111-1 535313

進階可程式邏輯系統設計與應用

Advanced Programmable Logic System Design and Application

實驗編號 : LAB 04

實驗單元 : Finite State Machines

實驗名稱 : Counter-like circuit

結報完成日期 : 2022.11.05

姓名 : 王語

系級 : 機械四

學號 : 0811127

1. 實驗目的

了解有限狀態機之學理，並且藉由練習設計有限狀態機，繪製每個狀態下對應輸入與下一個狀態，完成有+1、+2、+0、-1功能的計數器。

1. Verilog 程式碼

module Lab04\_0811127( input [0:0] KEY,

input [2:0] SW,

output [6:0] HEX0 );

reg [3:0] \_hex0 ;

wire [1:0] control;

ssd ssd\_0(.Din(\_hex0),.Dout(HEX0));

assign control = {SW[2],SW[1]};

always @( negedge KEY[0] ) begin

if (!SW[0]) begin:Active\_low\_reset

\_hex0 <= 4'd0 ;

end

else begin

case ( \_hex0 )

4'd0: begin

case (control)

2'd0: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

2'd1: begin

\_hex0 <= \_hex0 + 4'd1 ;

end

2'd2: begin

\_hex0 <= \_hex0 + 4'd2 ;

end

2'd3: begin

\_hex0 <= 4'd9 ;

end

default: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

endcase

end

4'd1: begin

case (control)

2'd0: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

2'd1: begin

\_hex0 <= \_hex0 + 4'd1 ;

end

2'd2: begin

\_hex0 <= \_hex0 + 4'd2 ;

end

2'd3: begin

\_hex0 <= \_hex0 - 4'd1 ;

end

default: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

endcase

end

4'd2: begin

case (control)

2'd0: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

2'd1: begin

\_hex0 <= \_hex0 + 4'd1 ;

end

2'd2: begin

\_hex0 <= \_hex0 + 4'd2 ;

end

2'd3: begin

\_hex0 <= \_hex0 - 4'd1 ;

end

default: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

endcase

end

4'd3: begin

case (control)

2'd0: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

2'd1: begin

\_hex0 <= \_hex0 + 4'd1 ;

end

2'd2: begin

\_hex0 <= \_hex0 + 4'd2 ;

end

2'd3: begin

\_hex0 <= \_hex0 - 4'd1 ;

end

default: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

endcase

end

4'd4: begin

case (control)

2'd0: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

2'd1: begin

\_hex0 <= \_hex0 + 4'd1 ;

end

2'd2: begin

\_hex0 <= \_hex0 + 4'd2 ;

end

2'd3: begin

\_hex0 <= \_hex0 - 4'd1 ;

end

default: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

endcase

end

4'd5: begin

case (control)

2'd0: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

2'd1: begin

\_hex0 <= \_hex0 + 4'd1 ;

end

2'd2: begin

\_hex0 <= \_hex0 + 4'd2 ;

end

2'd3: begin

\_hex0 <= \_hex0 - 4'd1 ;

end

default: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

endcase

end

4'd6: begin

case (control)

2'd0: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

2'd1: begin

\_hex0 <= \_hex0 + 4'd1 ;

end

2'd2: begin

\_hex0 <= \_hex0 + 4'd2 ;

end

2'd3: begin

\_hex0 <= \_hex0 - 4'd1 ;

end

default: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

endcase

end

4'd7: begin

case (control)

2'd0: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

2'd1: begin

\_hex0 <= \_hex0 + 4'd1 ;

end

2'd2: begin

\_hex0 <= \_hex0 + 4'd2 ;

end

2'd3: begin

\_hex0 <= \_hex0 - 4'd1 ;

end

default: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

endcase

end

4'd8: begin

case (control)

2'd0: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

2'd1: begin

\_hex0 <= \_hex0 + 4'd1 ;

end

2'd2: begin

\_hex0 <= 4'd0 ;

end

2'd3: begin

\_hex0 <= \_hex0 - 4'd1 ;

end

default: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

endcase

end

4'd9: begin

case (control)

2'd0: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

2'd1: begin

\_hex0 <= 4'd0 ;

end

2'd2: begin

\_hex0 <= \_hex0 + 4'd1 ;

end

2'd3: begin

\_hex0 <= \_hex0 - 4'd1 ;

end

default: begin

\_hex0 <= \_hex0 + 4'd0 ;

end

endcase

end

default: \_hex0 <= \_hex0 ;

endcase

end

end

endmodule

// sub module

//seven segment display

module ssd ( input [3:0] Din,

output [6:0] Dout );

assign Dout[0] = ((!Din[3])&(!Din[2])&(!Din[1])&( Din[0]))|

((!Din[3])&( Din[2])&(!Din[1])&(!Din[0]))|

(( Din[3])&( Din[2])&(!Din[1])&( Din[0]))|

(( Din[3])&(!Din[2])&( Din[1])&( Din[0]));

assign Dout[1] = ((!Din[3])&( Din[2])&(!Din[1])&( Din[0]))|

( ( Din[2])&( Din[1])&(!Din[0]))|

(( Din[3])& ( Din[1])&( Din[0]))|

(( Din[3])&( Din[2])& (!Din[0]));

assign Dout[2] = ((!Din[3])&(!Din[2])&( Din[1])&(!Din[0]))|

(( Din[3])&( Din[2])&( Din[1]) )|

(( Din[3])&( Din[2])& (!Din[0]));

assign Dout[3] = ((!Din[3])&(!Din[2])&(!Din[1])&( Din[0]))|

((!Din[3])&( Din[2])&(!Din[1])&(!Din[0]))|

(( Din[3])&(!Din[2])&( Din[1])&(!Din[0]))|

( ( Din[2])&( Din[1])&( Din[0]));

assign Dout[4] = ((!Din[3])& &( Din[0]))|

( (!Din[2])&(!Din[1])&( Din[0]))|

((!Din[3])&( Din[2])&(!Din[1]) );

assign Dout[5] = (( Din[3])&( Din[2])&(!Din[1])&( Din[0]))|

((!Din[3])&(!Din[2])& ( Din[0]))|

((!Din[3])&(!Din[2])&( Din[1]) )|

((!Din[3])& ( Din[1])&( Din[0]));

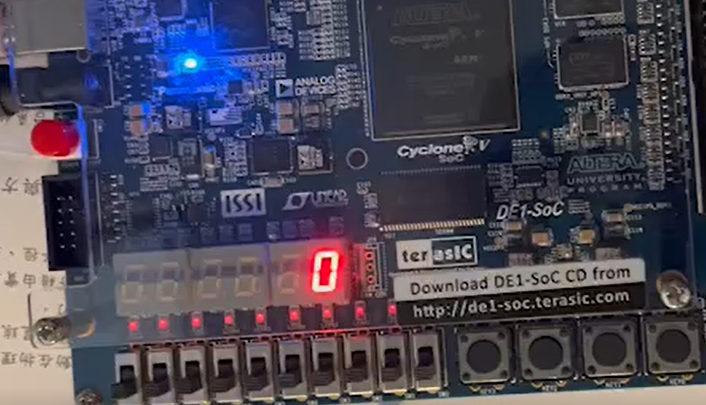
assign Dout[6] = ((!Din[3])&( Din[2])&( Din[1])&( Din[0]))|

(( Din[3])&( Din[2])&(!Din[1])&(!Din[0]))|

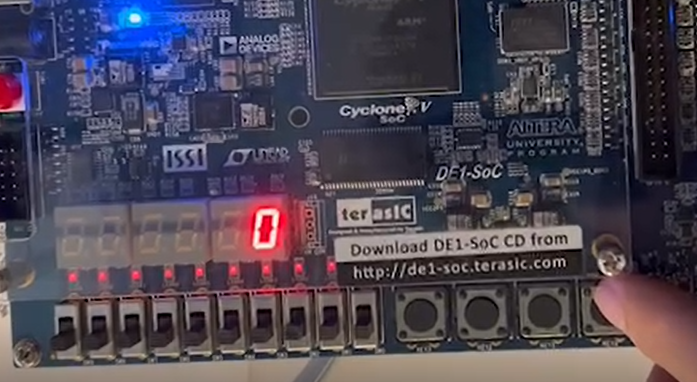
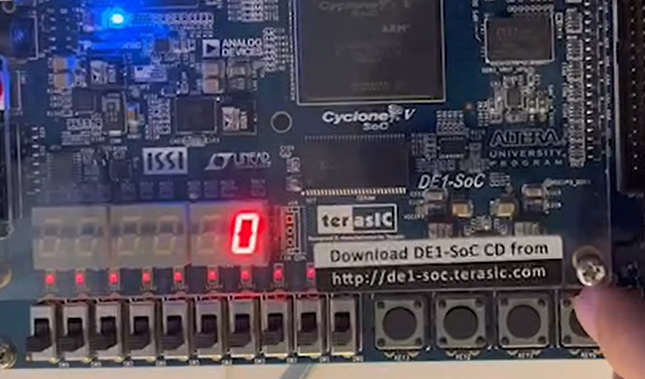
((!Din[3])&(!Din[2])&(!Din[1]) );

endmodule

1. 實驗結果

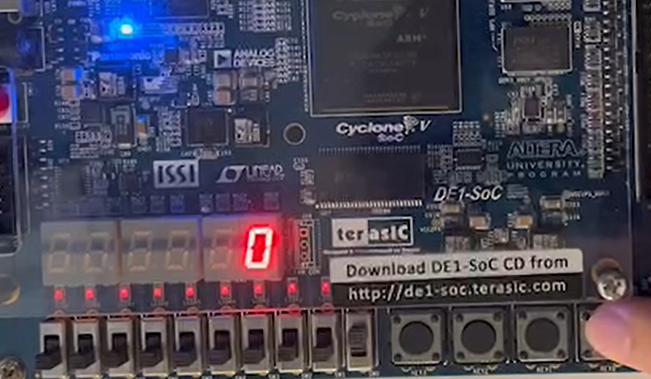
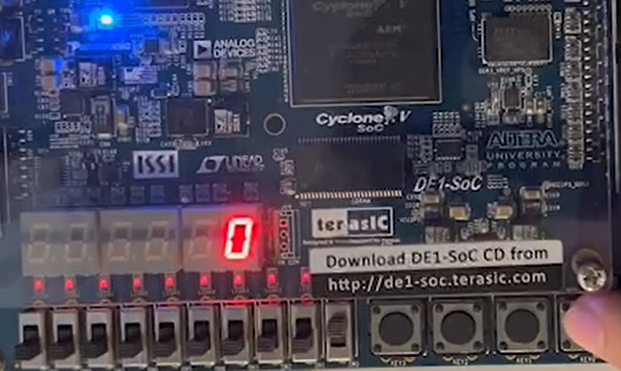
初始狀態

SW[2] = 0 (w1)，SW[1] = 0 (w0)，SW[0] = 0 (Active-Low reset)

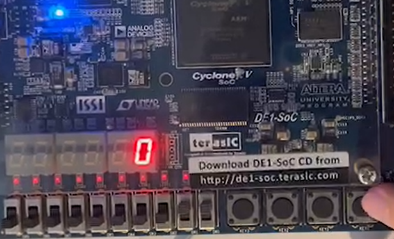


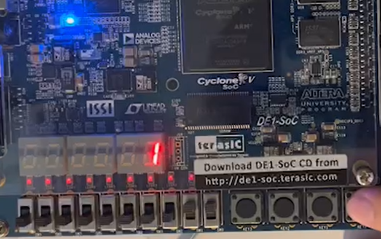
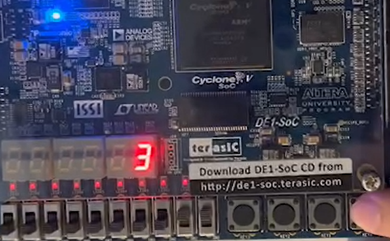
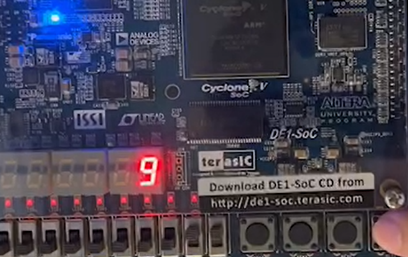
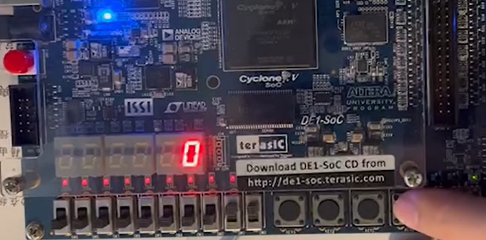
clocks

SW[2] = 0 (w1)，SW[1] = 0 (w0)，SW[0] = 1 (Active-Low reset)



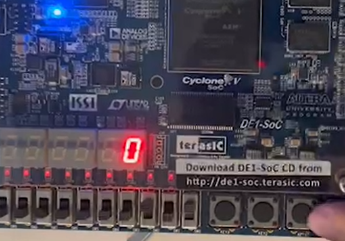
clocks

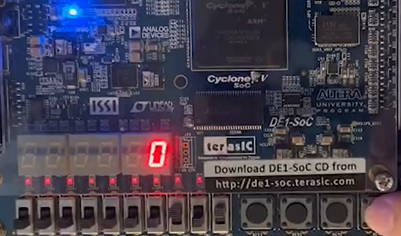
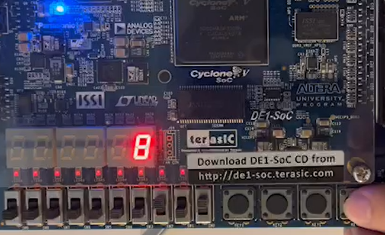
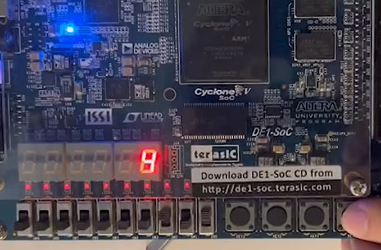
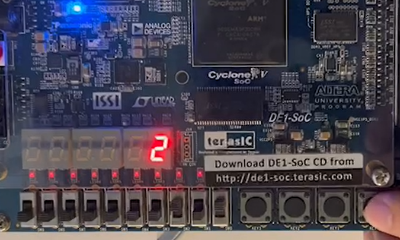
SW[2] = 0 (w1)，SW[1] = 1 (w0)，SW[0] = 1 (Active-Low reset)



…

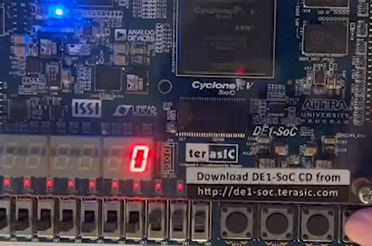
clocks

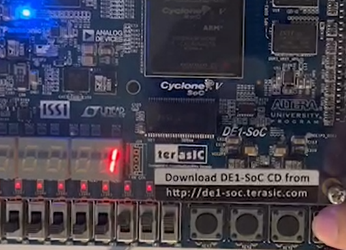
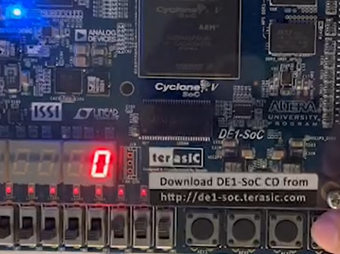
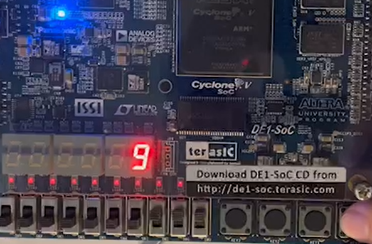
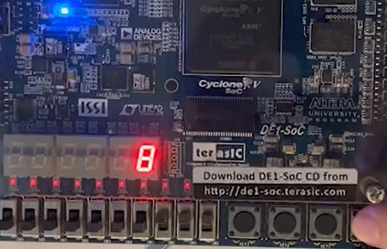
SW[2] = 1 (w1)，SW[1] = 0 (w0)，SW[0] = 1 (Active-Low reset)



…

clocks

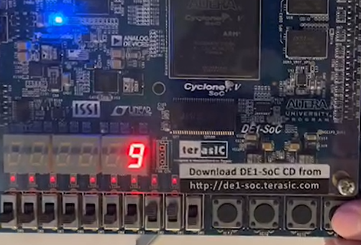
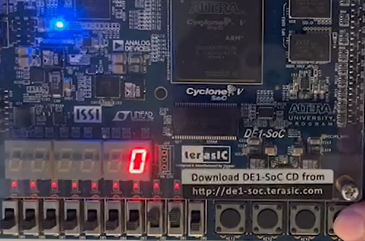
SW[2] = 0 (w1)，SW[1] = 0 (w0)，SW[0] = 1 (Active-Low reset)



…

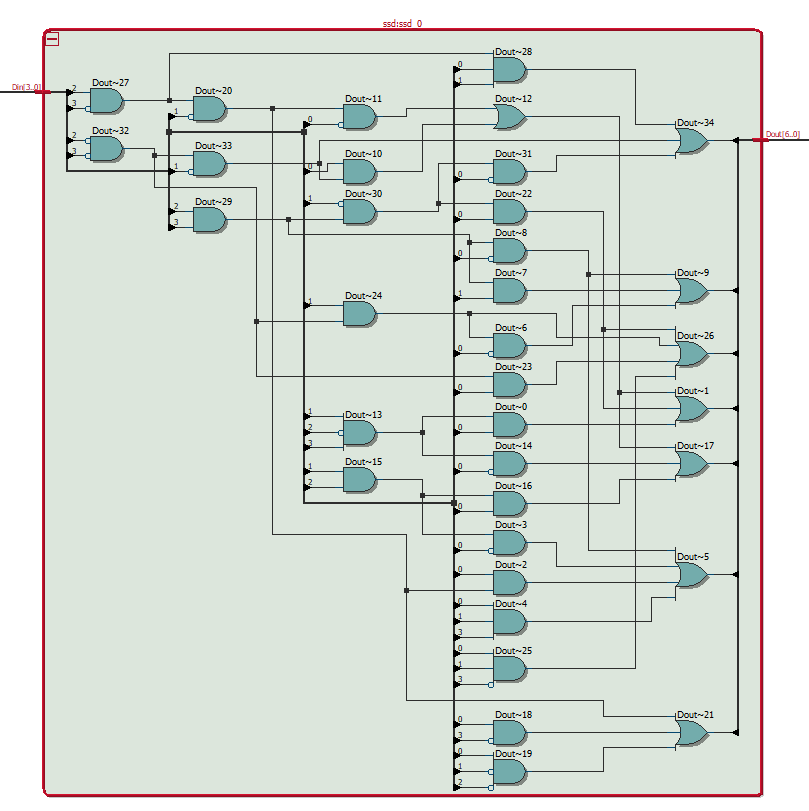
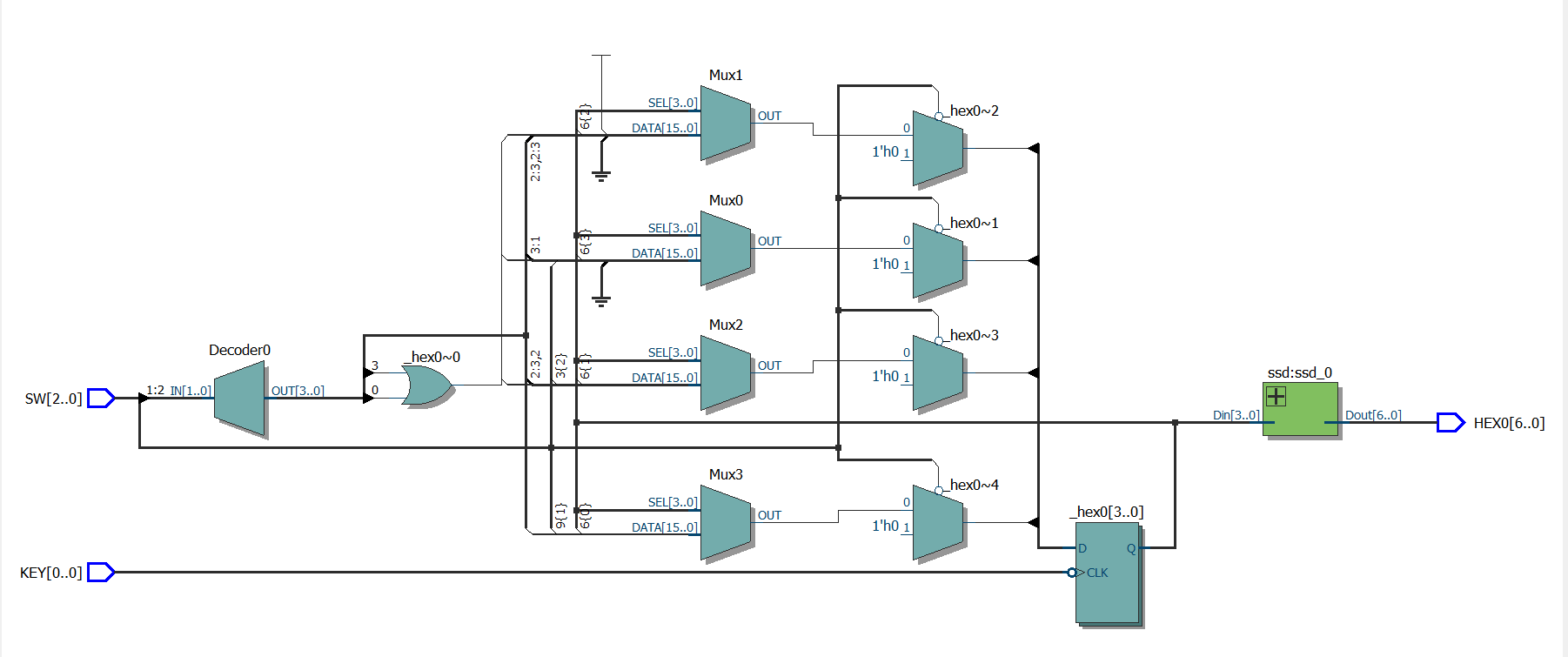
clocks

SW[2] = X (w1)，SW[1] = X (w0)，SW[0] = 1 (Active-Low reset)



clock

1. RTL



1. 問題與討論

本次實驗過程中發現DE1的硬體缺陷，周邊電路中的KEY按鈕有時按一下會連續觸發多次，可能是機械結構老舊，觸底的銅片防鏽度層脫落導致氧化所造成，也有可能是該元件的防彈跳設計不佳所致，不過該問題可以透過更換腳位解決，以其他KEY完成實驗即可避開。

此外，也觀察到Verilog不會自動匹配Interger的長度，如果

assign SW[3:0] = {0,0,0,1} ;會出現錯誤，0 和 1 的預設長度是32 bits，因此應該更正為 assign SW[3:0] = 4’b0001 ;。

1. 心得

這次實驗真的很順，寫有限狀態機跟填真值表一樣，雖然程式碼很多行，但是邏輯重複率高，只要照著步驟一步一步把下一個狀態填入，就能順利完成，獲益良多，謝謝助教批閱。