DD LAB10

Finite-state machine

助教: 陳冠良、方泰翔、郭政頡、趙文駿、陳韋廷

Outline

- 課程目的
- Finite-state machine介紹
- 課堂範例－信號偵測器
- LAB作業
- 作業繳交事項

課程目的

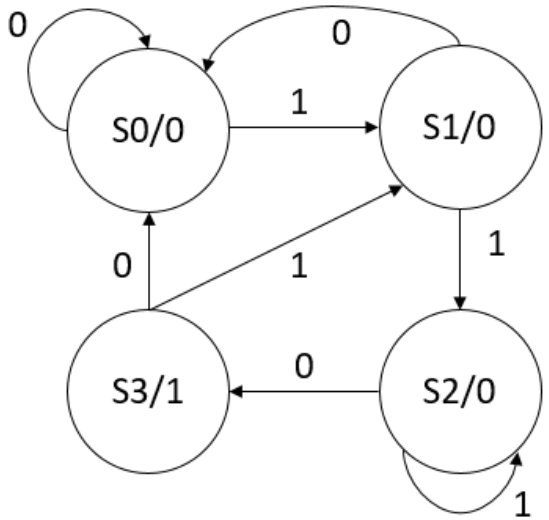
在數位系統導論課程中，同學們學習了Finite-state machine(FSM)的概念。

此LAB要教大家如何使用verilog描述FSM。

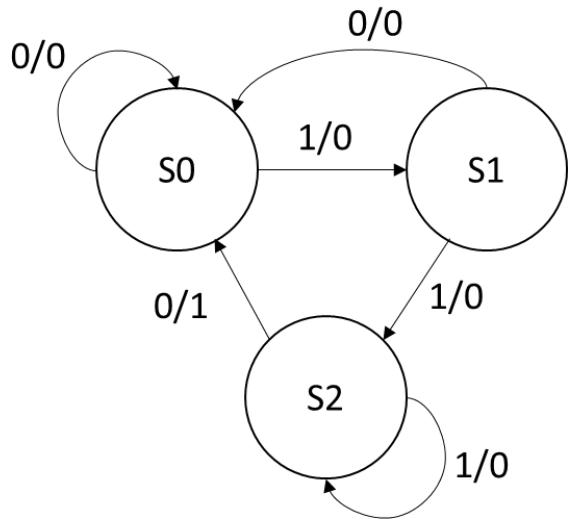
Finite-State Machine

- 有限狀態機(Finite-state machine, FSM），簡稱狀態機，是表示有限個狀態以及在這些狀態之間的轉換和動作等行為的數學模型。

- FSM可透過Moore Machine或Mealy Machine來表示



Moore Machine



Mealy Machine

Moore Machine 的輸出只和當前狀態有關
Mealy Machine 的輸出則不只取決於當前狀態，還與輸入有關

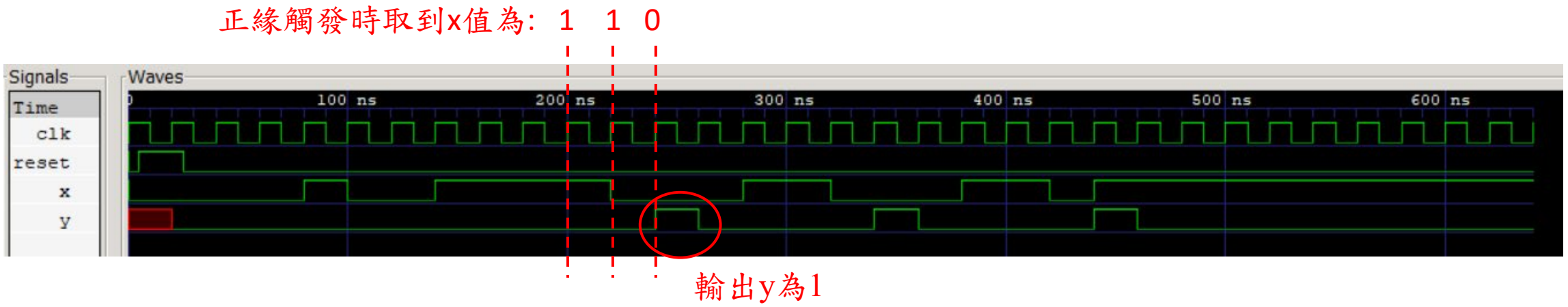
課堂範例-信號偵測器

- 設計一信號偵測器，當位元串流(bitstream)出現”110”時，則輸出y＝1，否則輸出y＝0。(下方波型圖可供參考)

e.g.

Input x : 0010011110001100001101

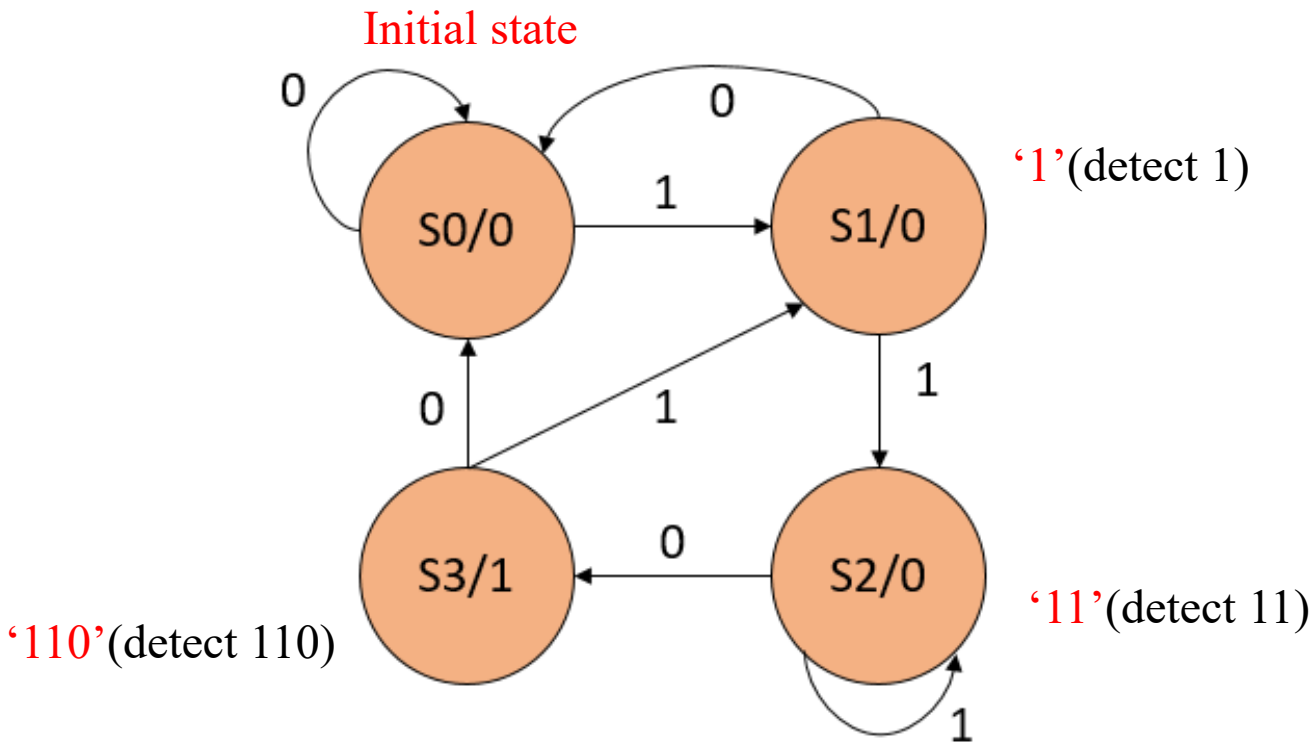
Output y : 0000000000100001000001



- 後續投影片分別使用Moore Machine及Mealy Machine來設計

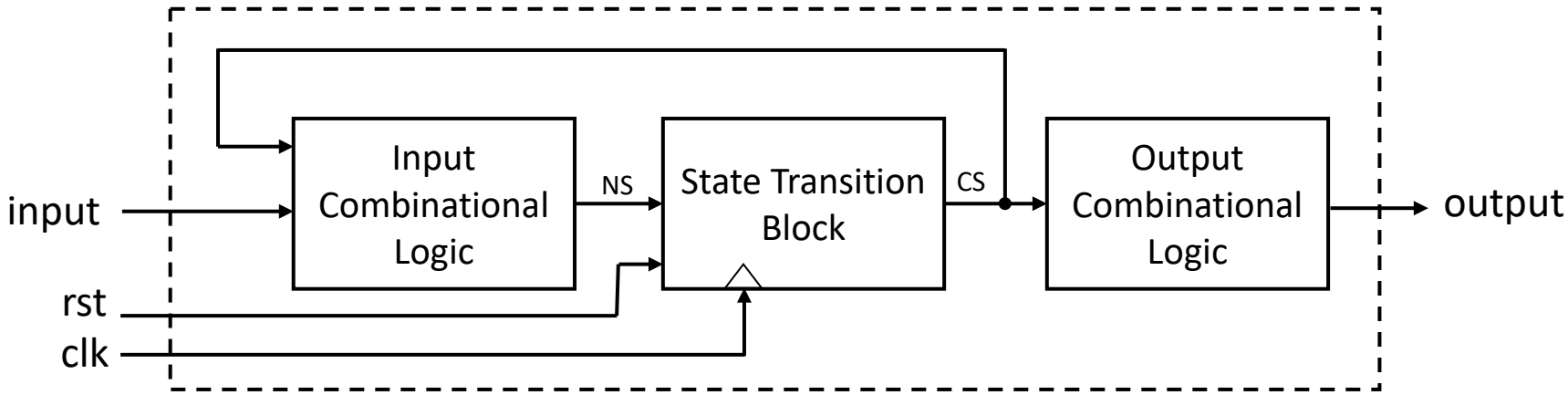
課堂範例–信號偵測器(Moore Machine)(1/5)

- 附圖為使用moore machine所表示的state diagram



課堂範例–信號偵測器(Moore Machine)(2/5)

- 以下為Moore Machine的架構圖及說明



- Input Combinational Logic：根據input與current state (CS)產生next state (NS)。
- State Transition Block ：負責傳遞state。
- Output Combinational Logic：根據current state產生output。

課堂範例–信號偵測器(Moore Machine)(3/5)

- I/O definition & state declaration

```
module moore_ex( input clk,
                 input reset,
                 input x,
                 output reg y );

    reg [1:0] current_state, next_state;

    parameter S0 = 2'd0;
    parameter S1 = 2'd1;
    parameter S2 = 2'd2;
    parameter S3 = 2'd3;
```

定義I/O

需要4個state來表示，current state與next state分別使用2 bit register 儲存

4個state(S0~S3)，使用parameter語法分別賦予值為0,1,2,3，讓state在module中取代常數表示，增加易讀性

課堂範例–信號偵測器(Moore Machine)(4/5)

- Input Combinational Logic

```
always@(*) begin
    case(current_state)
        S0:
            if(x) begin
                next_state = S1;
            end
            else begin
                next_state = S0;
            end
        S1:
            if(x) begin
                next_state = S2;
            end
            else begin
                next_state = S0;
            end
        S2:
            if(~x) begin
                next_state = S3;
            end
            else begin
                next_state = S2;
            end
        S3:
            if(x) begin
                next_state = S1;
            end
            else begin
                next_state = S0;
            end
        default:
            next_state = S0;
    endcase
end
```

根據input與current state產生next state
使用case來完成，列出每個state在收到input為0及為1時，next state分別為何

以S0 state為例
input為1時next state為S1
input為0時next state則維持S0

課堂範例–信號偵測器(Moore Machine)(5/5)

- State Transition Block

```
always@(posedge clk) begin
    if(reset) begin
        current_state <= S0;
    end
    else begin
        current_state <= next_state;
    end
end
```

負責傳遞 state

reset 時回到initial state S0

其餘則傳遞狀態，將next state送至current state

- Output Combinational Logic

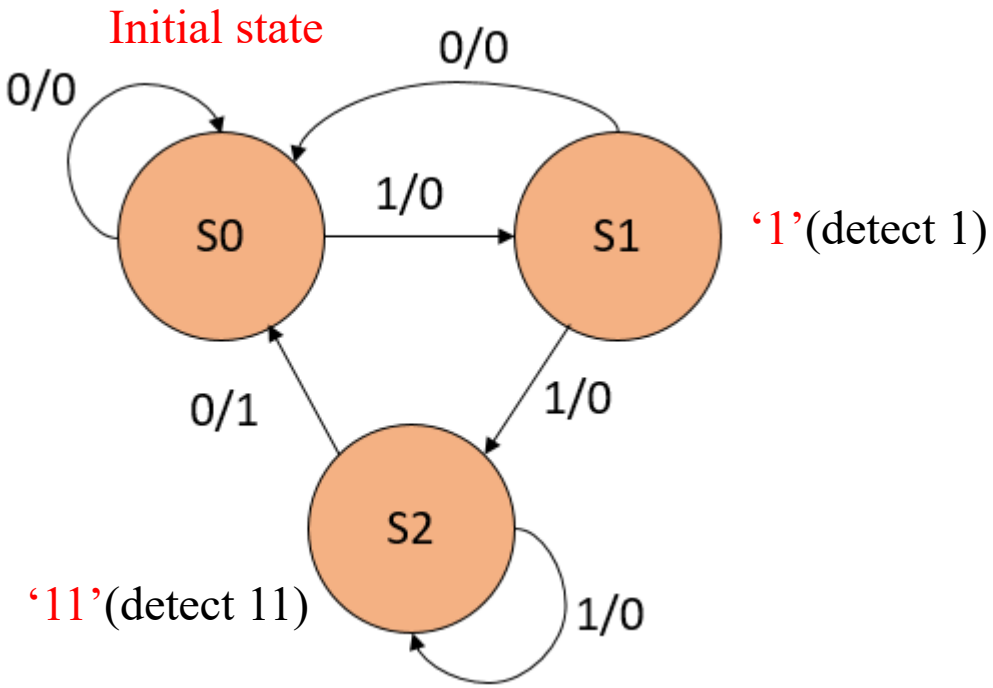
```
always@(*) begin
    case(current_state)
        S0:
            y = 1'b0;
        S1:
            y = 1'b0;
        S2:
            y = 1'b0;
        S3:
            y = 1'b1;
        default:
            y = 1'b0;
    endcase
end
```

根據current state產生output
同樣使用case來完成，因moore machine的output僅和state有關，
所以直接列出每個state的output為何即可

僅S3 state時output為1，其餘皆為0

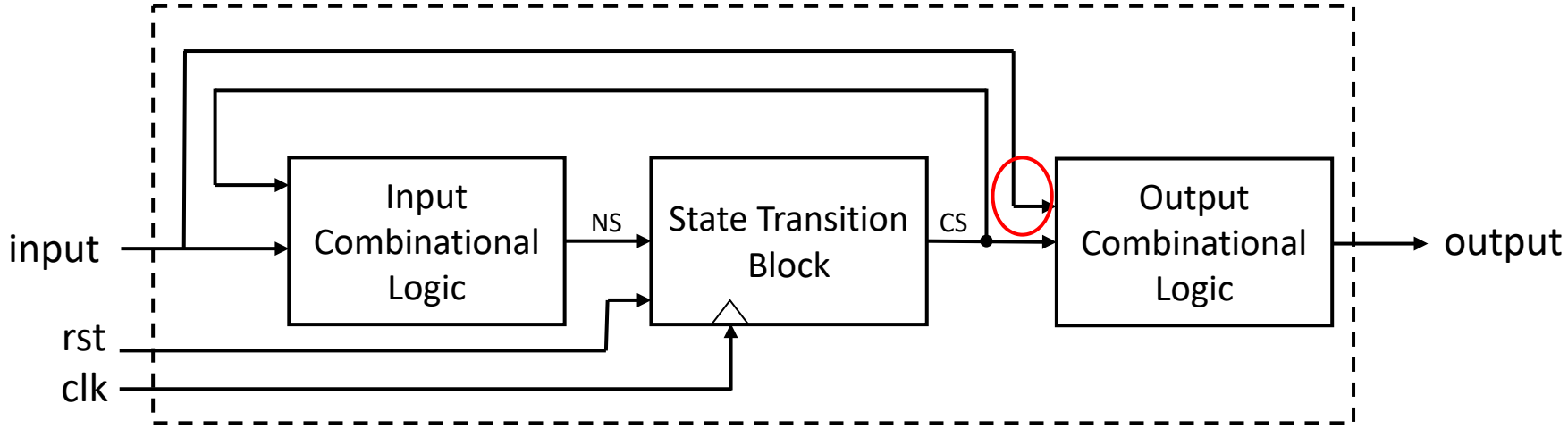
課堂範例–信號偵測器(Mealy Machine)(1/5)

- 附圖為使用mealy machine所表示的state diagram



課堂範例–信號偵測器(Mealy Machine)(2/5)

- 以下為Mealy Machine的架構圖及說明



- 與moore machine的差別在: output combinational logic根據current state及input來產生output

課堂範例–信號偵測器(Mealy Machine)(3/5)

- I/O definition & state declaration

```
module mealy_ex( input clk,
                 input reset,
                 input x,
                 output reg y );

    parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10;
    reg [1:0] current_state, next_state;
    reg y_temp;
```

需要3個state來表示，current state與next state分別使用2 bit register 儲存

課堂範例-信號偵測器(Mealy Machine)(4/5)

- Input Combinational Logic

```
always@(*) begin
    case(current_state)
        S0:
            if(x) begin
                next_state = S1;
            end
            else begin
                next_state = S0;
            end
        S1:
            if(x) begin
                next_state = S2;
            end
            else begin
                next_state = S0;
            end
        S2:
            if(~x) begin
                next_state = S0;
            end
            else begin
                next_state = S2;
            end
        default:
            next_state = S0;
    endcase
end
```

- State Transition Block

```
always@(posedge clk) begin
    if(reset) begin
        current_state <= S0;
    end
    else begin
        current_state <= next_state;
    end
end
```

這兩部分一樣根據所畫的state diagram來描述

課堂範例–信號偵測器(Mealy Machine)(5/5)

- Output Combinational Logic

```
always@(*) begin
    case(current_state)
        S0:
            if(x) begin
                y_temp = 1'b0;
            end
            else begin
                y_temp = 1'b0;
            end
        S1:
            if(x) begin
                y_temp = 1'b0;
            end
            else begin
                y_temp = 1'b0;
            end
        S2:
            if(~x) begin
                y_temp = 1'b1;
            end
            else begin
                y_temp = 1'b0;
            end
        default:
            y_temp = 1'b0;
    endcase
end
```

Mealy machine的output須由current state及input來決定

僅有當狀態S2、input為1時，output為1，其餘皆為0

- Output register

```
always@(posedge clk) begin
    if(reset) begin
        y <= 1'b0;
    end
    else begin
        y <= y_temp;
    end
end
```

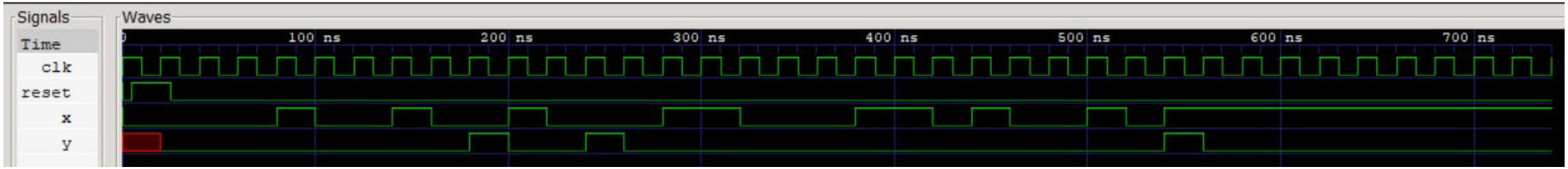
寫這個block的目的為避免output y提早一個clock輸出。並使用y_temp暫存

LAB作業

- 請同學在不更改testbench內容的情況下，修改”moore_ex.v”及”mealy_ex.v”，分別使用moore machine、mealy machine的方式，設計一偵測位元串流”10010”的信號偵測器，並使用所提供的testbench驗證，驗證結果皆須與下方的波型圖一致

- 完成一種方式: 60%，完成兩種方式: 100%

- 將作業檔案名稱分別取名為”moore_hw.v”、”mealy_hw.v”



作業繳交事項

- 繳交時間
  - 2021/6/23 (三) 23:59前
- 繳交方式
  - 上傳至eCourse2
- 繳交檔案
  1. moore_hw.v
  2. mealy_hw.v
  3. testbench.v