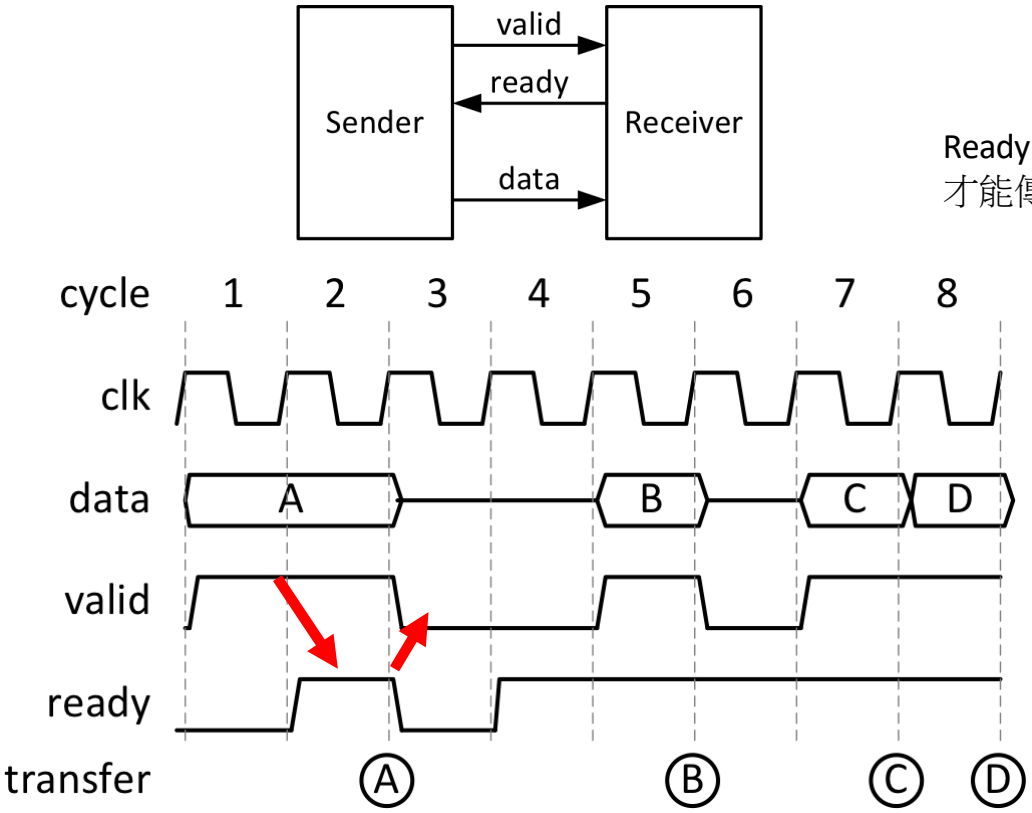
DCS Lab 5
AHB interconnect

Finite state machine

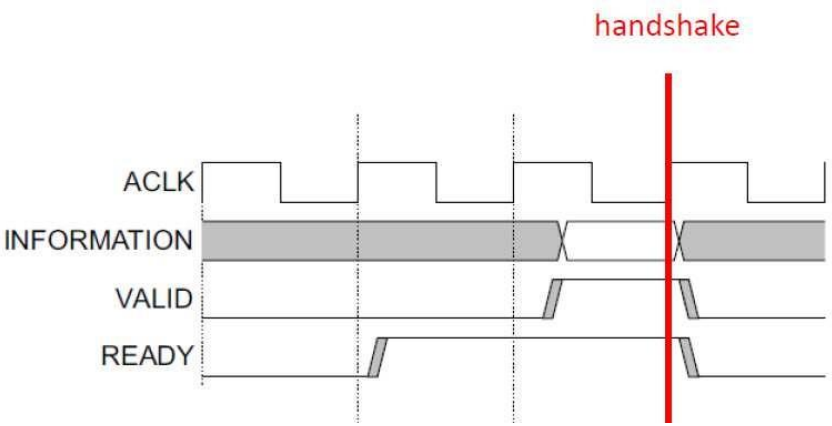
Flow Control: ready-valid flow

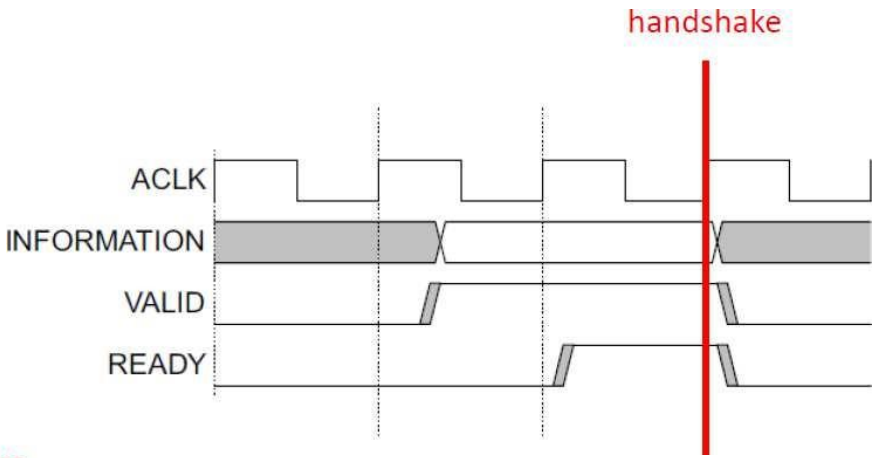


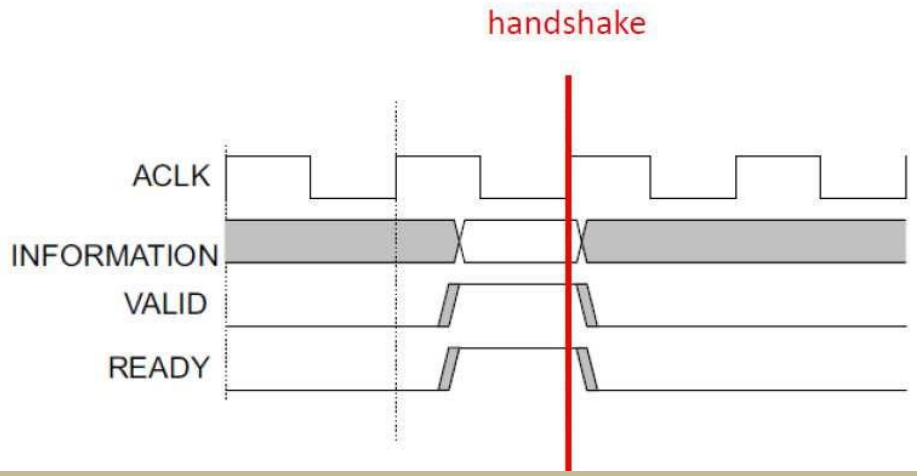
Ready 或valid 兩者都為1才能傳

Handshake process

- Ready before valid / Valid before ready
- Valid with ready

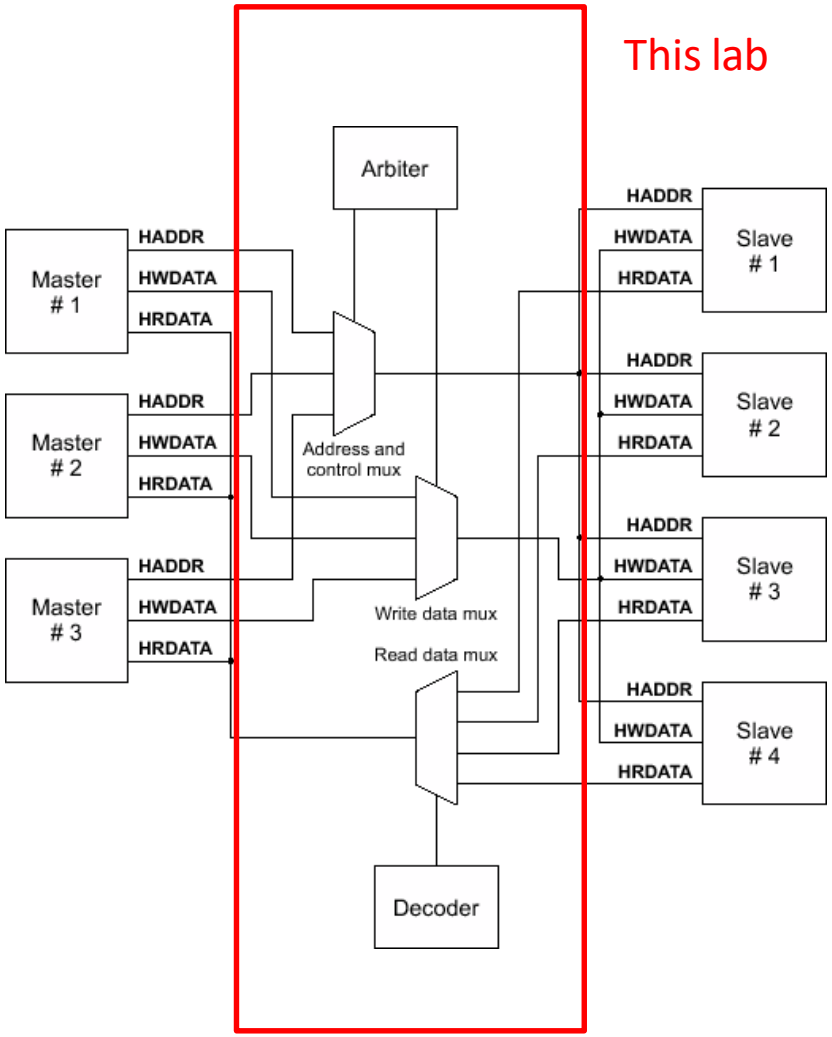






If(valid && ready)

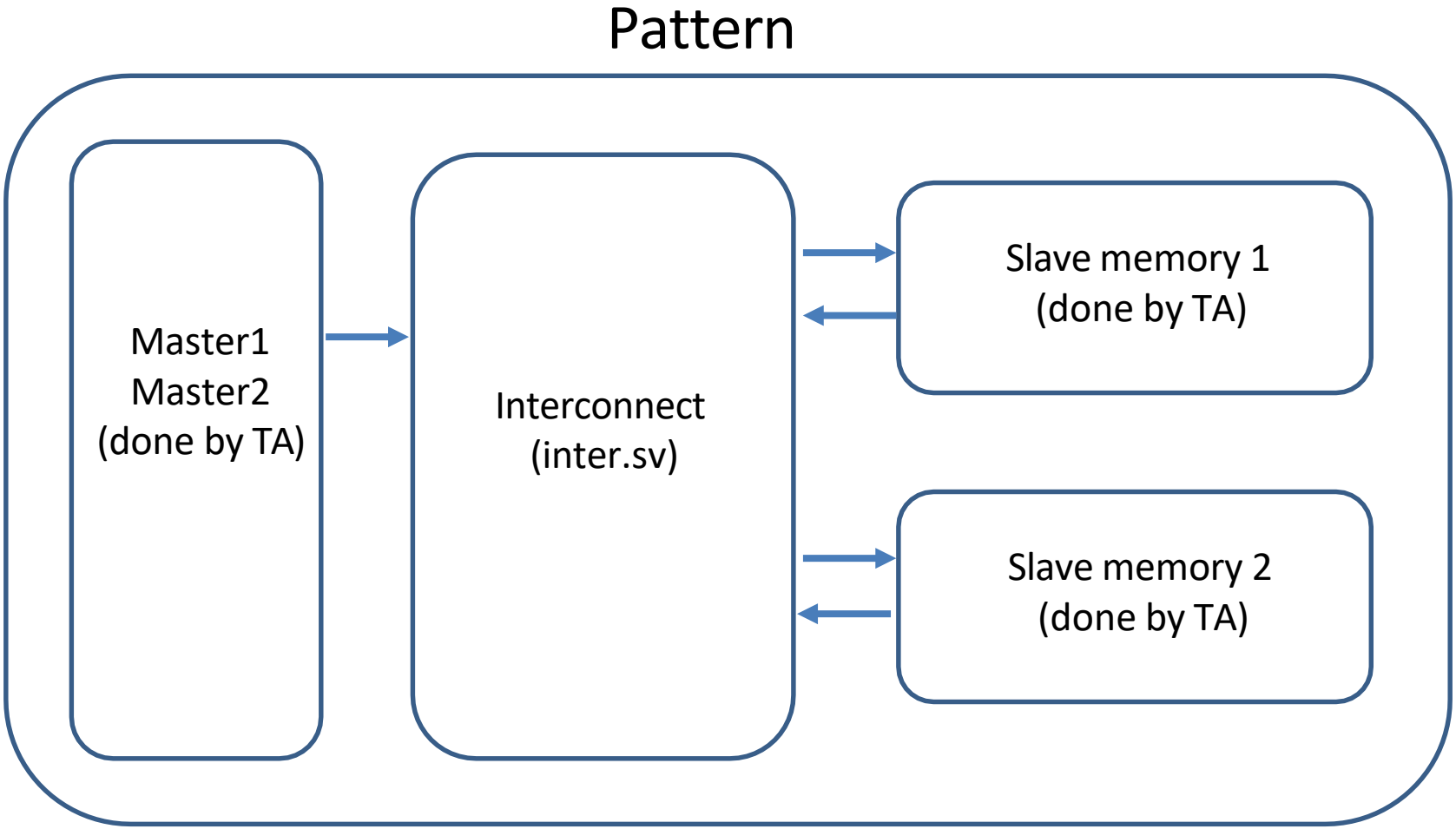
AHB Interconnect



Basic components:

- Master
- Slave
- Arbiter
- Decoder
- Mux

Lab - block diagram



Lab

- Masters will send input data to interconnect
- You should decode input data for valid, address and value
- Based on master priority(2->1), send data to slave memory
- Output handshake signal

Lab - decode

- [6:0]data_in_1 (from Master1)
- data_in_1[6]: 0->slave1 1->slave2
- data_in_1[5:3]: address
- data_in_1[2:0]: value
- Ex: 7'b0101001 for slave1, addr=5, value=1
- Ex: 7'b1011110 for slave2, addr=3, value=6

inter.sv

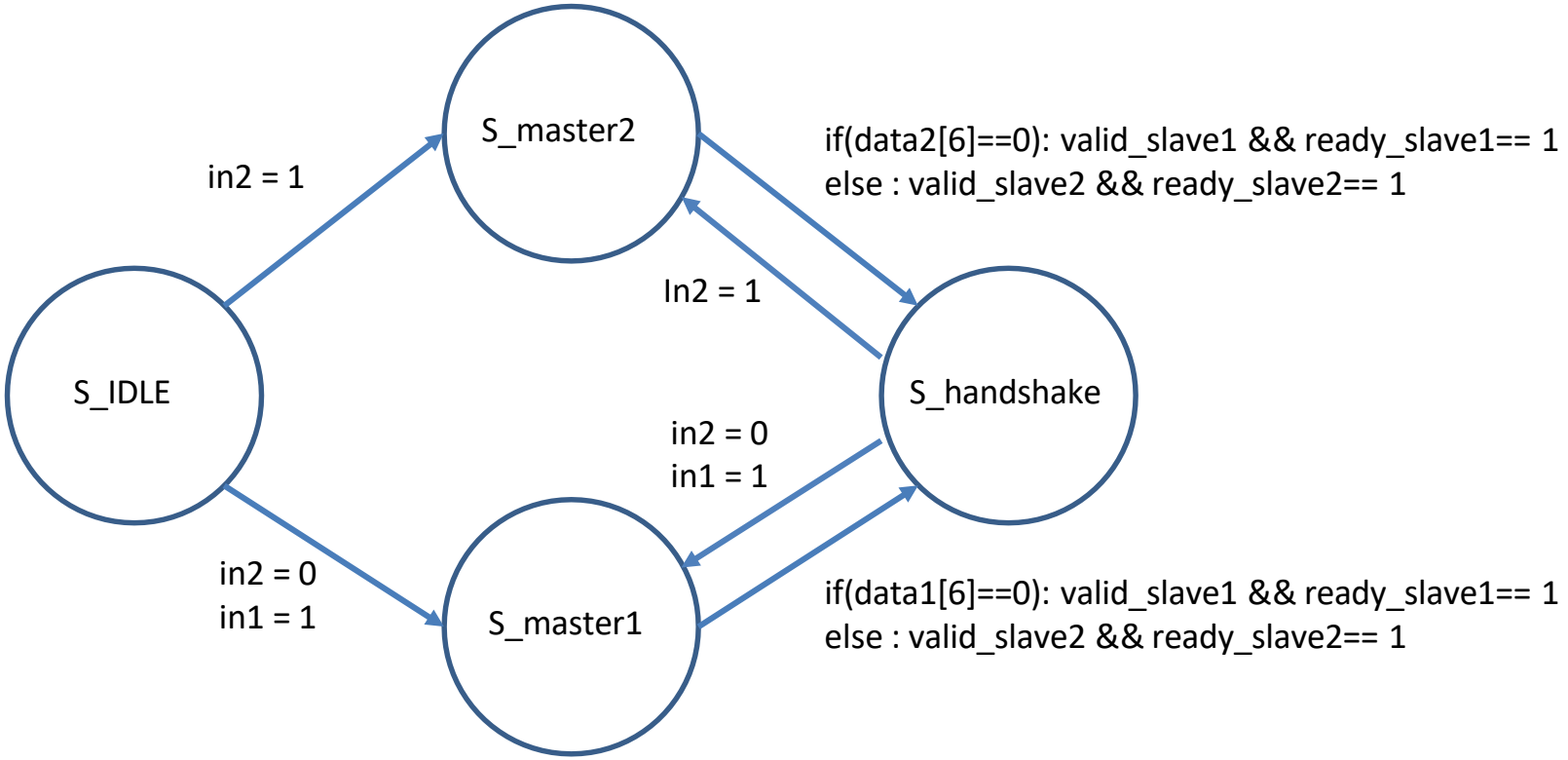
Input Signal	Bit Width	Definition
clk	1	Clock
rst_n	1	Asynchronous active-low reset
in_valid_1	1	in_valid from master1
in_valid_2	1	in_valid from master2
data_in_1	7	Data from master1
data_in_2	7	Data from master2
ready_slave1	1	Ready signal from slave1
ready_slave2	1	Ready signal from slave2

inter.sv

Output Signal	Bit Width	Definition
valid_slave1	1	valid signal to slave1
valid_slave2	1	valid signal to slave2
addr_out	3	Address you want to write
value_out	3	Value you want to save
handshake_slave1	1	High for 1 cycle after handshake with slave1
handshake_slave2	1	High for 1 cycle after handshake with slave2

Lab – Arbiter FSM (ref.)

in1 = in_valid_1訊號的暫存器
in2 = in_valid_2訊號的暫存器

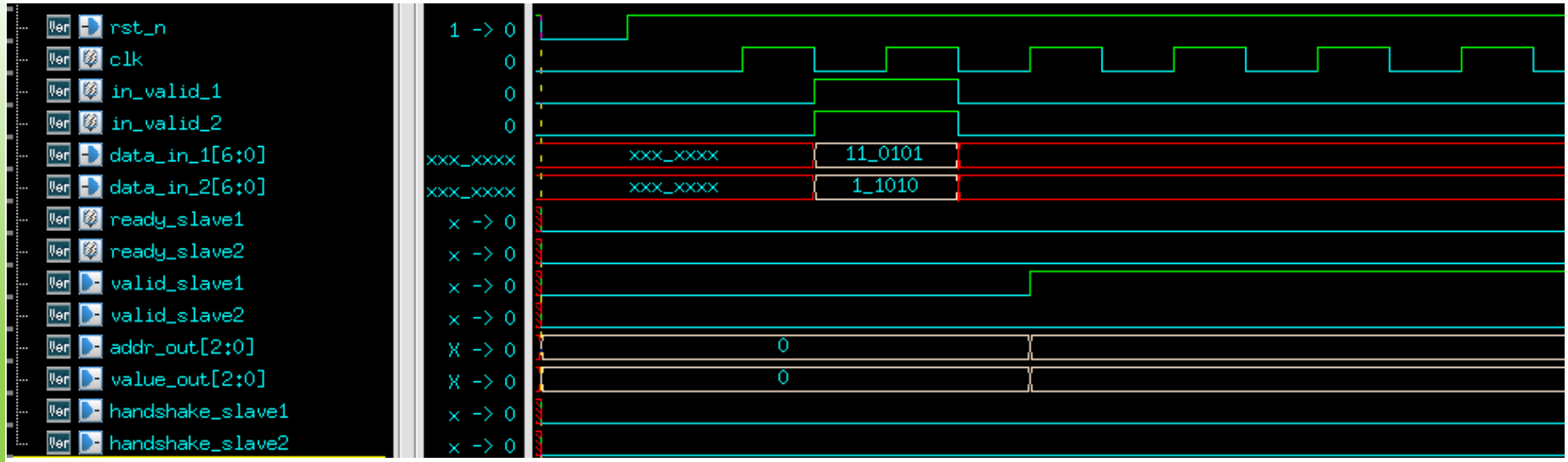


S_master1 and S_master2:
- Set valid_slave
- Set value_out and addr_out

Spec

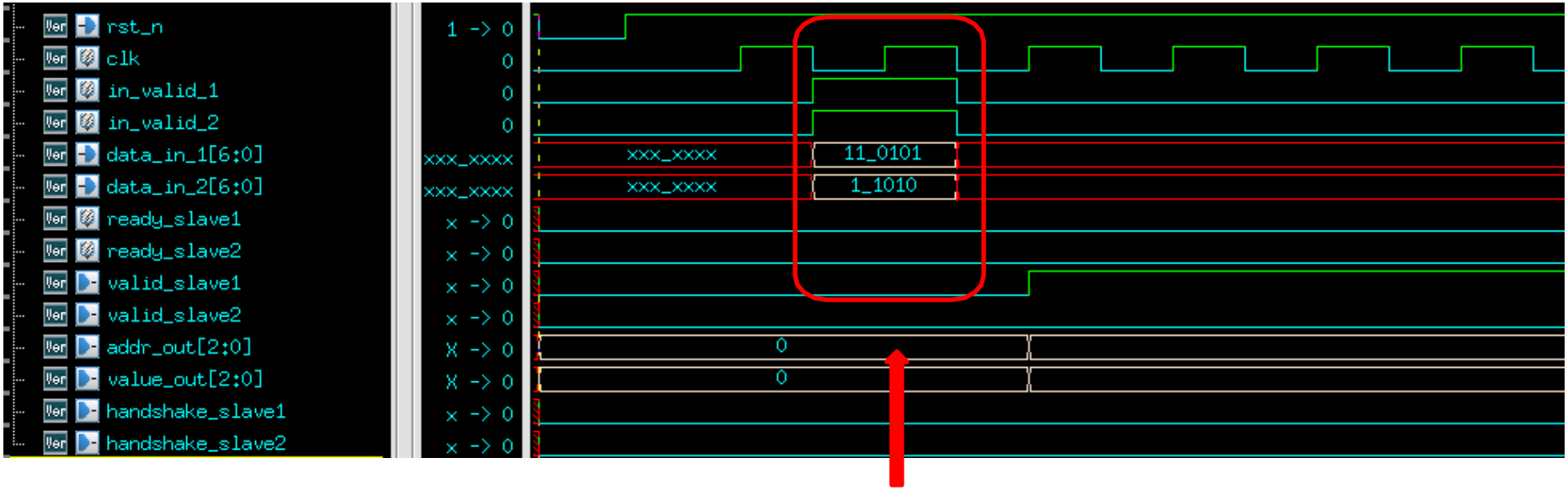
- 不可以超過30個cycles沒有完成handshake(從invalid或上一次handshake開始數)
- 拉起handshake signal(only 1 cycle)後,pattern會檢查 全部slave memory的值，必須依照master priority
- 可參考助教提供的FSM。
- 所有output必須非同步負準位reset。
- 01_RTL 需要PASS。
- 02_SYN不能有error跟latches。
- 02_SYN時間timing slack必須為MET。
- 03_GATE 需要PASS
- 宣告logic/wire/reg/submodule/parameter時，名稱勿包含*error*, *congratulation*, *latch* or *fail*等詞，字串裡含有上述關鍵詞也不行，例如"error_note", "read_fail"都會造成demo錯誤。**(FAIL!!!!)**

Waveform



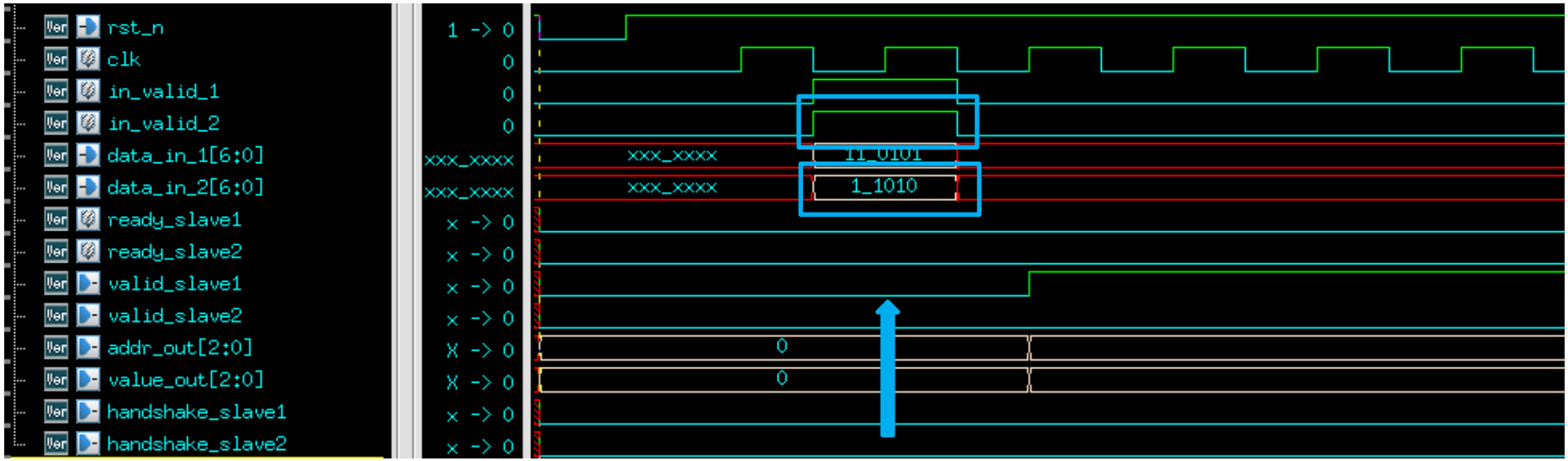
negative trigger asynchronous reset

Waveform



Input data

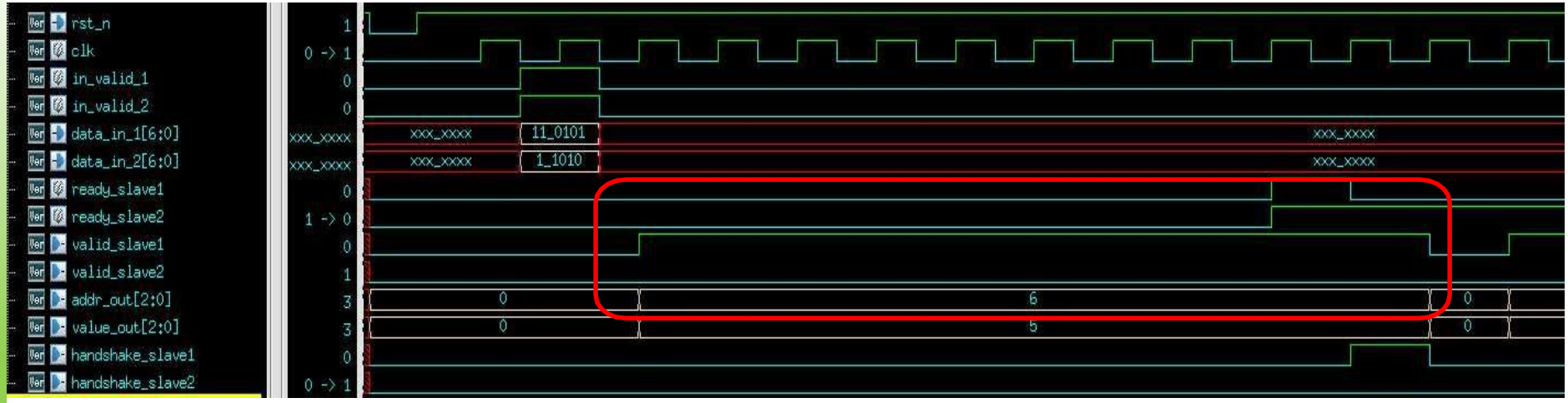
Waveform



Master2 first

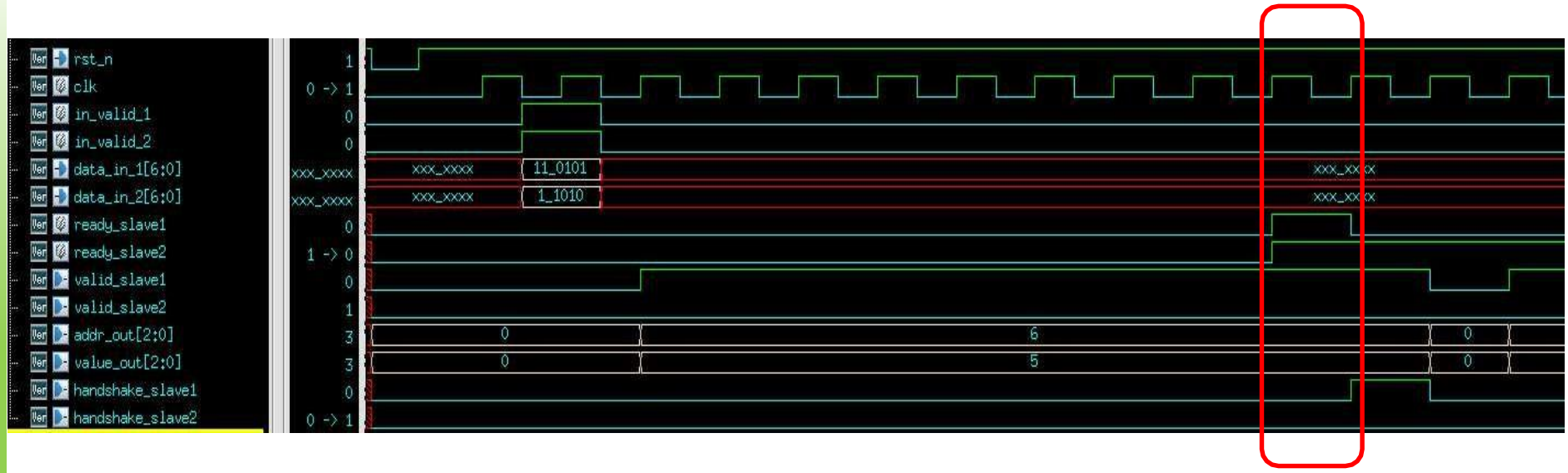
Decode: 0 011 010 => slave1 address:3 value:2

Waveform



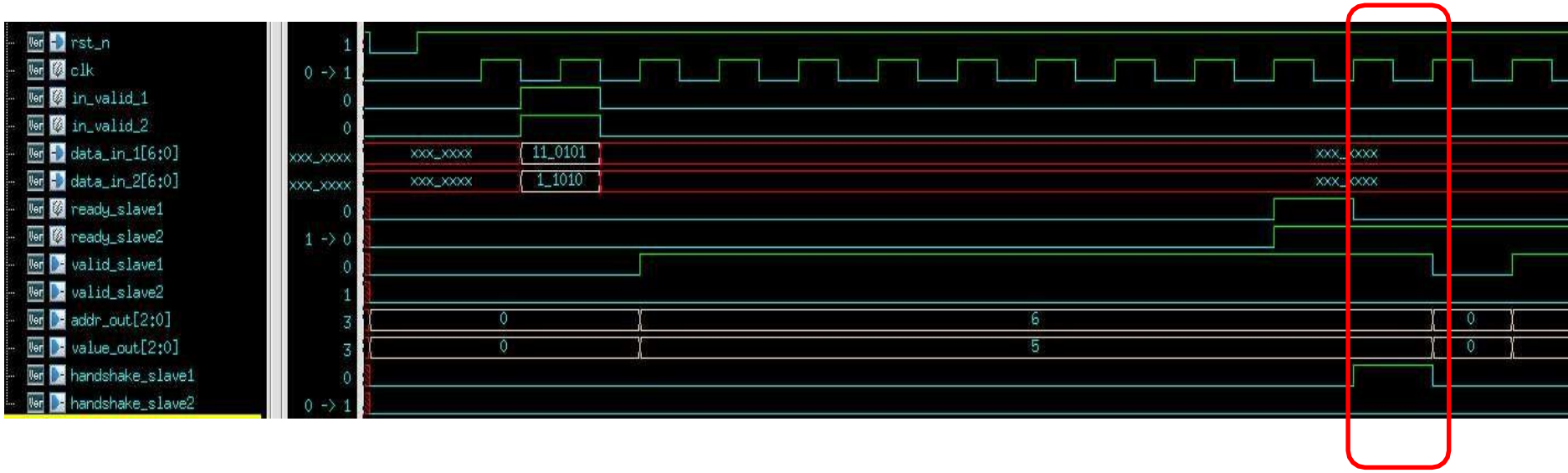
Set valid high, wait slave1 ready

Waveform



Handshake!

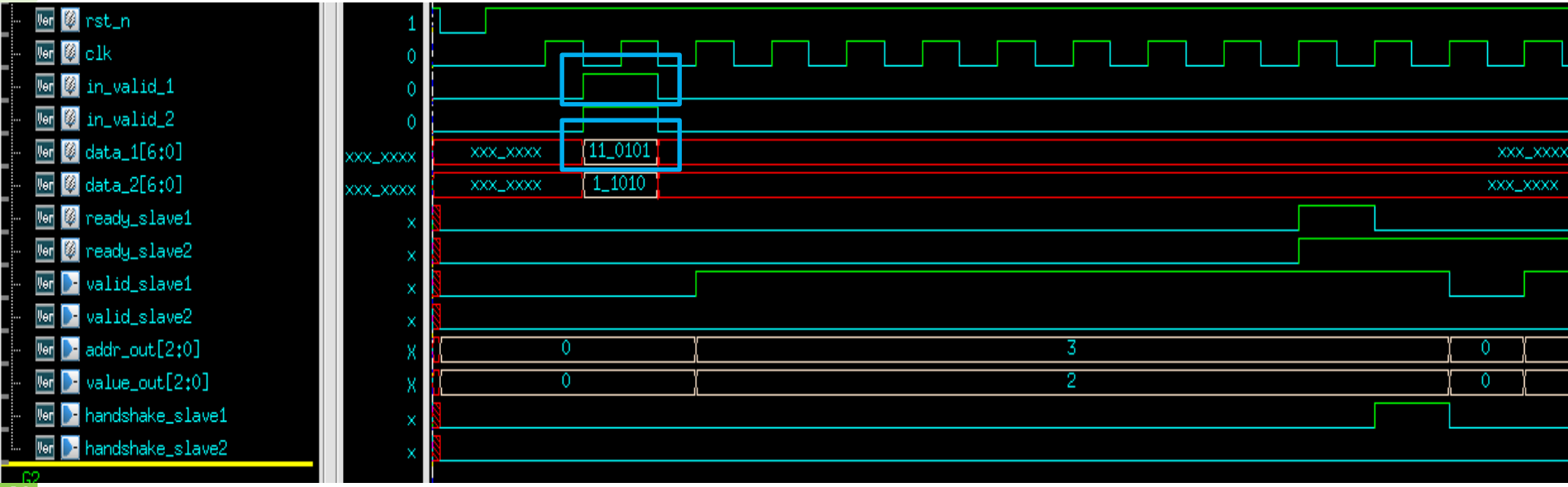
Waveform



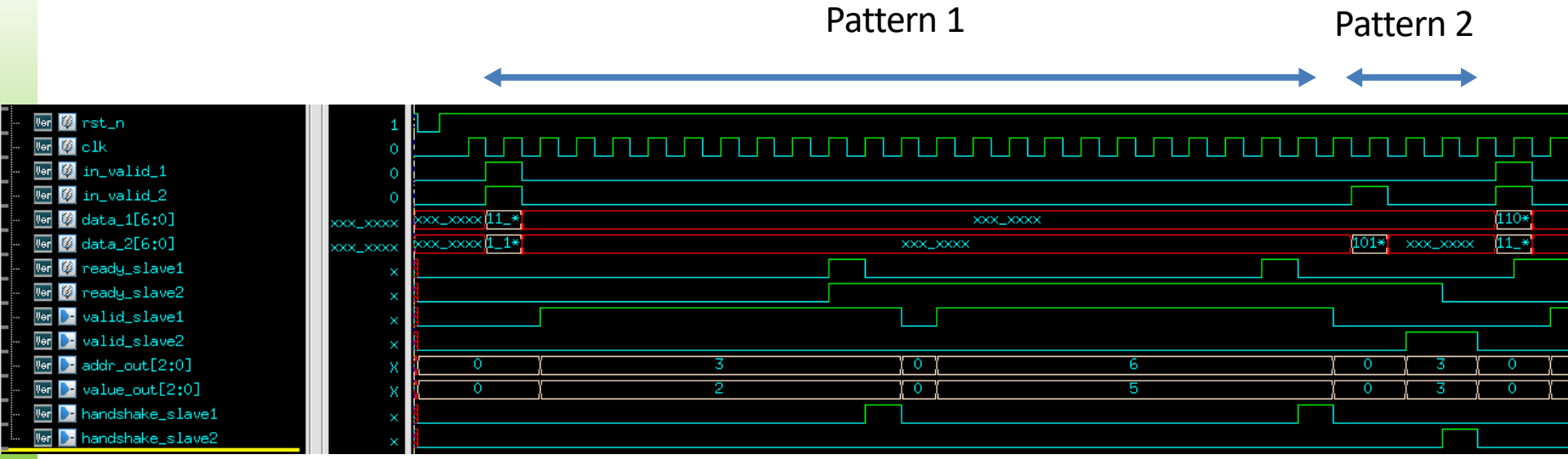
output handshake signal

Waveform

Continue for Master1



Waveform


Pattern 1
Pattern 2
rst_n
clk
in_valid_1
in_valid_2
data_1[6:0]
data_2[6:0]
ready_slave1
ready_slave2
valid_slave1
valid_slave2
addr_out[2:0]
value_out[2:0]
handshake_slave1
handshake_slave2
xxx_xxxx
11_*
1_1*
101*
110*
0
3
6
2
5

Command

- tar -xvf ~dcsTA01/Lab05.tar
- cd Lab05/01_RTL/
- Need 02_SYN
  - No Latch
  - No error
  - No timing violation (MET)
- Need 03_GATE

Demo1: 3/28(四), 17:30:00
Demo2: 3/29(五), 23:59:59

FSM simple example

```
parameter S_idle = 'd0;
parameter S_master1 = 'd1;
parameter S_master2 = 'd2;
parameter S_handshake = 'd3;
logic [1:0] cur_state, next_state;
//--------------------------------------------------------------------
//   YOUR DESIGN
//--------------------------------------------------------------------
always_ff @(posedge clk or negedge rst_n) begin
    if(!rst_n)
        cur_state <= S_idle;
    else
        cur_state <= next_state;
end

always_comb begin
    case(cur_state)
        S_idle :
            //statement
        S_master1:
            //statement
        S_master2:
            //statement
        S_handshake:
            //statement
        default:    //statement
    endcase
end
```

宣告

state register

next state logic