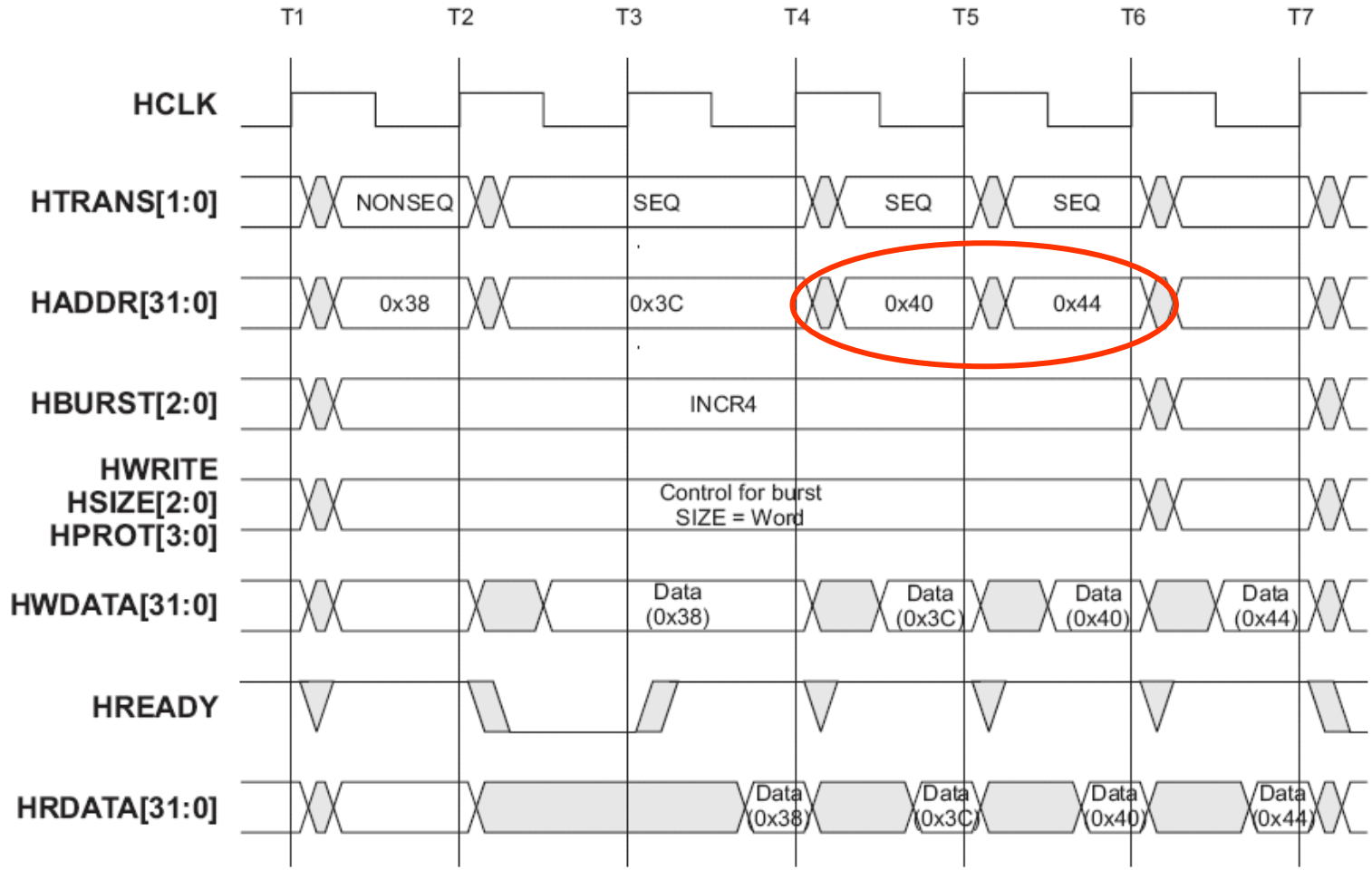
# Important!!

- Burst transfer can't cross the 1K boundary
  - Because the minimal address range for a slave is 1 KB
  - NONSEQ → SEQ → 1KB Boundary → NONSEQ → SEQ…
- The master do not attempt to start a fixed-length incrementing burst which would cause this boundary to be crossed

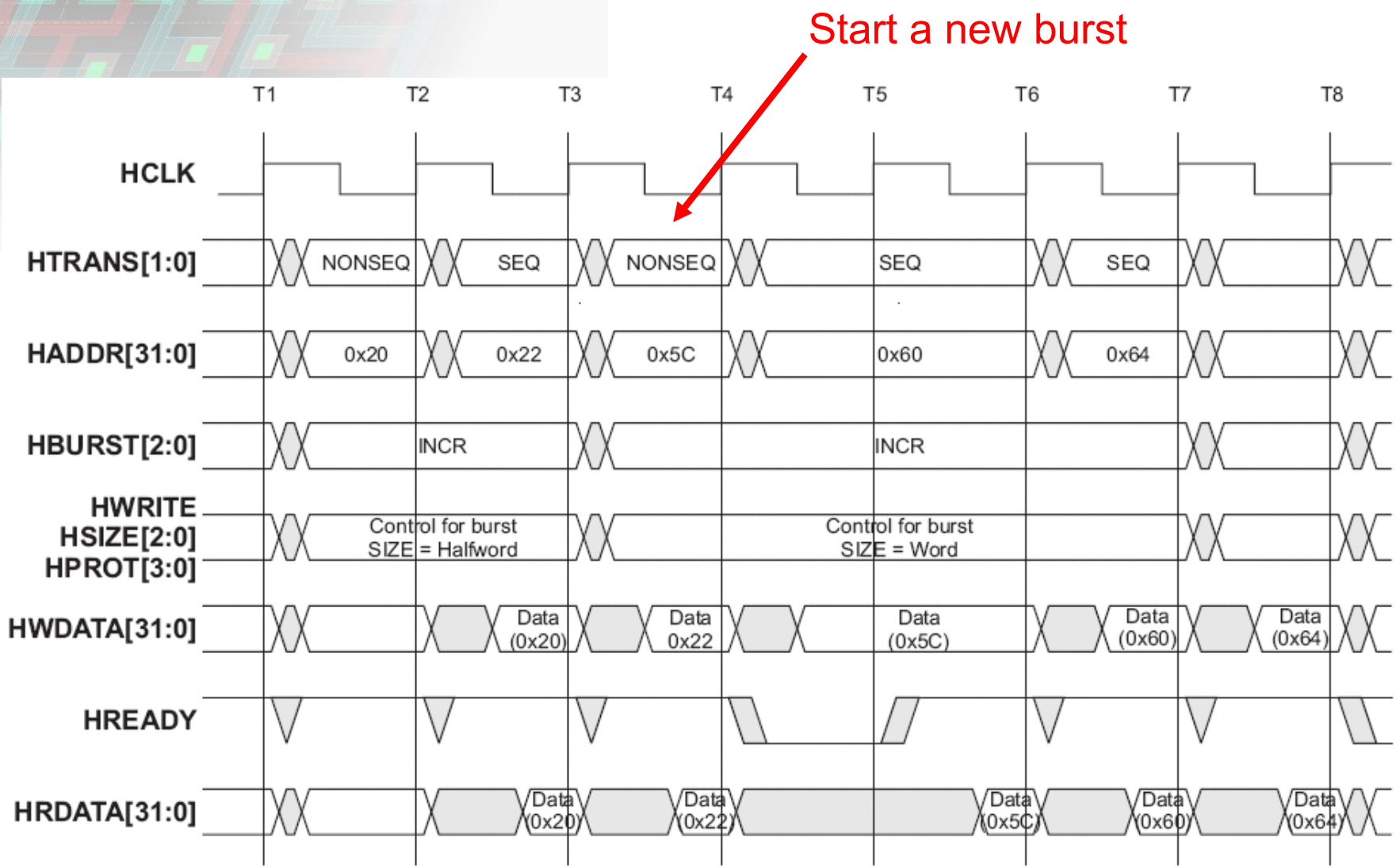
# Example: 4-Beat Wrapping Burst



# Example: 4-Beat Incrementing Burst



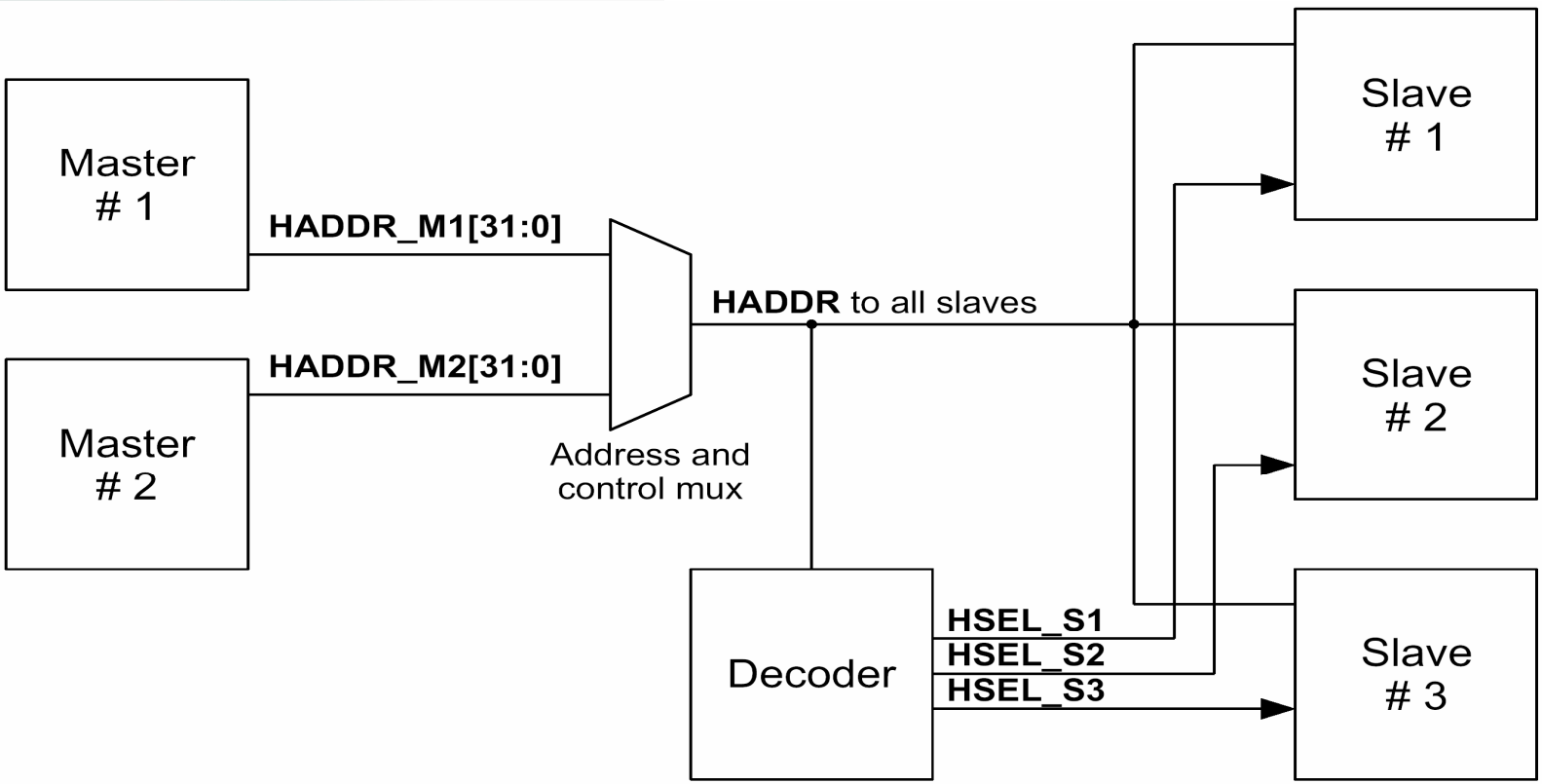
# Example: Undefined-Length Burst



# Address Decoding

- HSELx : slave select
  - Indicate the slave is selected by a master
- A central address decoder is used to provide the select signal
- A slave should occupy at least 1KB of memory space
- An additional default slave is used to fill up the memory map

# Address Decoding (cont.)



# Slave Response

- The slave accessed must respond the transfer
- The slave may
  - Complete the transfer
  - Insert wait state
  - Signal an error to indicate the transfer failure
  - Delay the transfer, leave the bus available for other transfer (split)

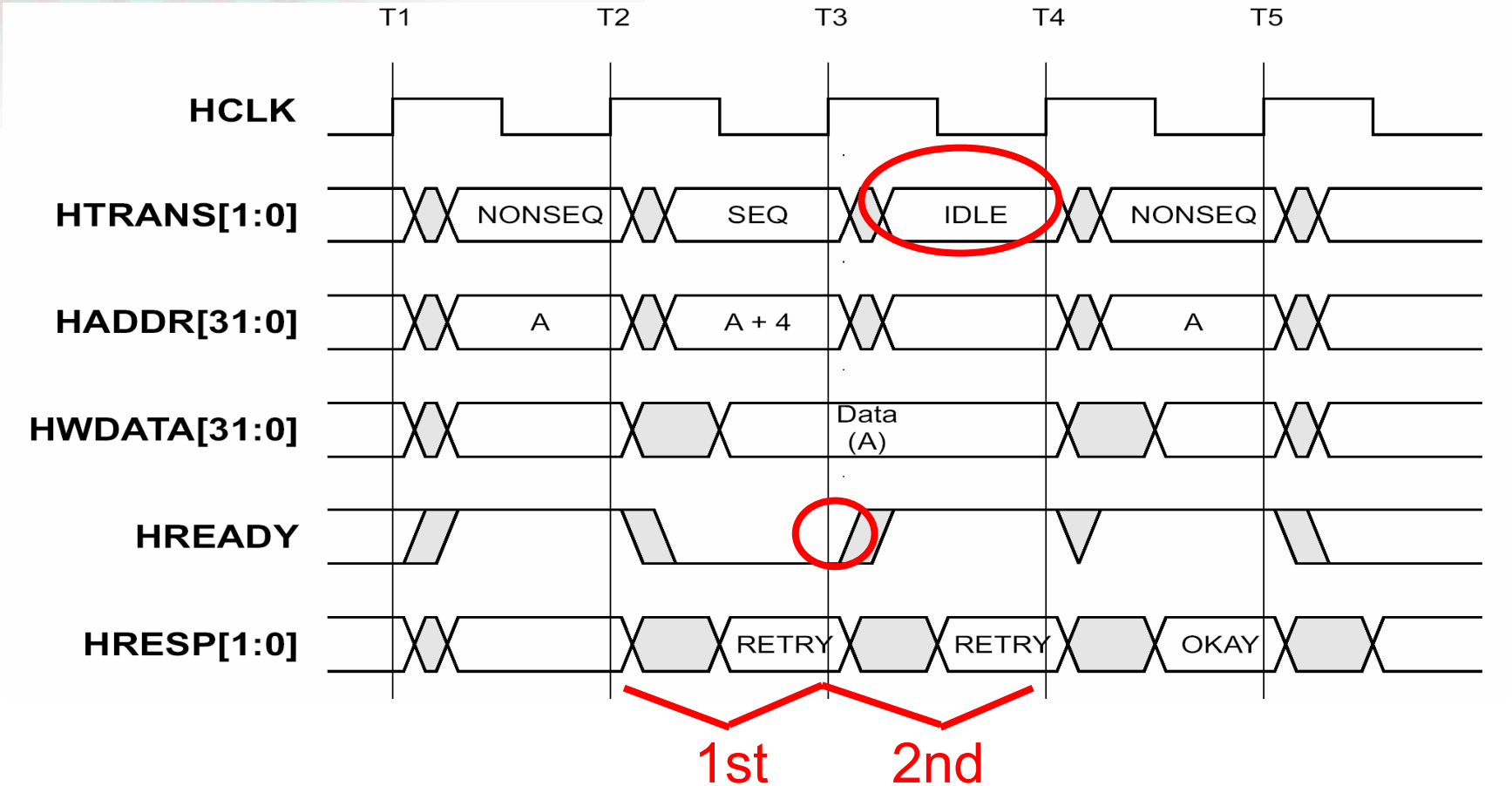
# Slave Response Signals

- HREADY : transfer done
- HRESP[1:0] : transfer response
- 00 : OKAY
  - Successful
- 01 : ERROR
  - Error
- 10 : RETRY
  - The transfer is not completed
  - Ask the master to perform a retry transfer
- 11 : SPLIT
  - The transfer is not completed
  - Ask the master to perform a split transfer

# Two-cycle Response

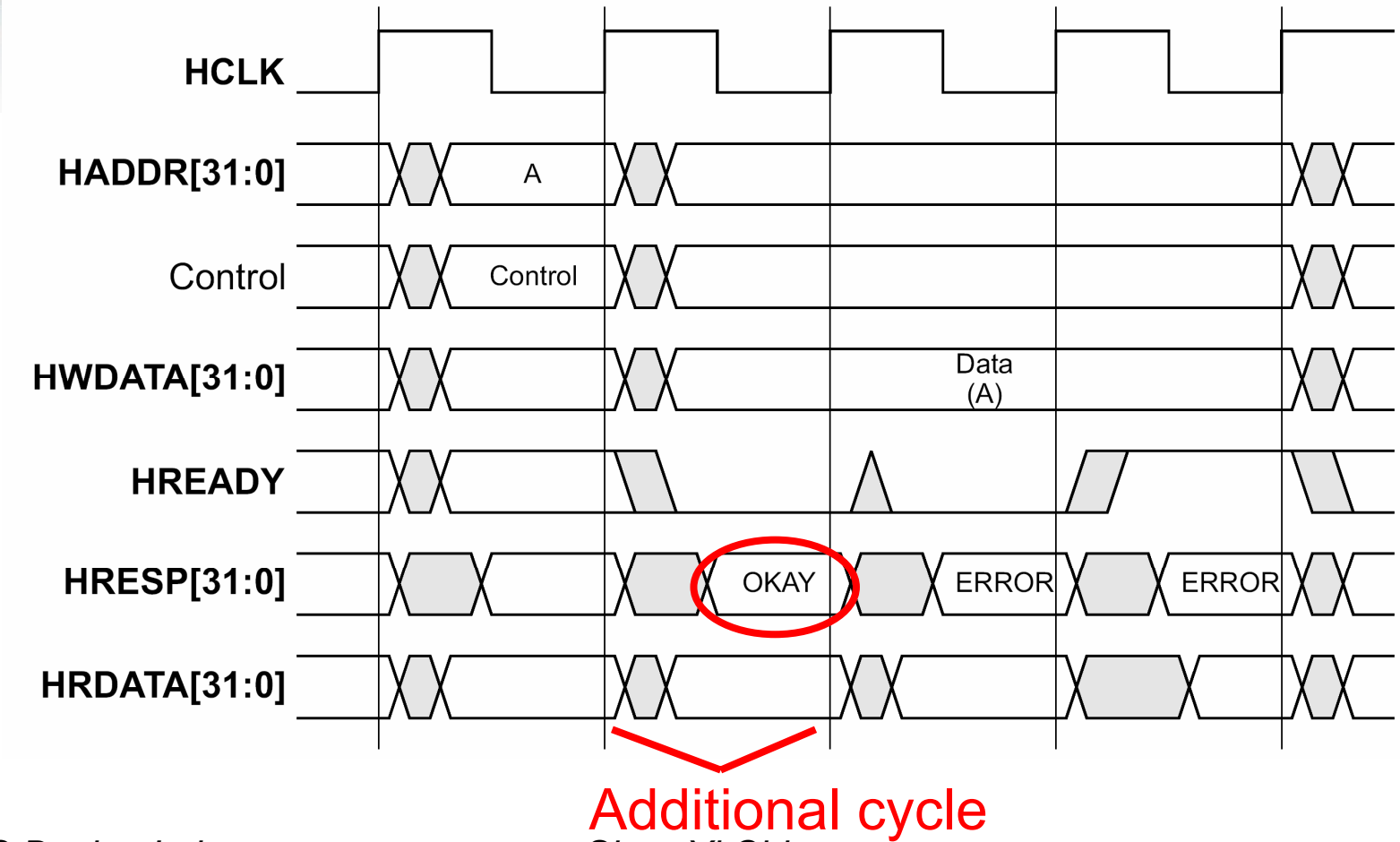
- HRESP[1:0]
  - OKAY: single cycle response
  - ERROR : two-cycle response
  - RETRY : two-cycle response
  - SPLIT : two-cycle response
- Two-cycle response is required because of the pipeline nature of the bus. This allows sufficient time for the master to handle the next transfer

# Retry Response Example



# ERROR Response Example

- An error response which needs three cycles



# Different Between Retry and Split

- The major difference is the way of arbitration
  - RETRY: the arbiter will continue to use the normal priority
  - SPLIT: the arbiter will adjust the priority scheme so that any other master requesting the bus will get access
    - Requires extra complexity in both the slave and the arbiter
- A bus master should treat RETRY and SPLIT in the same manner

# Data Bus

- Non-tri-state, separate read & write buses
- Endianness
  - Not specified in the AMBA spec.
  - **All the masters and slaves should of the same endianness**
  - Dynamic endianness is not supported
- For IP design, only IPs which will be used in wide variety of applications should be made bi-endian

# Active Bytes Lanes for a 32-bit Little-Endian Data Bus

| Transfer size | Address offset | DATA [31:24] | DATA [23:16] | DATA [15:8] | DATA [7:0] |
|---|---|---|---|---|---|
| Word | 0 | ✓ | ✓ | ✓ | ✓ |
| Halfword | 0 | - | - | ✓ | ✓ |
| Halfword | 2 | ✓ | ✓ | - | - |
| Byte | 0 | - | - | - | ✓ |
| Byte | 1 | - | - | ✓ | - |
| Byte | 2 | - | ✓ | - | - |
| Byte | 3 | ✓ | - | - | - |

# Active Bytes Lanes for a 32-bit Big-Endian Data Bus

| Transfer size | Address offset | DATA [31:24] | DATA [23:16] | DATA [15:8] | DATA [7:0] |
|---|---|---|---|---|---|
| Word | 0 | ✓ | ✓ | ✓ | ✓ |
| Halfword | 0 | ✓ | ✓ | - | - |
| Halfword | 2 | - | - | ✓ | ✓ |
| Byte | 0 | ✓ | - | - | - |
| Byte | 1 | - | ✓ | - | - |
| Byte | 2 | - | - | ✓ | - |
| Byte | 3 | - | - | - | ✓ |

# AHB Arbitration Signals

| Name | Source | Description |
|---|---|---|
| HBUSREQx | Master | Bus request |
| HLOCKx | Master | Locked transfers |
| HGRANTx | Arbiter | Bus grant |
| HMASTER[3:0] | Arbiter | Master number |
| HMASTLOCK | Arbiter | Locked sequence |
| HSPLITx[15:0] | Slave (SPLIT-capable) | Split completion request |

# Arbitration Signals (cont.)

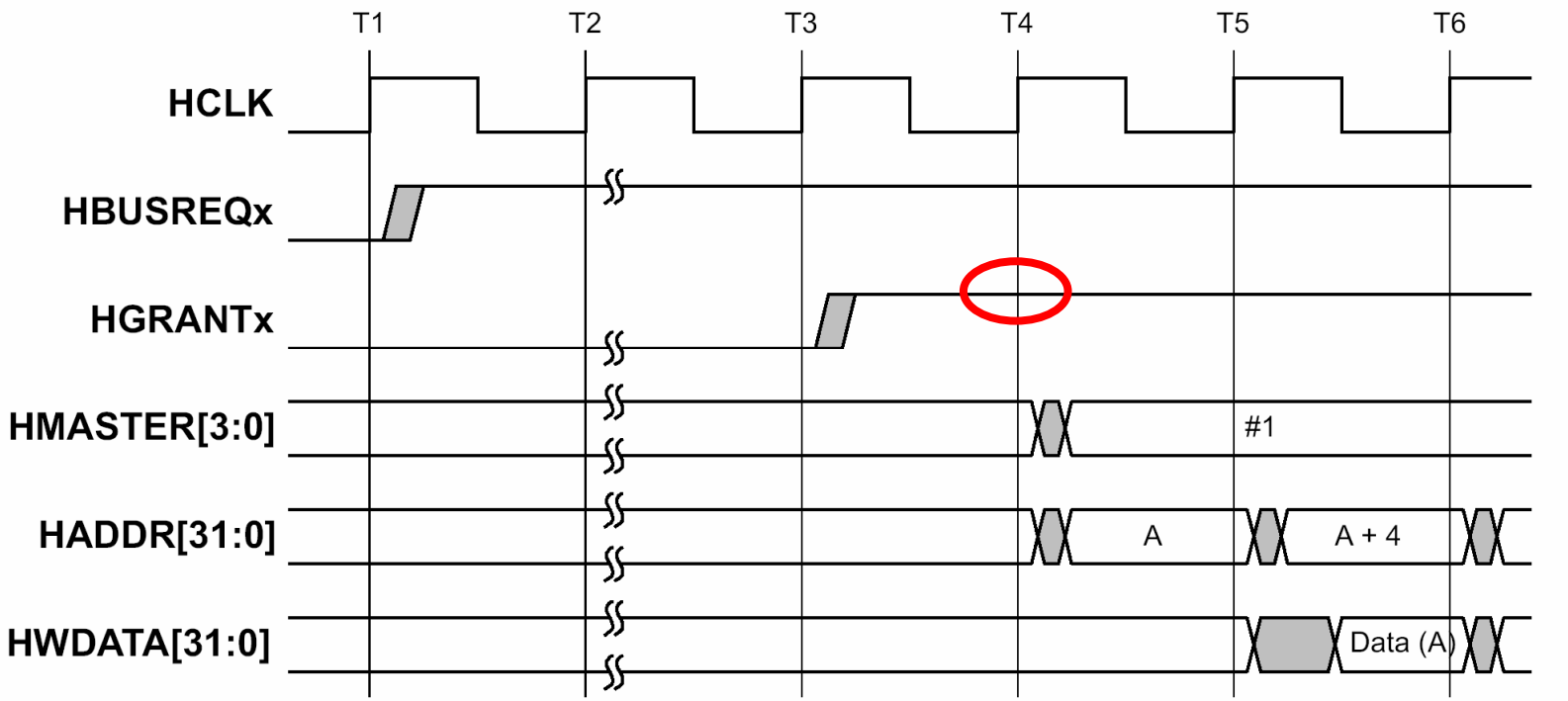
- HBUSREQ
  - Bus request
- HLOCKx :
  - High: the master requires locked access to the bus
- HGRANTx
  - Indicate the master x  accessible to the bus
  - Master x gains ownership: HGRANTx=1 & HREADY=1

# Arbitration Signals (cont.)

- HMASTER[3:0]
  - Indicate which master is transferring, information for splitting
- HMASTLOCK
  - Indicate the master is performing a locked transfer
- HSPLITx[15:0]
  - Used by the slave to indicate the arbiter which master should be allowed to re-attempt a split transaction
  - Each bit corresponds to a single master

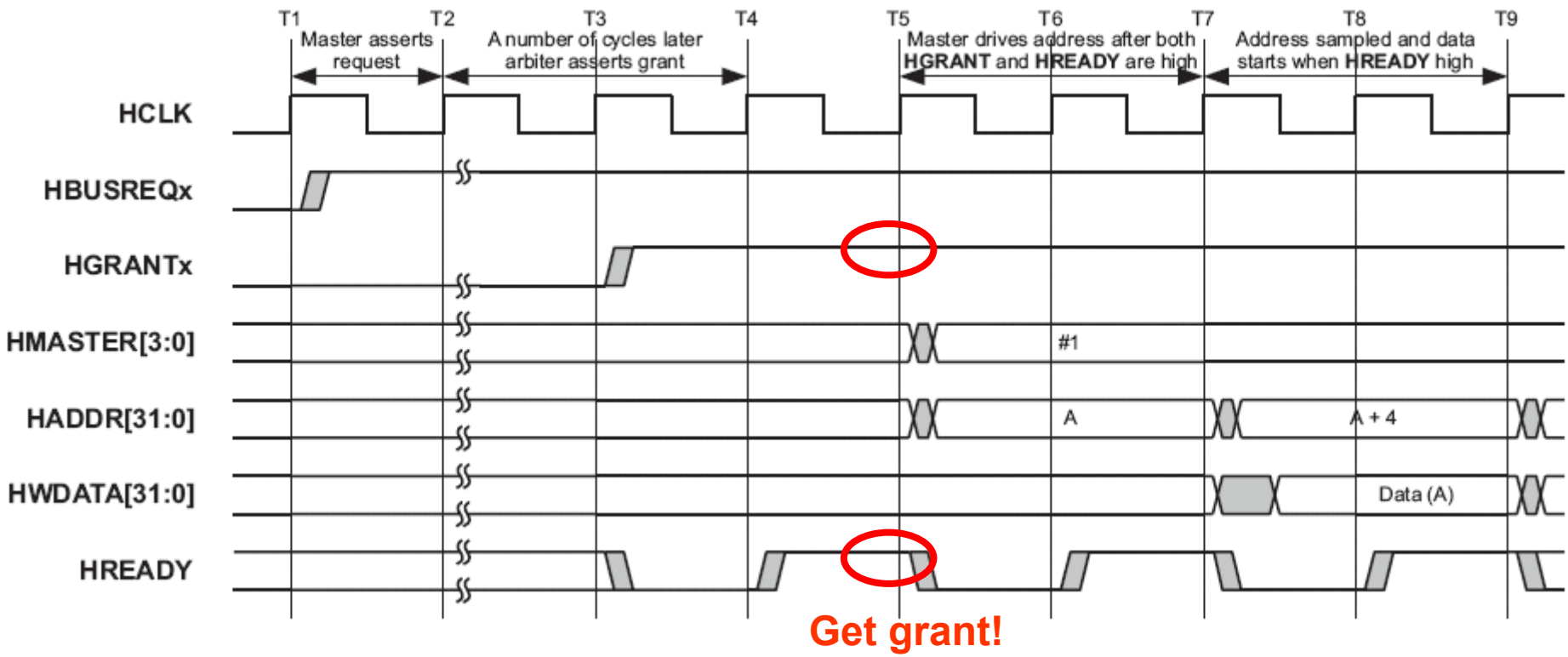
# Arbitration Example (1)

- Granting access with no wait state



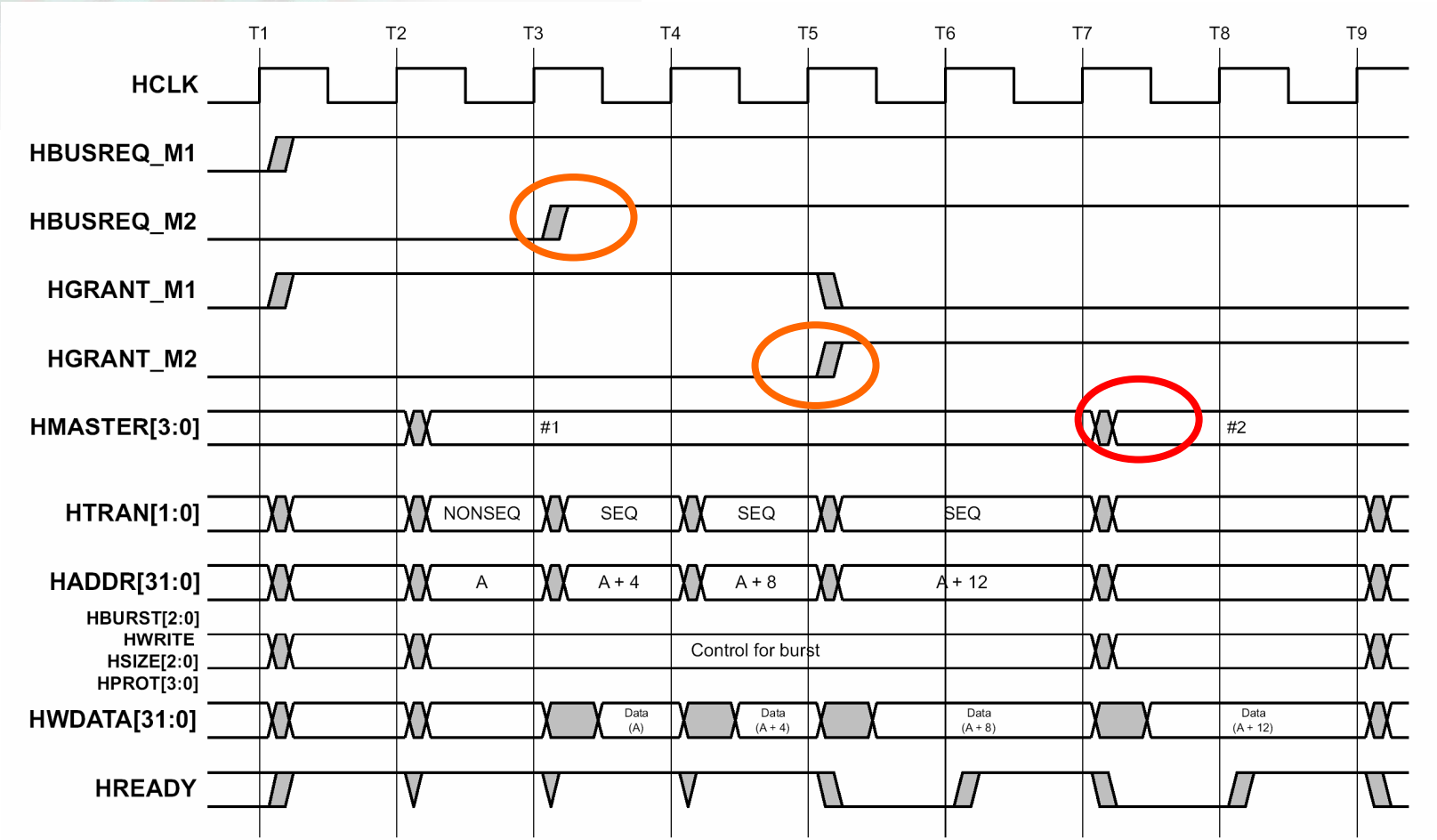
# Arbitration Example (2)

- Granting access with wait states

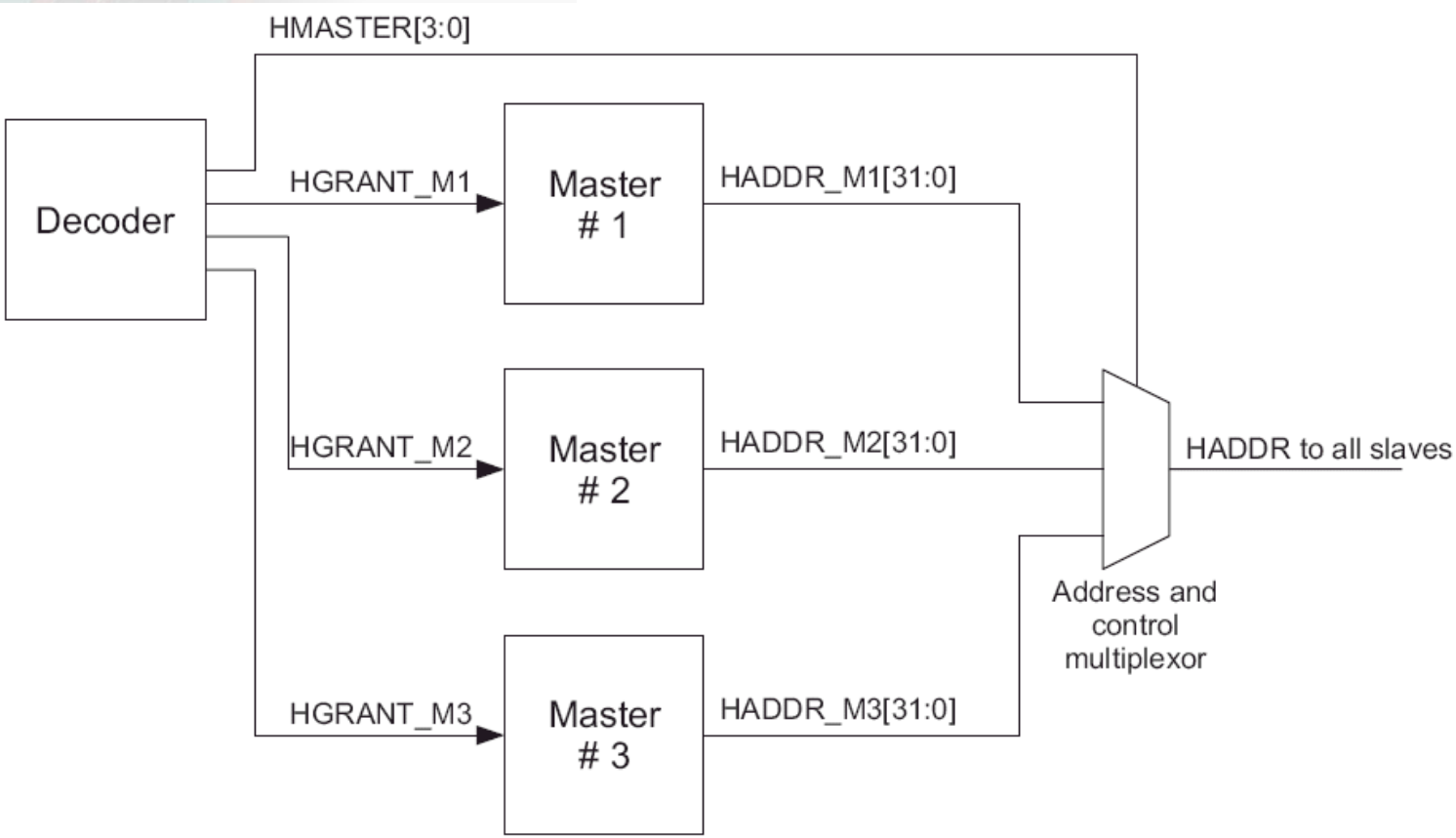


# Arbitration Example (3)

- Handover after burst



# Bus Master Grant Signals



# Notes

- For a fixed length burst, it is not necessary to continue request the bus
- For a undefined length burst, the master should continue to assert the request until it has started the last transfer
- If no master requests the bus, grant to the default master with HTRANS=IDLE
- It is recommended that the master inserts an IDLE transfer after any locked sequence to provide the opportunity for changing arbitration

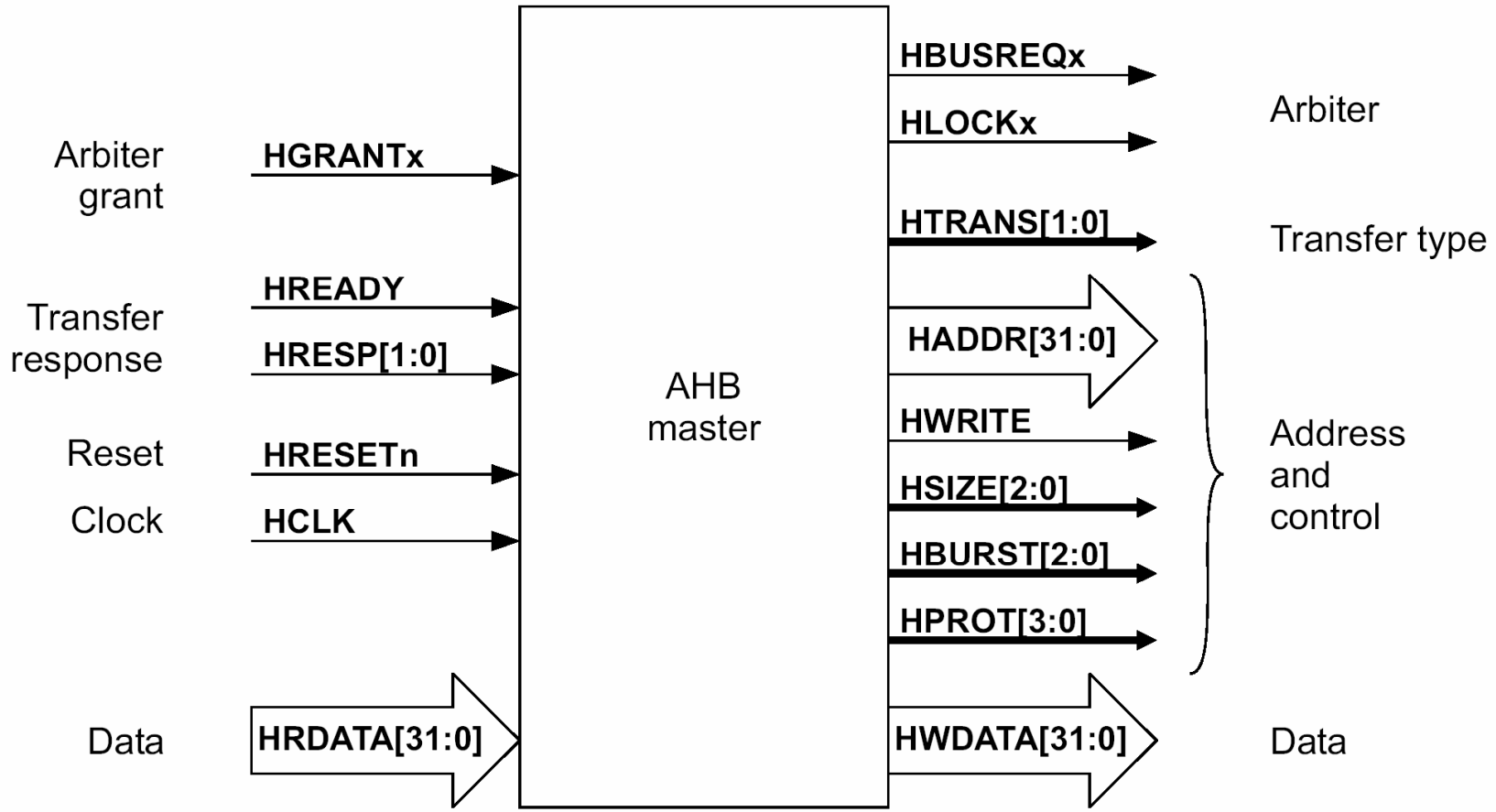
# Split Transfer Sequence

- The master starts the transfer.
- If the slave decides that it may take a number of cycles to obtain the data, it **gives a SPLIT transfer response**. The slave **record the master number**, HMASTER. Then the arbiter **change the priority** of the masters.
- The arbiter grants other masters, bus master handover.
- When the slave is ready to complete the transfer, it asserts the appropriate bit of the HSPLITx bus to the arbiter.
- The arbiter restores the priority
- The arbiter will grant the master so it can re-attempt the transfer
- Finish

# Preventing Deadlock

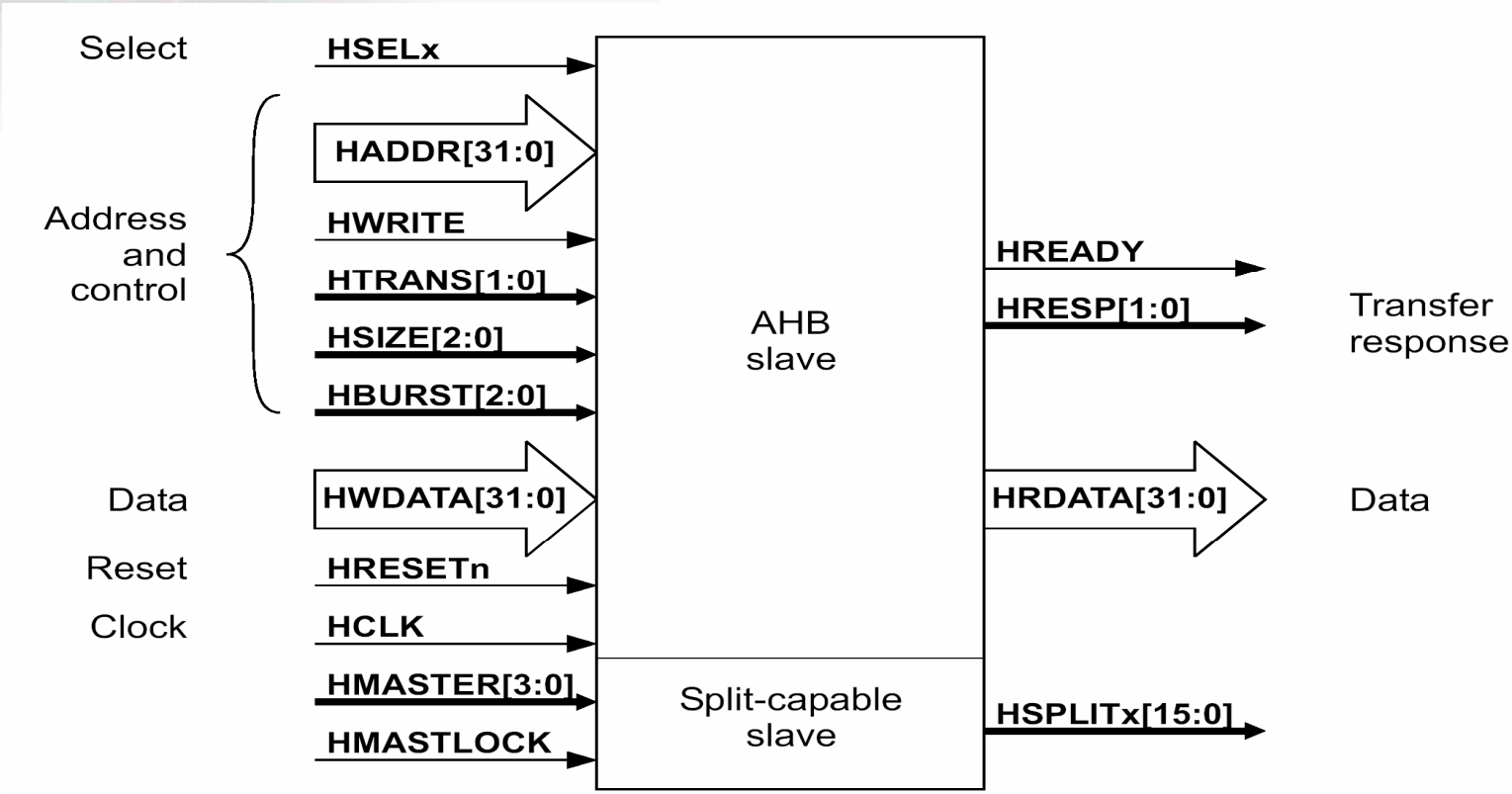
- It is possible for deadlock if a number of different masters attempt to access a slave which issues SPLIT or RETRY
- **The slave can withstand a request from every master in the system, up to a maximum of 16.** It only needs to record the master number. (can ignore address and control)
- **A slave which issues RETRY responses must only be accessed by one master at a time.**
  - Some hardware protection mechanisms, such as ERROR message, can be used.

# AHB Master Interface

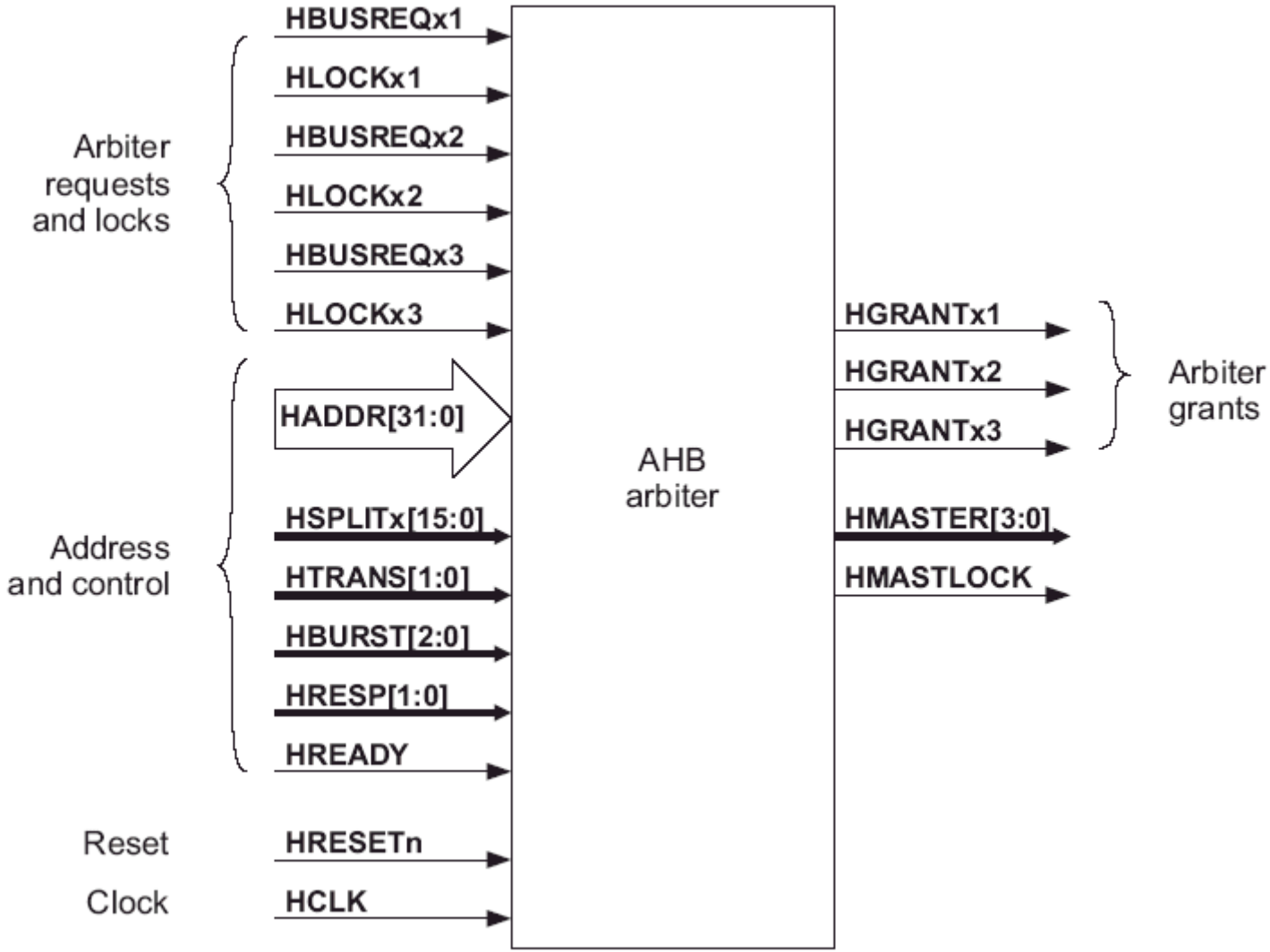


# AHB Slave Interface

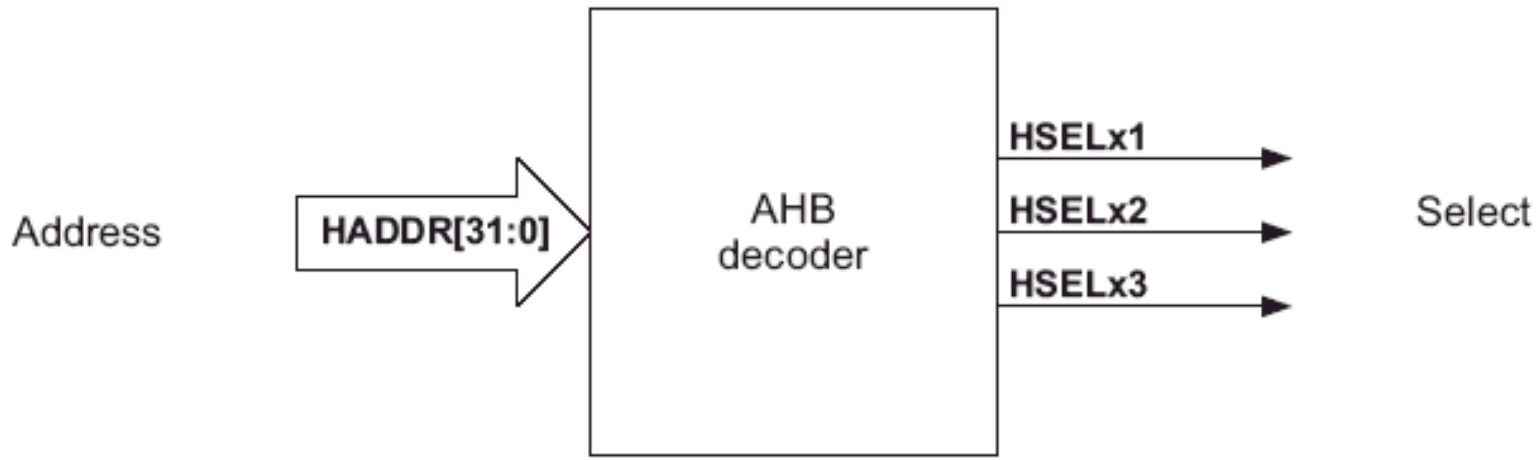
HREADY_IN →



# AHB Arbiter


HBUSREQx1
HLOCKx1
HBUSREQx2
HLOCKx2
HBUSREQx3
HLOCKx3
Arbiter requests and locks
HADDR[31:0]
HSPLITx[15:0]
HTRANS[1:0]
HBURST[2:0]
HRESP[1:0]
HREADY
Address and control
HRESETn
Reset
HCLK
Clock
AHB arbiter
HGRANTx1
HGRANTx2
HGRANTx3
Arbiter grants
HMASTER[3:0]
HMASTLOCK

# AHB Decoder

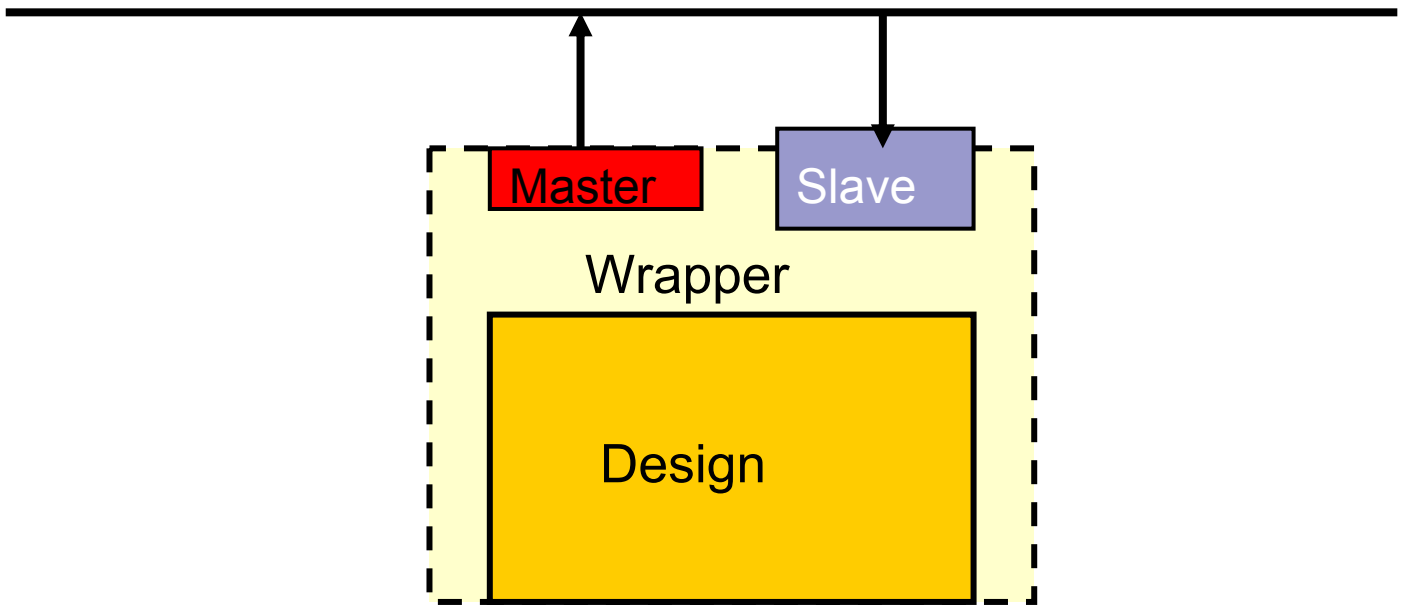


# Review on AHB

- Main components
  - Master, slaves, arbiter, decoder
- How the transfer progress
  - The pipelined scheme
- How to increase the performance
  - Burst read/write
- Arbitration
  - Bus ownership handover

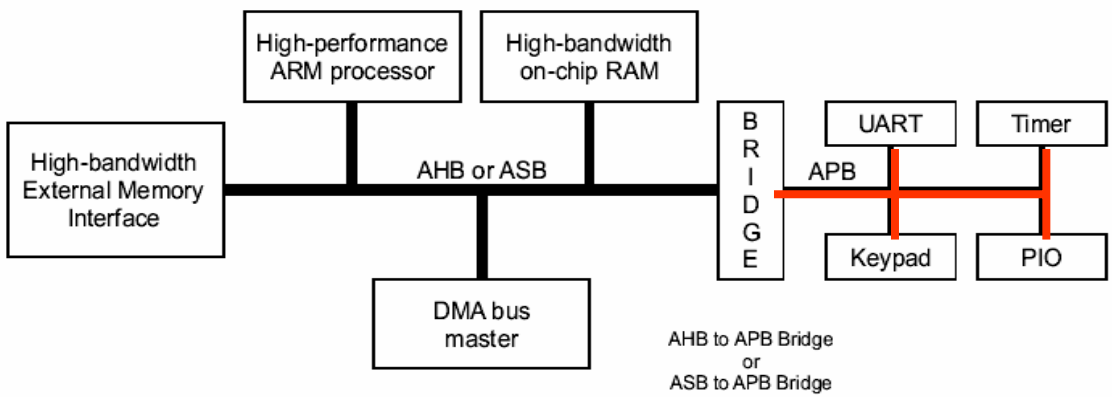
# Implementation

- Both master port & slave port in your design
  - Read/write data via master port
  - Configured by the others via slave port
  - Such as some parameters set by the processor



# Outline

- Overview
- AHB
- APB
- Test methodology



# APB Signals

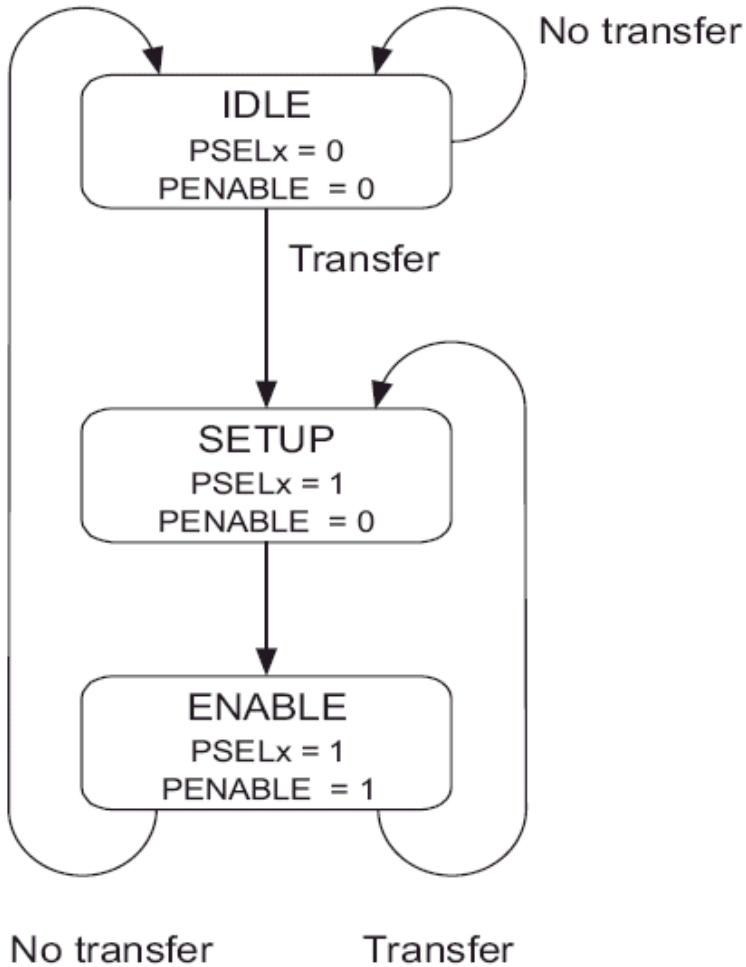
| Name | Description |
|---|---|
| PCLK | Bus clock |
| PRESETn | APB reset |
| PADDR[31:0] | APB address bus |
| PSELx | APB select |
| PENABLE | APB strobe |
| PWRITE | APB transfer direction |
| PRDATA | APB read data bus |
| PWDATA | APB write data bus |

# APB Signals (cont.)

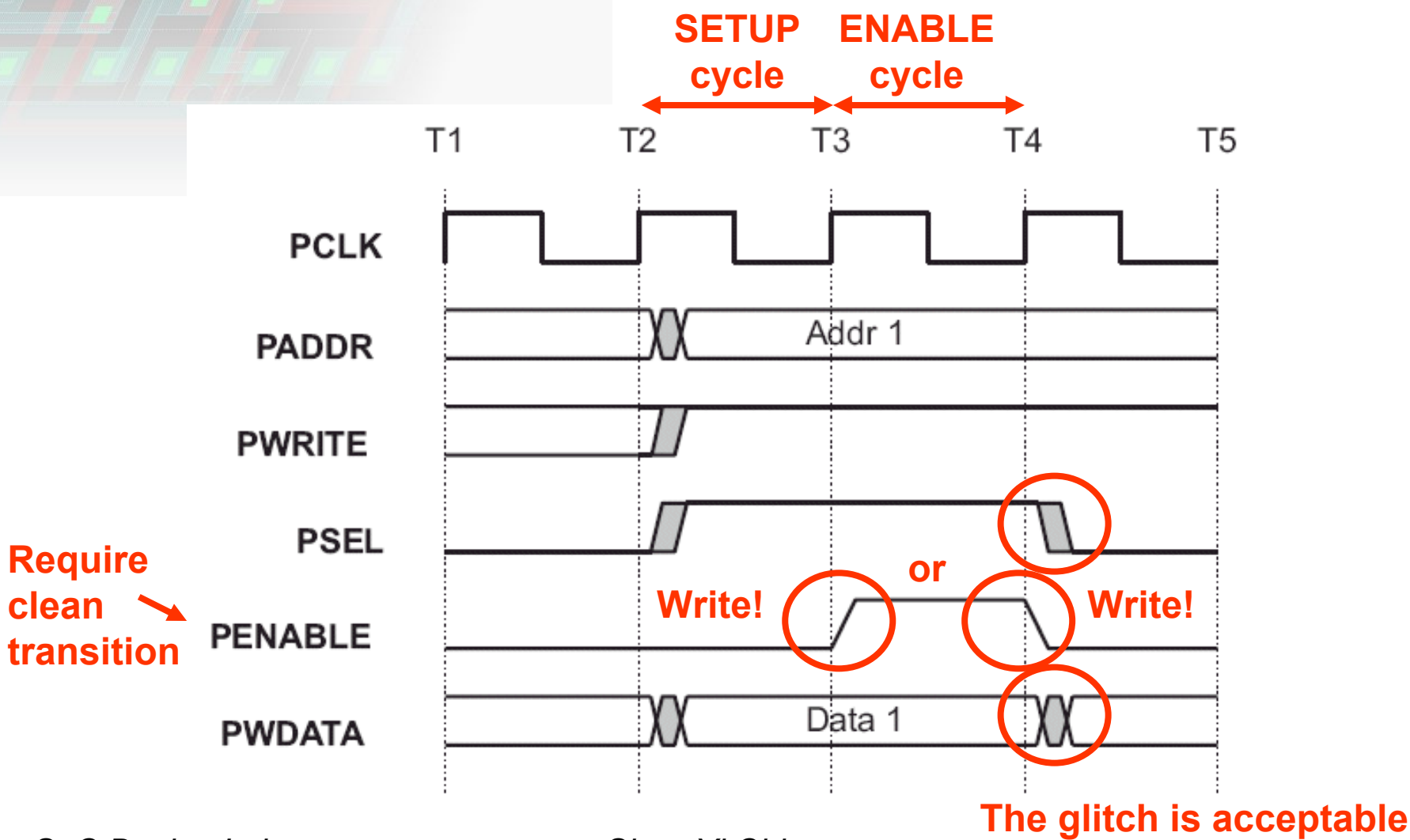
- PADDR[31:0]
  - Address bus, driven by the peripheral bus bridge unit
- PSELx
  - A signal from the secondary decoder to each peripheral bus slave x
- PENABLE
  - Used to time all accesses on the peripheral bus
- PWRITE
  - High: write
  - Low: read
- PRDATA and PWDATA
  - Up to 32-bits wide

# Stage Diagram

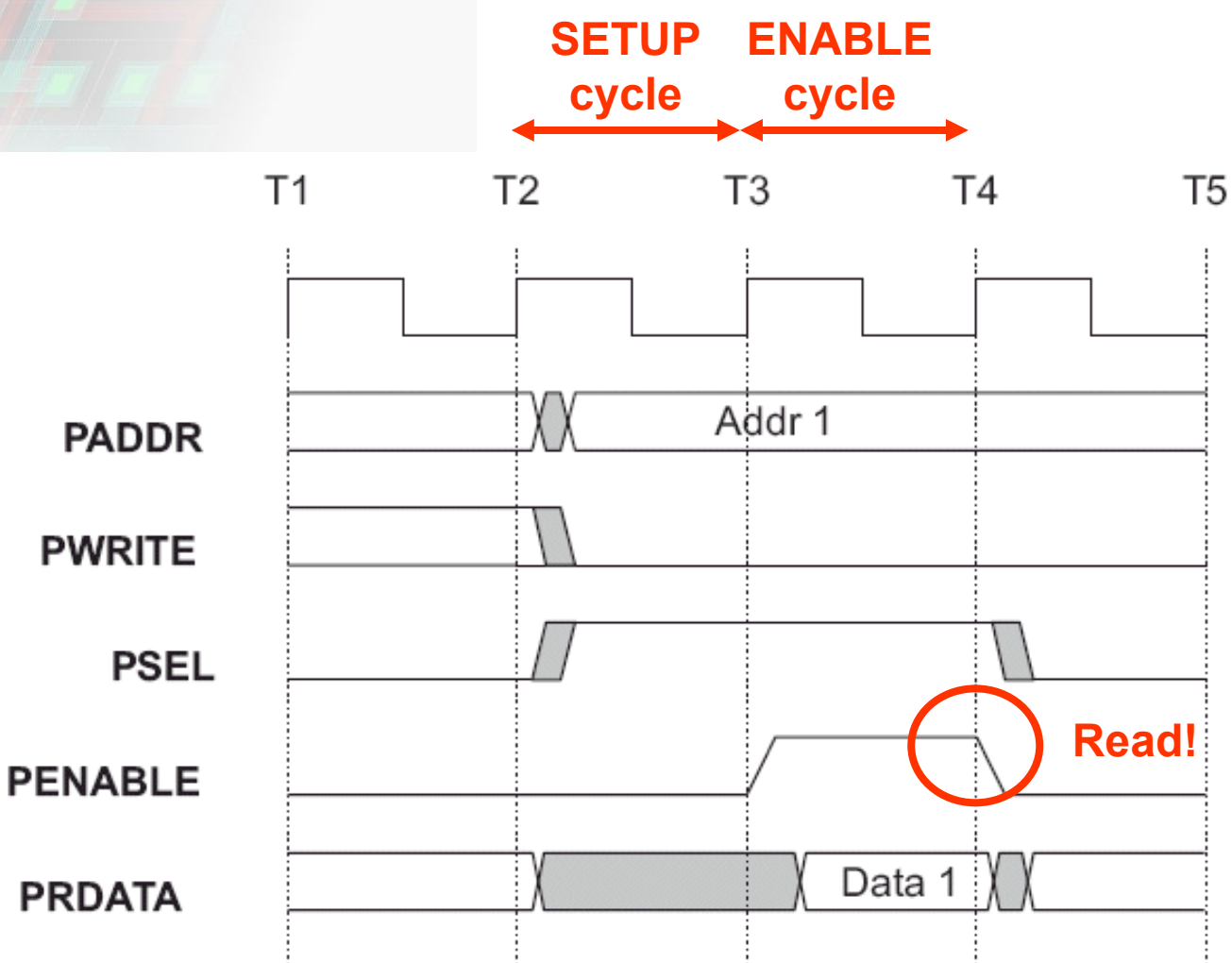
- SETUP and ENABLE state: last for only one cycle



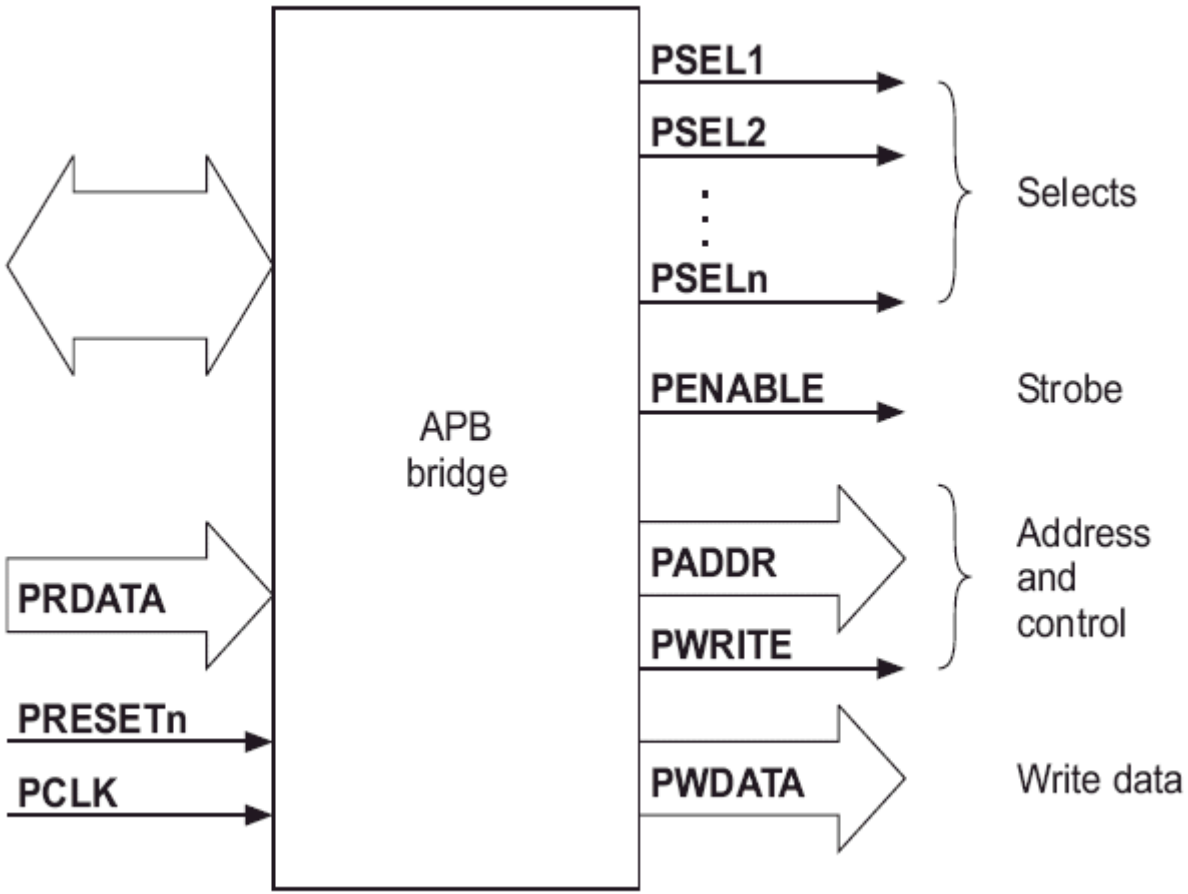
# Write Transfer



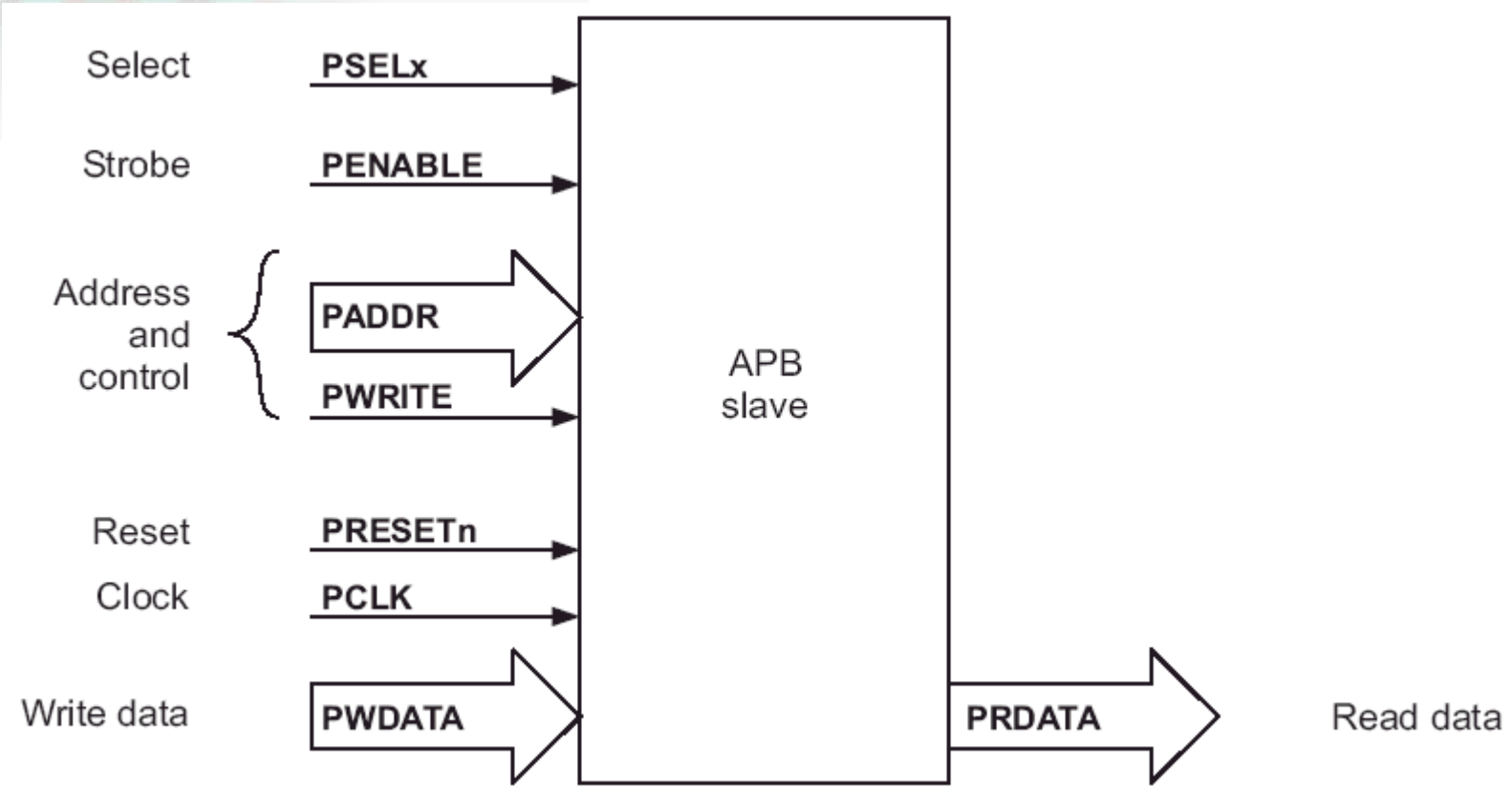
# Read Transfer



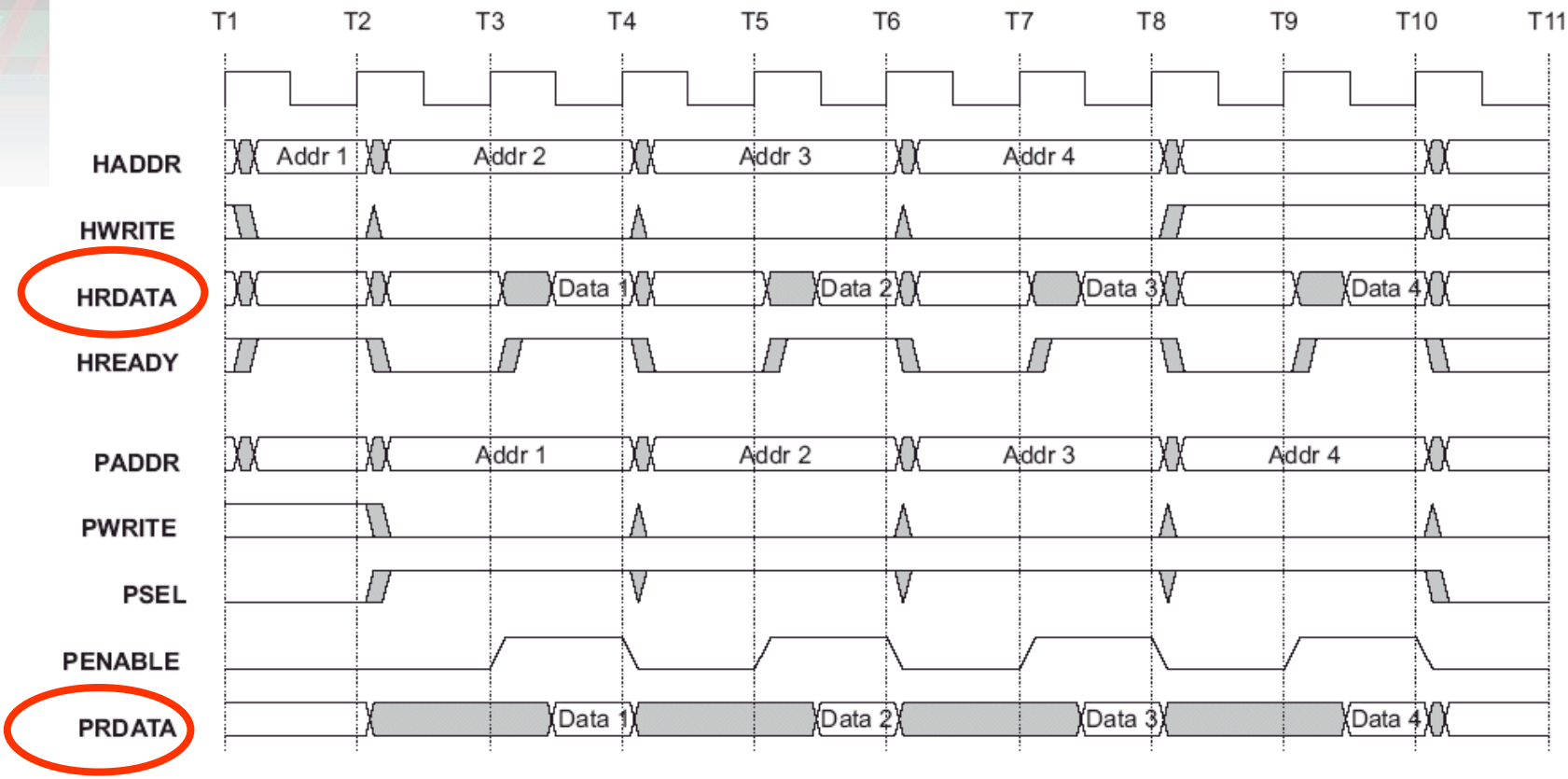
# APB Bridge


PSEL1
PSEL2
PSELn
Selects
PENABLE
Strobe
APB
bridge
PADDR
PWRITE
Address
and
control
PRDATA
PRESETn
PCLK
PWDATA
Write data

# APB Slave

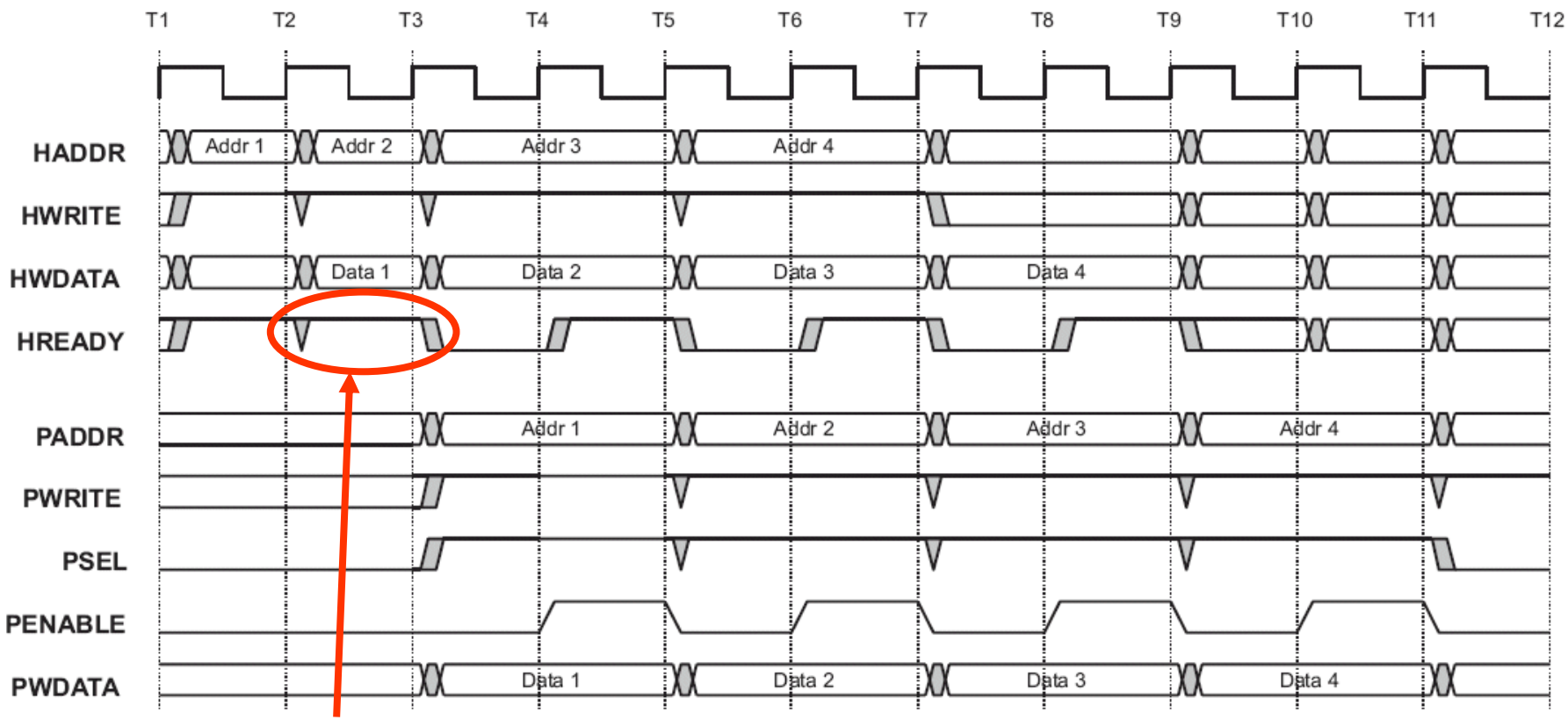

Select
PSELx
Strobe
PENABLE
Address and control
PADDR
PWRITE
Reset
PRESETn
Clock
PCLK
Write data
PWDATA
APB slave
PRDATA
Read data

# Interfacing APB to AHB – Read



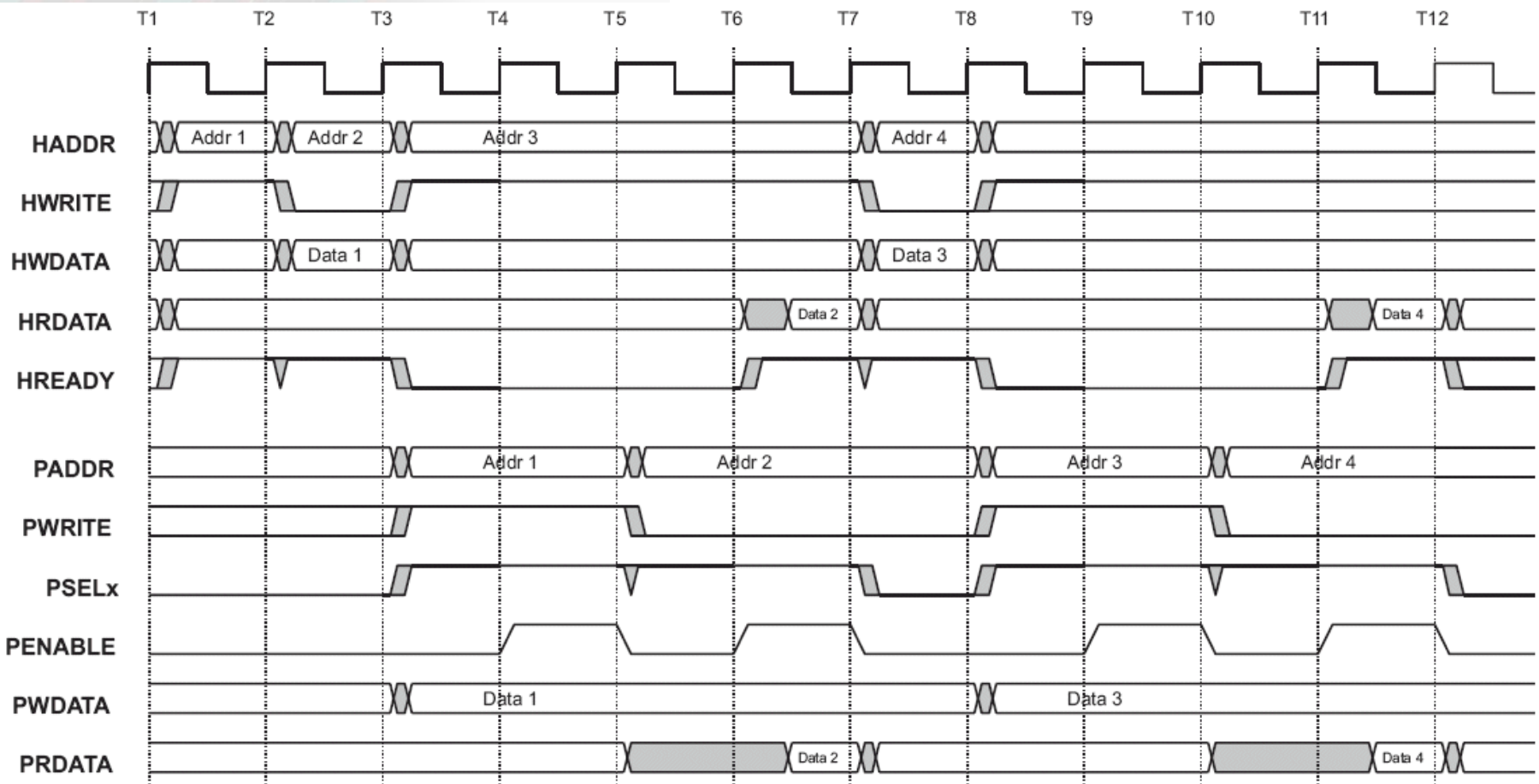

**PRDATA can be directly routed back to HRDATA,**
**or registering the read data at the end of enable cycle then drive back allows the AHB to run at a higher clock**

# Interfacing APB to AHB – Write




**The first write transfer can occur with zero wait stages**

# Interfacing APB to AHB – Back to Back Transfers

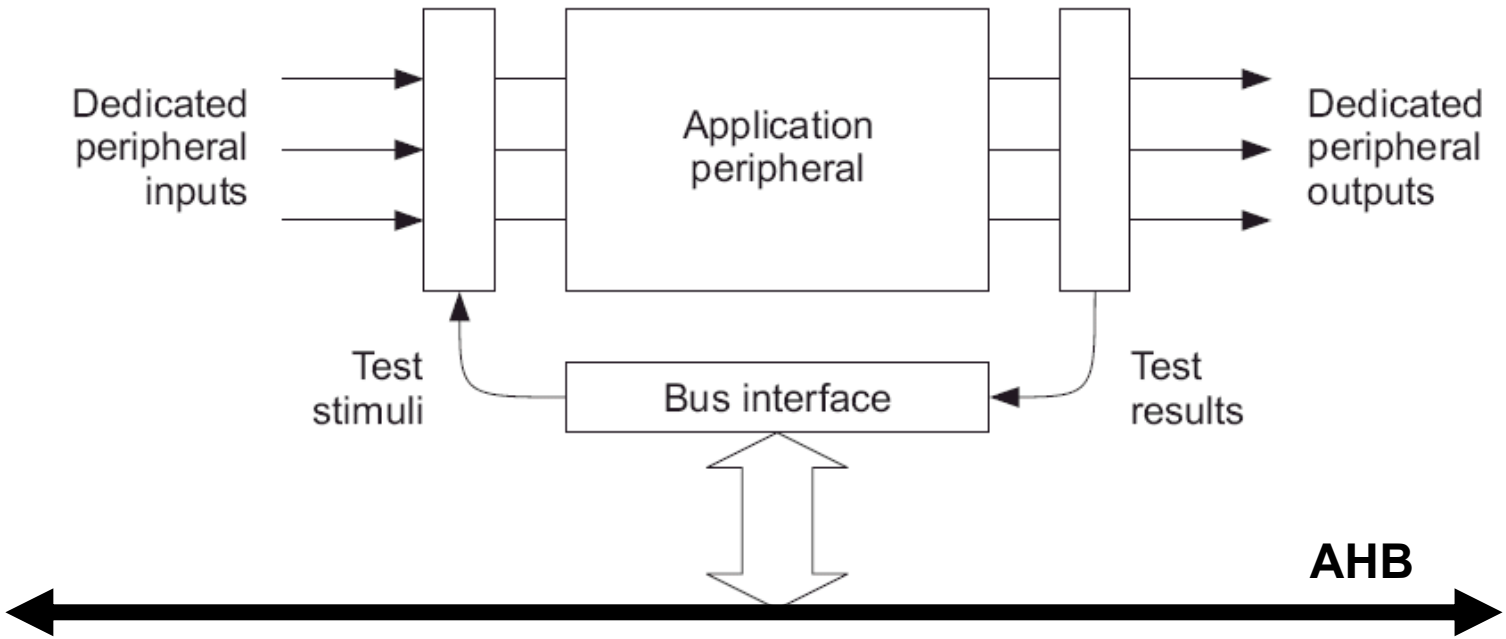


# Outline

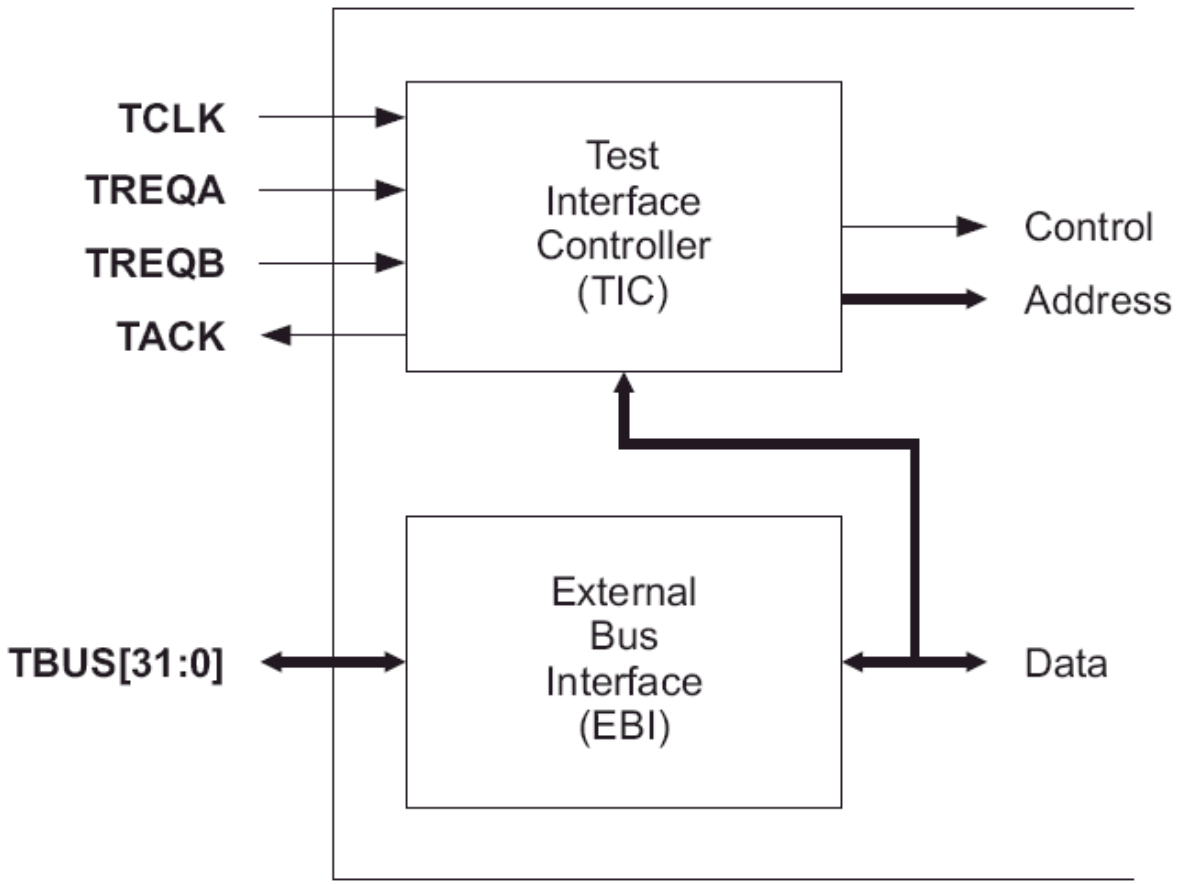
- Overview
- AHB
- APB
- Test methodology

# AMBA Test Interface

- With the test interface, each module can be tested only using transfers from the bus



# AMBA Test Interface



# Signals for Test Interface

- TCLK
  - Test clock
  - Assigned HCLK during test mode
- TREQA
  - Test burst request A, is required as a dedicated device pin
  - Used to request entry into the test mode
- TREQB
  - TREQA/TREQA can indicate the type of test vector
- TACK
  - Test acknowledge, required as a dedicated device pin
  - TREQA and TREQB signals are only sampled by the TIC when TACK is HIGH

# Control Signals

- During normal operation

| TREQA | TREQB | TACK | Description |
|---|---|---|---|
| 0 | 0 | 0 | Normal operation |
| 1 | 0 | 0 | Enter test mode request |
| 0 | 1 | 0 | Reserved |
| - | - | 1 | Test mode entered |

# Control Signals

- During test operation

| TREQA | TREQB | TACK | Description |
|---|---|---|---|
| - | - | 0 | Current access incomplete |
| 1 | 1 | 1 | Address vector, control vector, or turnaround vector (to switch the direction of TBUS) |
| 1 | 0 | 1 | Write vector |
| 0 | 1 | 1 | Read vector |
| 0 | 0 | 1 | Exit test mode |

# How to Distinguish Address, Control, and Turnaround Vectors?

- Single address/control vector is **address vector**
- Burst of address/control vectors are all **address vectors**, but the last one is **control vector**
- A read vector, or burst of read vectors, is always followed by a **turnaround vector** (AHB requires two turnaround vectors)

# Incremental Addressing

- The TIC may support incrementing of the bus address
- A typical implementation would use an 8-bit address incrementer, up to 1kB boundary
- Only supports undefined-length transfer

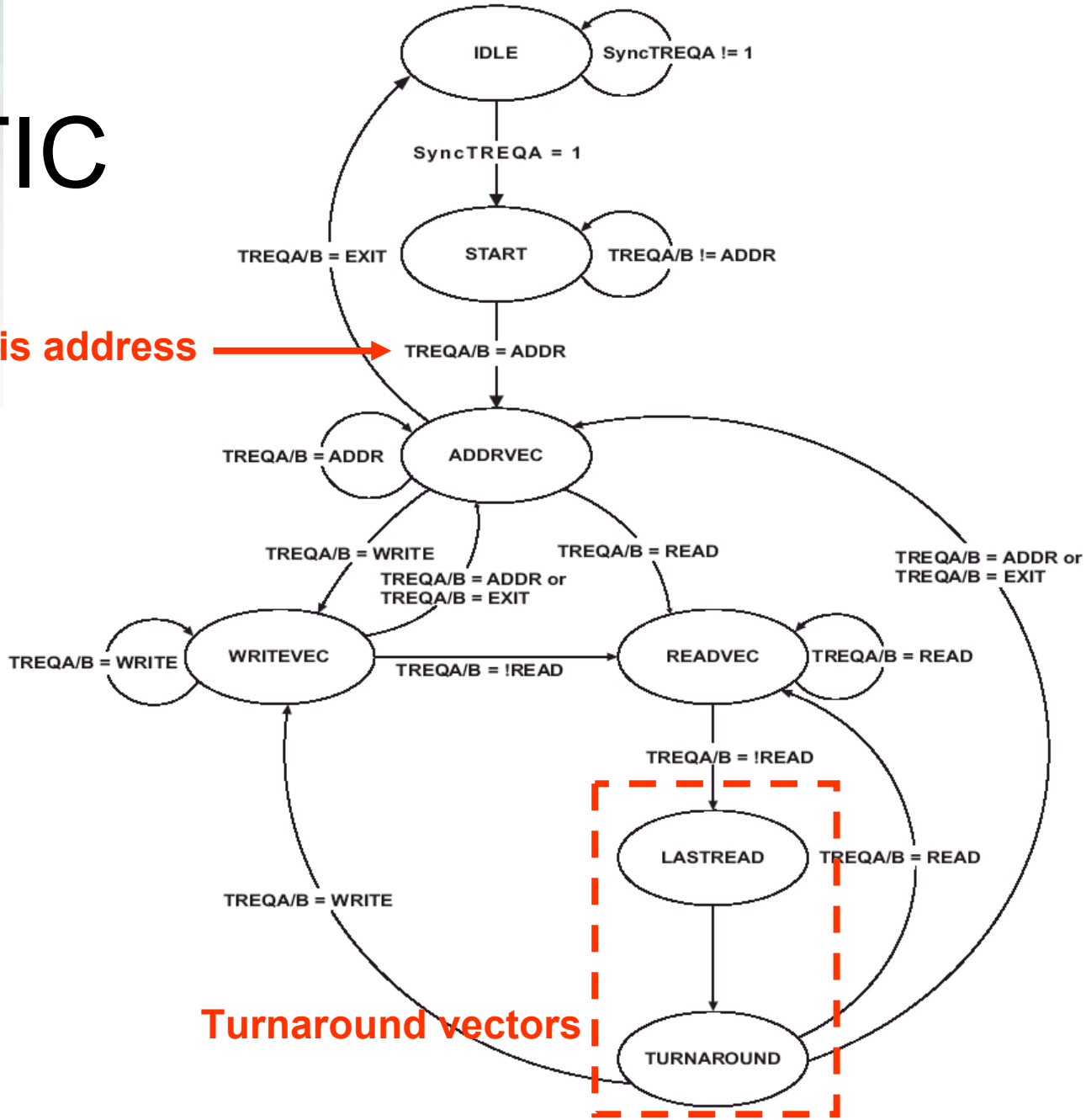
# Control Vector

| Bit Position | Description |
|---|---|
| 0 | Control vector valid (valid: 1) |
| 1 | Reserved |
| 2 | HSIZE[0] |
| 3 | HSIZE[1] |
| 4 | HLOCK |
| 5 | HPROT[0] |
| 6 | HPROT[1] |
| 7 | Address increment enable |
| 8 | Reserved |
| 9 | HPROT[2] |
| 10 | HPROT[3] |

- HSIZE[2]=0: only support 8, 16, 32-bit
- Use HLOCK signal carefully, it may lock up the bus

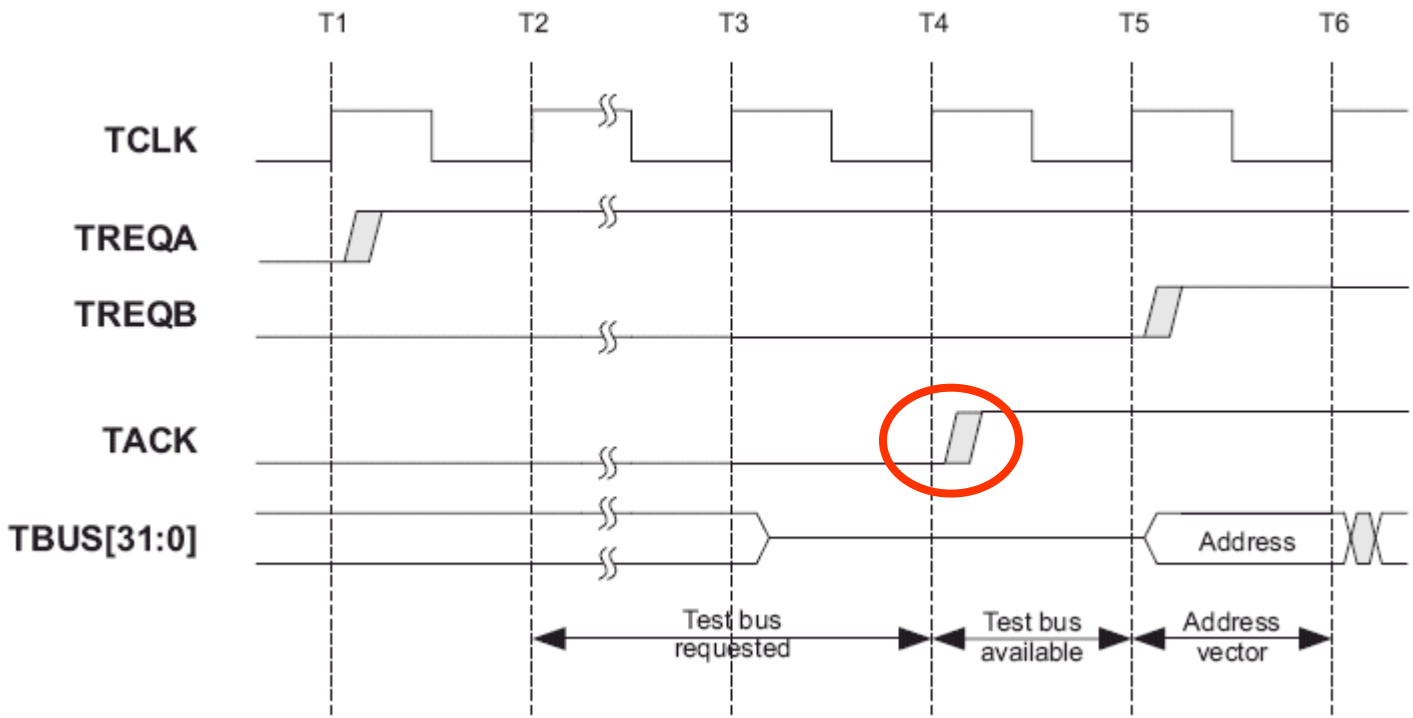
# FSM of TIC

The first vector is address



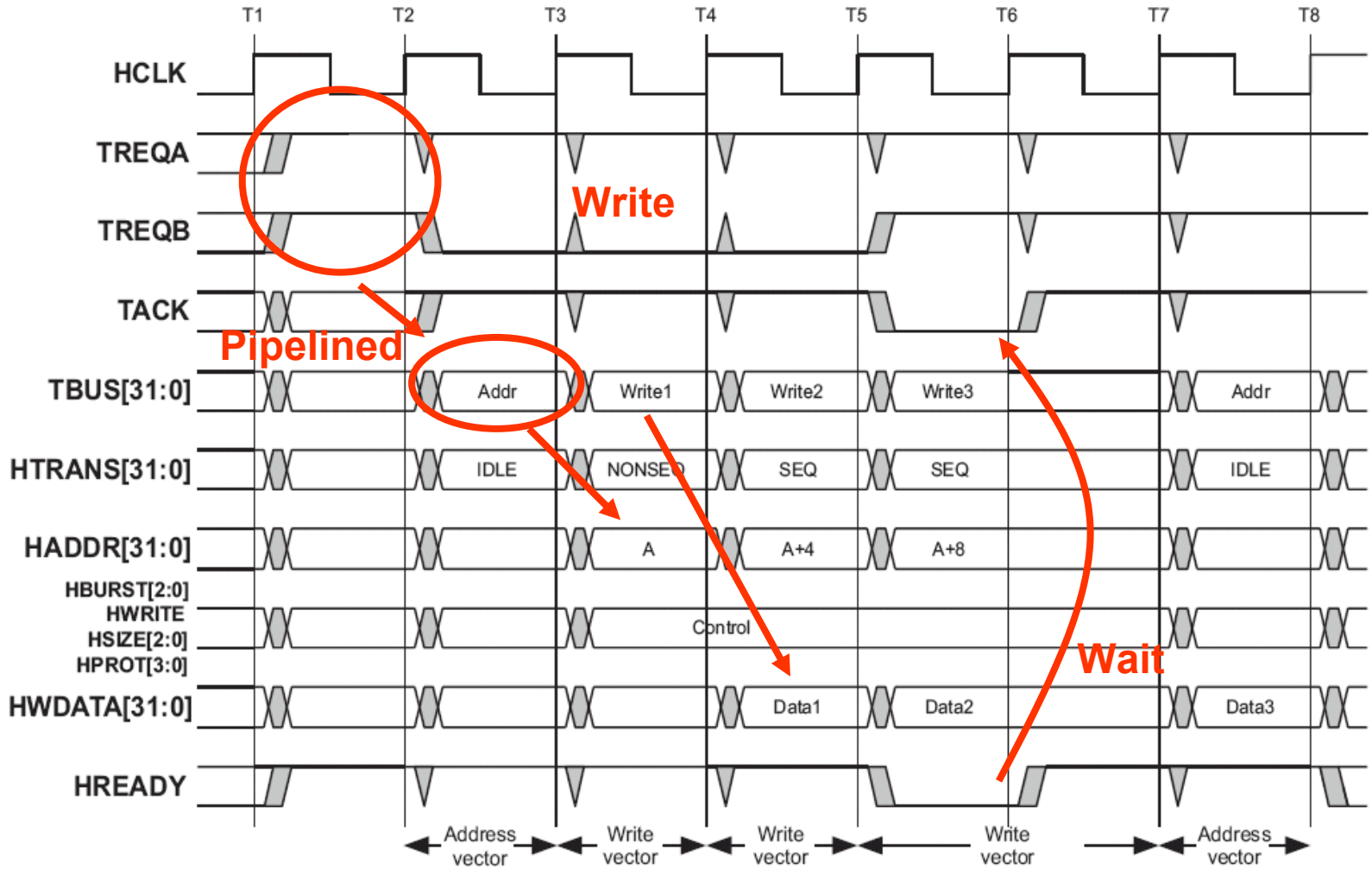

Turnaround vectors

# Enter Test Mode

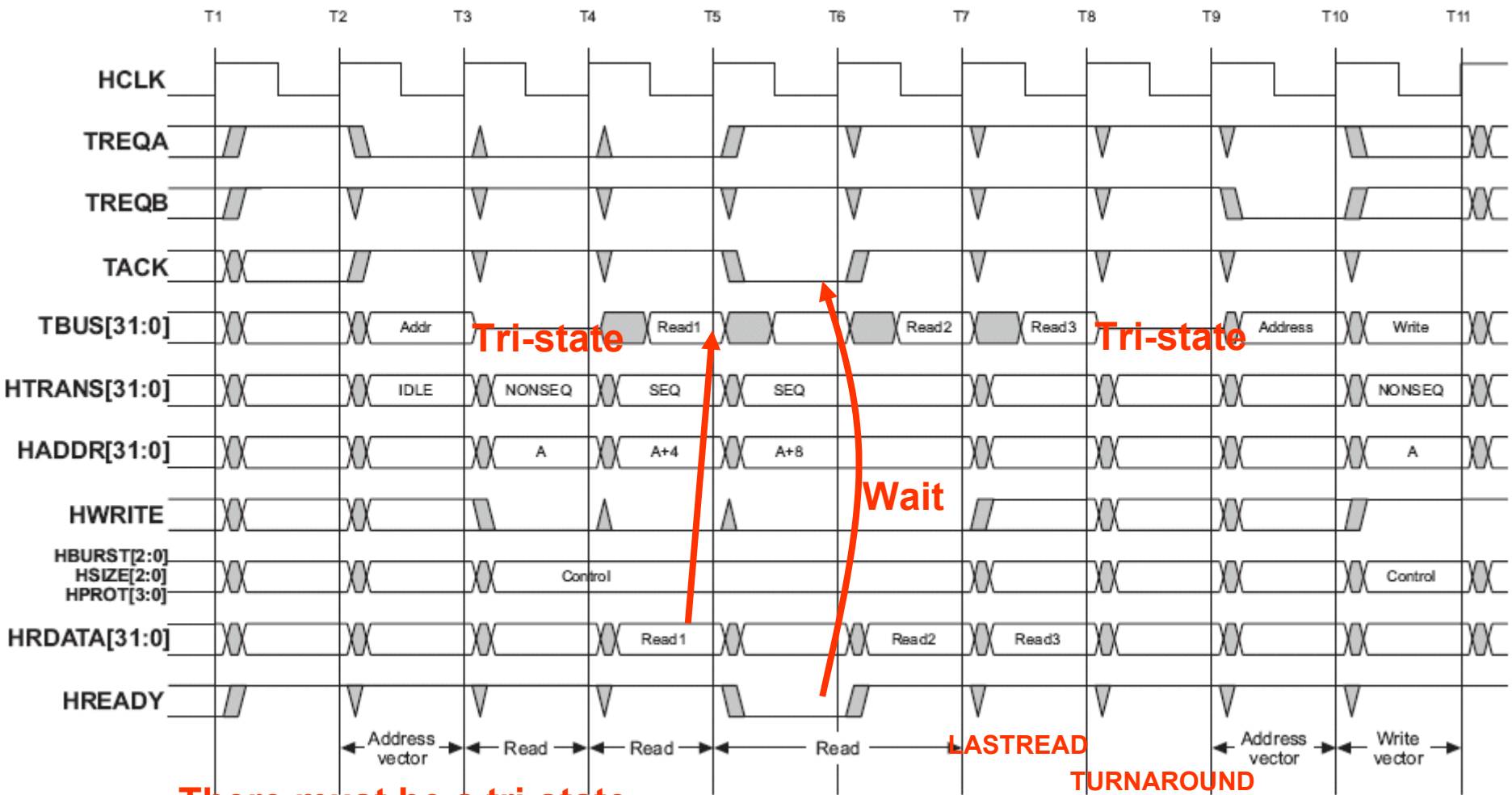



**Enter test mode**
**HCLK←TCLK**

# Write Test Vectors

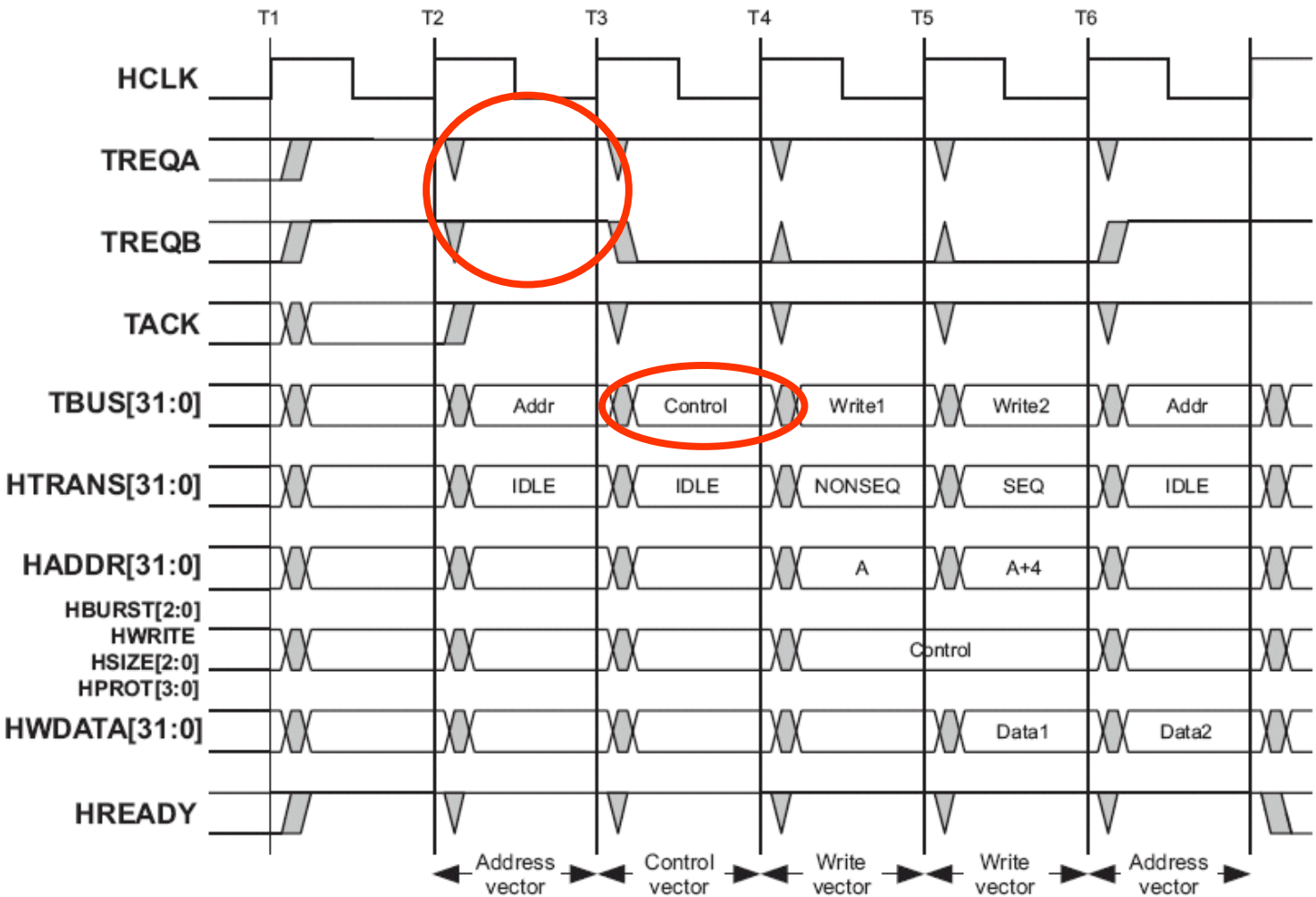


# Read Test Vectors

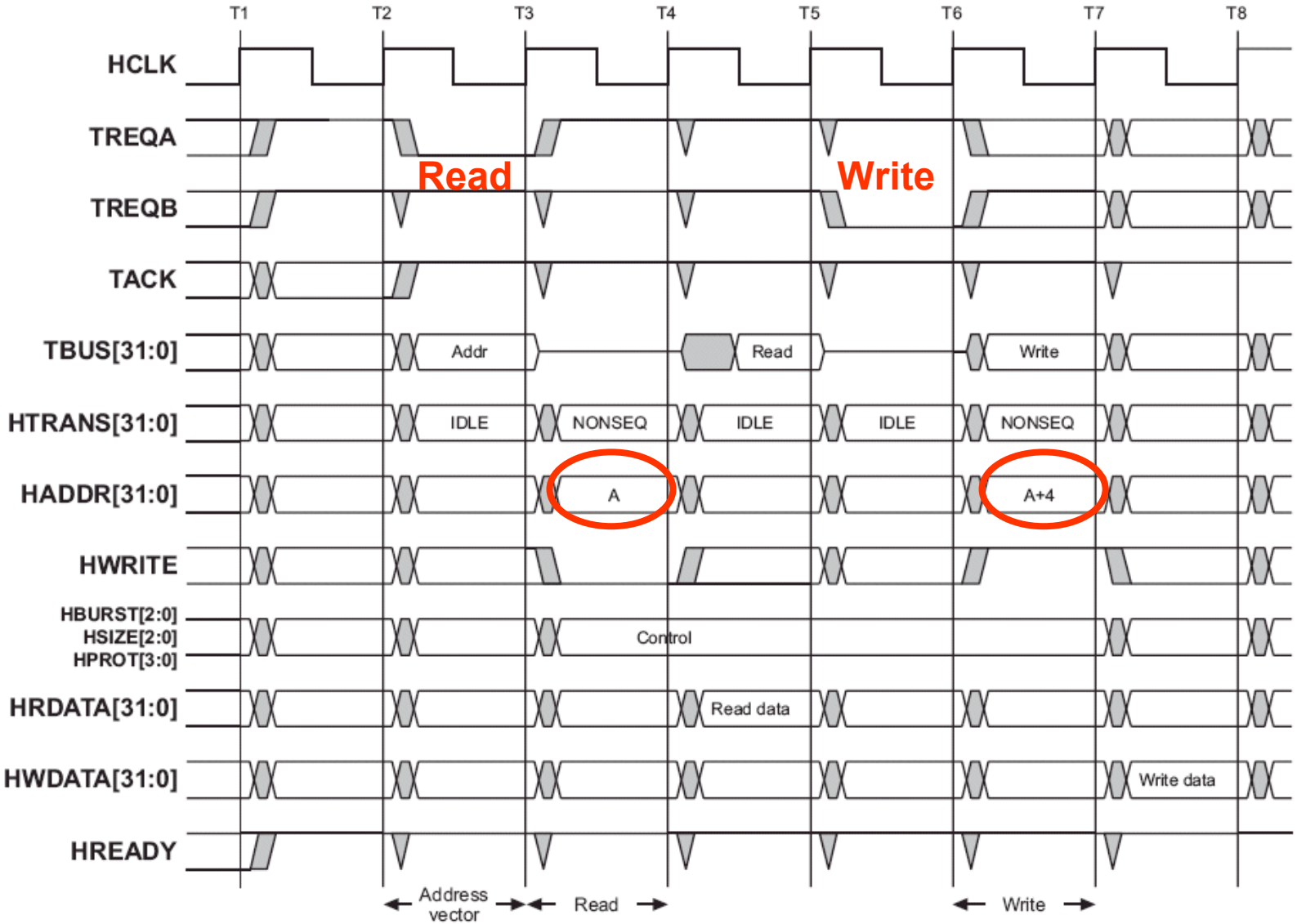



**There must be a tri-state after address and read vector**

# Control Vector



# Read-to-Write and Write-to-Read



# Other Topics

- AHB-Lite
- Multi-layer AHB

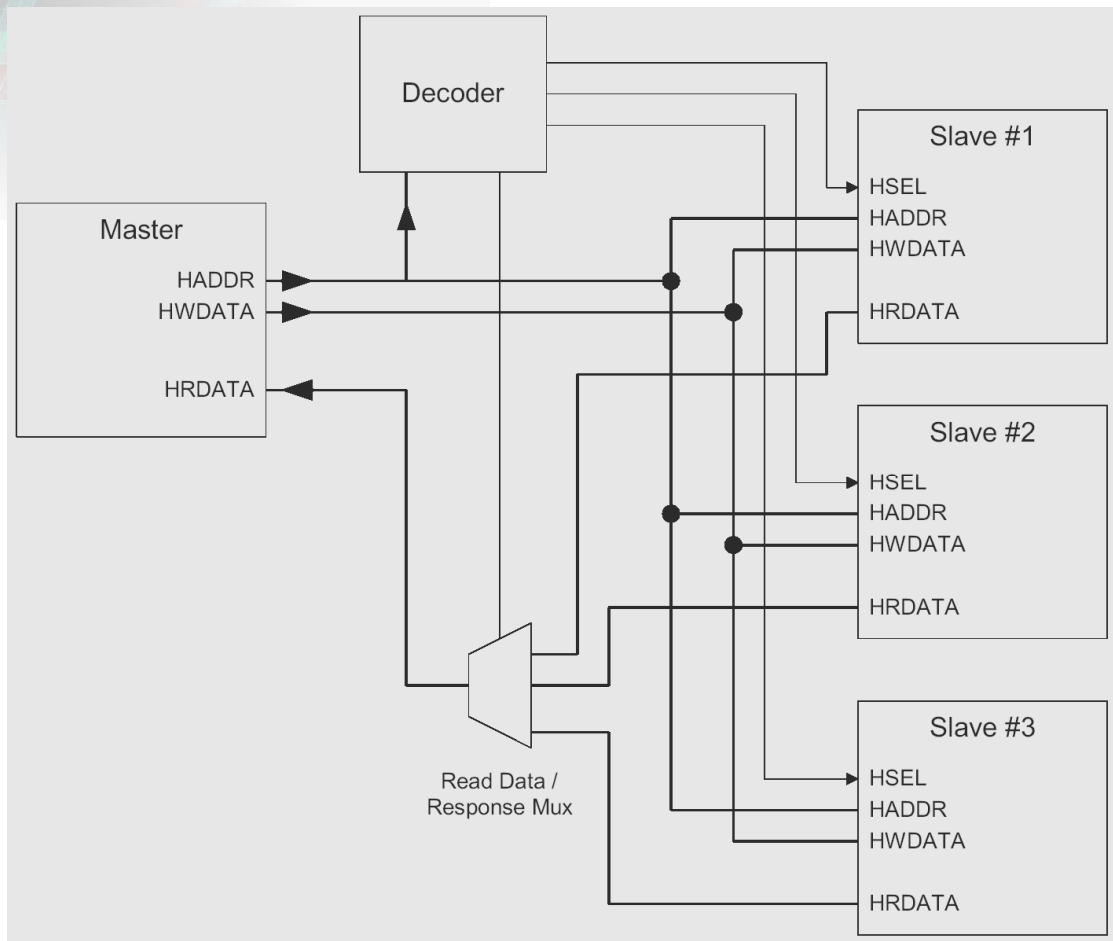
# AHB-Lite

- AHB-Lite
  - A subset of the full AHB specification
  - Only one single bus master used
  - No need of the request/grant protocol to the arbiter
  - No arbiter
  - No Retry/Split response from slaves
  - No master-to-slave multiplexor

# AHB-Lite Interchangeability

| Component | Full AHB system | AHB-Lite system |
|---|---|---|
| Full AHB master | ok | ok |
| AHB-Lite master | Need standard AHB master wrapper | ok |
| AHB slave (no Retry/Split) | ok | ok |
| AHB slave with Retry/Split | ok | Need standard AHB slave wrapper |

# AHB-Lite Block Diagram

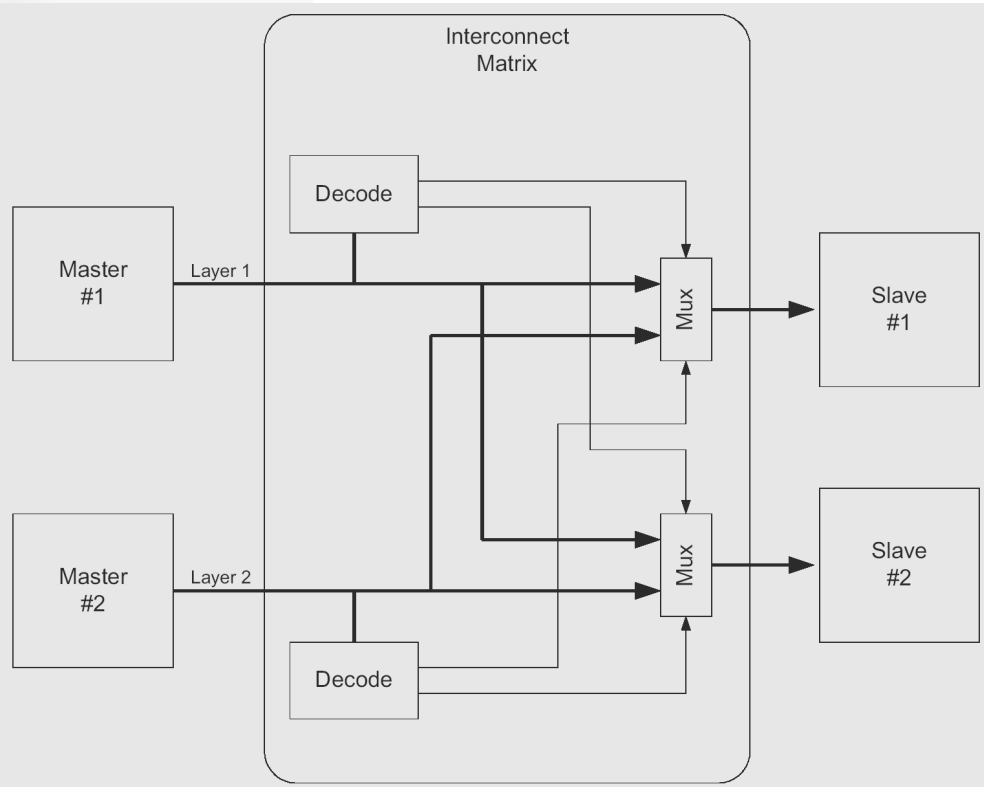


# Multi-layer AHB

- Multi-layer AHB
  - Enables parallel access paths between multiple masters and slaves by an interconnection matrix
  - Increase the overall bus bandwidth
  - More flexible system architecture
    - Make slaves local to a particular layer
    - Make multiple slaves appear as a single slave to the interconnection matrix
    - Multiple masters on a single layer

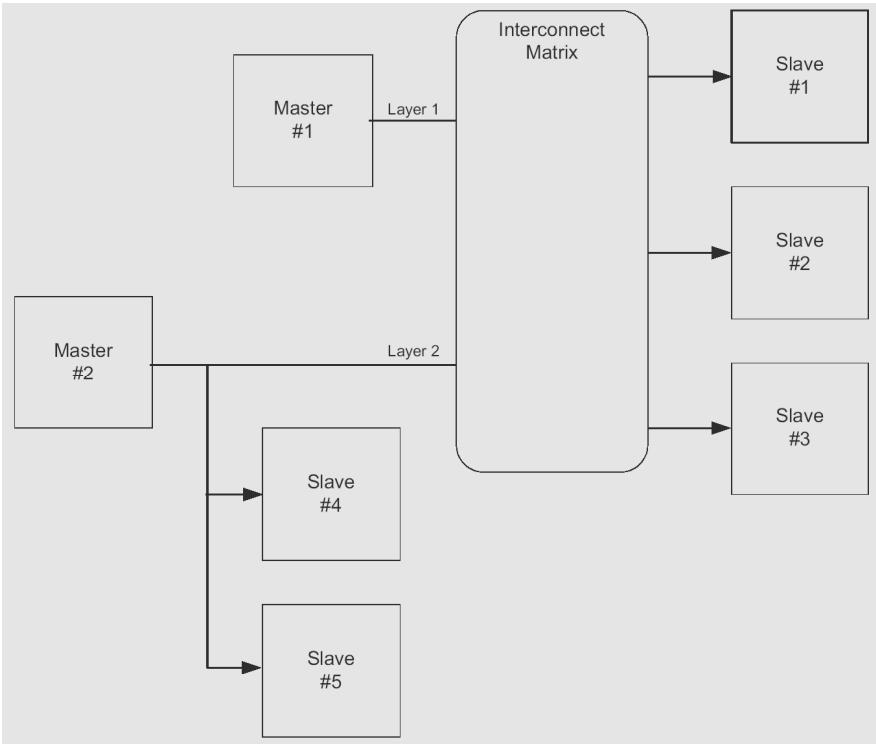
# Multi-layer AHB

- A simple multi-layer system



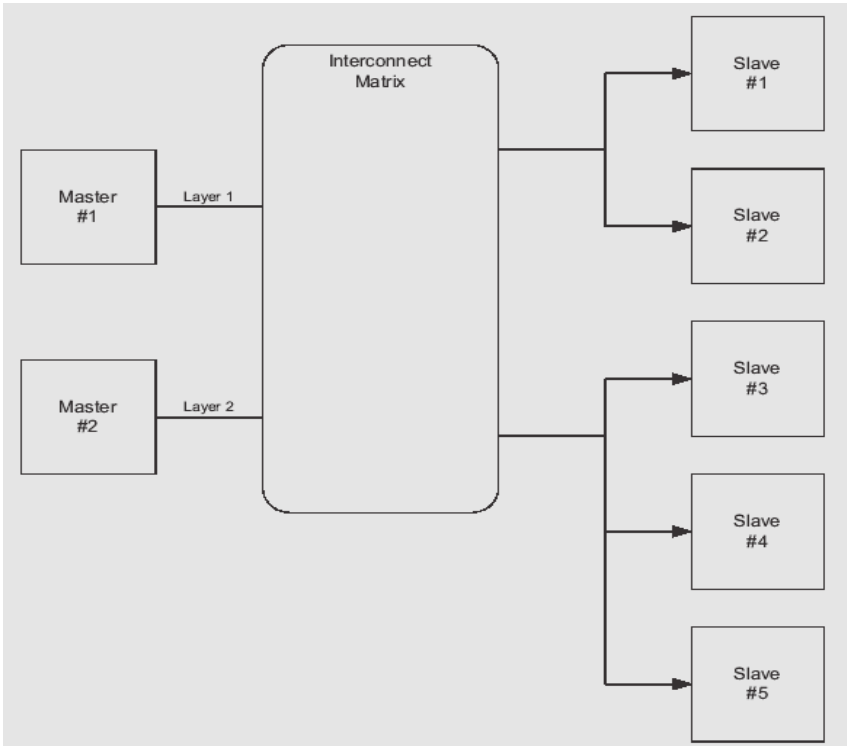
# Multi-layer AHB

- Local slaves
  - Slave #4 and Slave #5 can only be accessed by Master #2



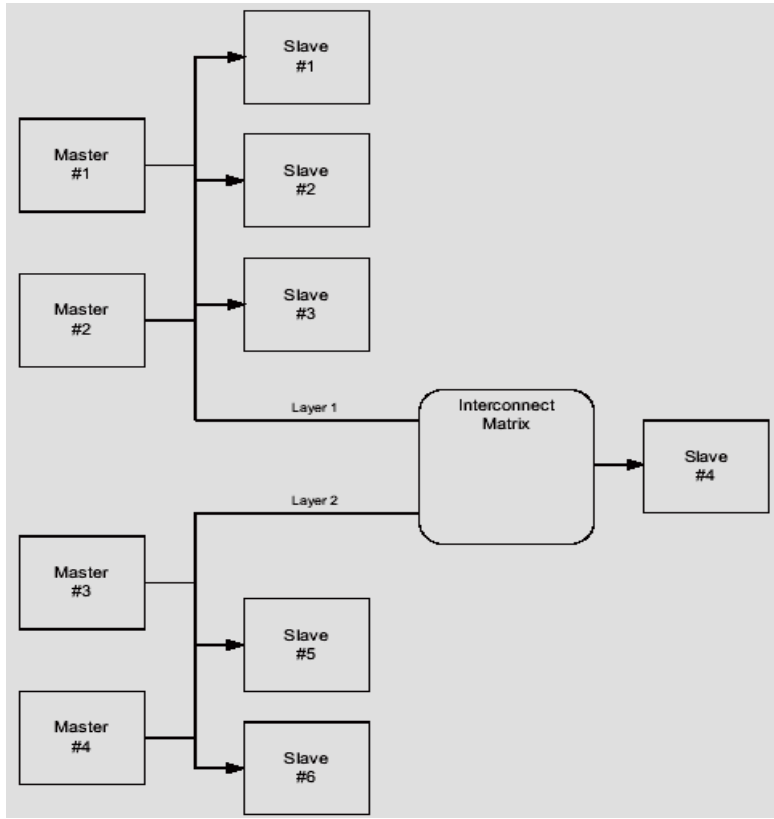
# Multi-layer AHB

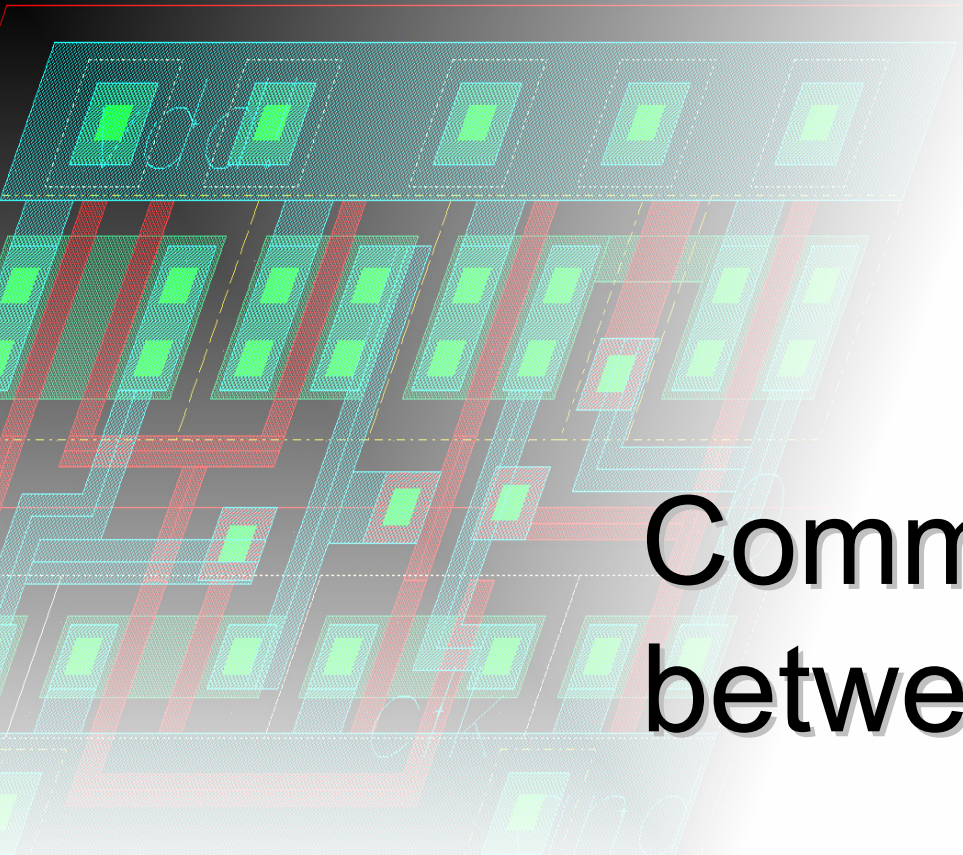
- Multiple slaves on one slave port
  - Combine low-bandwidth slaves together
  - Combine salves usually accessed by the same master together



# Multi-layer AHB

- Multiple masters on one layer
  - Combine masters which have low-bandwidth requirements together
  - Combine special masters together





# Communications between Different IPs

# Communications

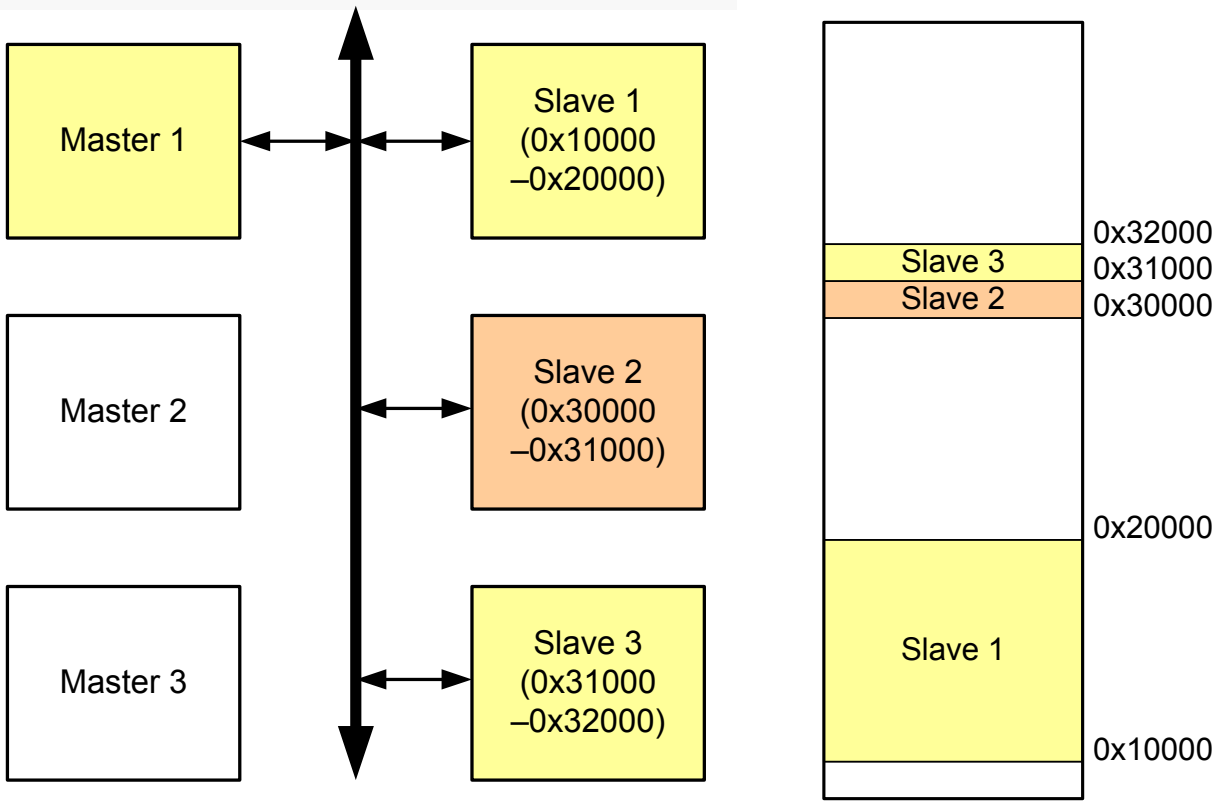
- CPU (master) ←→ IP (slave)
- IP (master) ←→ IP (slave)

# Memory Mapped I/O

- Each **slave** occupies a range of (>1KB) address space in the system
- All the slaves are **addressable**
- Memory mapped register/memory
- CPU/IP and read/write data to other IP **as read/write data from/to memory**

# Communication between IPs

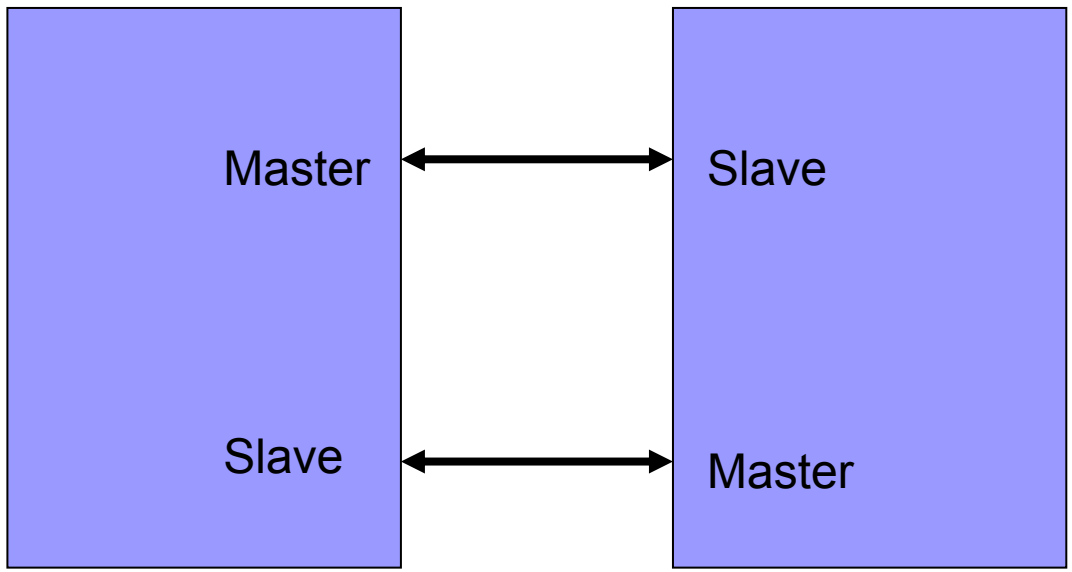
- After the master is granted by the arbiter, it can access all the slaves on the bus




Write(address, data)
Read(address, data)

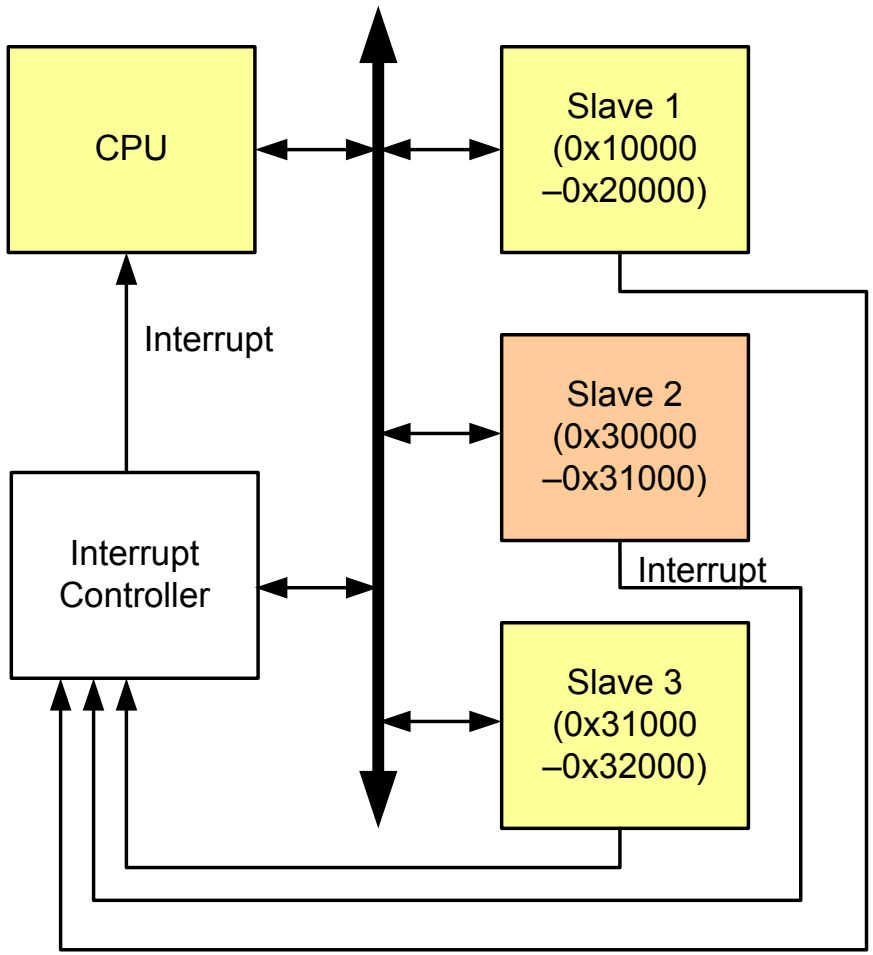
Ex:
Write(0x30020, 0x0)
Read(0x30100, &temp)

# An IP Can Have Both Mater and Slave I/F

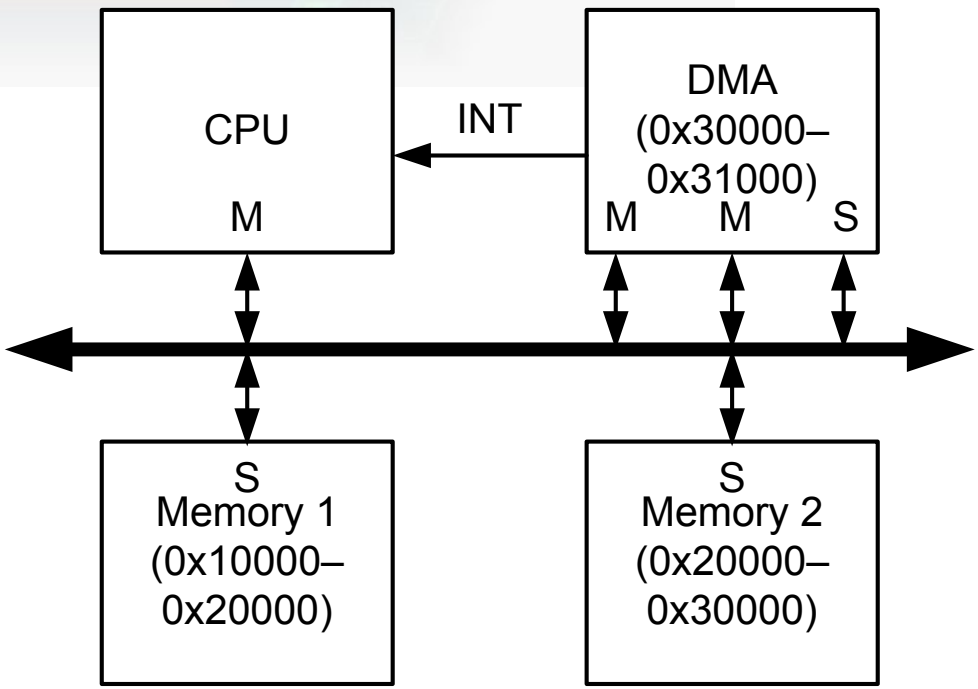


# Communication between CPU and IP

- CPU is always the master
- The IP is always the slave
- The IP can initiate the feedback with **interrupt**
- After interrupt, the CPU enters interrupt mode, and the interrupt is handled with **interrupt service routine (ISR)**



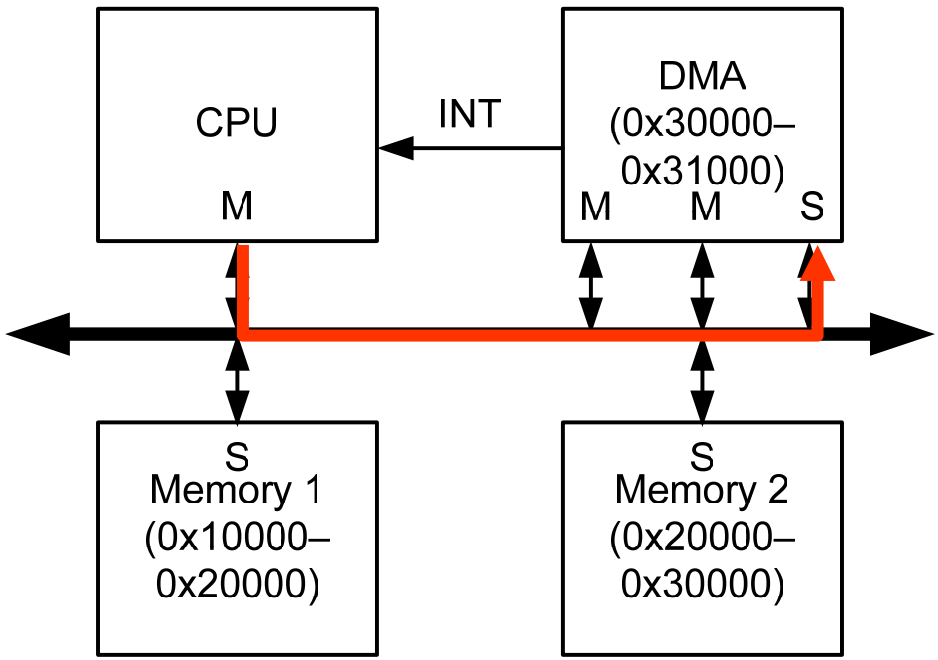
# Example: DMA




| Address | Function |
|---|---|
| 0x00 | 1: start, 0: stop |
| 0x04 | Status.<br>0: ready;<br>1: busy |
| 0x08 | Source address |
| 0x0C | Destination address |
| 0x10 | Size |

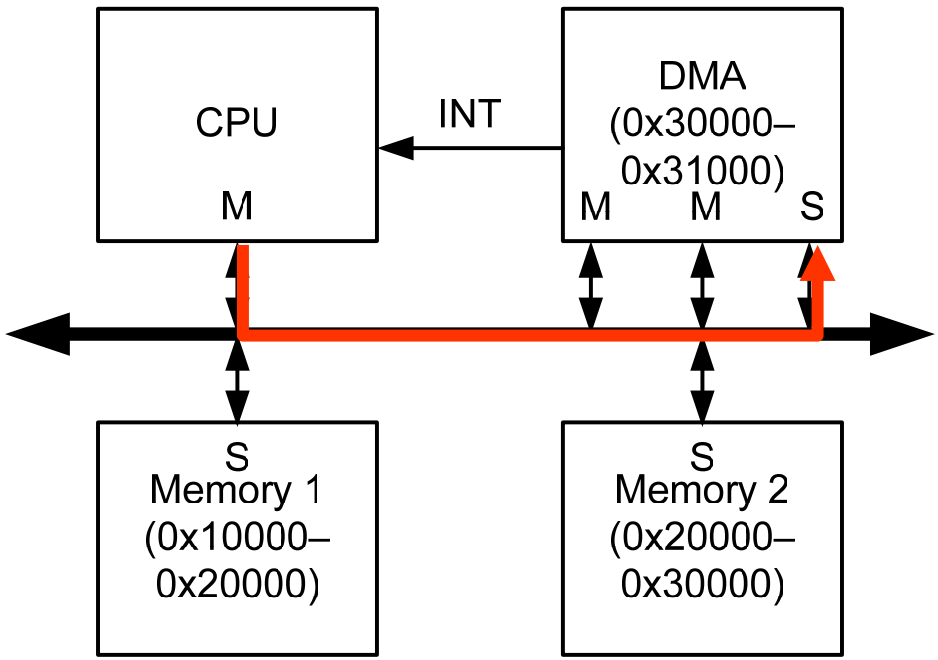
# Example: DMA

- Step 0: CPU check the status of DMA to make sure it is ready to be used
  - While(1)
  - {
  - Read(0x30004, &status)
  - if(status == 0)
  - break;
  - }



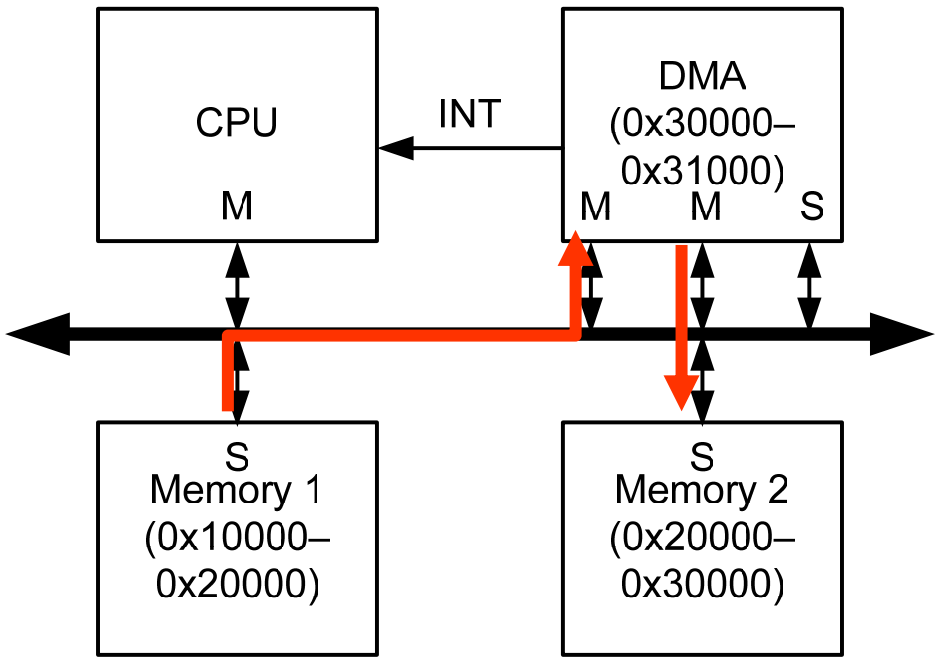
# Example: DMA

- Step 1: CPU sets the (source address), (destination address), and (size) with the slave I/F
  - Write(0x30008, 0x10000)
  - Write(0x3000C, 0x20000)
  - Write(0x30010, 0x100)
- Step 2: starts DMA
  - Write(0x30000, 0x1)



# Example: DMA

- Step 3: DMA moves data from memory 1 to memory 2 with the two master I/F



# Example: DMA

- Step 4: DMA interrupts CPU
- Step 5: CPU checks the status of DMA
  - Read(0x30004, &status)