



# 邏輯系統實習

## 實驗八

**Verilog語法介紹(五)：行為層次-有限狀態機**

國立成功大學 電機系

**2016**

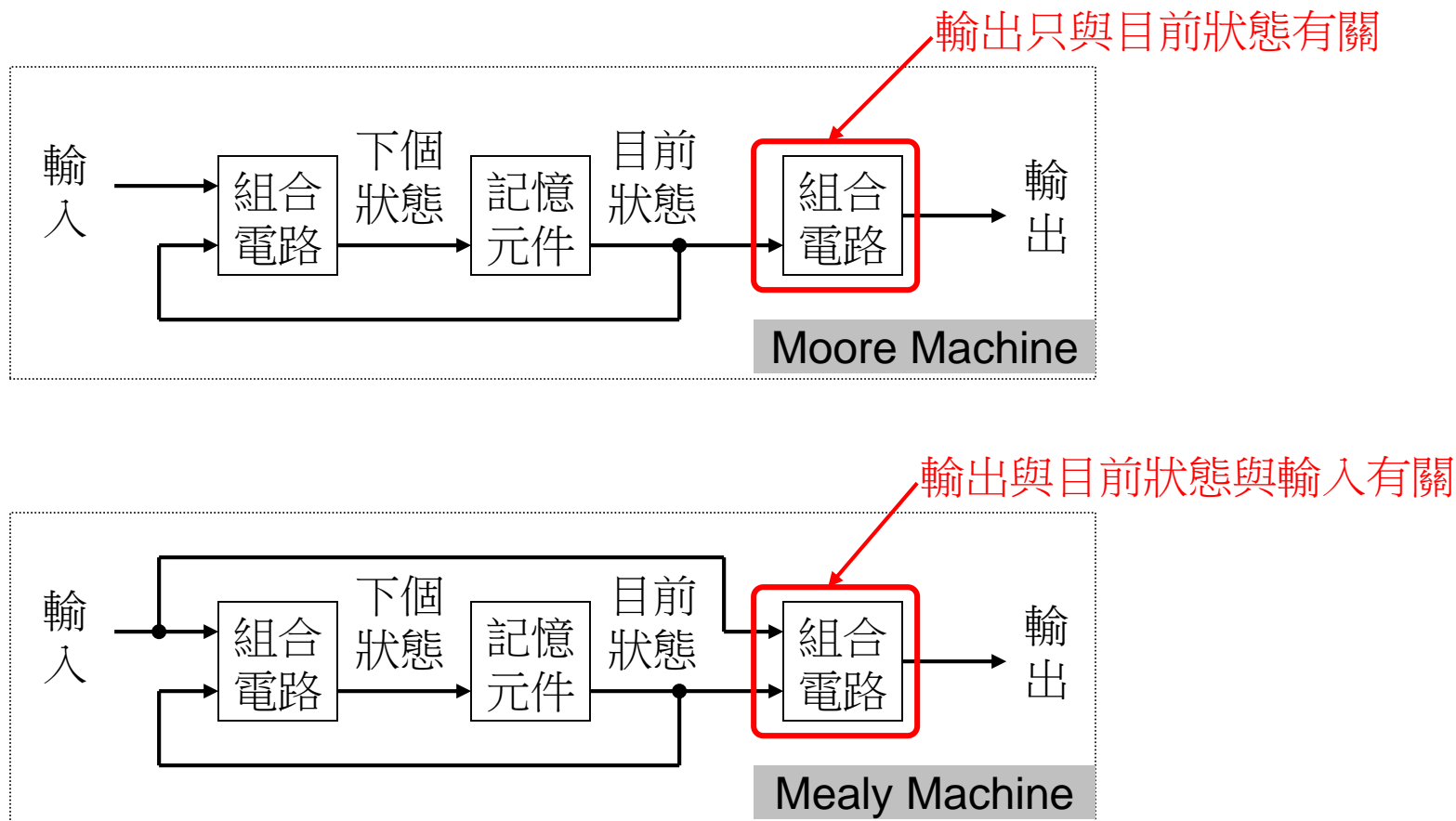
# 大綱

- 有限狀態機
- **Moore Machine與Mealy Machine之比較**
- 狀態圖與狀態表
- 有限狀態機範例
- 基礎題 (一)
  - 有限狀態機範例
- 基礎題 (二)
  - 交通燈
- 挑戰題
  - 販賣機
- 實驗結報繳交

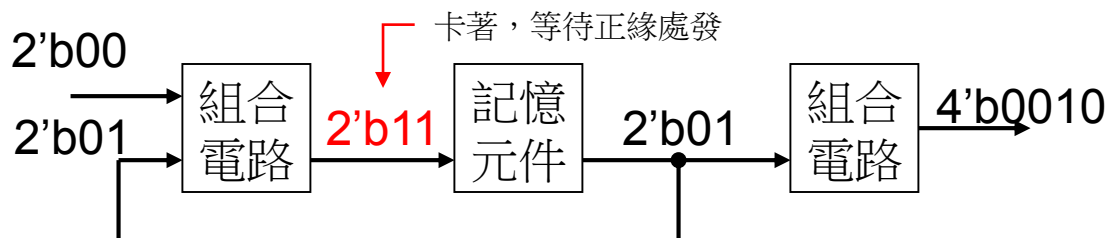
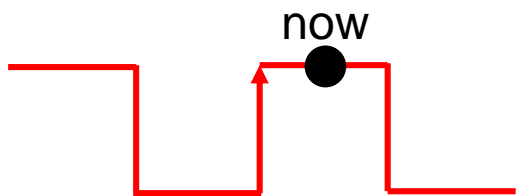
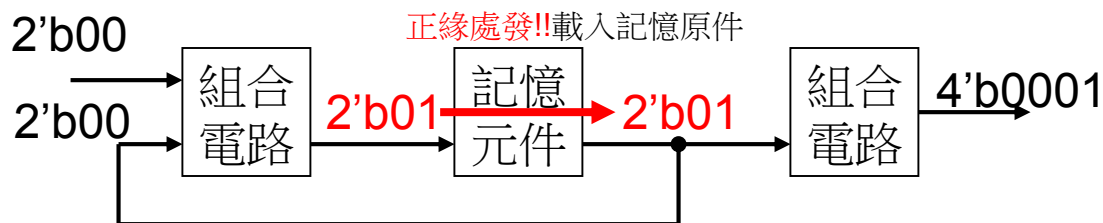
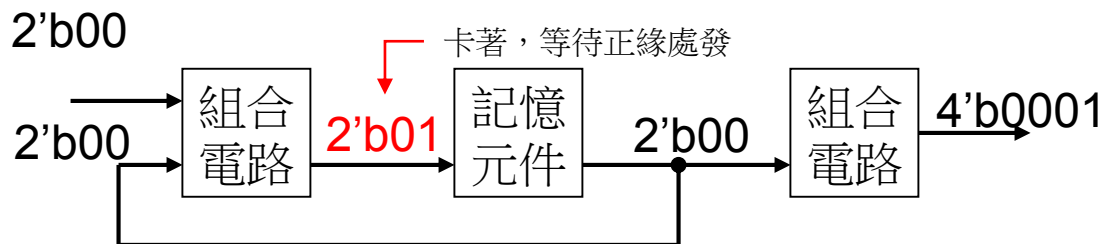
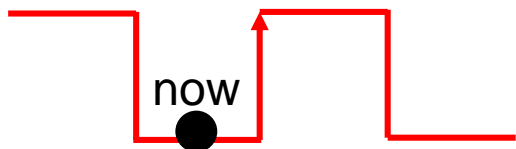
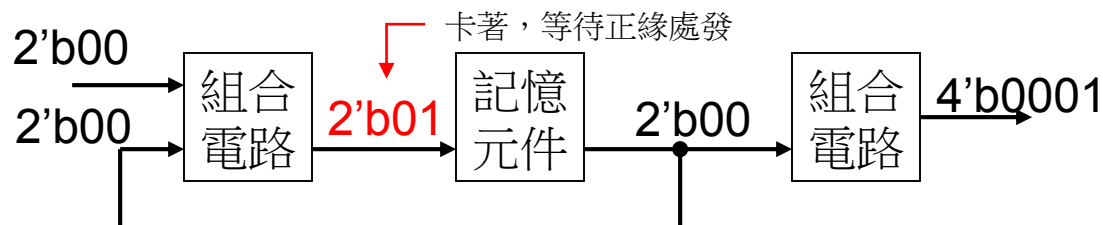
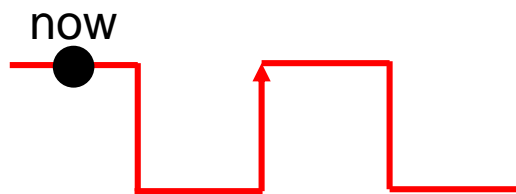
# 有限狀態機

- 有限狀態機(finite state machine) (FSM)。
- 有限狀態機是由一組狀態、一個起始狀態、輸入、將輸入與目前狀態轉換為下個狀態的條件函數所組成。
- 有限狀態機分為兩種模形：
  - Moore machine：輸出只與目前狀態有關。
  - Mealy machine：輸出與目前狀態與輸入有關。
- 通常我們會使用狀態圖或狀態表來描述有限狀態機。
- 序向電路有時也被稱為有限狀態機。

# Moore Machine與Mealy Machine之比較

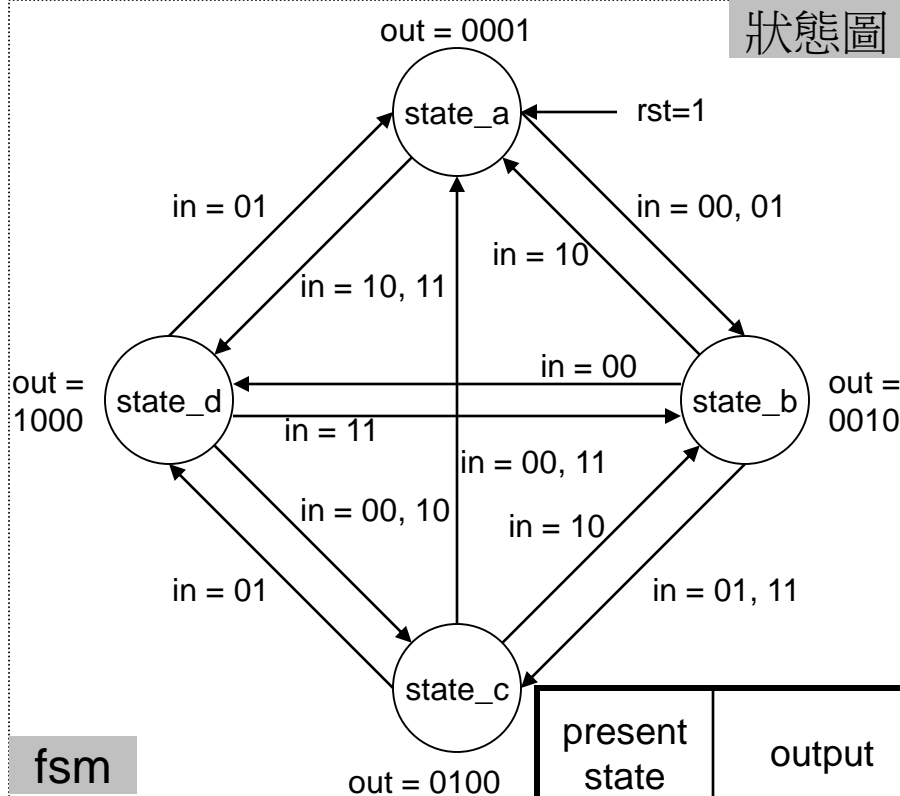


# 正緣觸發 Moore Machine 行為介紹



# 狀態圖與狀態表

狀態圖



狀態表

fstm

present state	output	next state				
		rst=1	in = 00	in = 01	in = 10	in = 11
state_a	out = 0001	state_a	state_b	state_b	state_d	state_d
state_b	out = 0010	state_a	state_d	state_c	state_a	state_c
state_c	out = 0100	state_a	state_a	state_d	state_b	state_a
state_d	out = 1000	state_a	state_c	state_a	state_c	state_b

# 有限狀態機範例 (1/3)

```
module fsm(clk, rst, in, out); fsm (continued)
```

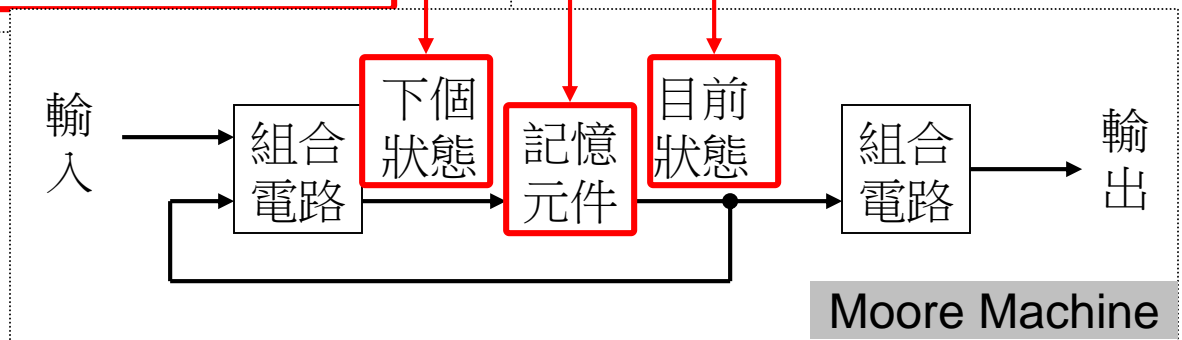
```
input clk, rst;  
input [1:0]in;  
output [3:0]out;
```

```
parameter STATE_A=2'd0,  
STATE_B=2'd1,  
STATE_C=2'd2,  
STATE_D=2'd3;
```

```
reg [1:0]p_s; n_s;  
reg [3:0]out;
```

```
always@(posedge clk or posedge rst)begin  
if(rst)p_s<=STATE_A;  
else p_s<=n_s;  
end
```

在此範例，記憶元件為2bits  
的flip-flop (reg [1:0]p\_s;)



# 有限狀態機範例 (2/3)

```
always@(*)begin
```

```
  case(p_s)
```

```
    STATE_A : n_s = (in==2'b00) ? STATE_B :  
                    (in==2'b01) ? STATE_B :  
                    (in==2'b10) ? STATE_D :  
                    STATE_D ;
```

```
    STATE_B : n_s = (in==2'b00) ? STATE_D :  
                    (in==2'b01) ? STATE_C :  
                    (in==2'b10) ? STATE_A :  
                    STATE_C ;
```

```
    STATE_C : n_s = (in==2'b00) ? STATE_A :  
                    (in==2'b01) ? STATE_D :  
                    (in==2'b10) ? STATE_B :  
                    STATE_A ;
```

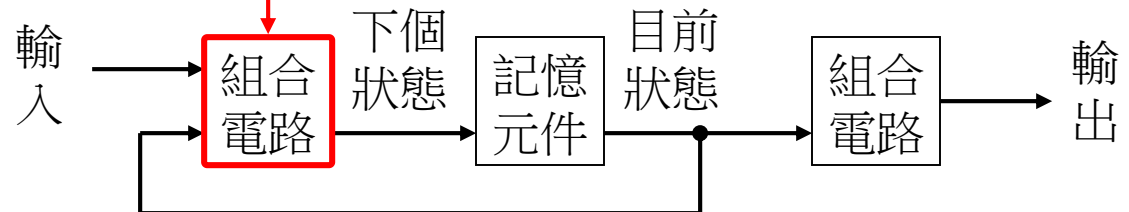
```
    STATE_D : n_s = (in==2'b00) ? STATE_C :
```

```
  default : n_s = s;  
endcase
```

```
end
```

fsm (continued)

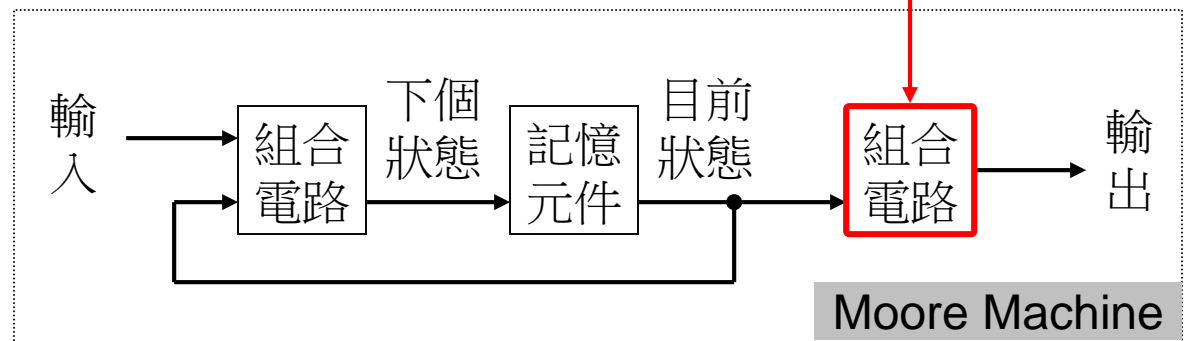
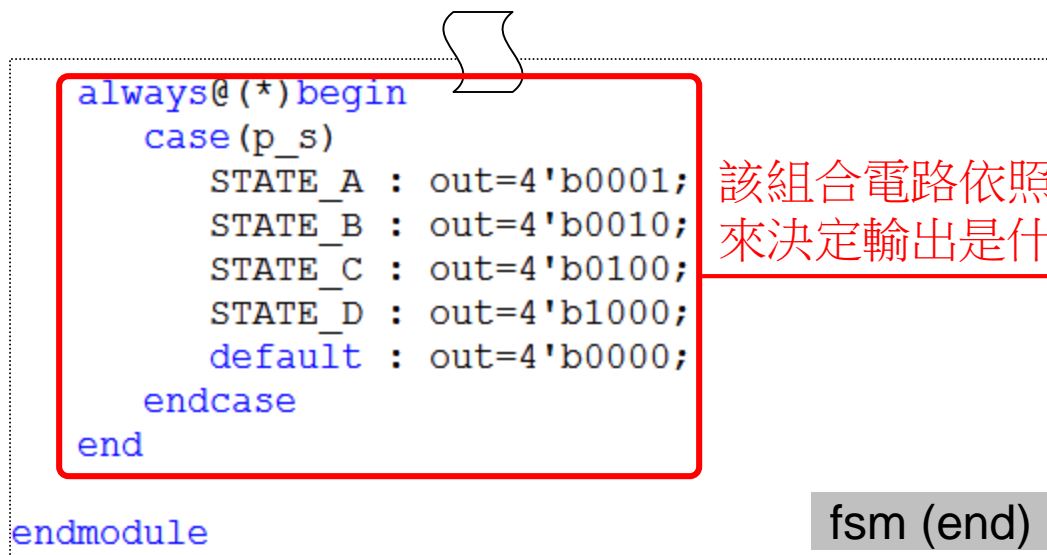
該組合電路依照不同的輸入，  
來決定下個狀態是什麼



Moore Machine



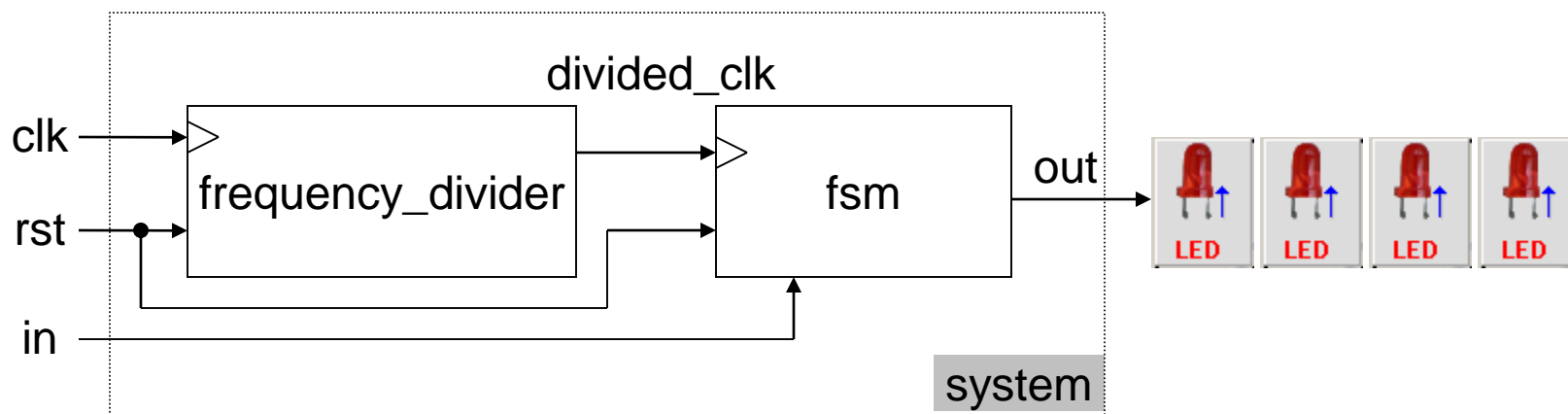
# 有限狀態機範例 (3/3)



# 基礎題 (一)

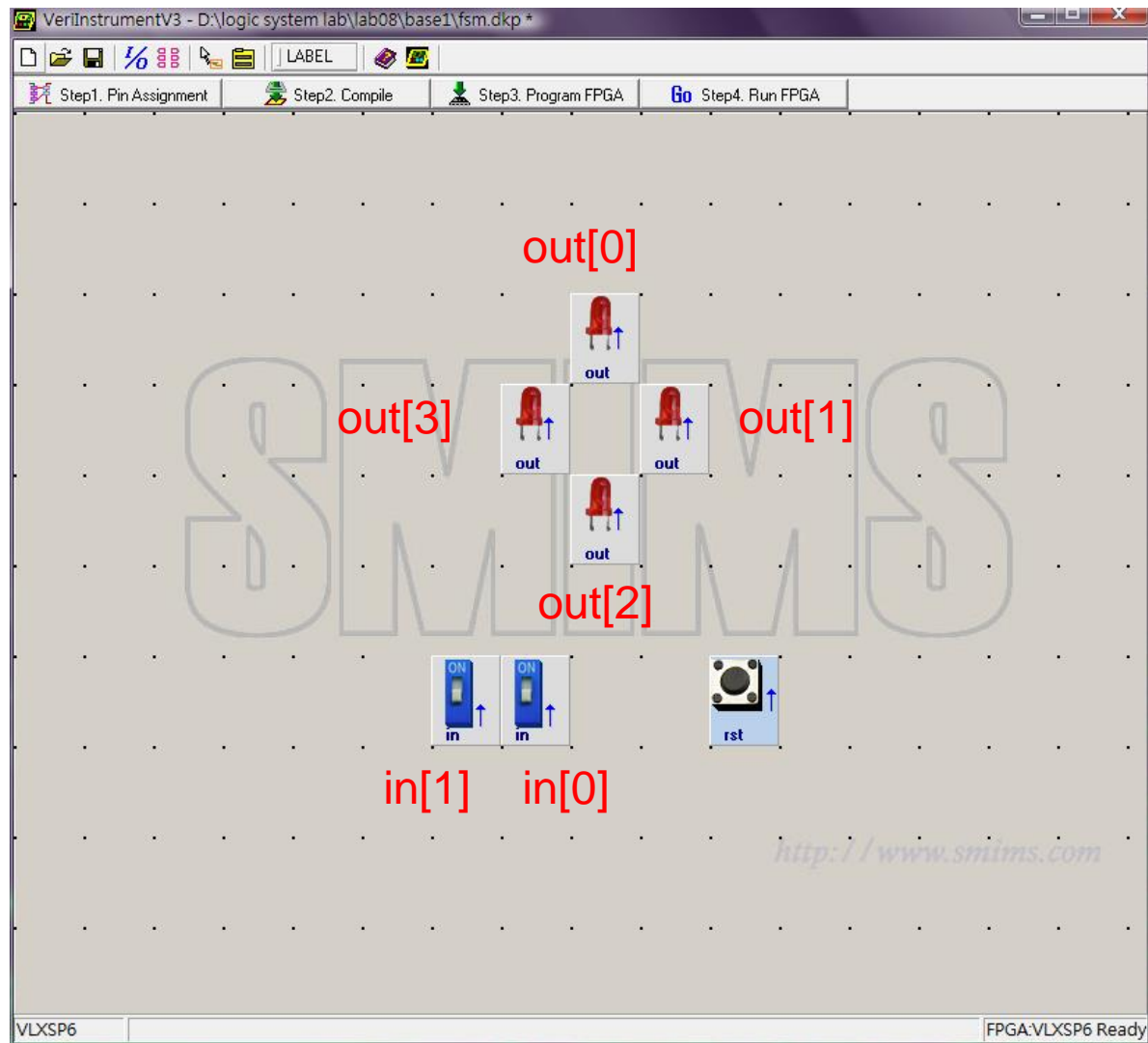
## 有限狀態機範例 (1/4)

- 請參考fsm.v的範例原始碼與操作流程，寫出以下的電路。
  - 設計時，請寫出以下的電路模組，並且確認編譯後沒有error或warning產生。
  - (模擬時，請自行撰寫testbench.v，並觀察波形結果或主控台的螢幕顯示結果是否如期望一般。)
  - 驗證時，請觀察虛擬儀器之運作是否如期望一般。



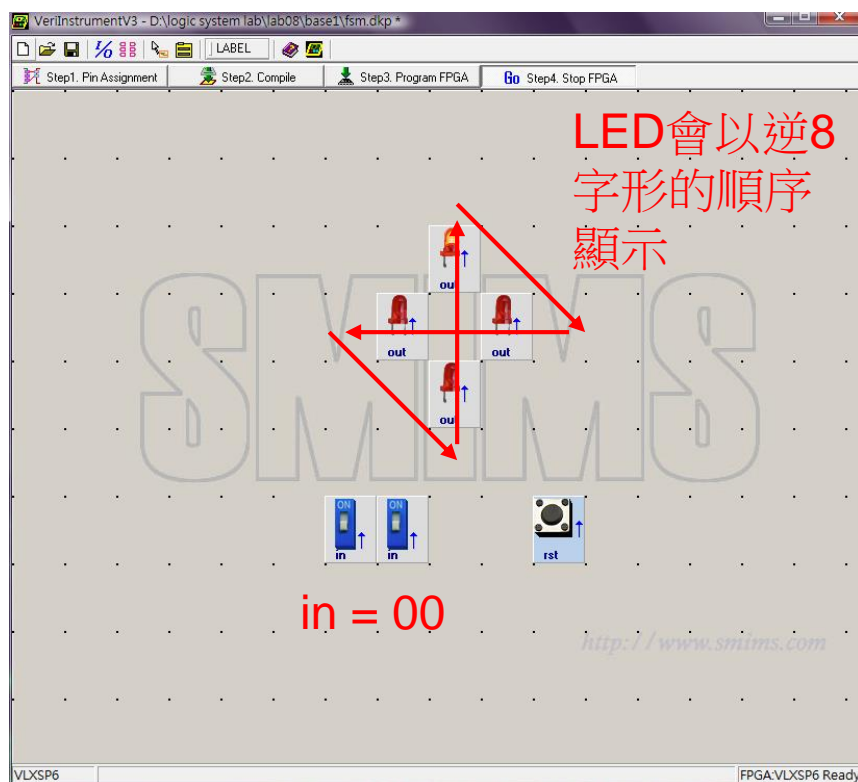
# 基礎題 (一)

## 有限狀態機範例 (2/4)

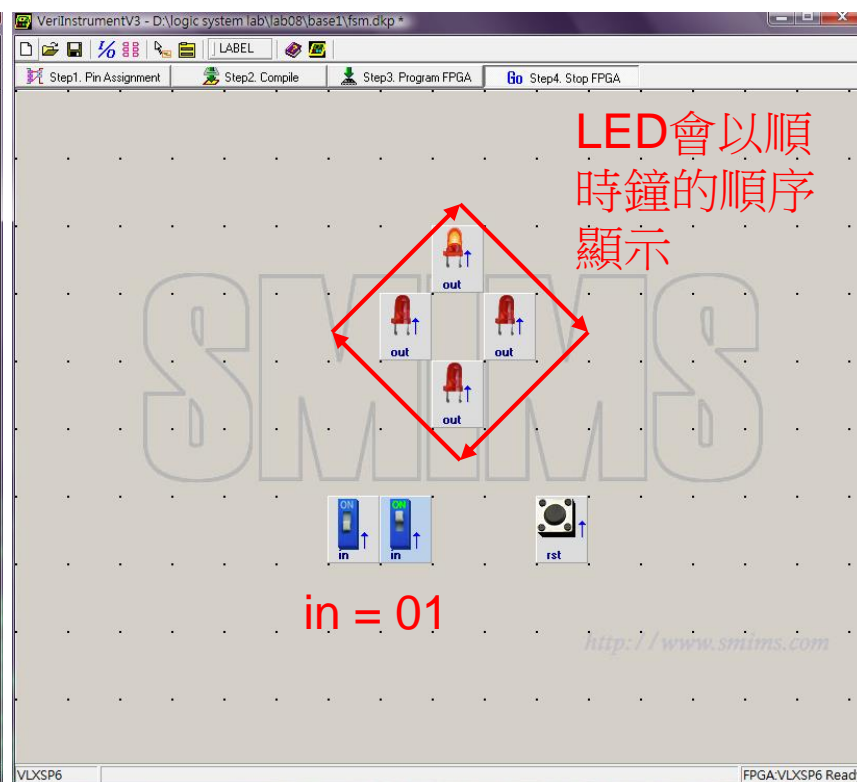


# 基礎題 (一)

## 有限狀態機範例 (3/4)



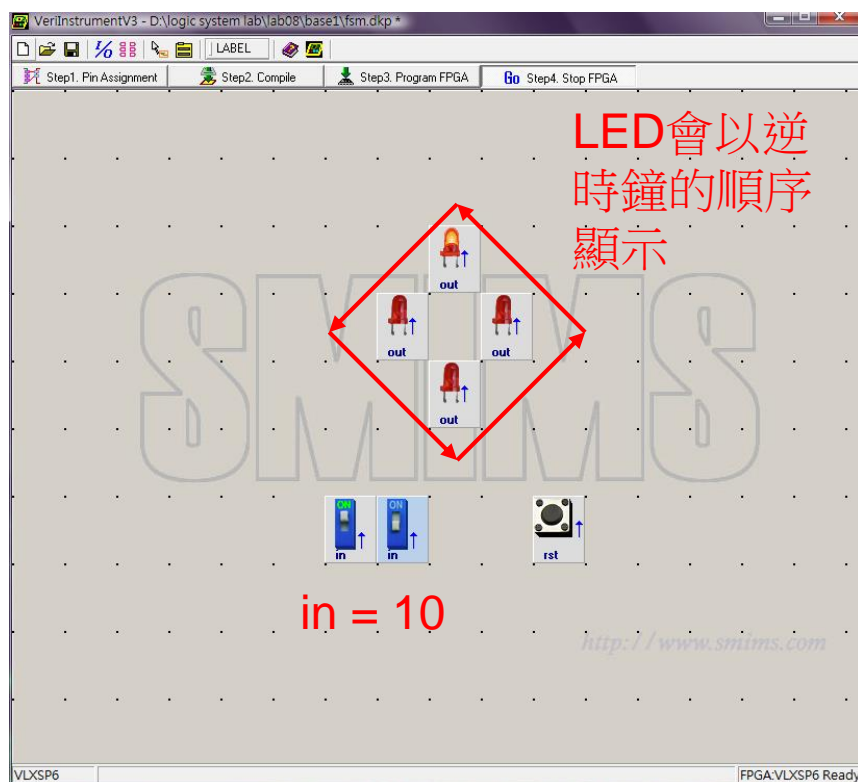
state\_a -> state\_b -> state\_d ->  
state\_c -> state\_a ...



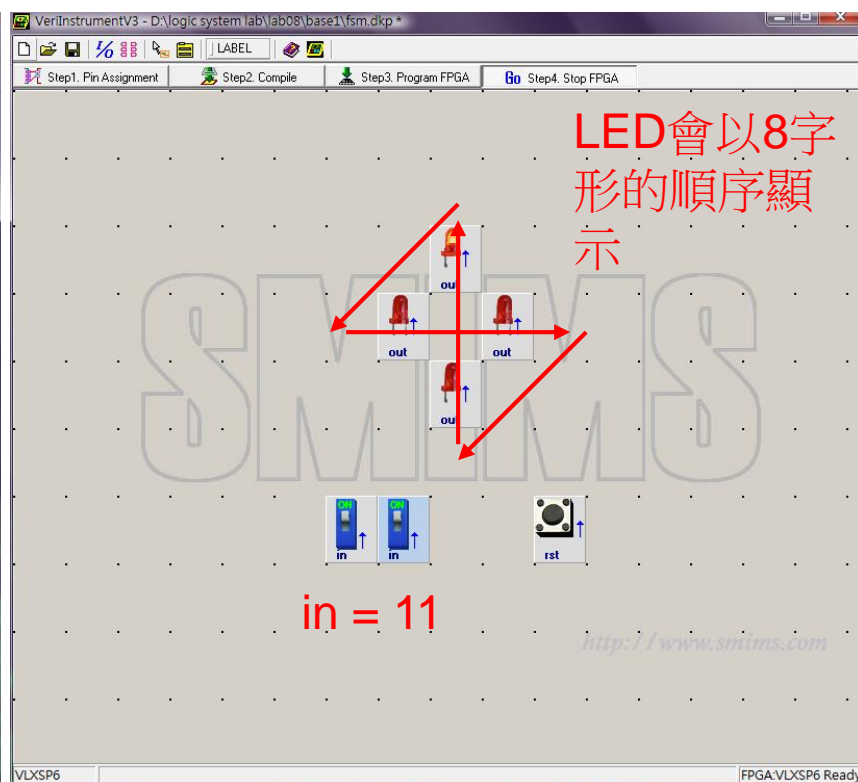
state\_a -> state\_b -> state\_c ->  
state\_d -> state\_a ...

# 基礎題 (一)

## 有限狀態機範例 (4/4)



state\_a -> state\_d -> state\_c ->  
state\_b -> state\_a ...

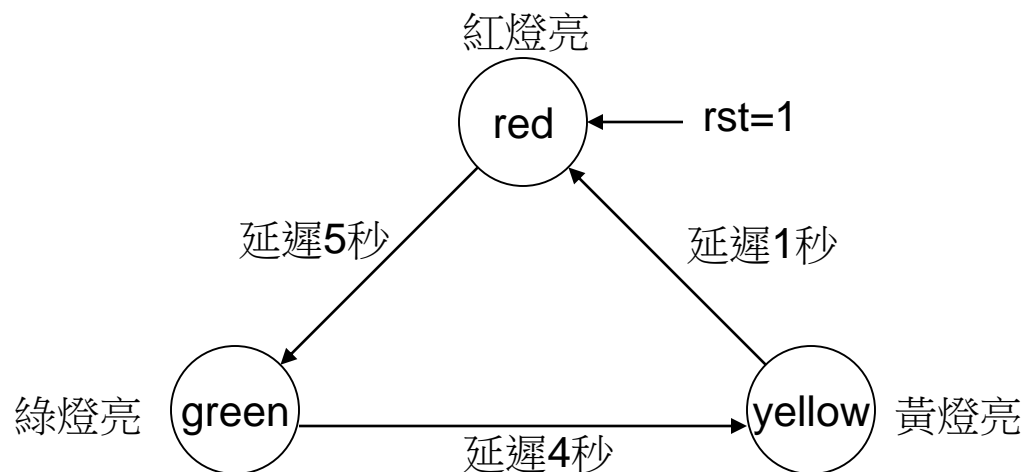


state\_a -> state\_d -> state\_b ->  
state\_c -> state\_a ...

# 基礎題 (二)

## 交通燈 (1/5)

- 請寫出具有以下功能的交通燈電路。
  - 設計時，請寫出具有以下功能的電路模組，並且確認編譯後沒有error或warning產生。
  - (模擬時，請自行撰寫testbench.v，並觀察波形結果或主控台的螢幕顯示結果是否如期望一般。)
  - 驗證時，請觀察虛擬儀器之運作是否如期望一般。



# 基礎題 (二)

## 交通燈 (2/5)

交通燈部分

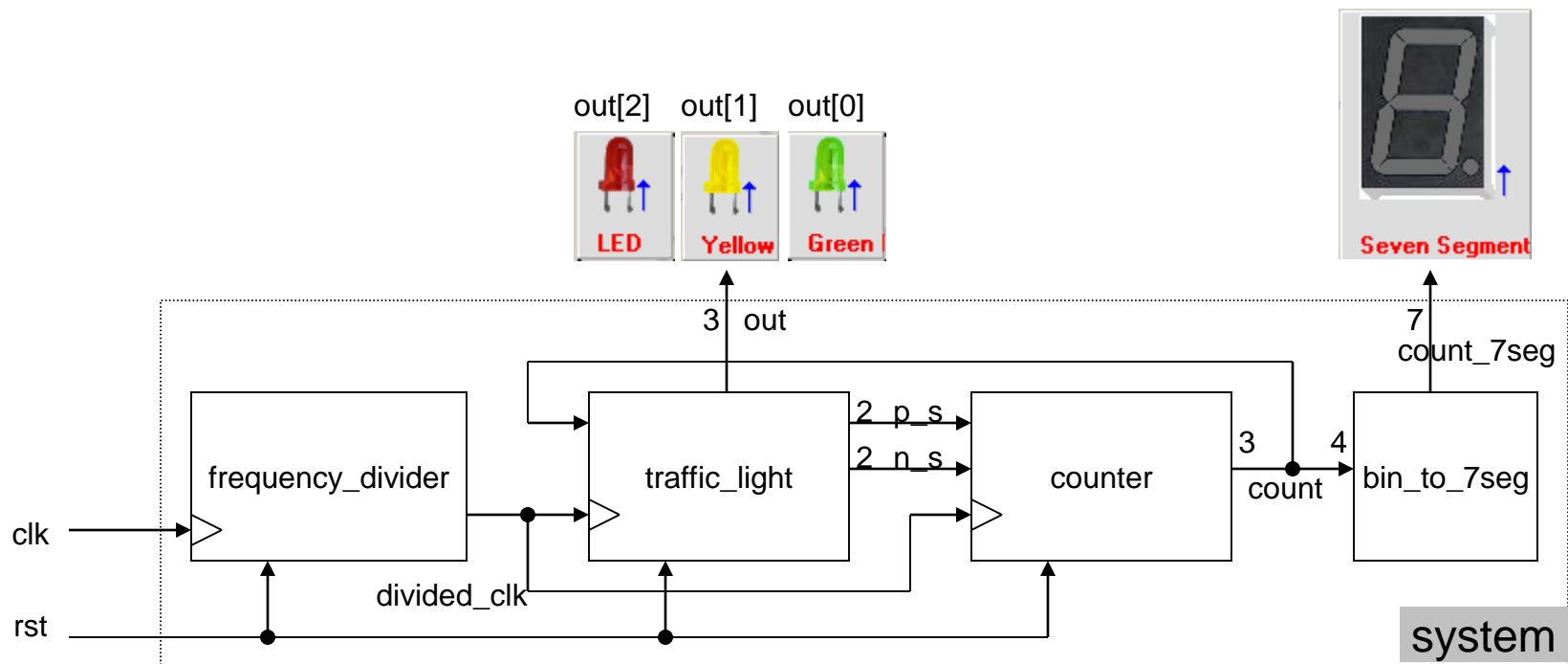
present state	output	next state						
		rst=1	count=1	count != 1	count=4	count != 4	count=5	count != 5
green	out = 001	red			yellow	green		
yellow	out = 010	red	red	yellow				
red	out = 100	red					green	red

計數器部分

present state	output	next state		
		rst=1	p_s = n_s	p_s != n_s
count (1~5)	count	1	count+1	1

# 基礎題 (二)

## 交通燈 (3/5)





# 基礎題 (二)

## 交通燈 (4/5)

```
module traffic_light(clk, rst, count, p_s, n_s, out);  
    input clk, rst;  
    input [2:0]count;  
    output [1:0]p_s, n_s;  
    output [2:0]out;
```

```
    parameter GREEN  =2'd0,  
                YELLOW=2'd1,  
                RED   =2'd2;
```

```
    reg [1:0]p_s, n_s;  
    reg [2:0]out;
```

```
    always@(posedge clk or p_rst)begin  
        記憶元件  
    end
```

```
    always@(*)begin
```

產生下個狀態的組合電路

```
    end
```

```
    always@(*)begin
```

產生輸出的組合電路



按讚才能看隱藏內容

```
    end
```

```
endmodule
```

traffic\_light

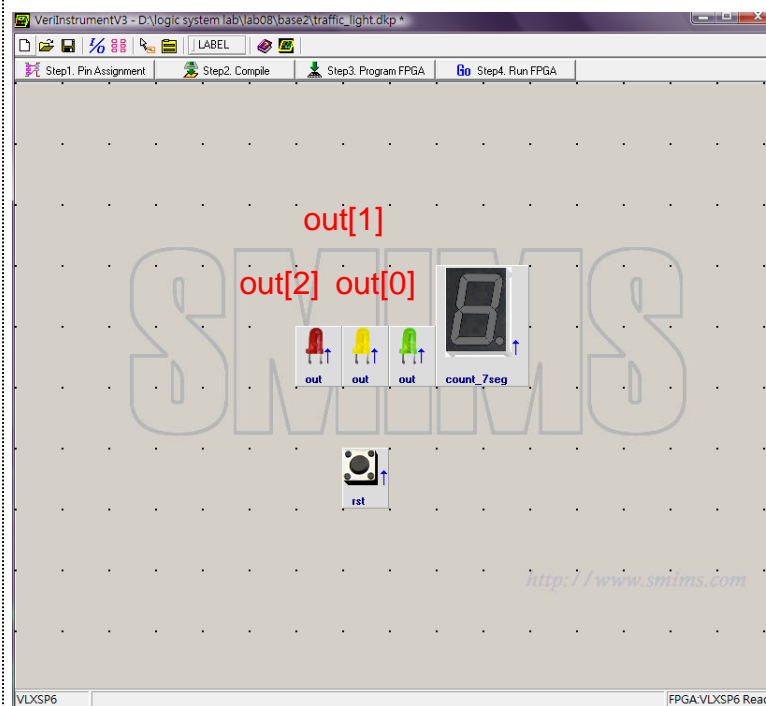
這三部分的程式碼，請同學參考狀態表自行撰寫

# 基礎題 (二)

## 交通燈 (5/5)

```
module counter(clk, rst, p_s, n_s, count);  
    input clk, rst;  
    input [1:0]p_s, n_s;  
    output [2:0]count;  
  
    reg [2:0]count, n_count;  
  
    always@(posedge clk or posedge rst)begin  
        記憶元件  
    end  
  
    always@(*)begin  
        產生下個狀態的組合電路  
    end  
  
endmodule
```

counter



這兩部分的程式碼，請同學參考狀態表自行撰寫

# 挑戰題

## 販賣機 (1/8)

- 請寫出具有下一頁功能的販賣機電路。
  - 設計時，請寫出具有下一頁功能的電路模組，並且確認編譯後沒有error或warning產生。
  - (模擬時，請自行撰寫testbench.v，並觀察波形結果或主控台的螢幕顯示結果是否如期望一般。)
  - 驗證時，請觀察虛擬儀器之運作是否如期望一般。

# 挑戰題

## 販賣機 (2/8)

VeriInstrumentV3 - D:\logic system lab\lab08\challenge\vending\_machine.dkp \*

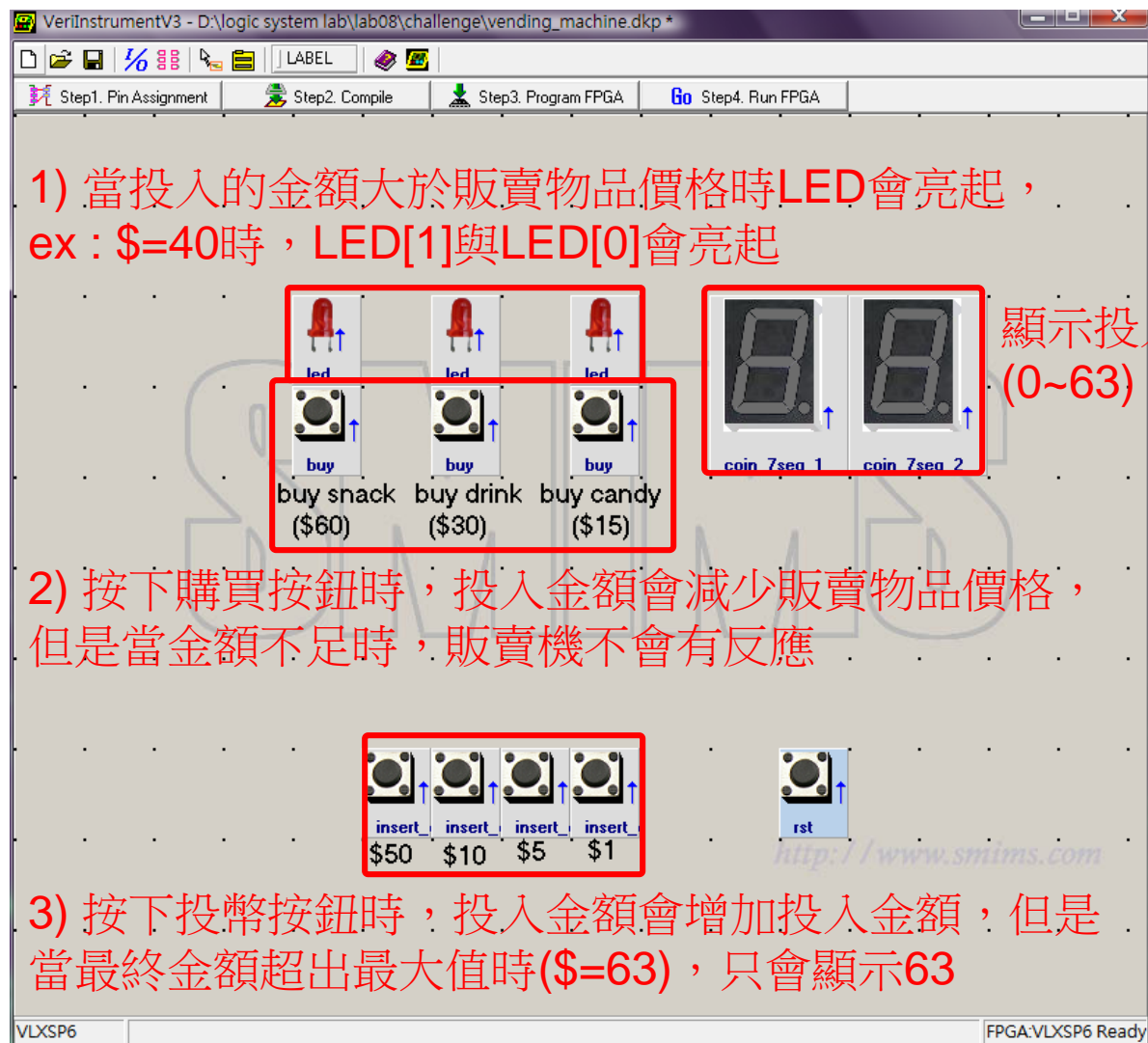
Step1. Pin Assignment Step2. Compile Step3. Program FPGA Go Step4. Run FPGA

1) 當投入的金額大於販賣物品價格時LED會亮起，  
ex：\$=40時，LED[1]與LED[0]會亮起

顯示投入的金額 (0~63)

2) 按下購買按鈕時，投入金額會減少販賣物品價格，  
但是當金額不足時，販賣機不會有反應

3) 按下投幣按鈕時，投入金額會增加投入金額，但是  
當最終金額超出最大值時(\$=63)，只會顯示63



# 挑戰題

## 販賣機 (3/8)

### 販賣機部分

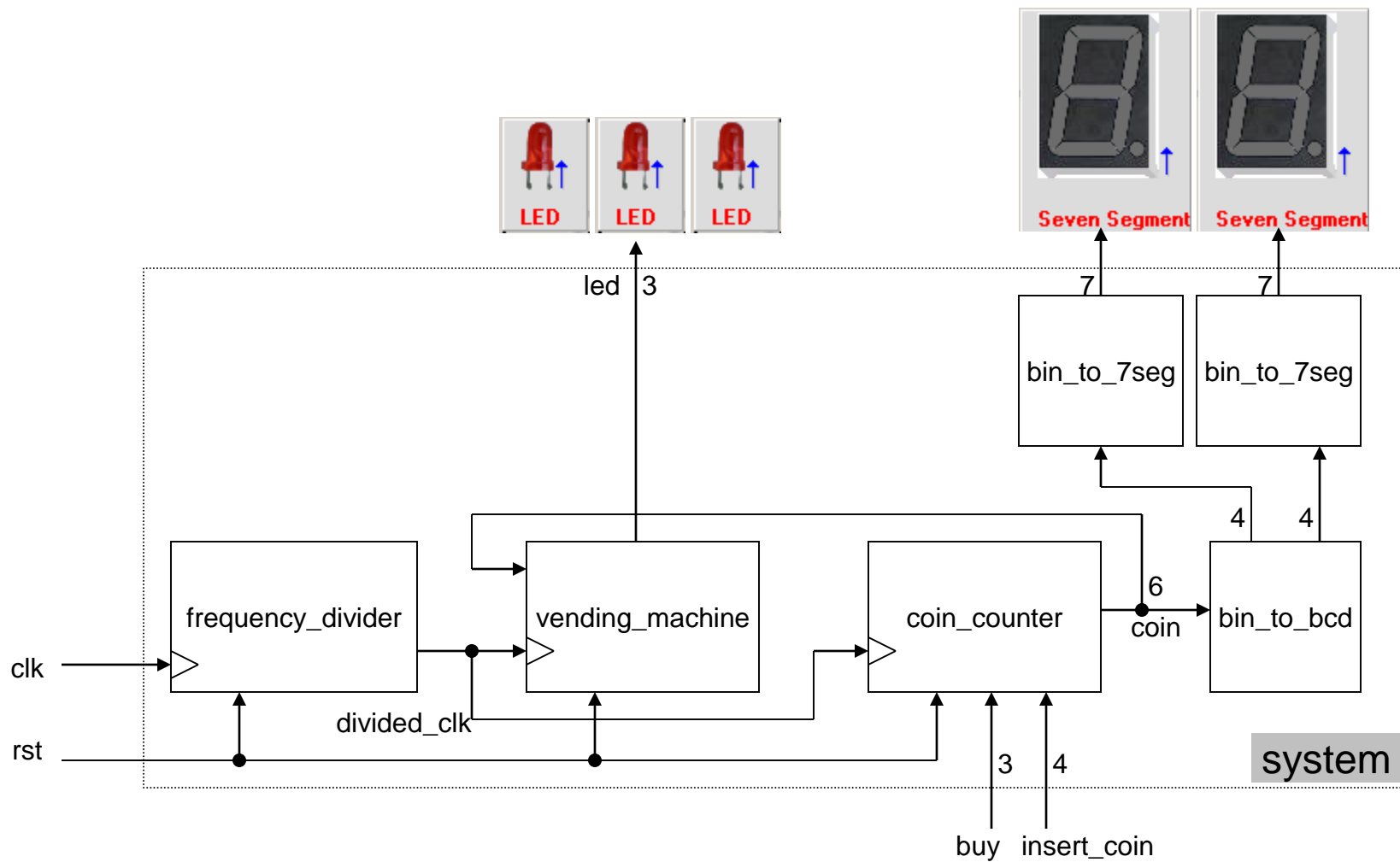
present state	output	next state				
		rst=1	15>coin>=0	30>coin>=15	60>coin>=30	64>coin>=60
insert	led=000	insert	insert	candy	drink	snack
candy	led=001	insert	insert	candy	drink	snack
drink	led=011	insert	insert	candy	drink	snack
snack	led=111	insert	insert	candy	drink	snack

### 金錢計數器部分

present state	output	next state							
		rst=1	buy candy	buy drink	buy snack	insert \$1	insert \$5	insert \$10	insert \$50
coin (\$) (0~63)	coin (\$)	0	\$>=15	\$>=30	\$>=60	\$<=62	\$<=58	\$<=53	\$<=13
			\$-15	\$-30	\$-60	\$+1	\$+5	\$+10	\$+50
			\$<15	\$<30	\$<60	\$>62	\$>58	\$>53	\$>13
			\$	\$	\$	\$	\$	\$	\$

# 挑戰題

## 販賣機 (4/8)



# 挑戰題

## 販賣機 (5/8)

```
module vending_machine(clk, rst, coin, led);  
    input clk, rst;  
    input [5:0]coin;  
    output [2:0]led;  
  
    reg [1:0]p_s, n_s;  
    reg [2:0]led;  
  
    wire cond1 = (coin>=0 && coin<15);  
    wire cond2 = (coin>=15 && coin<30);  
    wire cond3 = (coin>=30 && coin<60);  
    wire cond4 = (coin>=60 && coin<64);  
  
    parameter INSERT = 2'd0,  
               CANDY  = 2'd1,  
               DRINK  = 2'd2,  
               SNACK  = 2'd3;  
  
    always@(posedge clk or posedge rst)begin  
        記憶元件  
    end
```

這部分的程式碼，請同學  
參考狀態表自行撰寫

vending\_machine (continued)



# 挑戰題

## 販賣機 (6/8)

```
always@(*)begin
```

產生下個狀態的組合電路

```
end
```

```
always@(*)begin
```

產生輸出的組合電路

```
end
```

```
endmodule
```

```
vending_machine (end)
```

這兩部分的程式碼，請同學參考狀態表自行撰寫



# 挑戰題

## 販賣機 (7/8)

```
module coin_counter(clk, rst, buy, insert_coin, coin);
    input clk, rst;
    input [2:0]buy;
    input [3:0]insert_coin;
    output [5:0]coin;

    reg [5:0]coin, n_coin;

    `define BUY_CANDY    buy[0]
    `define BUY_DRINK    buy[1]
    `define BUY_SNACK    buy[2]
    `define INSERT_NT1    insert_coin[0]
    `define INSERT_NT5    insert_coin[1]
    `define INSERT_NT10    insert_coin[2]
    `define INSERT_NT50    insert_coin[3]

    always@(posedge clk or posedge rst)begin
        記憶元件
    end
```


這部分的程式碼，請同學  
參考狀態表自行撰寫

coin\_counter (continued)



# 挑戰題

## 販賣機 (8/8)



```
always@(*)begin
    if(`BUY_CANDY )
    else if(`BUY_DRINK )
    else if(`BUY_SNACK )
    else if(`INSERT_NT1 )
    else if(`INSERT_NT5 )
    else if(`INSERT_NT10)
    else if(`INSERT_NT50)
    else n_coin=coin;
end
endmodule
```

產生下個狀態的組合電路

coin\_counter (end)

這部分的程式碼，請同學參考狀態表自行撰寫

# 實驗結報繳交

- 基礎題 (一)
  - 請附上原始碼、(模擬結果擷圖)、驗證結果擷圖與解釋。
- 基礎題 (二)
  - 請附上原始碼、(模擬結果擷圖)、驗證結果擷圖與解釋。
- 挑戰題
  - 請附上原始碼、(模擬結果擷圖)、驗證結果擷圖與解釋。
- 各自之心得報告