N26F300
VLSI SYSTEM DESIGN
(GRADUATE LEVEL)

Bus Interface

Outline

- □ Processor
- □ Custom processor – GCD example
- □ Peripherals
- □ **Interfacing via bus**

[Material partly adapted from Embedded System Design by F. Vahid & T. Givargis]

Interfacing

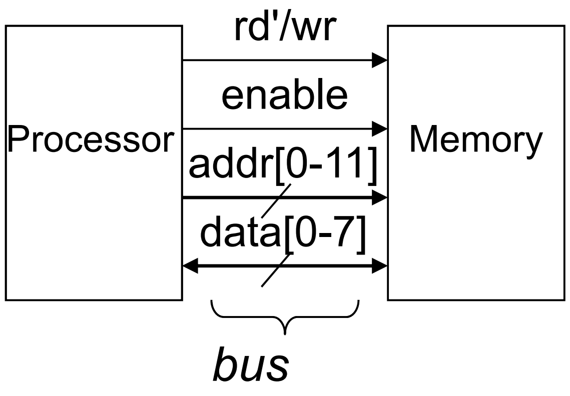
Bus Overview

AHB Bus

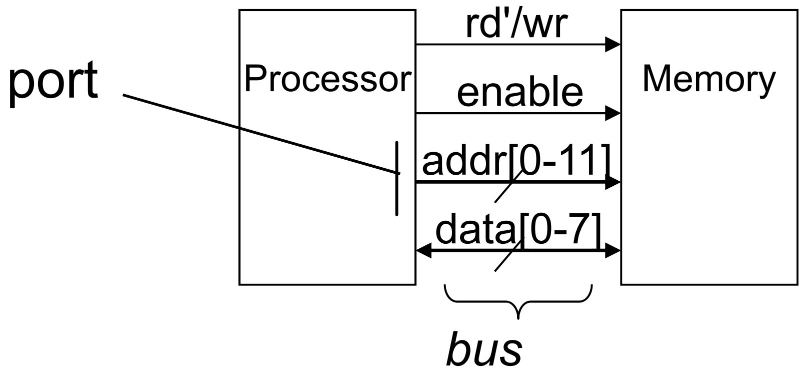
AXI Bus

A simple bus

- Wires:
  - Uni-directional or bi-directional
  - One line may represent multiple wires
- Bus
  - Set of wires with a single function
    - Address bus, data bus
  - Or, entire collection of wires
    - Address, data and control
    - Associated protocol: rules for communication



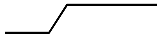
Ports



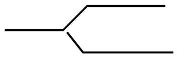
- Conducting device on periphery
- Connects bus to processor or memory
- Often referred to as a *pin*
  - Actual pins on periphery of IC package that plug into socket on printed-circuit board
  - Sometimes metallic balls instead of pins
  - Today, metal “pads” connecting processors and memories within single IC
- Single wire or set of wires with single function
  - E.g., 12-wire address port

Timing Diagrams

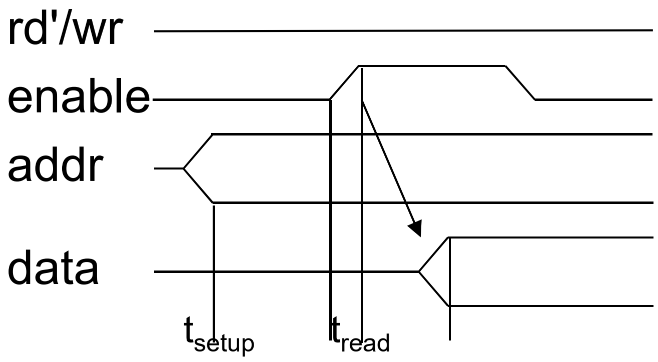
- Most common method for describing a communication protocol
- Time proceeds to the right on x-axis
- Control signal: low or high



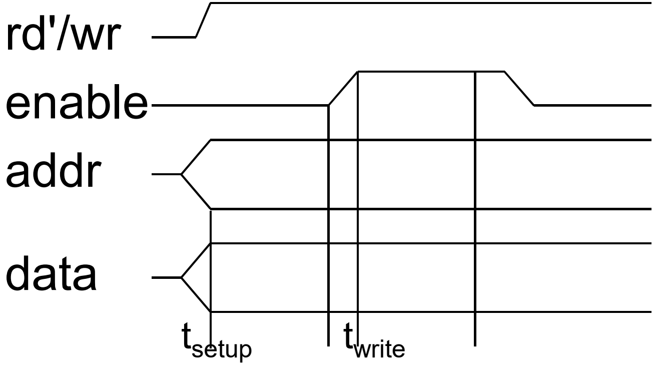
  - May be active low (e.g., go', /go, or go_L)
  - Use terms *assert* (active) and *deassert*
  - Asserting go' means go=0
- Data signal: not valid or valid



- Protocol may have subprotocols
  - Called bus cycle, e.g., read and write
  - Each may be several clock cycles
- Read example
  - *rd'/wr* set low,address placed on *addr* for at least t_{setup} time before *enable* asserted, enable triggers memory to place data on *data* wires by time t_{read}



read protocol

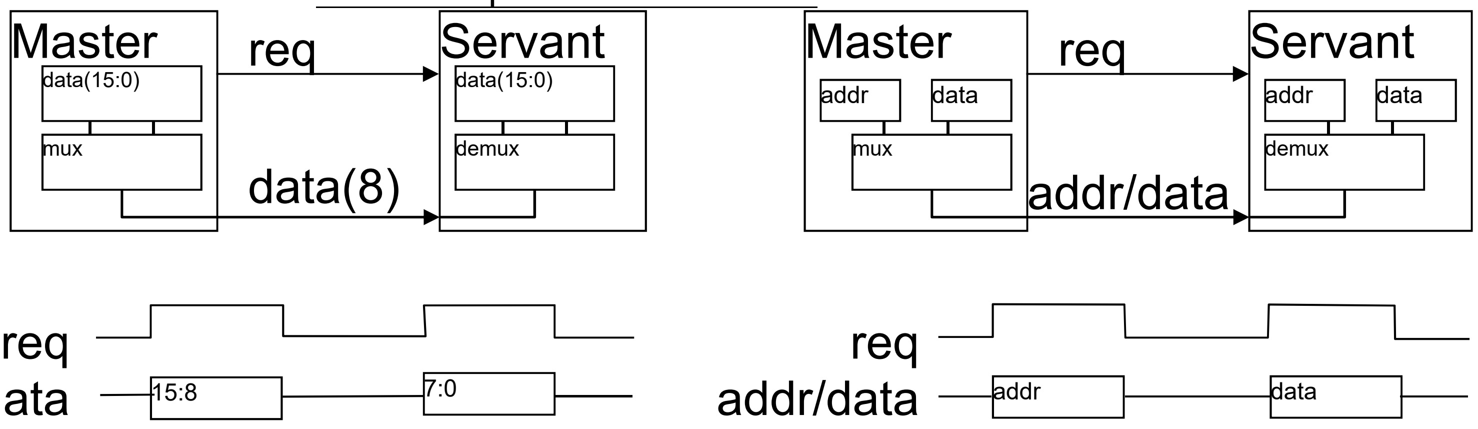


write protocol

Basic protocol concepts

- Actor: master initiates, servant (slave) respond
- Direction: sender, receiver
- Addresses: special kind of data
  - Specifies a location in memory, a peripheral, or a register within a peripheral
- Time multiplexing
  - Share a single set of wires for multiple pieces of data
  - Saves wires at expense of time

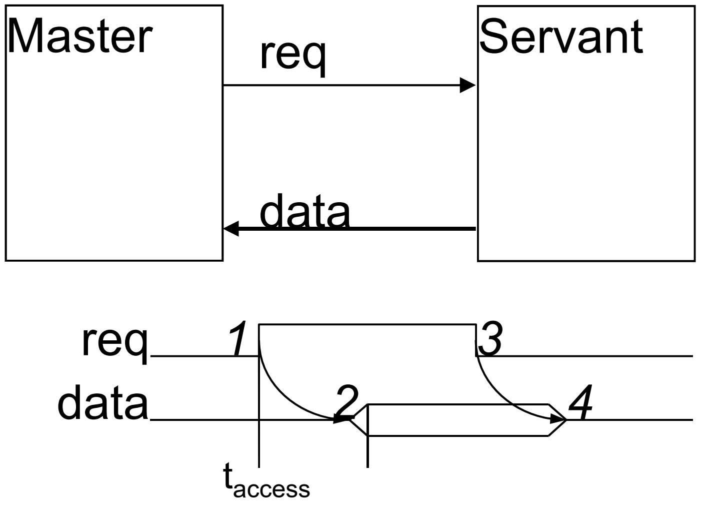
Time-multiplexed data transfer



data serializing

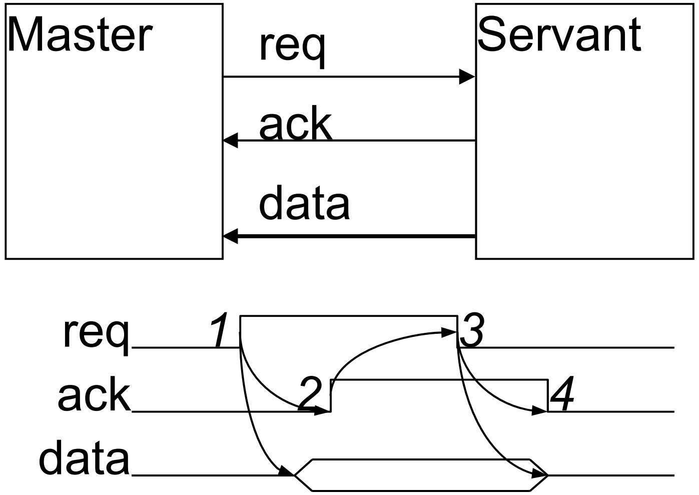
address/data muxing

Basic protocol concepts: control methods



1. Master asserts *req* to receive data
2. Servant puts data on bus **within time t_{access}**
3. Master receives data and deasserts *req*
4. Servant ready for next request

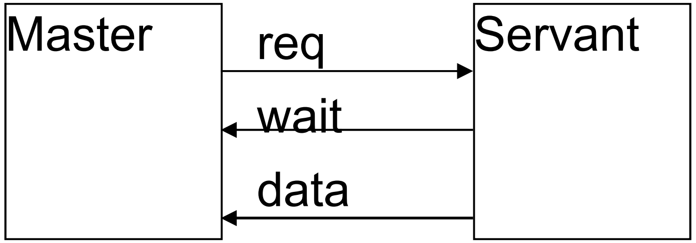
Strobe protocol

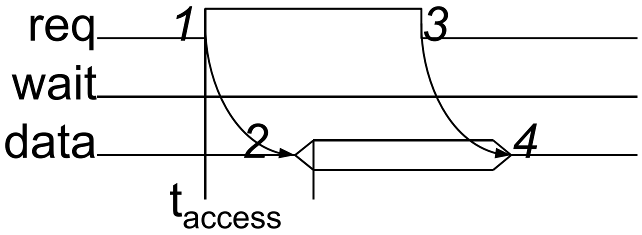


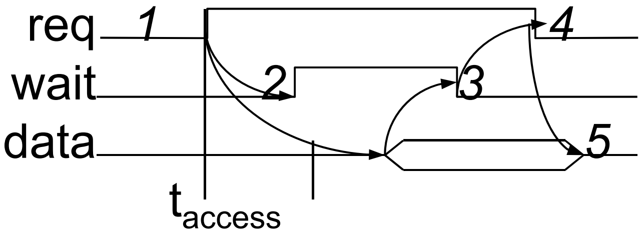
1. Master asserts *req* to receive data
2. Servant puts data on bus **and asserts *ack***
3. Master receives data and deasserts *req*
4. Servant ready for next request

Handshake protocol

A strobe/handshake compromise







1. Master asserts *req* to receive data
2. Servant puts data on bus **within time t_{access}** (wait line is unused)
3. Master receives data and deasserts *req*
4. Servant ready for next request

Fast-response case

1. Master asserts *req* to receive data
2. Servant can't put data within **t_{access}**, **asserts *wait*** ack
3. Servant puts data on bus and **deasserts *wait***
4. Master receives data and deasserts *req*
5. Servant ready for next request

Slow-response case

AHB Master Behavioral Model

Bus protocol

AMBA

AHB characteristics and infrastructure

Control signals

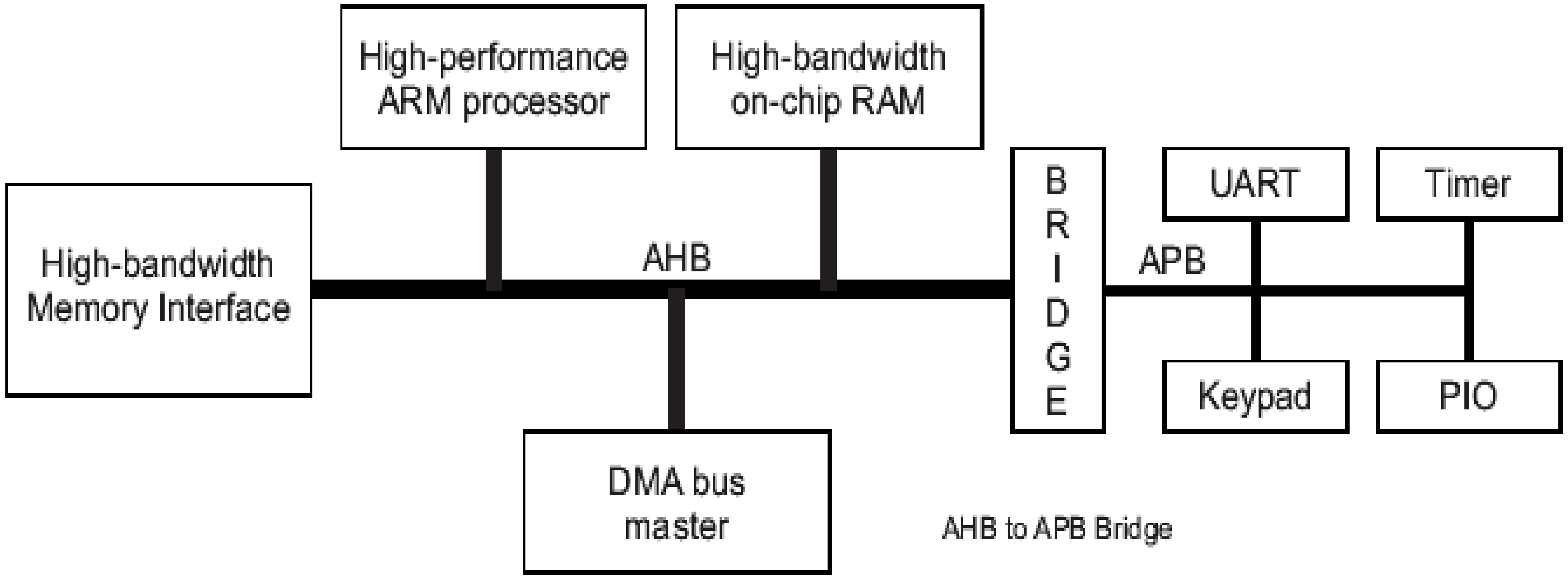
Ref:
-- AMBA 2.0
-- 簡弘倫, “Verilog 晶片設計,” 文魁資訊, 2005

Bus Protocols

- Specification of signals, timing, and sequencing of bus operations
  - Allows independent design of components
  - Ensures interoperability
- Standard bus protocols
  - PCI, VXI, …
    - For connecting boards in a system
  - AMBA (ARM), CoreConnect (IBM), Wishbone (Open Cores)
    - For connecting blocks within a chip

AMBA

- Advanced High-performance Bus(AHB)
- Advanced System Bus(ASB)
- Advanced Peripheral Bus


High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth Memory Interface
AHB
DMA bus master
B R I D G E
UART
Timer
APB
Keypad
PIO

AHB to APB Bridge

AHB characteristic

- □ Single cycle edge operation
- □ Non-tristate implementation
- □ Burst transfers
- □ Split transactions
- □ Single cycle bus master handover
- □ Wider data bus configurations(64/128bit)

AHB Simple framework

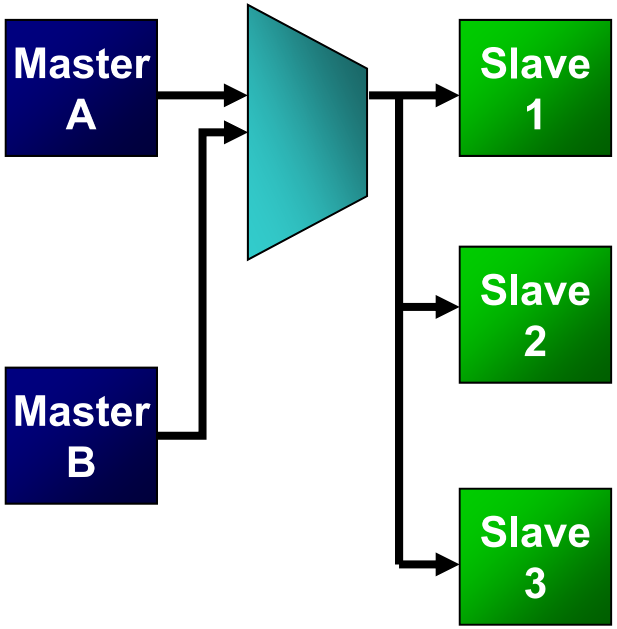
address
Control signal
Write data

clock

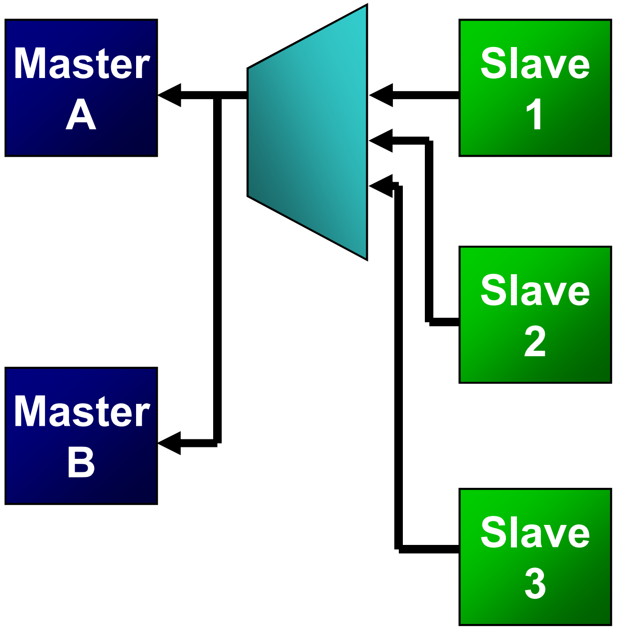
arbitration

Read data

Response signal

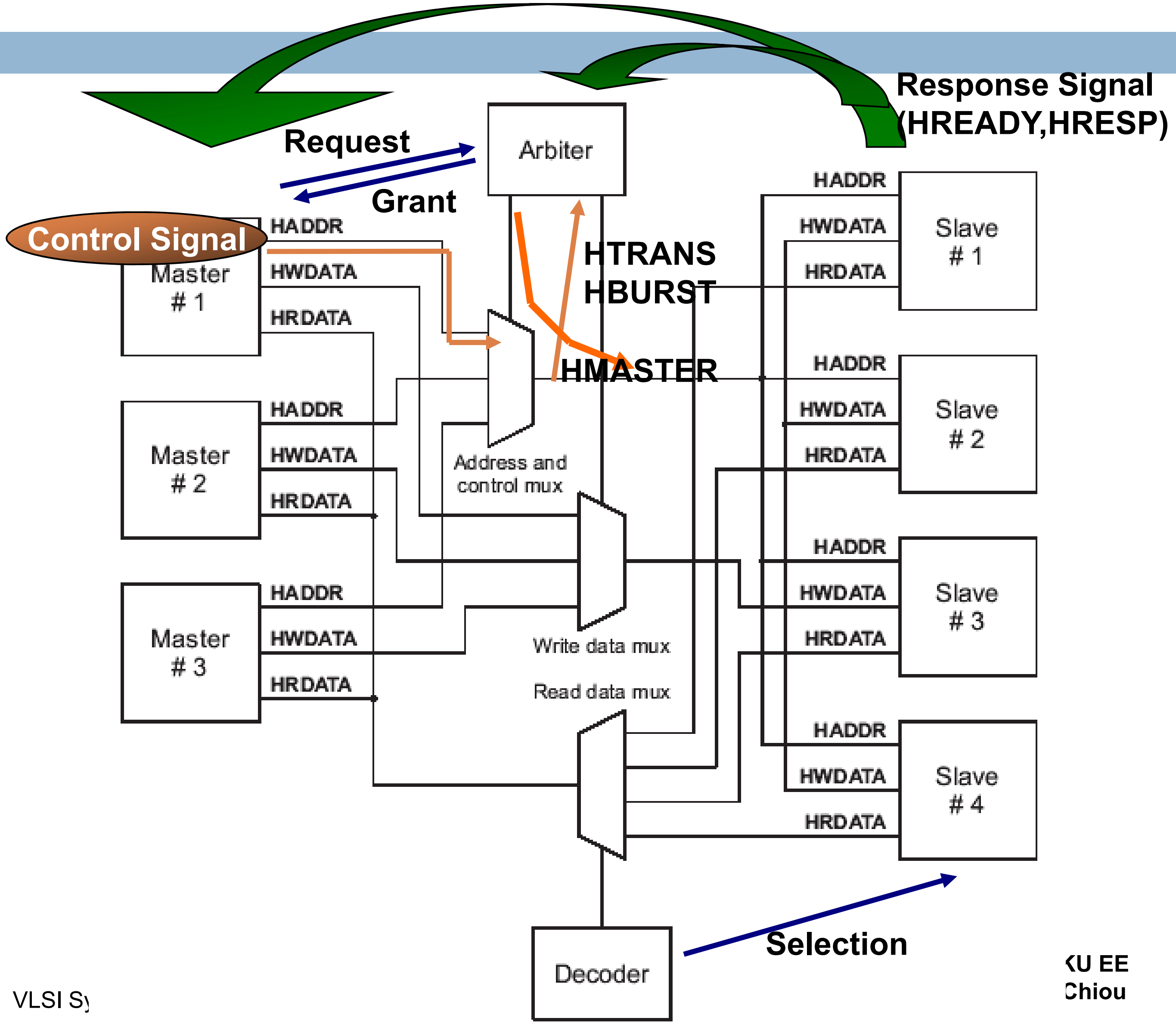


Master to Slave Multiplexor

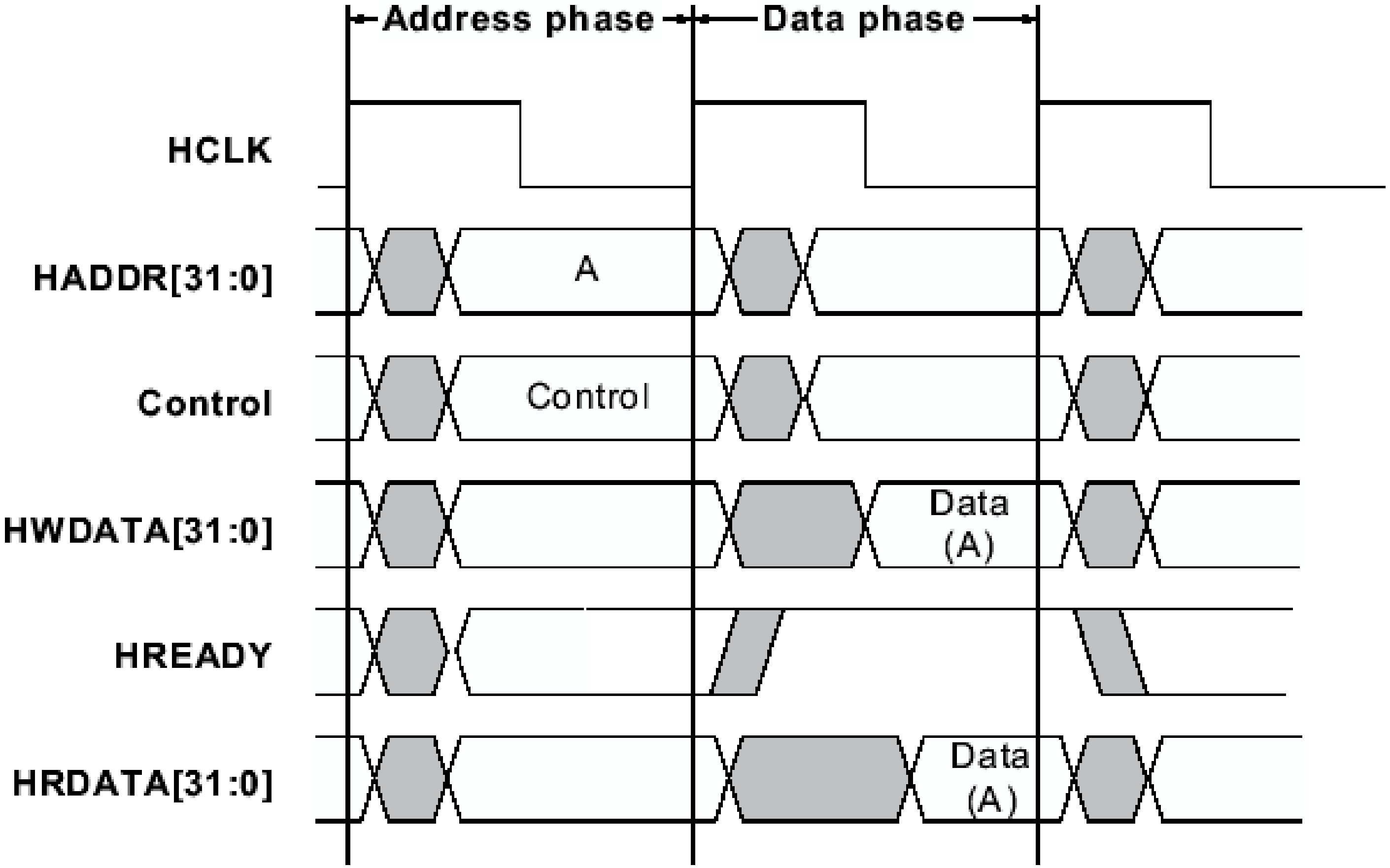


Slave to Master Multiplexor

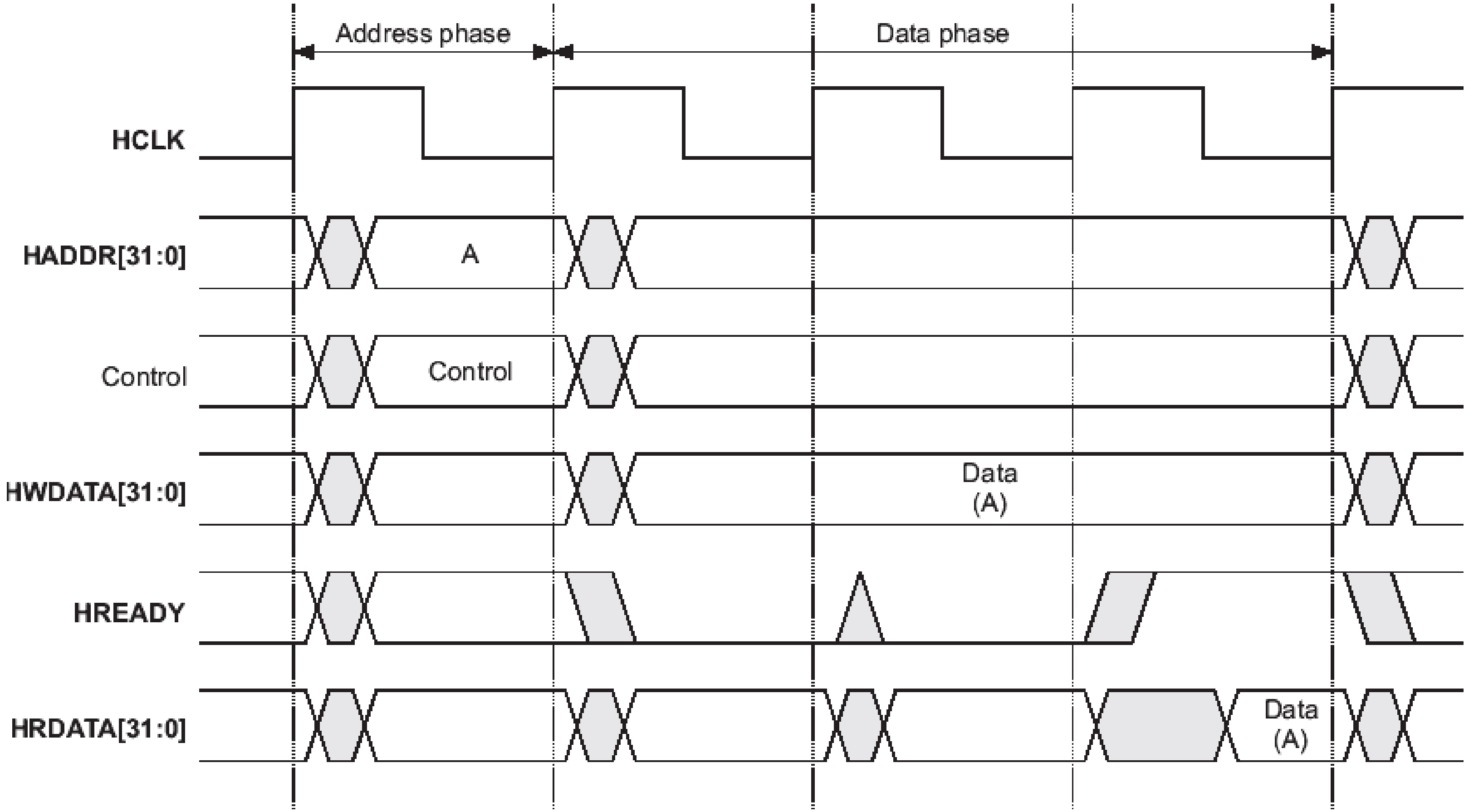
AHB Bus Interconnection



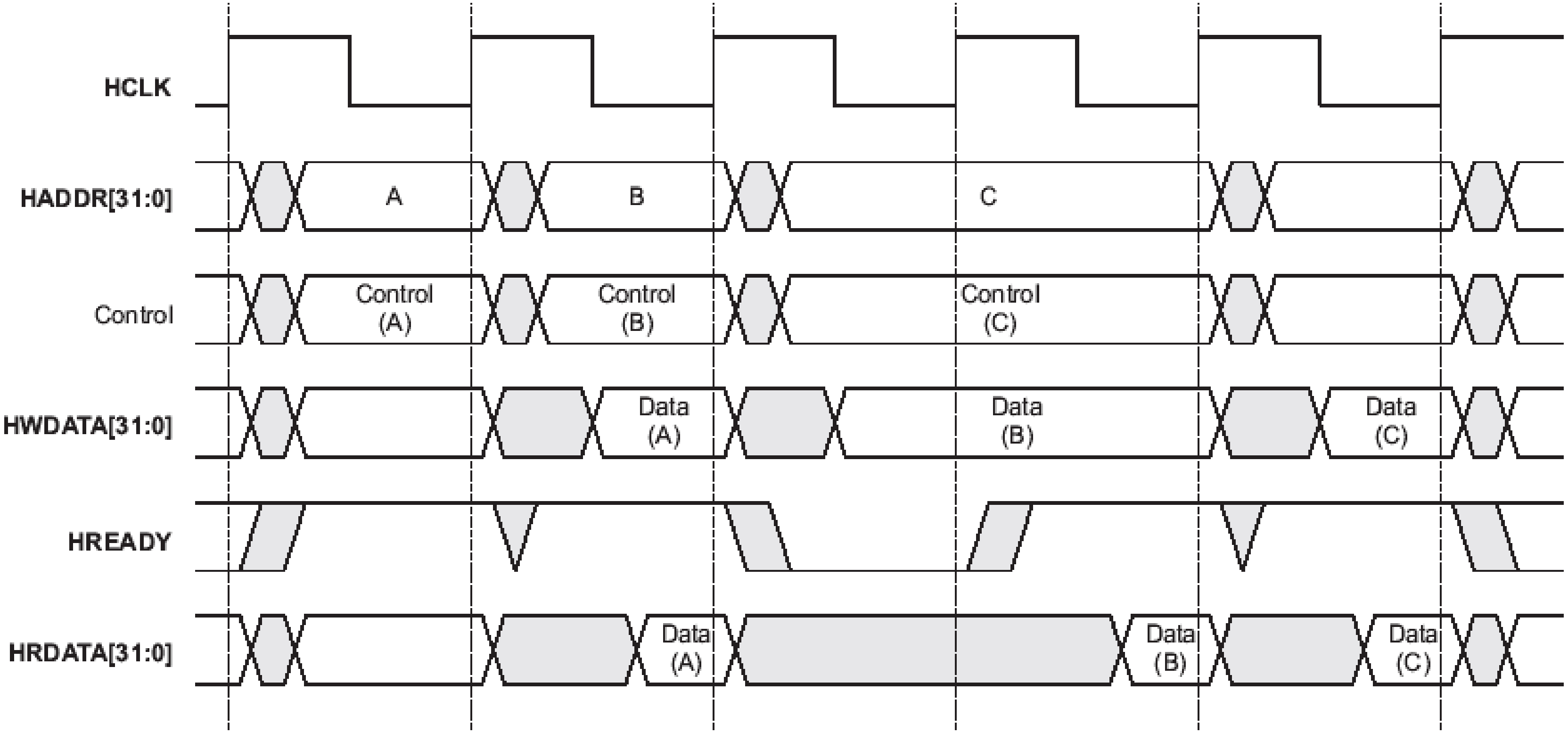
Basic Transfer (no wait state)



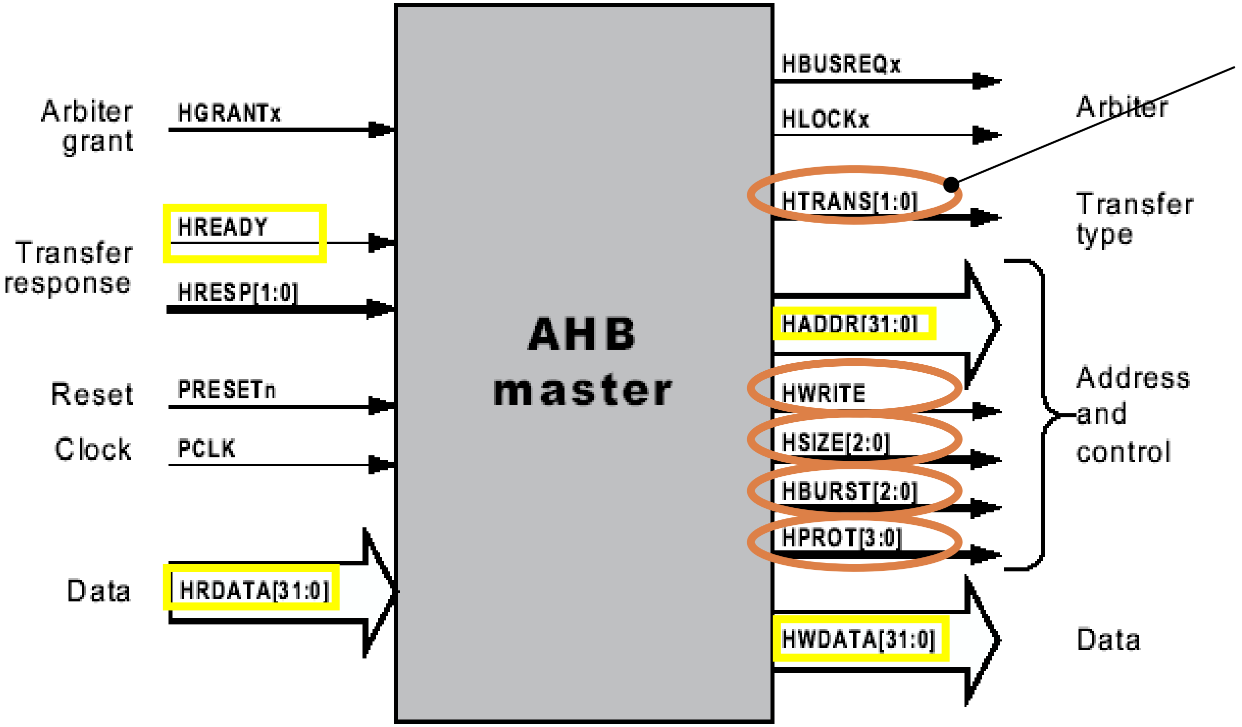
Basic Transfer (wait state)



Multiple Transfer



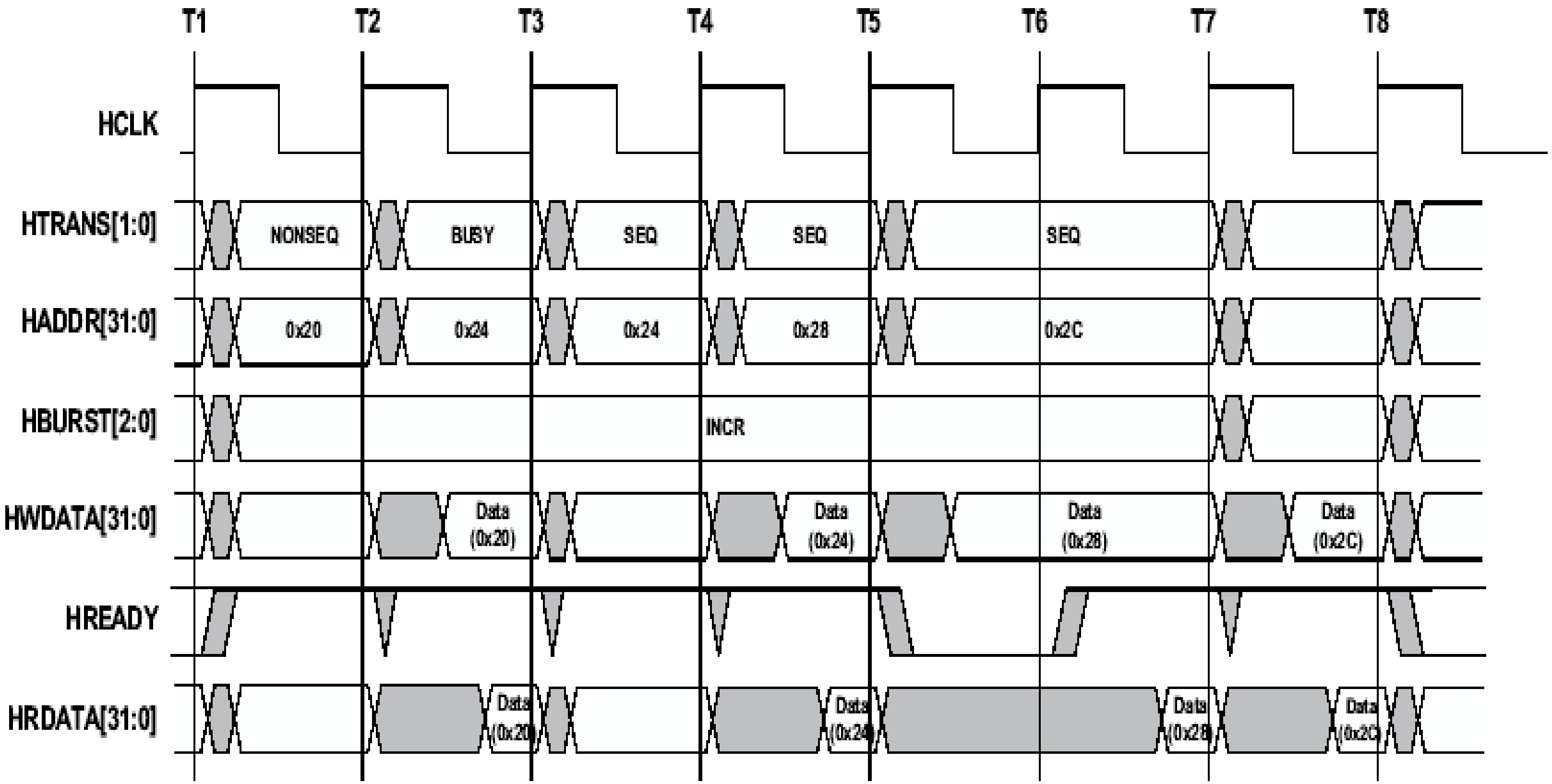
Master---Transfer Type



00:IDLE
01:BUSY
10:NONSEQ
11:SEQ

IDLE : Master has no data to transfer,而Slave會在data phase回應OKAY(response signal)
BUSY: Master無法準備好資料在下一個週期傳送,則Master發出BUSY訊號來延遲這筆資料的傳送,而Slave會在data phase回應OKAY(response signal)
NONSEQ : 表目前transfer的address/control signal和前週期被傳送的資料無關
SEQ :表目前transfer的address/control signal和前週期被傳送的資料相關) (用於Burst transfer)

Transfer type example

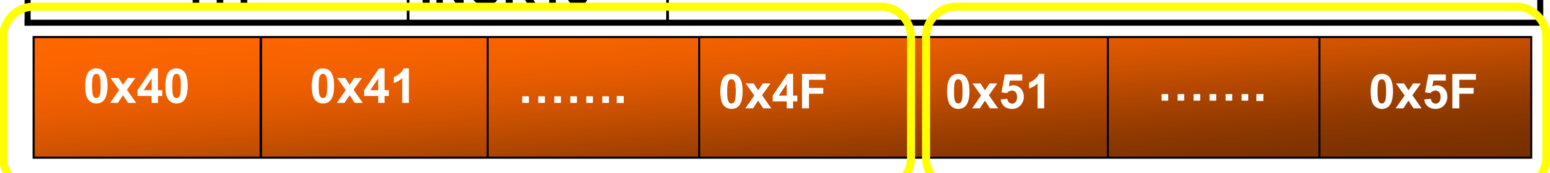


HSIZE Operation

HSIZE[2:0]	SIZE (bit)
000	8
001	16
010	32
011	64
100	128
101	256
110	512
111	1024

Burst Operation

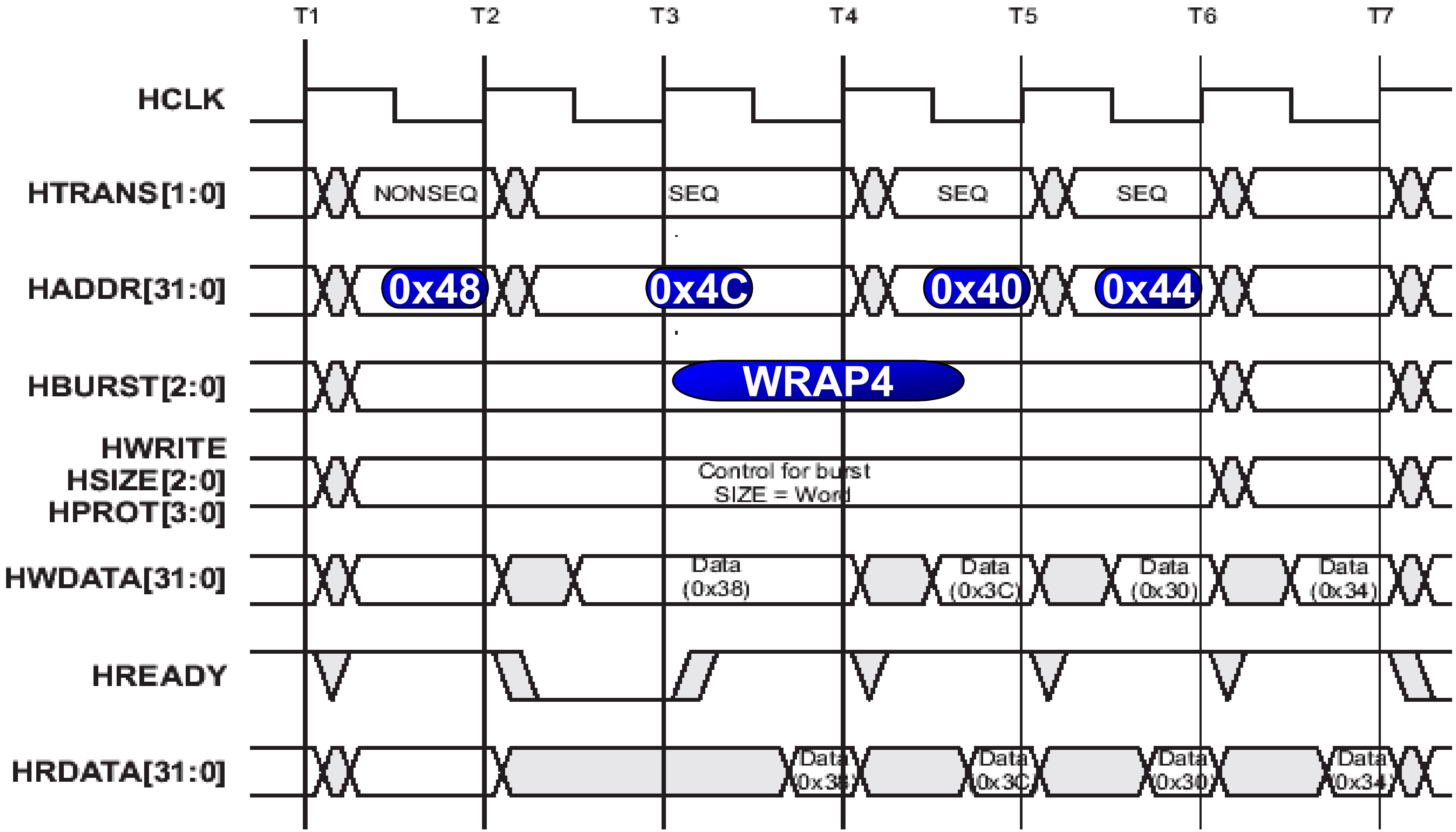
HBURST[2:0]	Type	Sample(HSIZE=4byte)
000	SINGLE	0x48
001	INCR	0x48,0x4c,0x50, …
010	WRAP4	0x48, 0x4c,0x40,0x44
011	INCR4	0x48, 0x4c,0x50,0x54
100	WRAP8	
101	INCR8	
110	WRAP16	
111	INCR16	



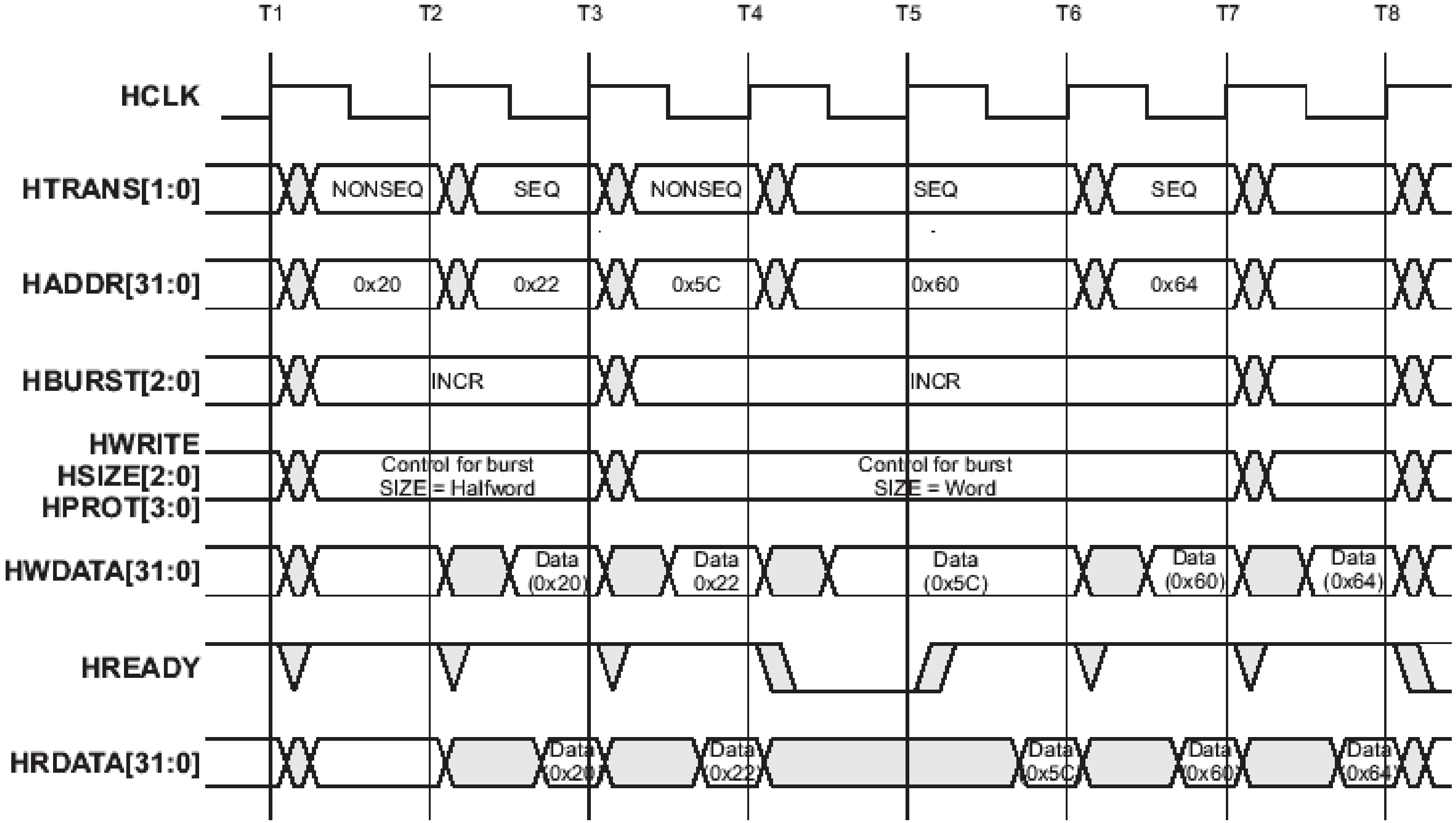
Increase (INCR) : 將前一筆的資料位址加上HSIZE的大小即為下一筆資料的位址

WRAP : wrapping burst將memory切割成某固定大小的一個個memory boundary,當transfer address要跨越此boundary時,下一筆transfer address會繞回原之boundary的起點

Burst Operation example

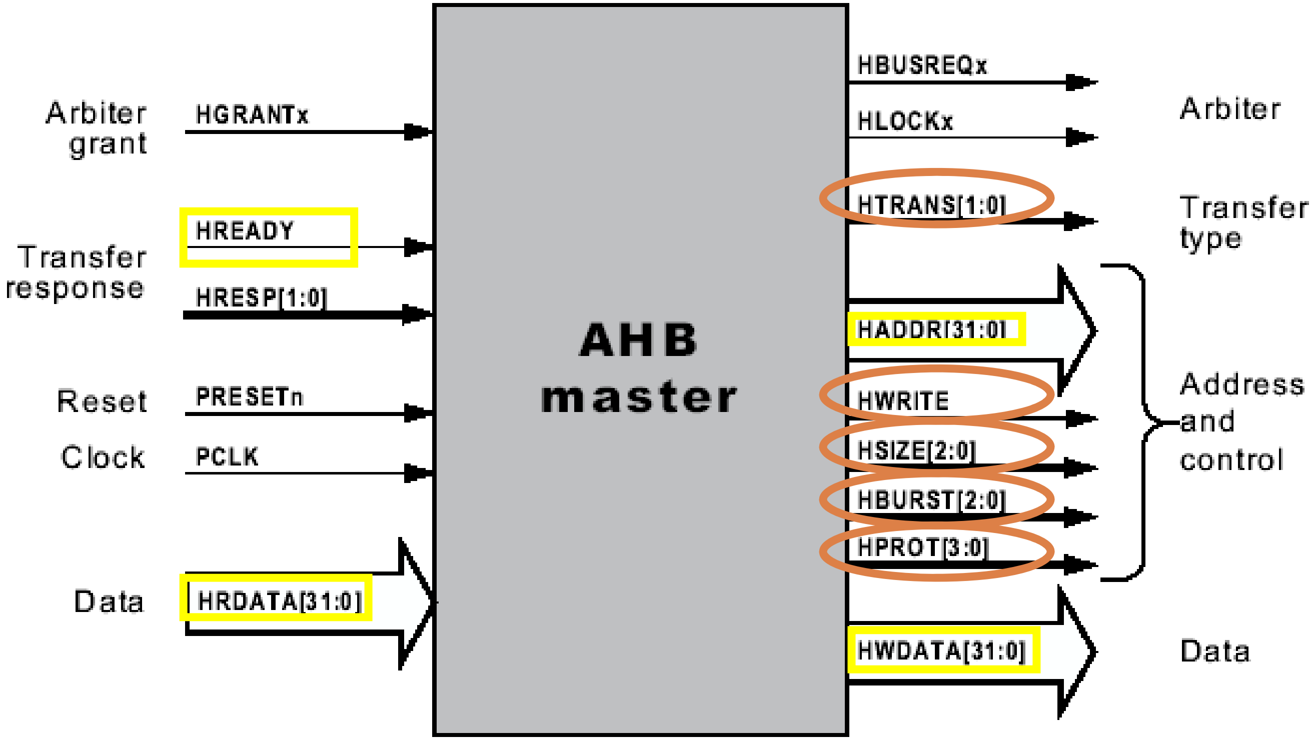


Burst Operation example



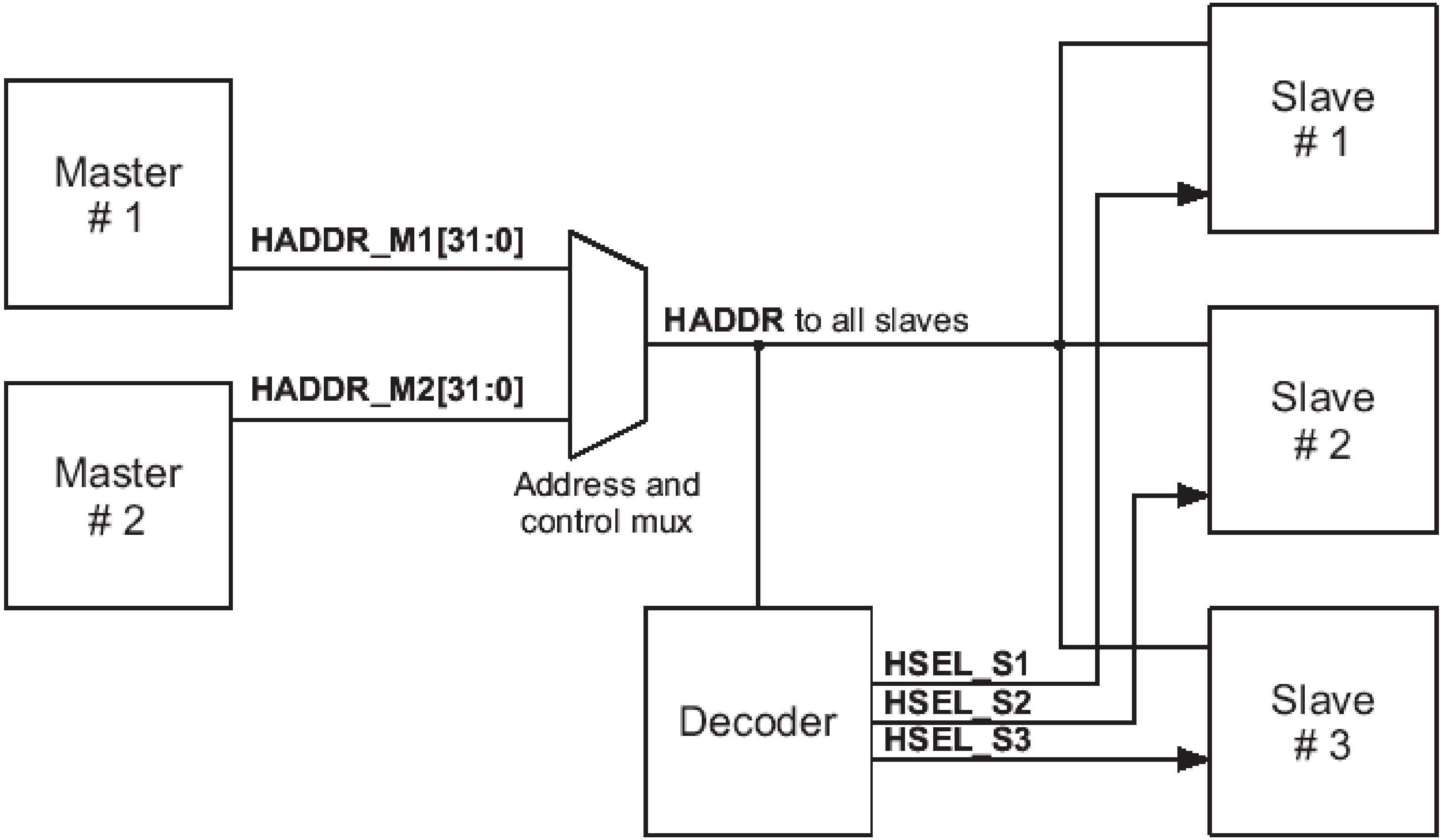
Undefined-length burst

Master---Others

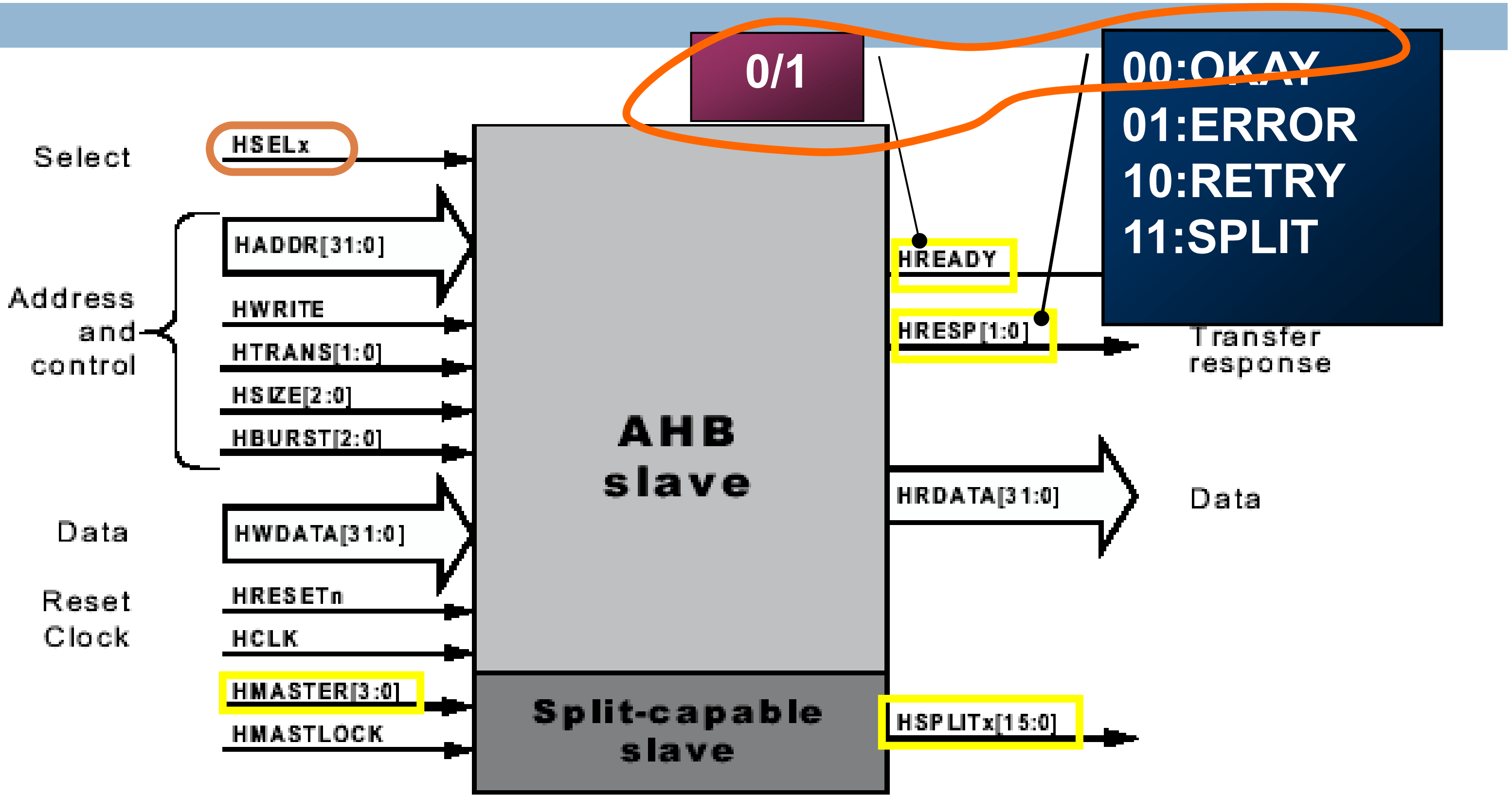


HBUSREQx : Master傳給Arbiter的編號
HWRITE : Write/Read (1/0)
HLOCKx : 要求完全的匯流排使用權

Address decoding

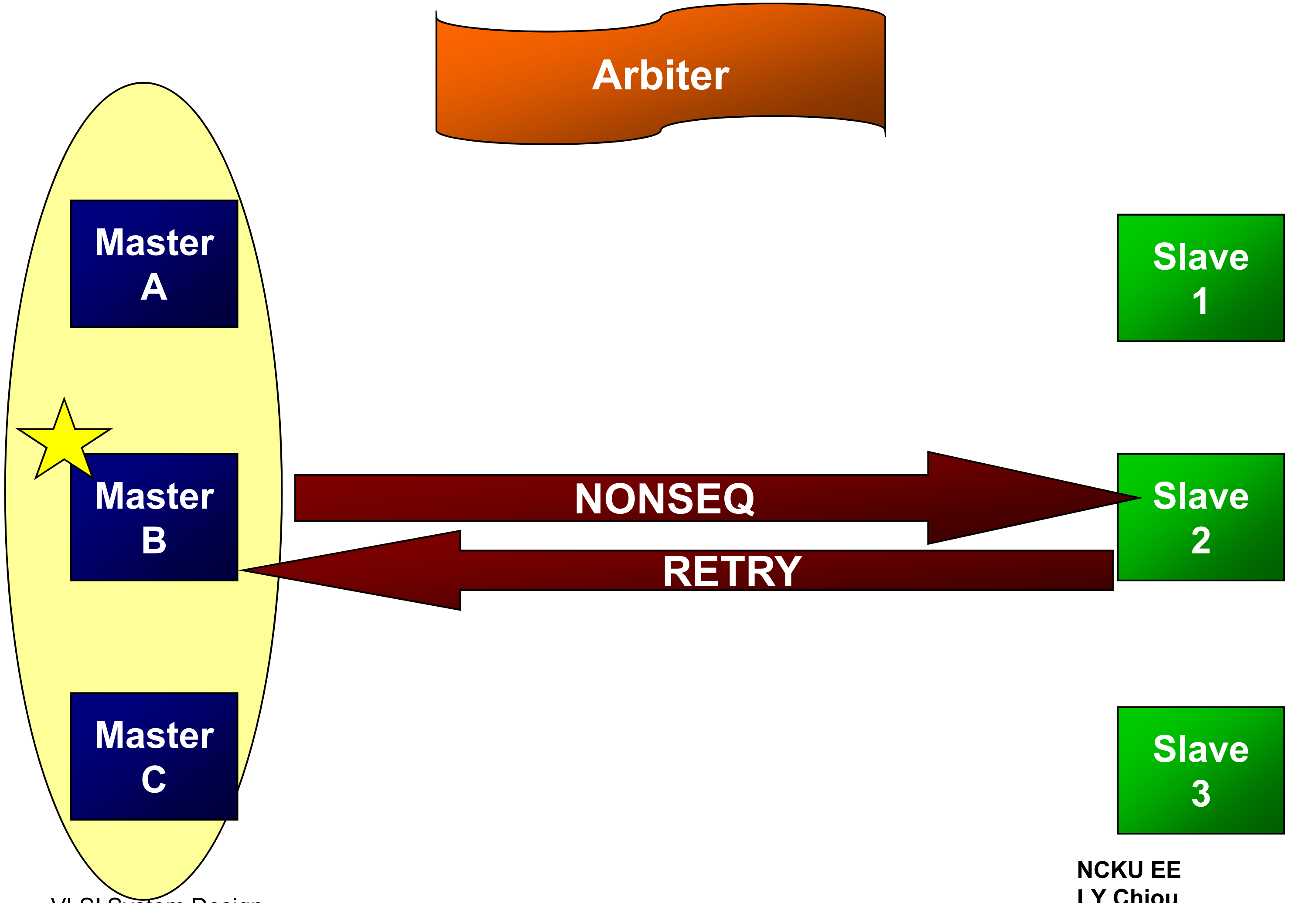


Slave --- transfer response

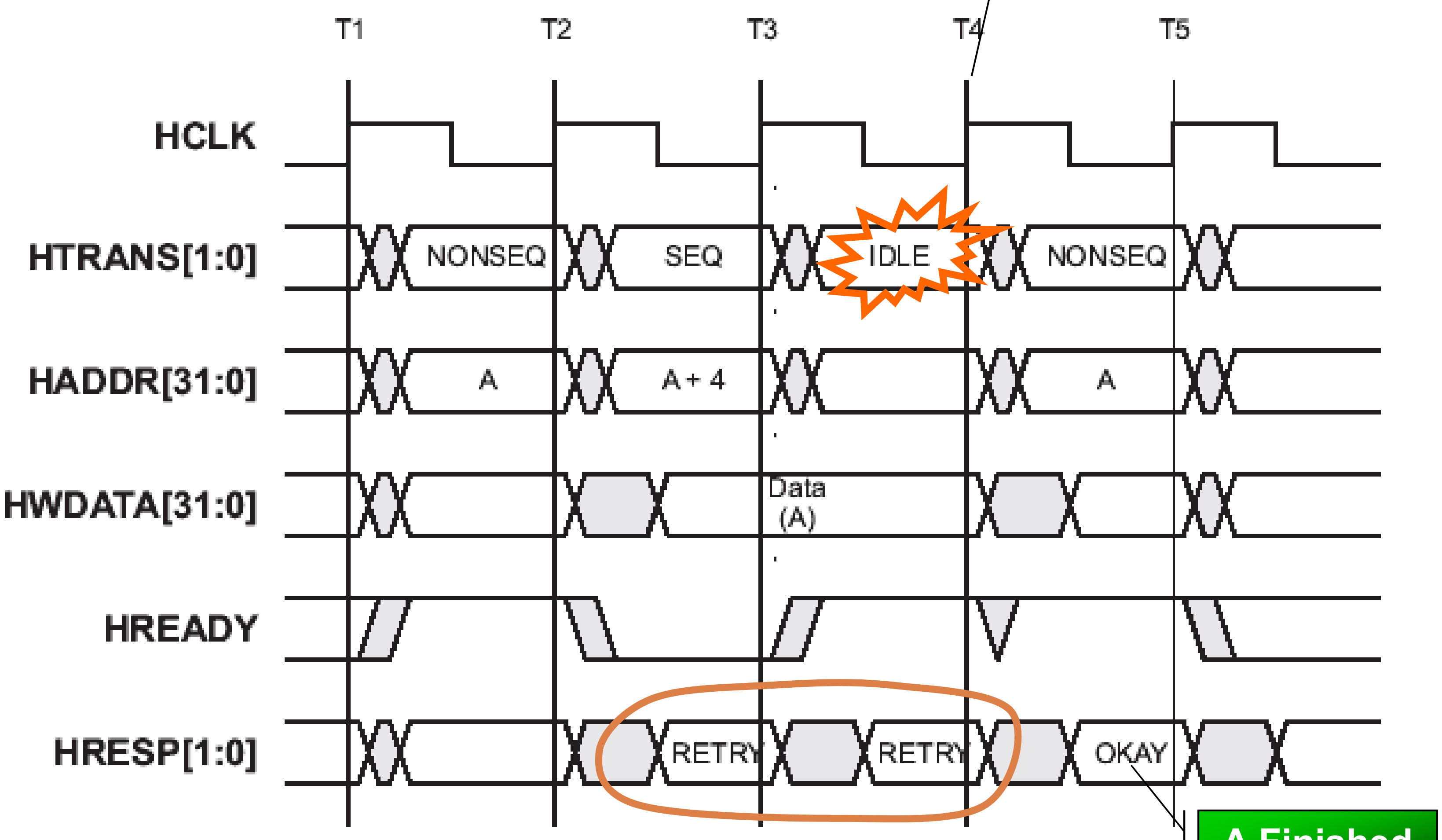


HREADY: extend transfer, end
HRESP : Slave結束時的status

RETRY


Arbiter
Master A
Master B
Master C
NONSEQ
RETRY
Slave 1
Slave 2
Slave 3

Retry Response

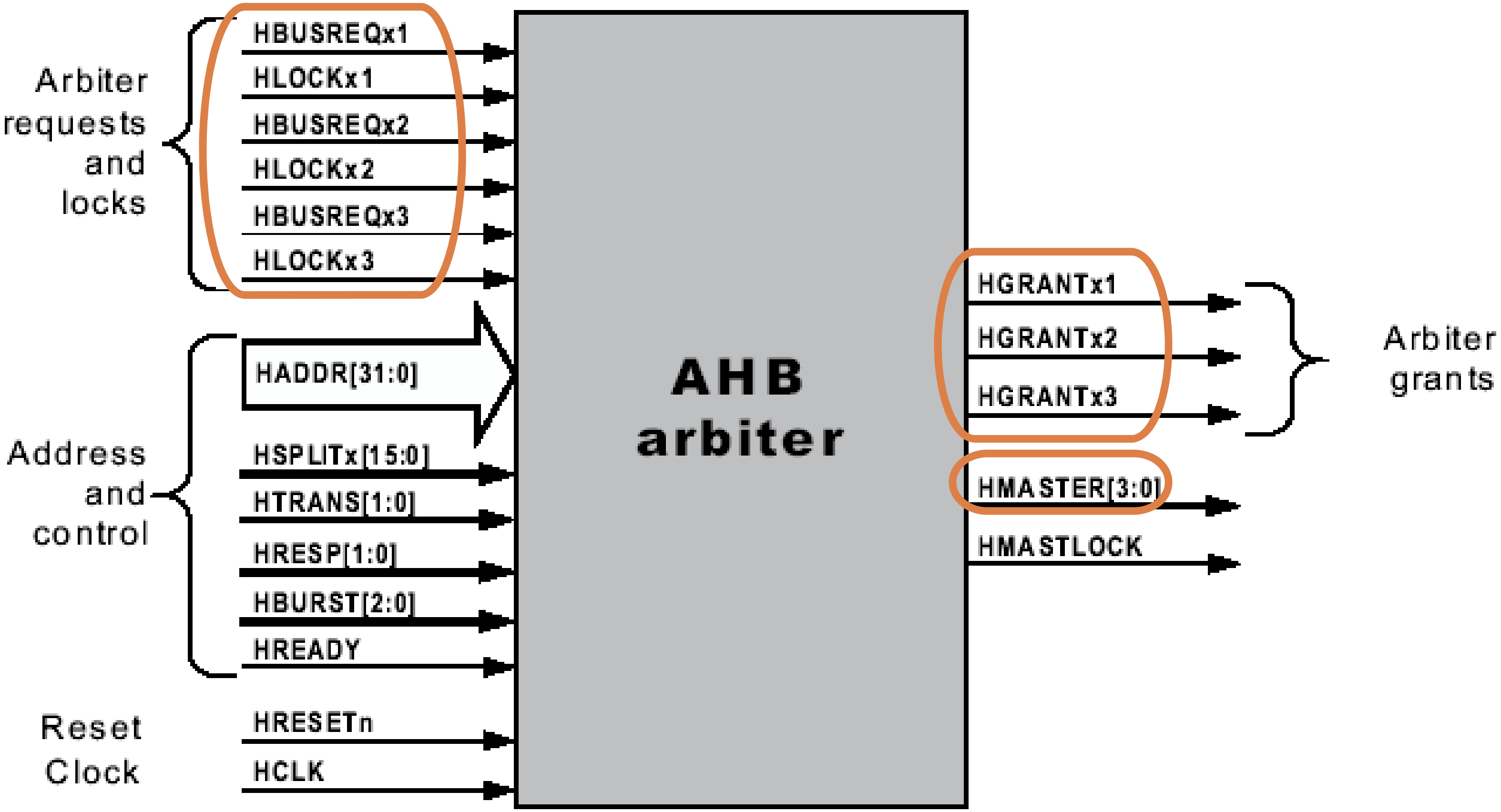


A : highest Priority

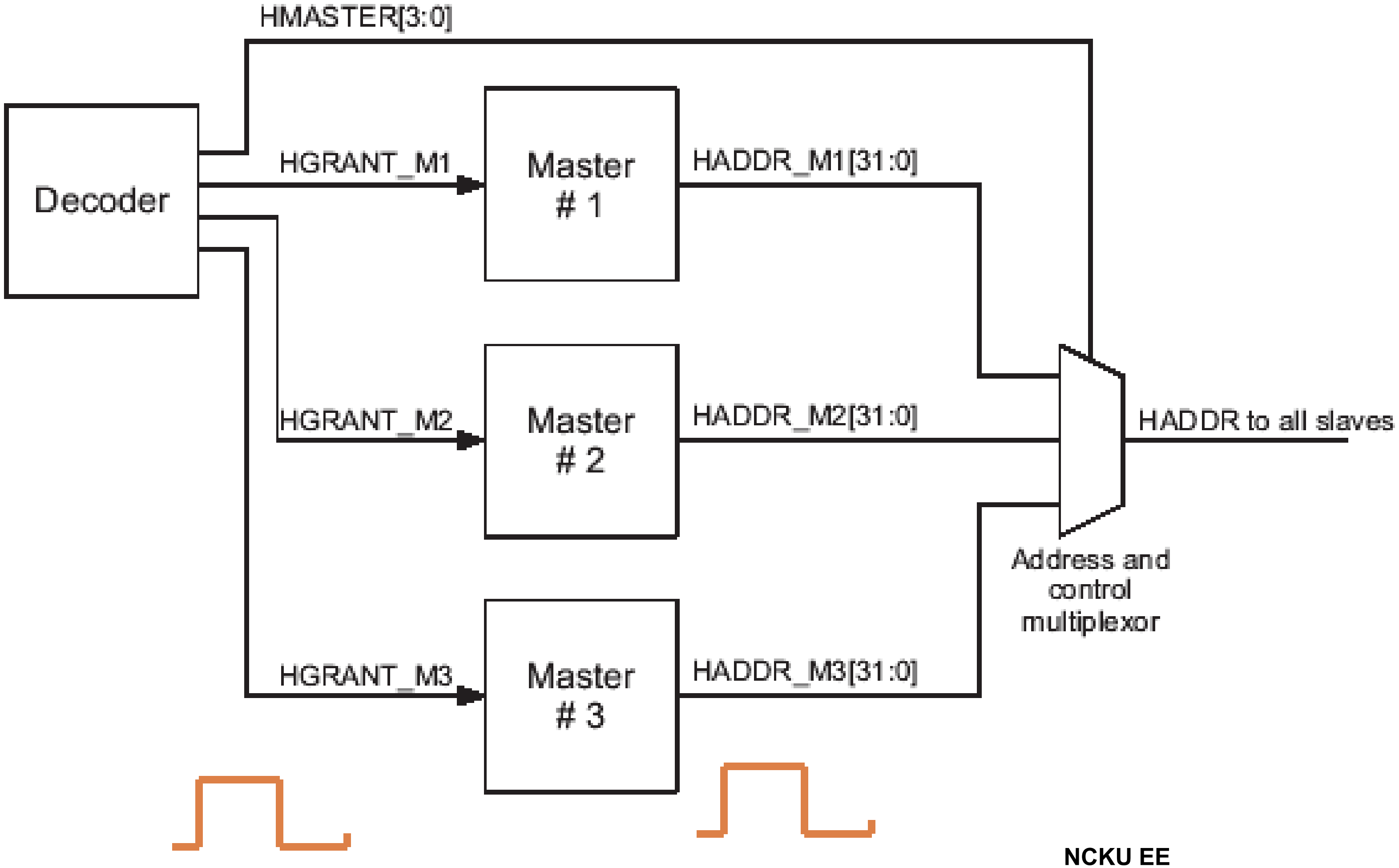
A Finished

OKAY:signal cycle; others more than one

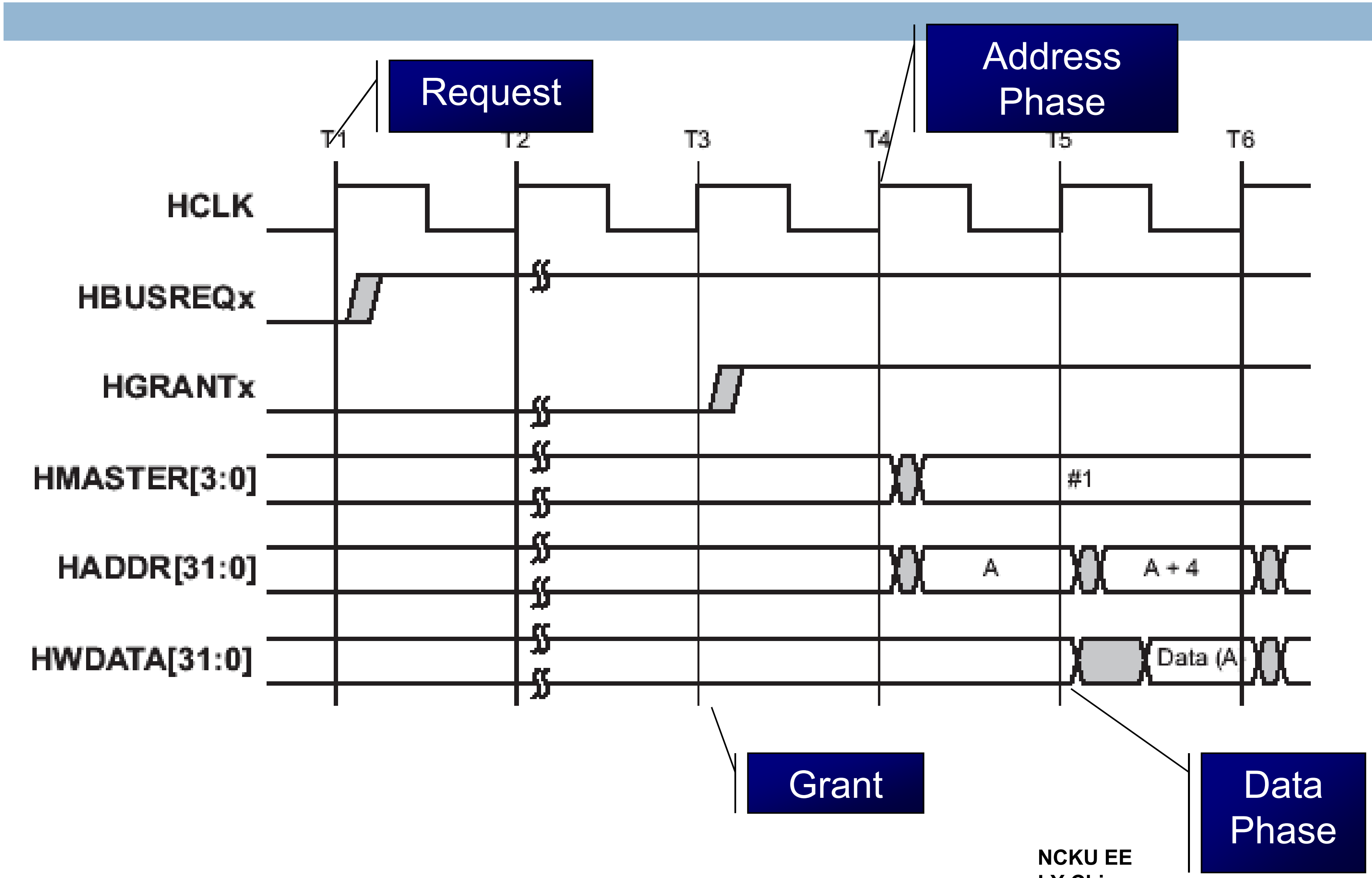
Arbiter


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

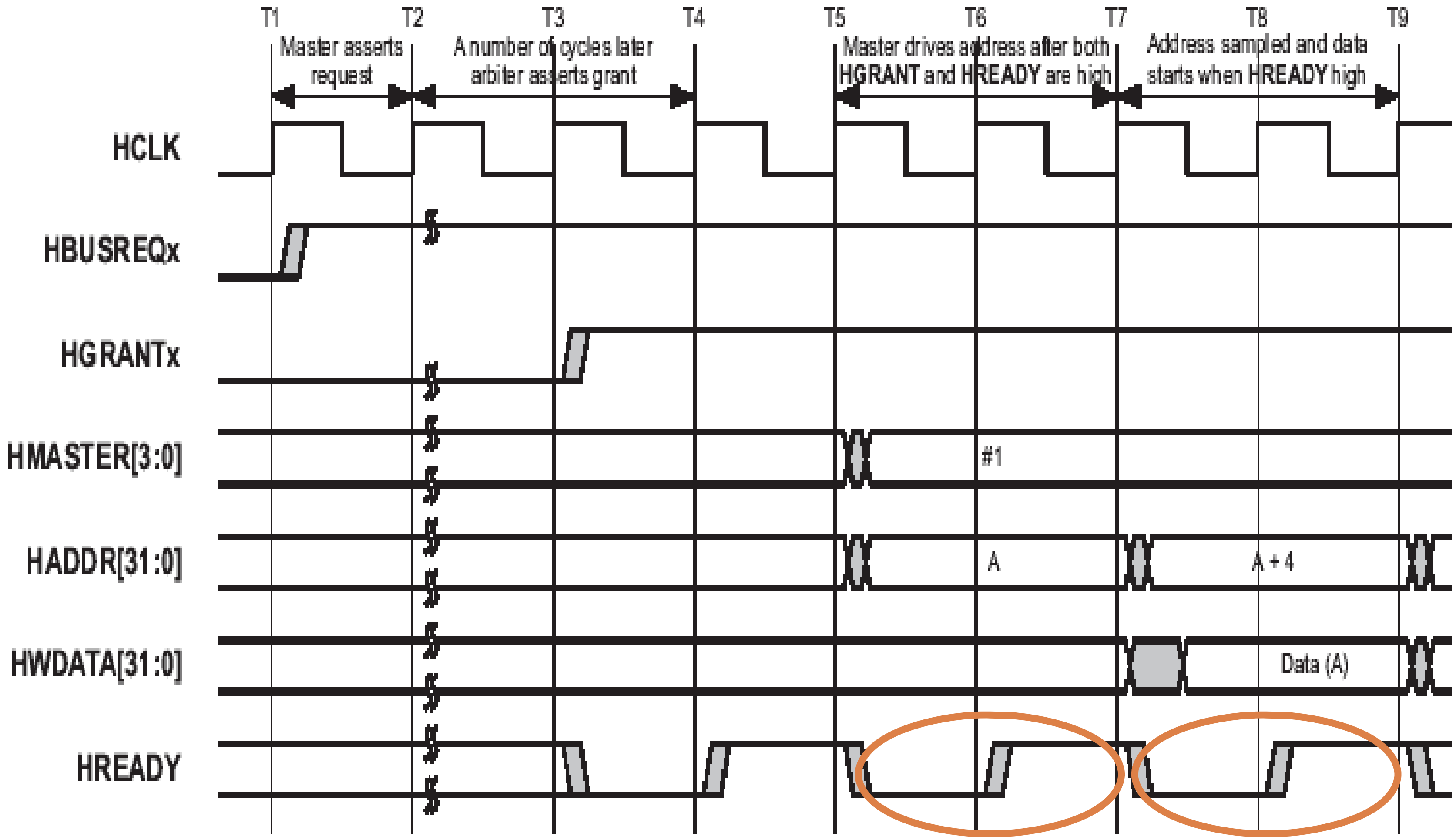
Bus master grant signals



Arbiter --- Grant access with no wait state



Arbiter --- Grant access with wait state

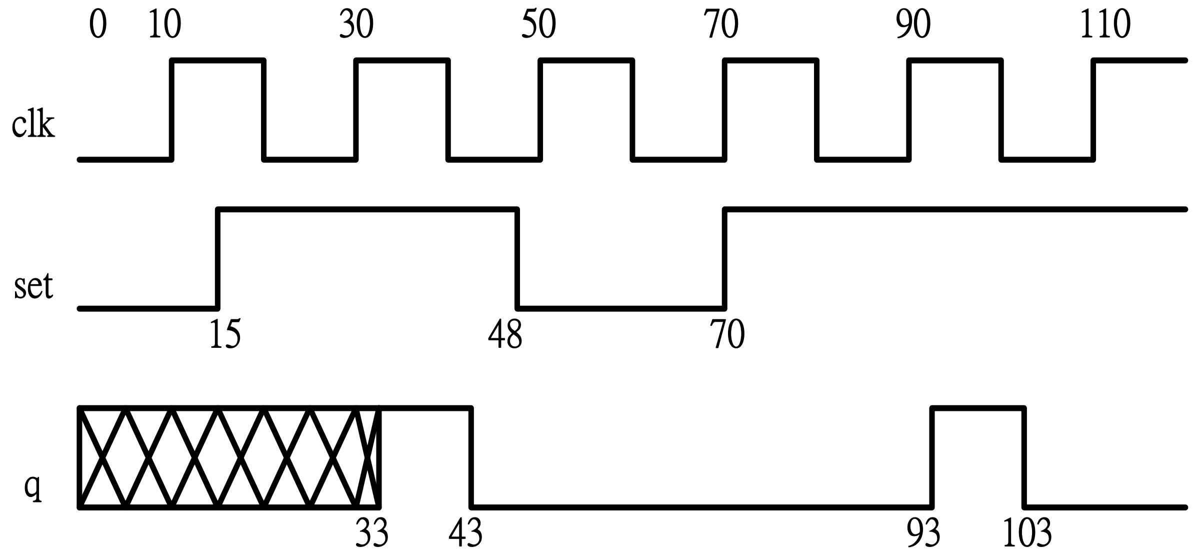

T1
T2
T3
T4
T5
T6
T7
T8
T9
Master asserts request
A number of cycles later arbiter asserts grant
Master drives address after both HGRANT and HREADY are high
Address sampled and data starts when HREADY high
HCLK
HBUSREQx
HGRANTx
HMASTER[3:0]
#1
HADDR[31:0]
A
A + 4
HWDATA[31:0]
Data (A)
HREADY

Timing Control in Procedural Blocks

- Three types of timing controls.
  - **Simple Delay**
    - **#10 rega = regb;**
    - **#(cycle/2) clk = ~clk; //cycle is declared as parameter**
  - **Edge-Trigger Timing Control**
    - **@(r or q) rega = regb; // controlled by "r" or "q"**
    - **@(posedge clk) rega = regb; // controlled by positive edge**
    - **@(negedge clk) rega = regb; // controlled by negative edge**
  - **Level-Triggered Timing Control**
    - **wait (!enable) rega = regb; // will wait until enable=0**

Timing Control in Procedural Blocks (continued)

```
always wait(set)
Begin @(posedge clk)
      #3 q = 1;
      #10 q = 0;
      wait(!set);
end
```



Read Task and Write Task

- Read
  - parameters: haddr, hsize, hburst
- Read(32’h48, `Hsize_Word, `Hburst_WRAP4)
  - address is 48, in word, burst by WRAP4
- Read(32’h64, `Hsize_Word, `Hburst_INCR8)
  - address is 48, in word, burst by INCR8

- Write
  - parameters: haddr, hsize, hburst, wdata
- Write(32’h48, `Hsize_Word, `Hburst_WRAP4, 32’ha0, 32’ha1,32’ha2, 32’ha3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
  - address is 48, in word, burst by WRAP4