

On Chip Bus

Speaker: 沈文中

Adopted from National
Taiwan University

SOC Course Material

Outline



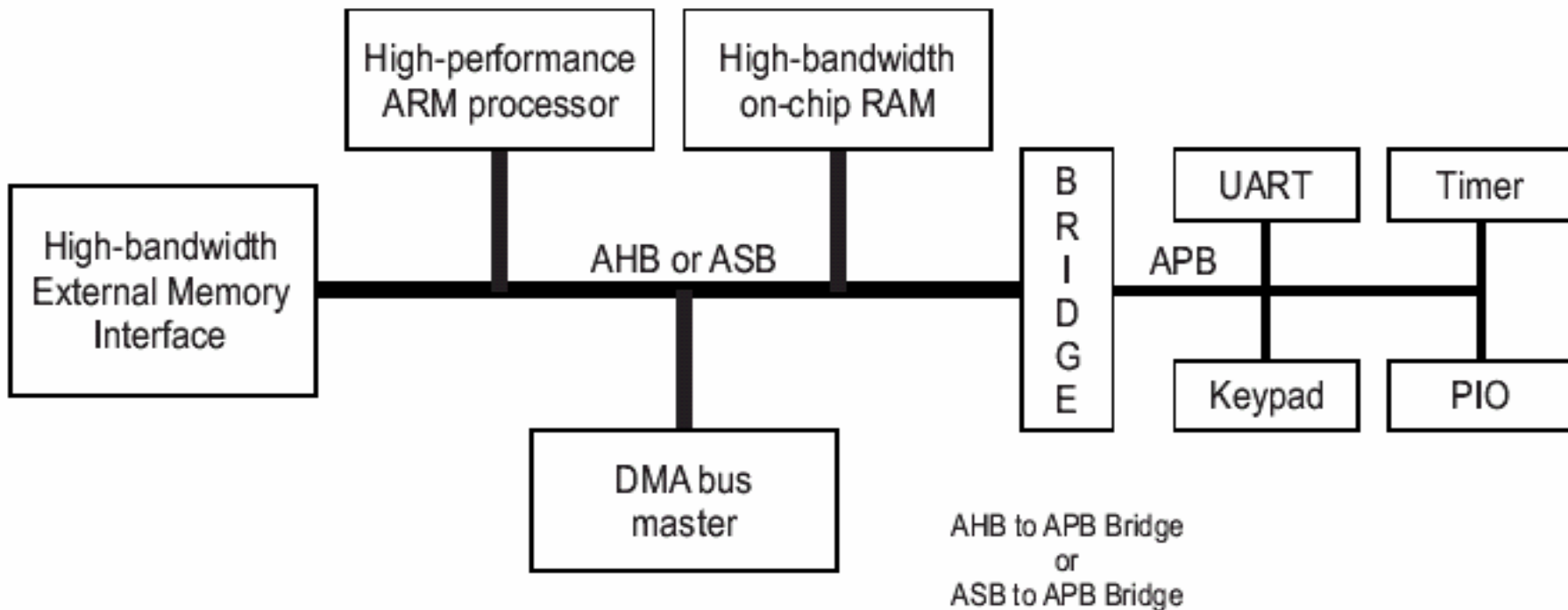
AMBA Bus

- Advanced System Bus
- Advanced High-performance Bus
- Advanced Peripheral Bus

IP Design flow

FPGA design flow

Bus Architecture



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

Outline



□ AMBA Bus

– Advanced System Bus

- High performance
- Pipelined operation
- Multiple bus master

– Advanced High-performance Bus

– Advanced Peripheral Bus

□ FPGA design flow

ASB characters



Negative edge trigger

Tri-state bus

- Drawback: More effort used to control timing
- Advantage: cost less area

Outline



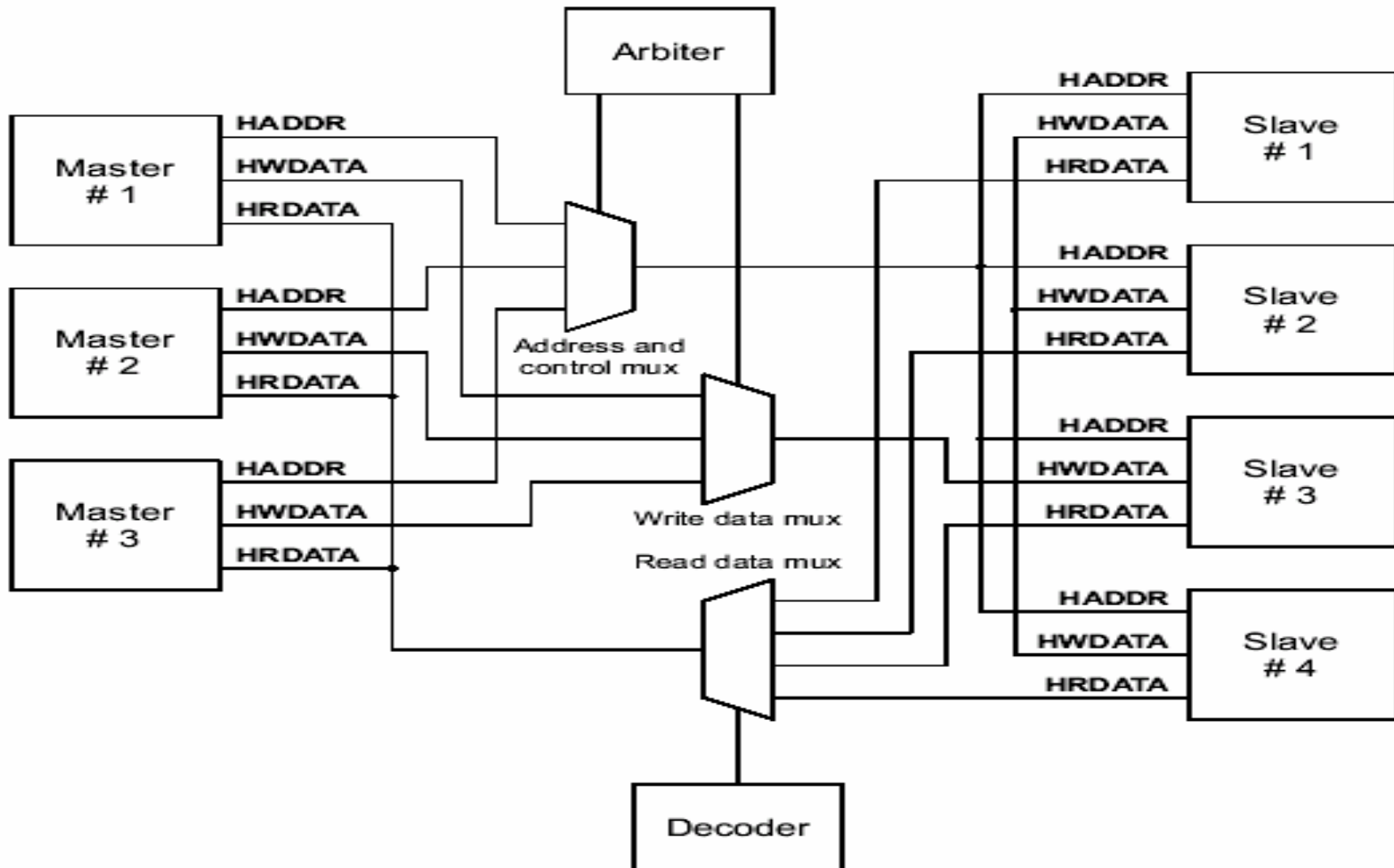
AMBA Bus

- Advanced System Bus
- **Advanced High-performance Bus**
 - High performance
 - Pipelined operation
 - Multiple bus master
 - Burst transfers
 - Split transactions
- Advanced Peripheral Bus

IP Design flow

FPGA design flow

AHB simple Arch.



AHB Components



□ AHB Components

- **AHB master** is able to initiate read and write operations by providing an address and control information. Only one bus master is allowed to actively use the bus at any one time.(max. 16)
- **AHB slave** responds to a read or write operation within a given address-space range. The bus slave signals back to the active master the success, failure or waiting of the data transfer.

AHB Components(ii)



□ AHB Components

- **AHB arbiter** ensures that only one bus master at a time is allowed to initiate data transfers.
- **AHB decoder** is used to decode the address of each transfer and provide a select signal for the slave that is involved in the transfer. A single centralized decoder is required in all AHB implementations.

AHB Signals(i)



□ AHB Signals can be classified as

- Clock (HCLK)
- Address and read/write data (HADDR, HRDATA, HWDATA)
- Arbitration (HGRANT_x, HMASTER, HMASTLOCK,...)
- Control signal (HRESET_n,...)
- Response signal(HREADY, HRESP)

□ Transfer signals

- **HCLK**
 - bus clock. All signal timings are related to the rising edge.
- **HADDR[31:0]**
 - 32 bits system bus
- **HWDATA/HRDATA [31:0]**
 - 32 bits write/read data bus
- **HWRITE**
 - High: write data
 - Low: read data
- **HREADY**
 - Transfer done

AHB Signals(ii)

Basic Transfer



- Each transfer consists of
 - An address and control cycle
 - One or more cycles for the data

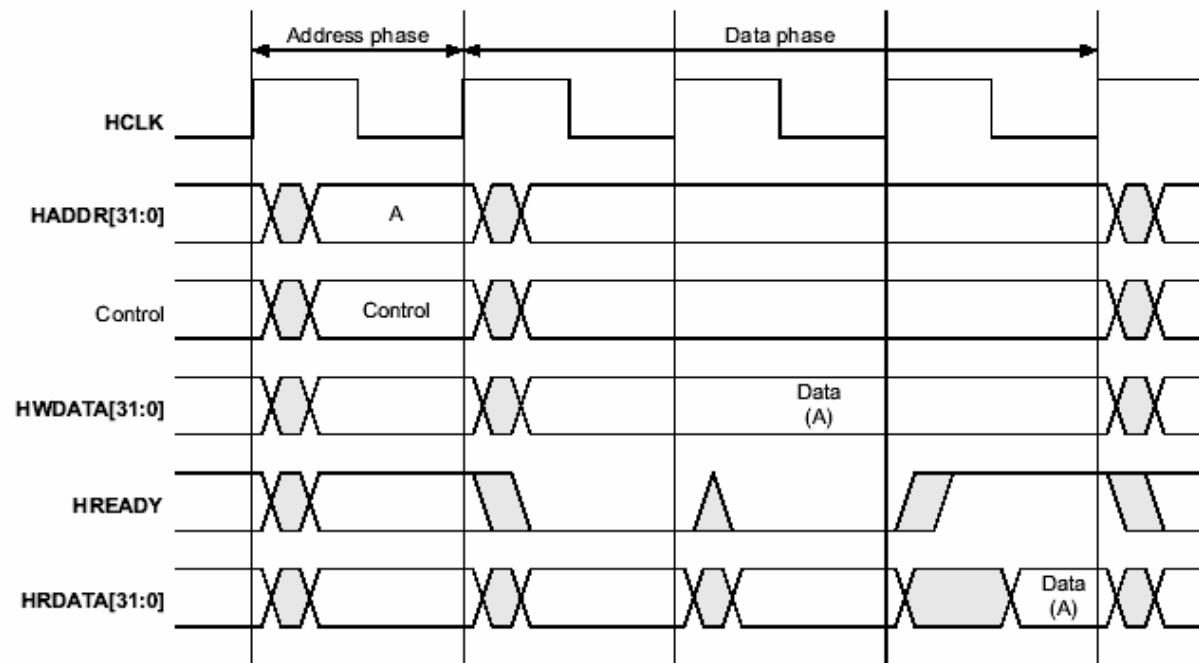


Figure 3-4 Transfer with wait states

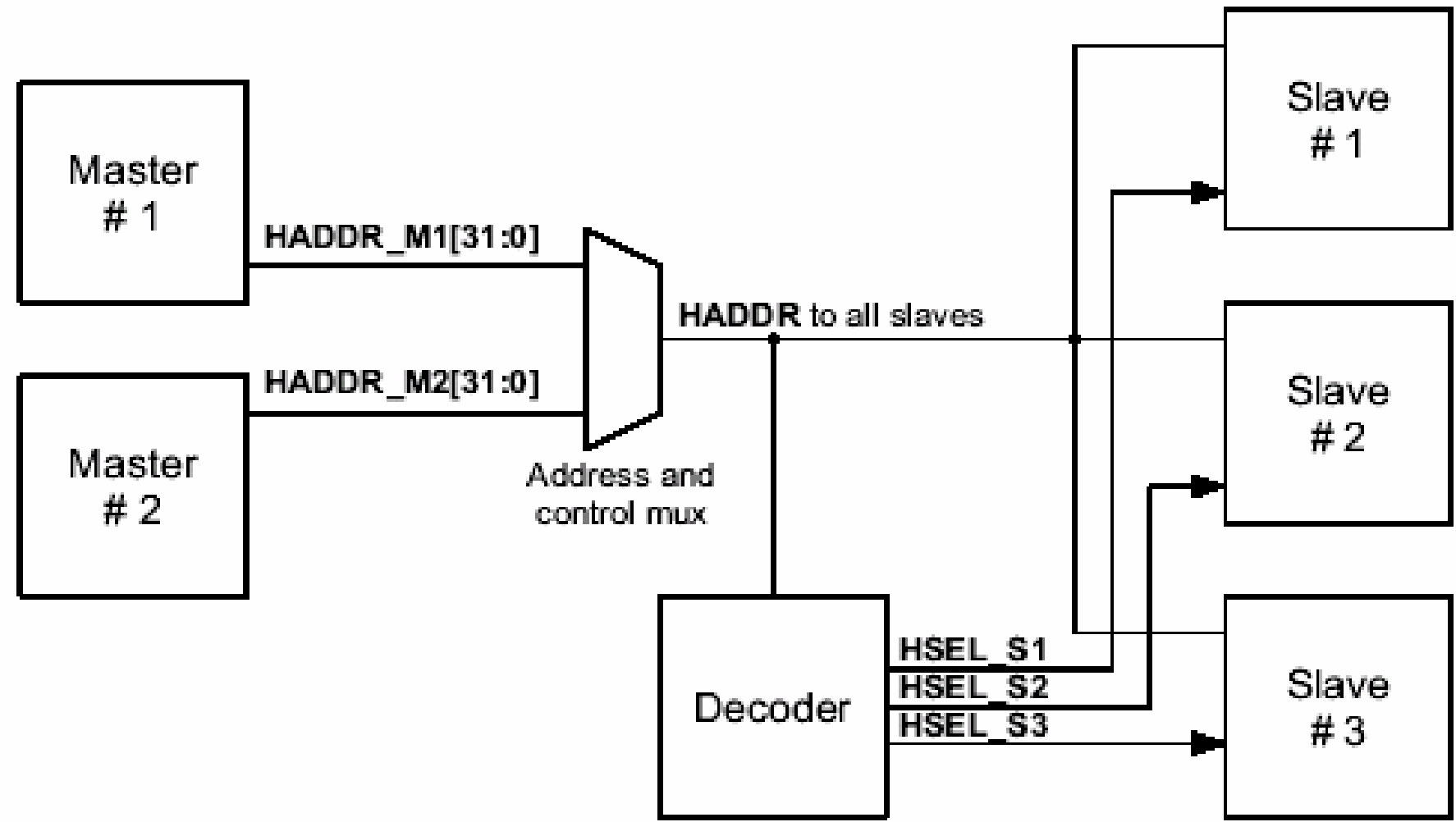


Figure 3-12 Slave select signals

AHB Signals(iii)



□ Control signals

- HTRANS[1:0]
 - Current transfer type
- HBURST[2:0]
 - When sequential transfer, control transfer relation
- HSIZE[2:0]
 - Control transfer size= 2^{HSIZE} bytes(max=1024bits)
- HPROT[3:0]
 - Protection data

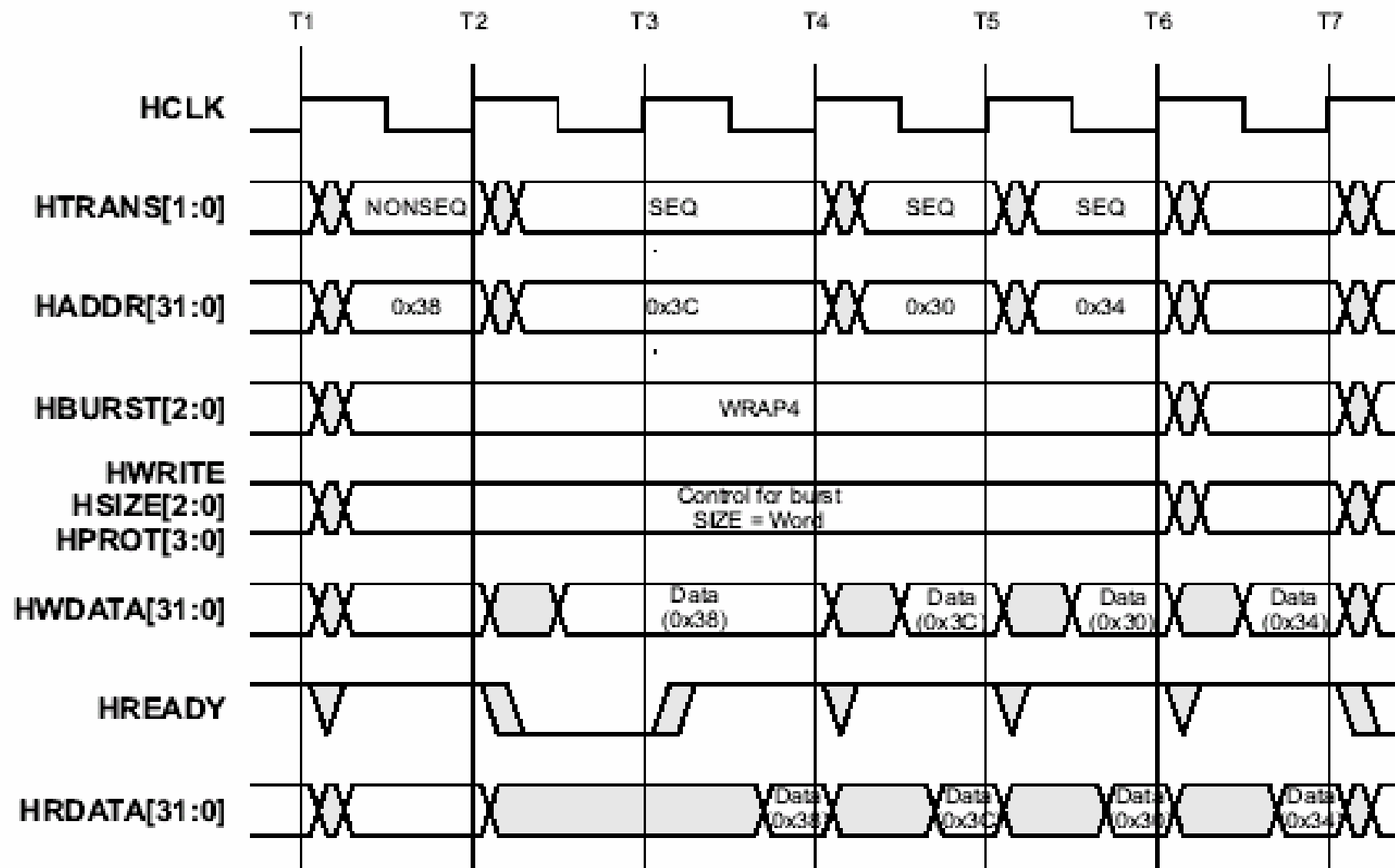
AHB Signals(iii)-HTRANS



□ HTRANS[1:0]

- IDLE: master don't need data to be transferred
- BUSY: allows bus masters to insert IDLE cycles in the middle of bursts of transfers.
- NONSEQ: The address and control signals are unrelated to the previous transfer.
- SEQ: the address is related to the previous transfer.

AHB Signals(iii)-HBURST



AHB Signals(iii)-HBURST



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

AHB Signals(iv)



□ Response signals

– HREADY

- Transfer done, ready for next transfer

– HRESP[1:0]

- **OKAY** transfer complete
- **ERROR** transfer failure(ex: write ROM)
- **RETRY** higher priority master can access bus
- **SPLIT** other master can access bus

AHB Signals(v)



□ Arbiter signals

- HGRANT_x
 - Select active bus master
- HMASTER[3:0]
 - Multiplex signals that sent from master to slave
- HMASTLOCK
 - Locked sequence

Master signal

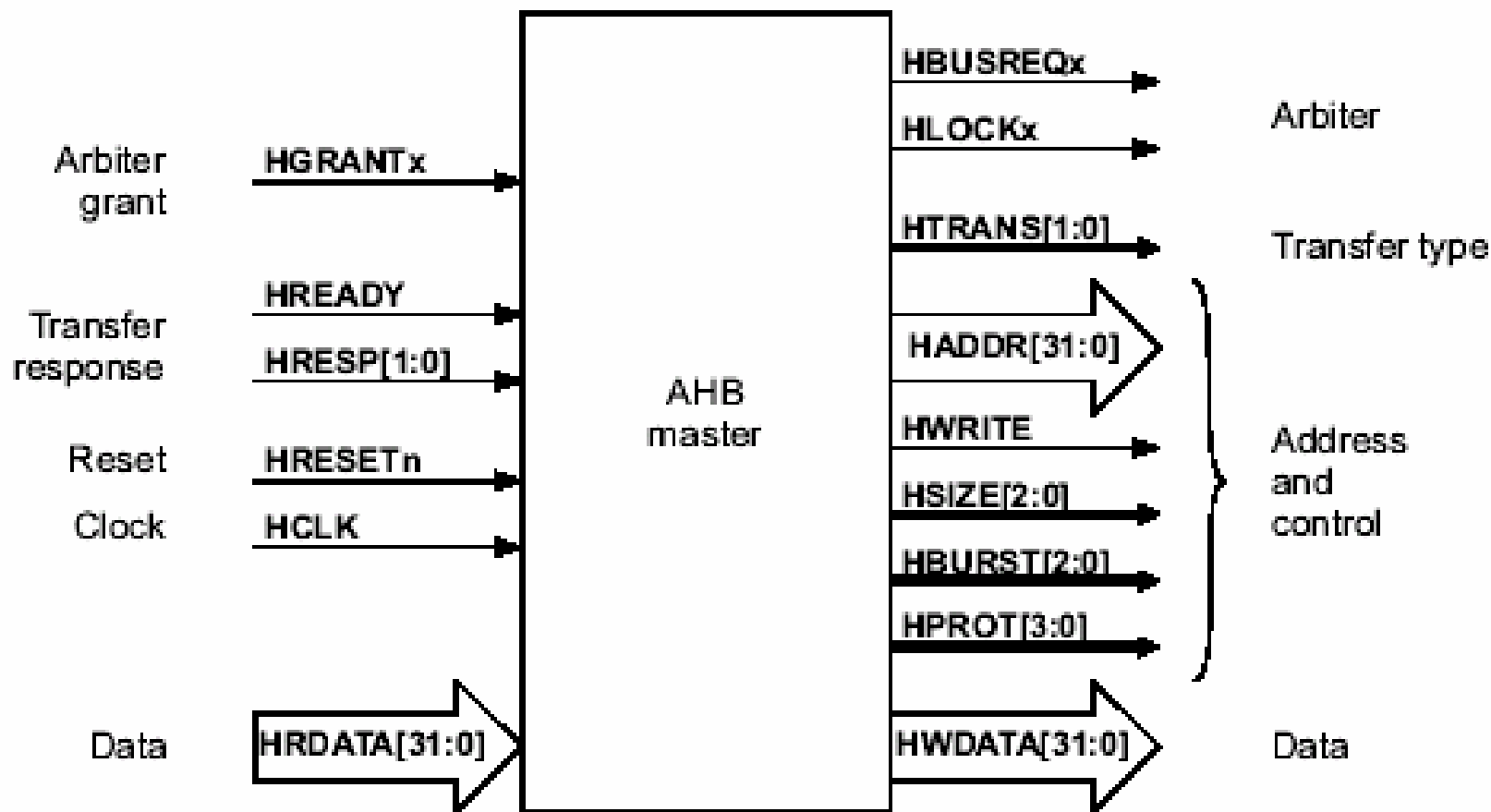
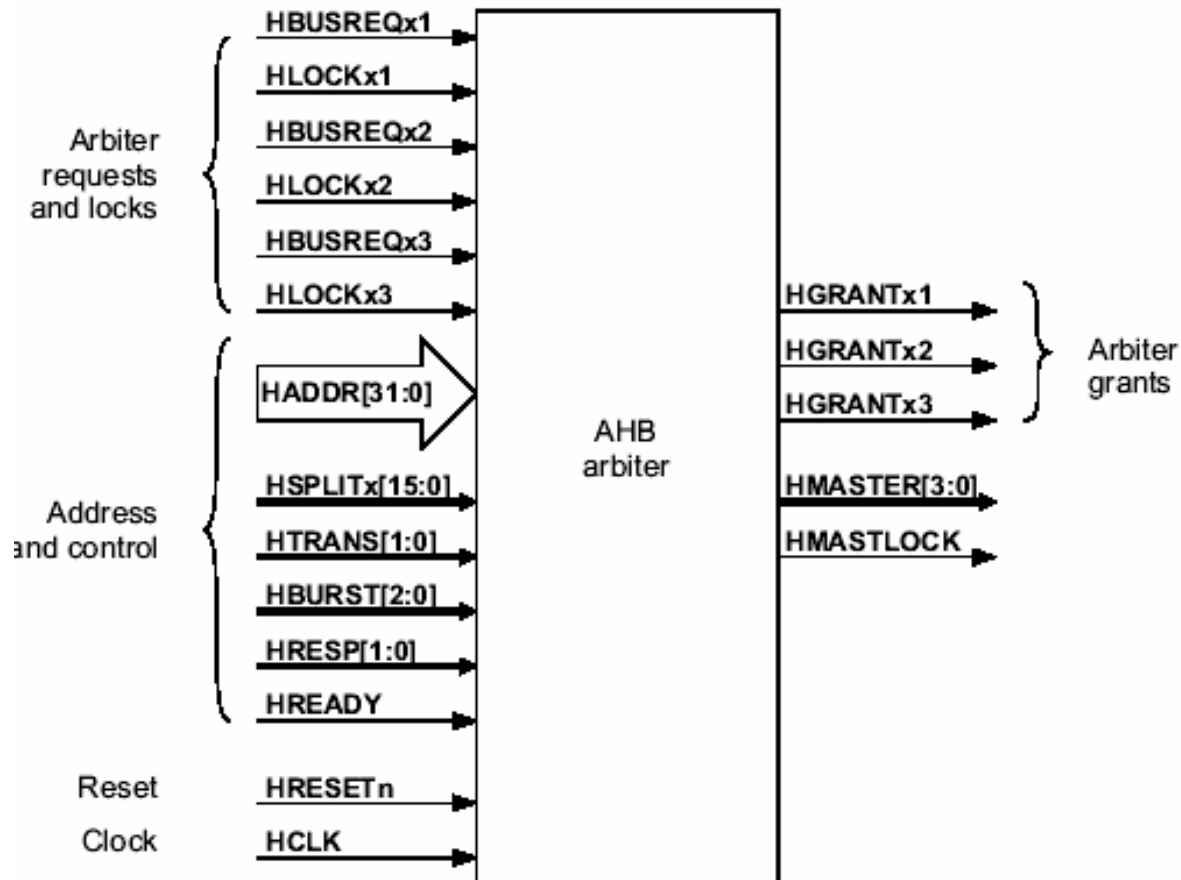
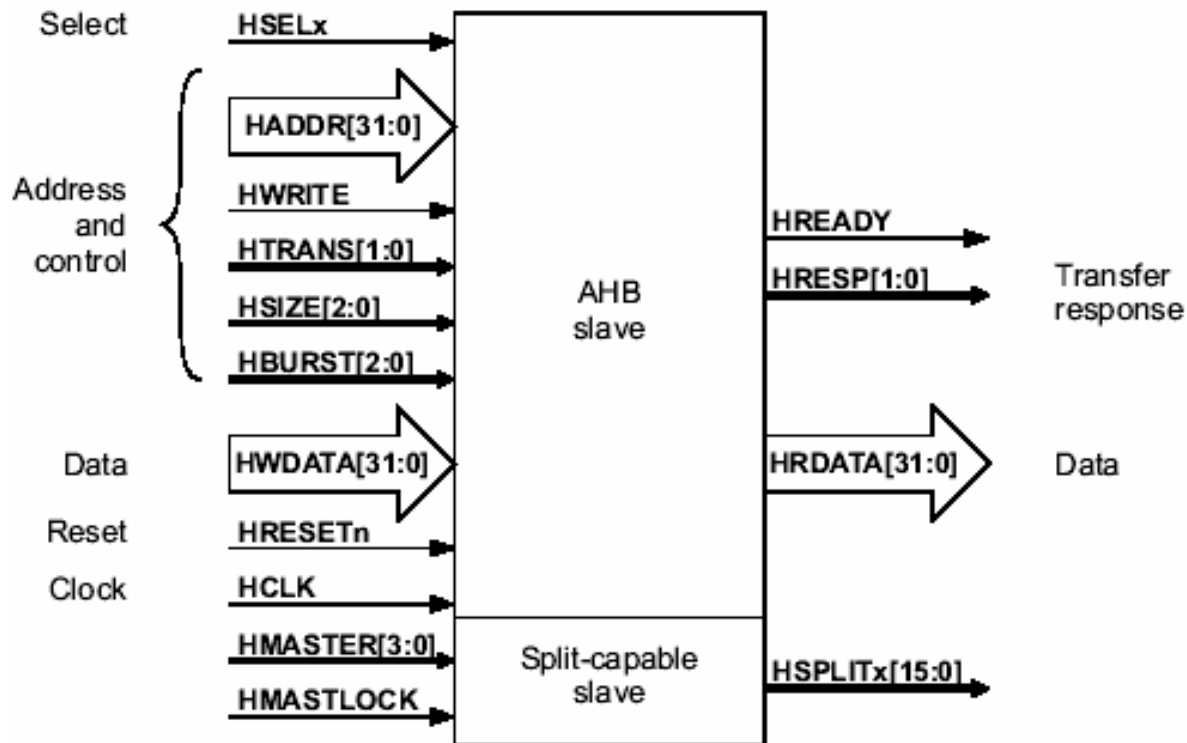


Figure 3-27 AHB bus master interface diagram

Arbiter signal



Slave signal



Outline



□ AMBA Bus

- Advanced System Bus
- Advanced High-performance Bus
- **Advanced Peripheral Bus**
 - Low power
 - Latched address and control
 - Simple interface
 - Suitable for many peripherals

□ IP Design flow

□ FPGA design flow

APB state diagram

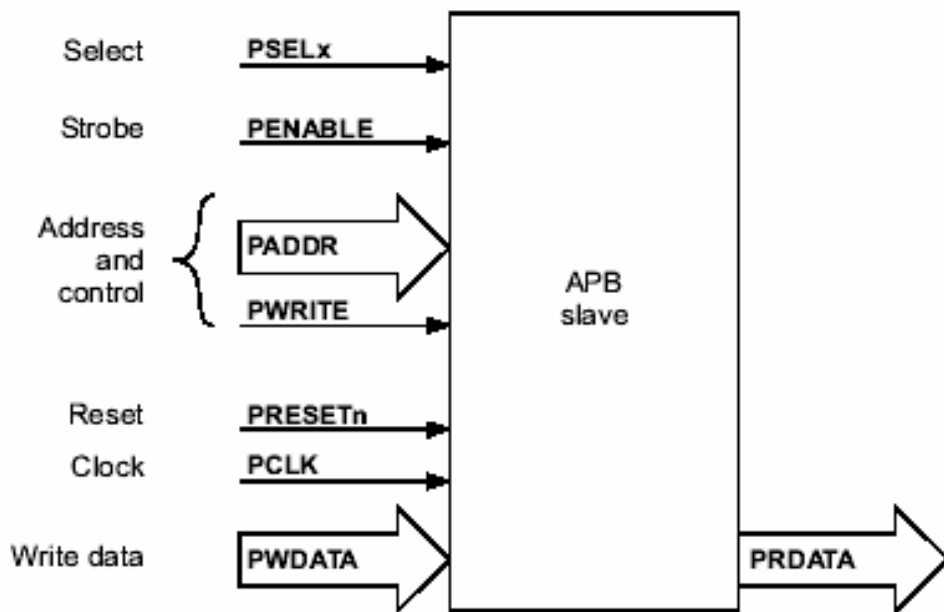


Figure 5-7 APB slave interface description

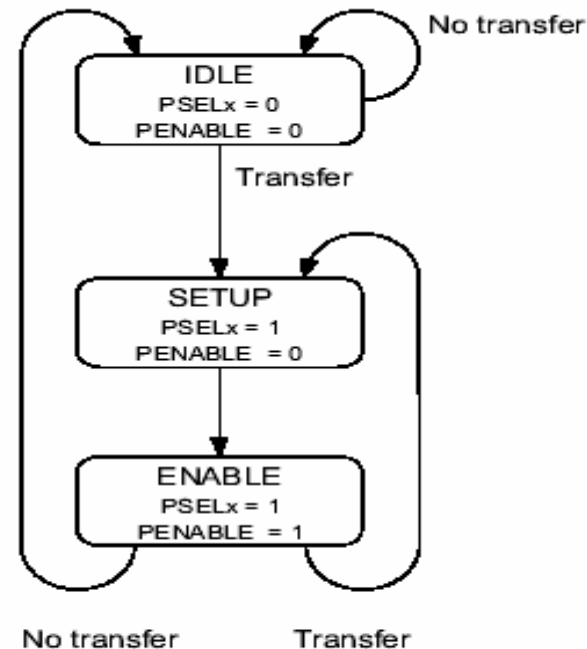


Figure 5-2 State diagram

APB signals



□ APB character

- Always two cycle transfer
- No wait cycle and response signal

□ APB signals

- **PCLK** Bus clock , rising edge is used to time all transfers.
- **PRESETn** APB reset, active Low.

APB signals



- **PADDR[31:0]** APB address bus.
- **PSELx** Indicates that the slave device is selected.
There is a PSELx signal for each slave.
- **PENABLE** Indicates the second cycle of an APB transfer.
- **PWRITE** Transfer direction. High for write access,
Low for read access.
- **PRDATA** Read data bus
- **PWDATA** Write data bus

Outline



□ AMBA Bus

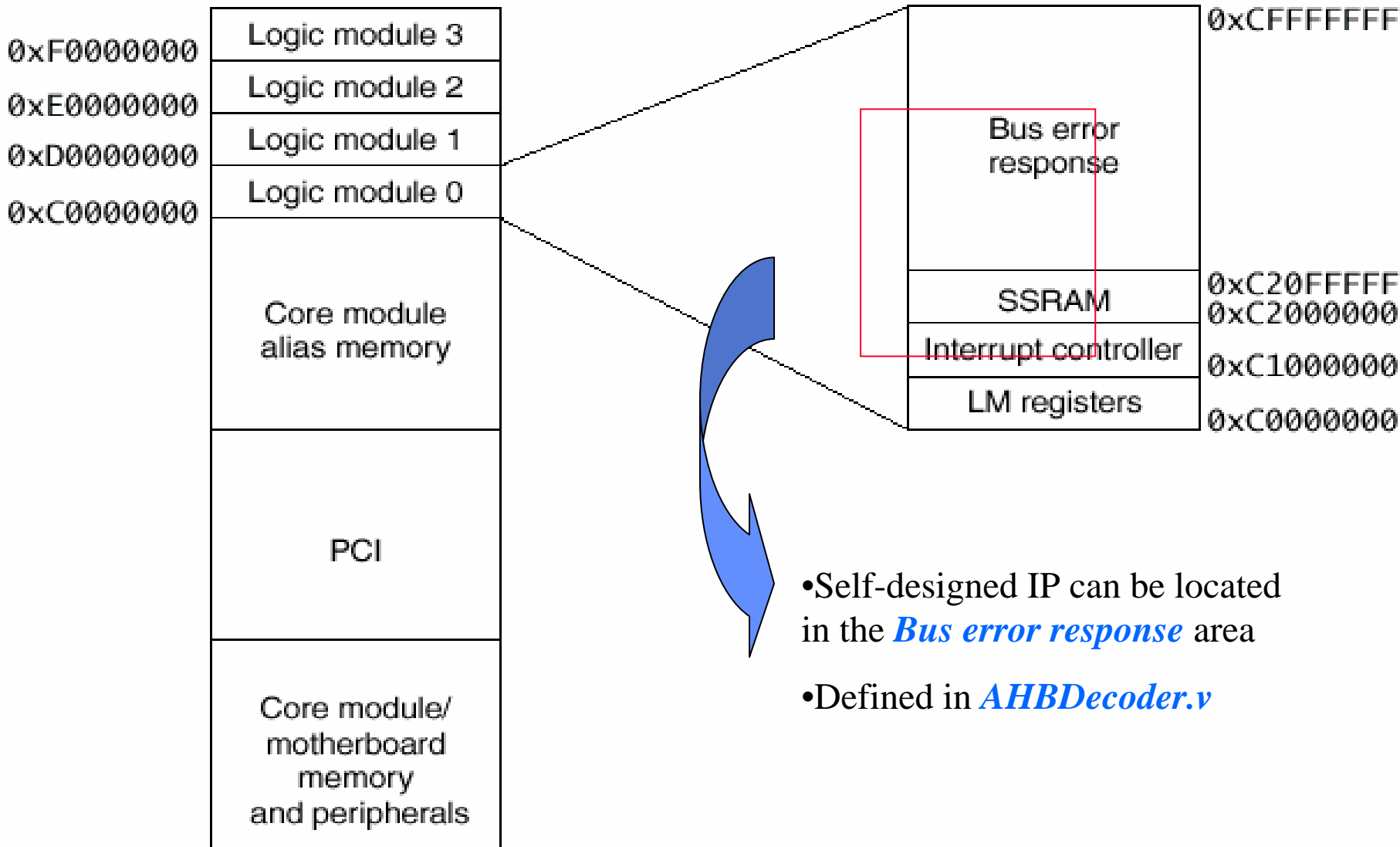
- Advanced System Bus
- Advanced High-performance Bus
- Advanced Peripheral Bus

□ IP Design flow

- Memory definition
- IP design-SW
- IP design-HW

□ FPGA design flow

Memory definition



□ Software part

- Write a function to control hardware
- Delay number of clocks by **NOOP (asm)** instruction

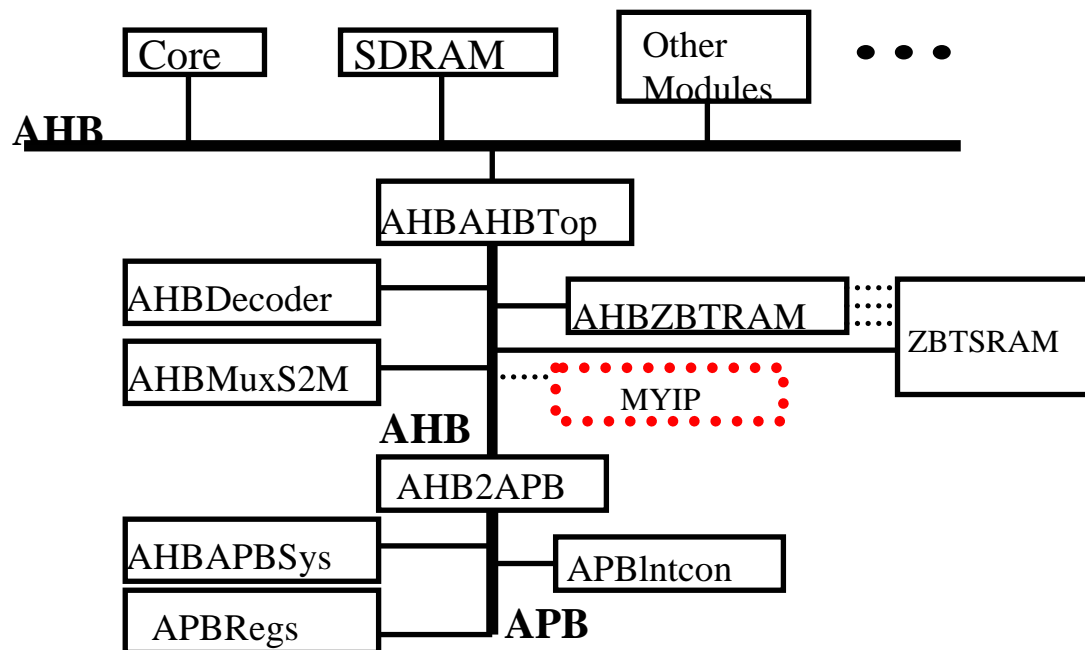
□ Hardware part

- Add MYIP.v into top module
- Change ahbdecoder.v
- Change AHBmuxS2M.v
- Change ahbahbtop.v

IP design-HW



- Add MYIP.v into top module
 - Write your own module in AMBA interface

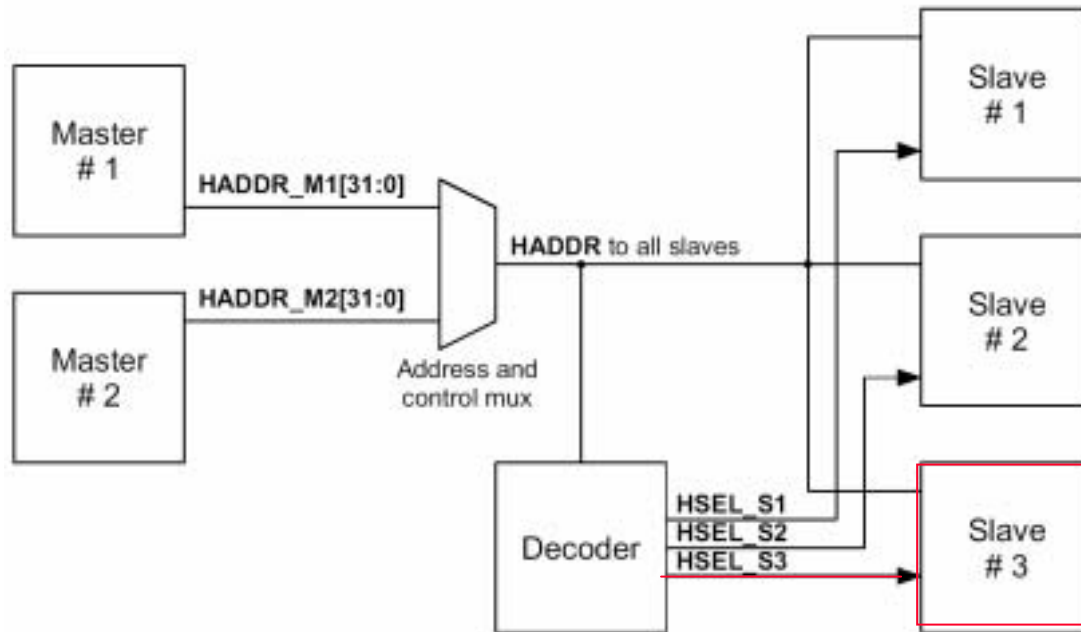


IP design-HW



□ Change ahbdecoder.v

- Add **HSELMYIP** signal to select your own slave IP to response
- Address are defined in decoder

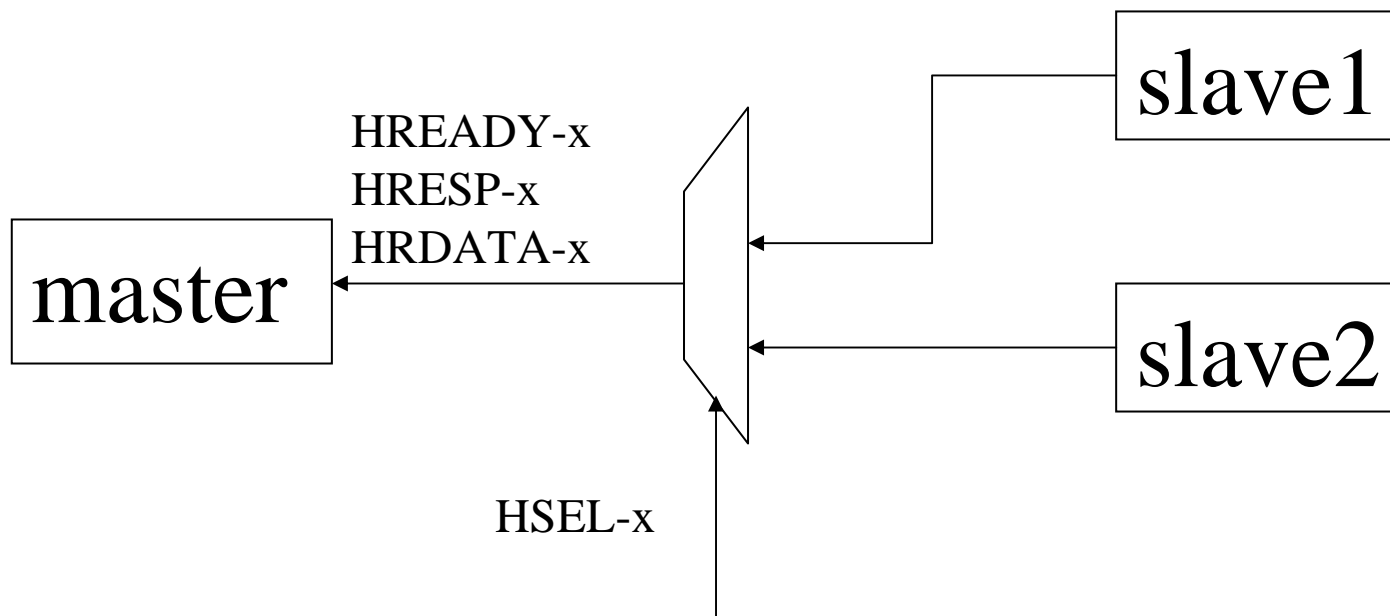


IP design-HW



□ Change AHBMuxS2M.v

- Use mux to select slave which can use HRDATA

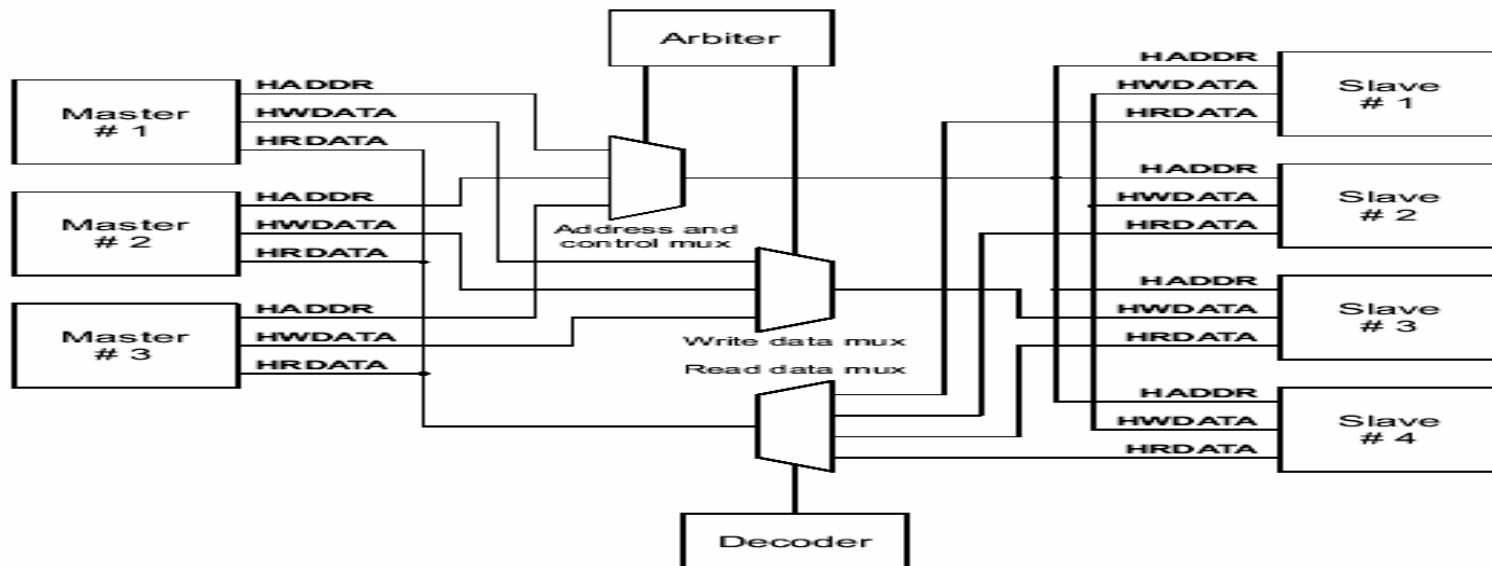


IP design-HW

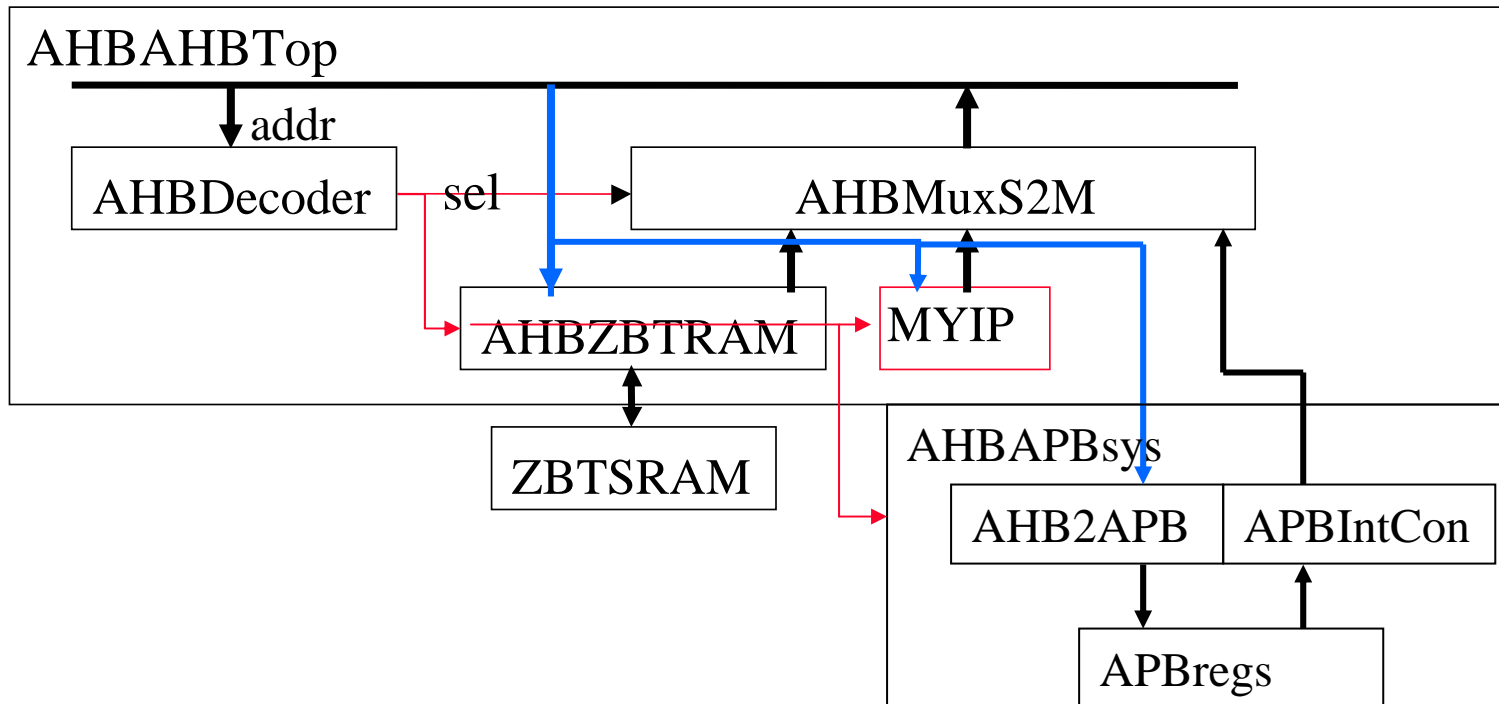


□ Change ahbahbtop.v

- Add your module in AMBA Bus
- Connect the above net connection



Architecture



Outline



□ AMBA Bus

- Advanced System Bus
- Advanced High-performance Bus
- Advanced Peripheral Bus

□ IP Design flow

□ FPGA design flow

- Compile flow
- Download flow

Compile flow(i)



- ❑ All verilog module must be synthesized by Xilinx Software

Project Name: lab5

Project Location: D:\SOC\lab5\lab5

Project Device Options:

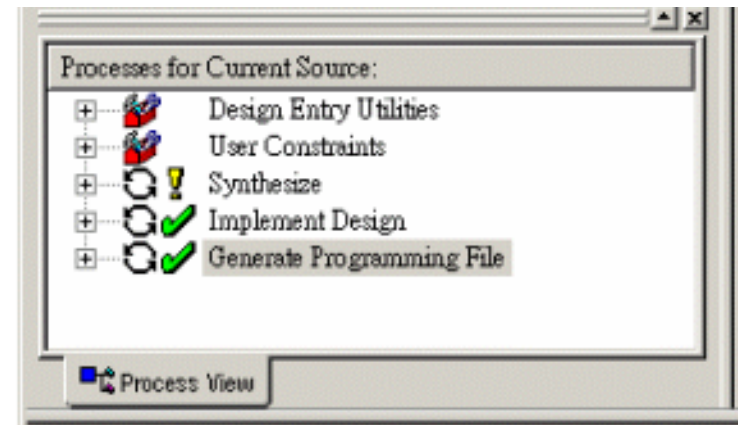
Property Name	Value
Device Family	VirtexE
Device	xcv2000e
Package	fg680
Speed Grade	-6
Design Flow	XST Verilog

OK Cancel Help

Compile flow(ii)



- ❑ Add **example2.ucf** (define the pin assignment) into project
- ❑ Double click **generate programming file** to generate *.bit (which can be downloaded into FPGA)



Outline



□ AMBA Bus

- Advanced System Bus
- Advanced High-performance Bus
- Advanced Peripheral Bus

□ IP Design flow

□ FPGA design flow

- Compile flow
- Download flow

Download flow



- Connect **config link**
- Connect Multi-ICE to **Logic Module**
- Power on
- Use **progcrd.exe** to download **example2.bit** file
- Remove **config link**
- Power off