第八次實驗報告

題目:8-1~8-4

姓名:羅名志

學號:0813228

繳交日期:2022/5/5

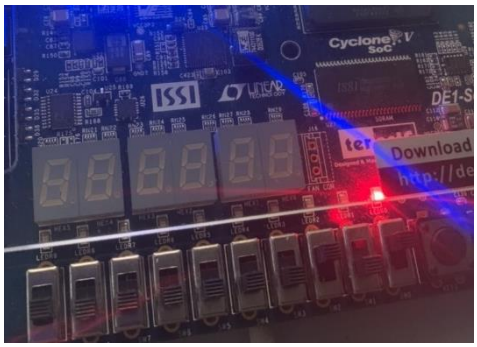
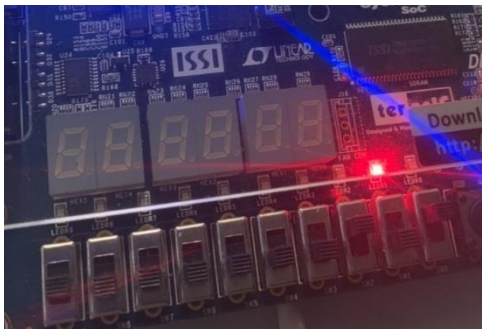
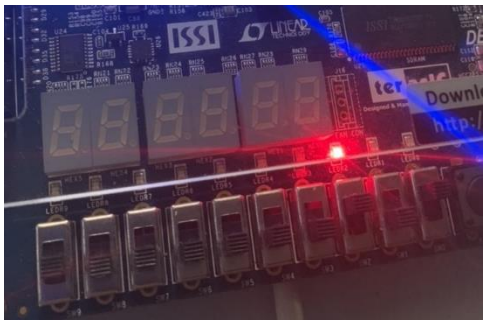
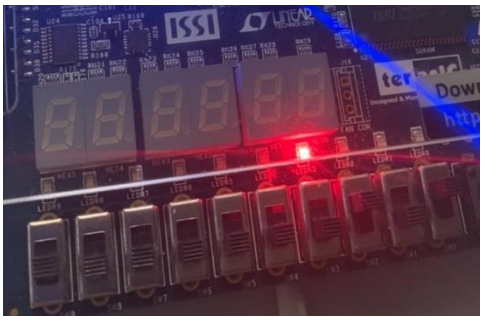
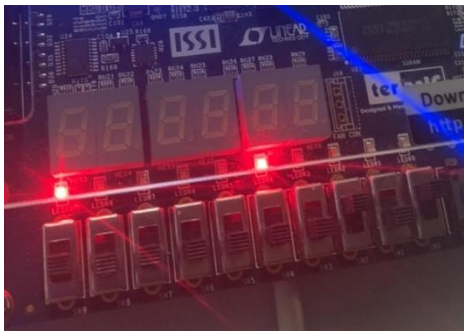
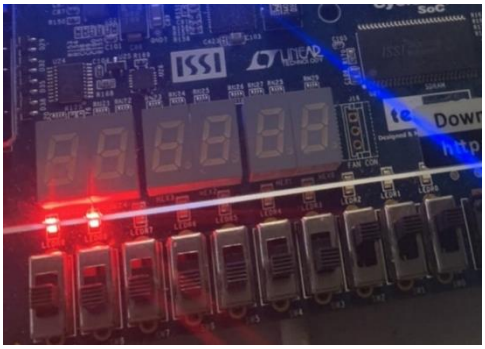
一、8-1

(1)實驗程式碼:

```
module labeight1(SW,KEY,LEDR);
	input [1:0] SW;
	input [0:0] KEY;
	output [9:0]LEDR;
	wire [8:0]out;
	wire w;

	assign w=SW[1];
	assign LEDR[8:0]=out;
	assign LEDR[9]=(out[4]|out[8]);

	init a(~KEY[0],SW[0],out[0]);
	d_f_f d1(~KEY[0],SW[0],(~w)&(out[0]|out[5]|out[6]|out[7]|out[8]),out[1]);
	d_f_f d2(~KEY[0],SW[0],(~w)&(out[1]),out[2]);
	d_f_f d3(~KEY[0],SW[0],(~w)&(out[2]),out[3]);
	d_f_f d4(~KEY[0],SW[0],(~w)&(out[3]|out[4]),out[4]);
	d_f_f d5(~KEY[0],SW[0],(w)&(out[0]|out[1]|out[2]|out[3]|out[4]),out[5]);
	d_f_f d6(~KEY[0],SW[0],(w)&(out[5]),out[6]);
	d_f_f d7(~KEY[0],SW[0],(w)&(out[6]),out[7]);
	d_f_f d8(~KEY[0],SW[0],(w)&(out[7]|out[8]),out[8]);
endmodule


module init(clk,reset,Q);
	input clk;
	input reset;
	output reg Q;
	always@(posedge clk) begin
		if(~reset) begin
			Q<=1;
		end
		else begin
			Q<=0;
		end
	end
```

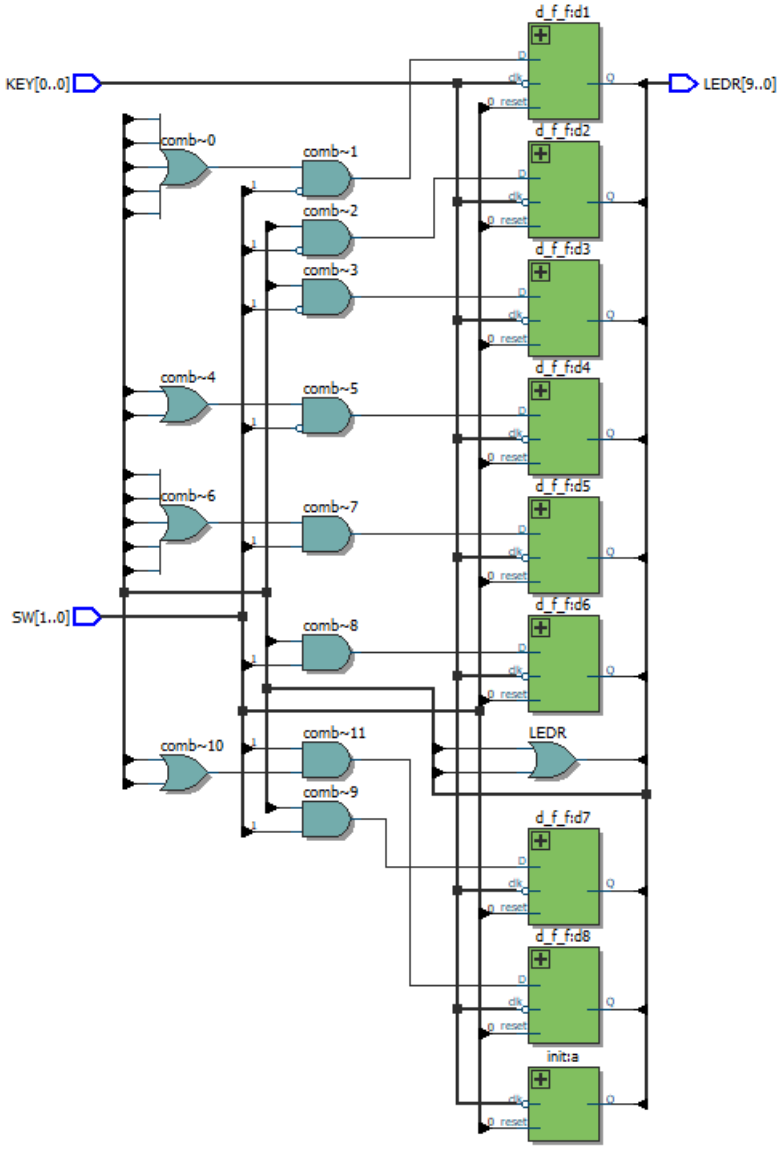
```
endmodule


module d_f_f(clk,reset,D,Q);
	input clk;
	input reset;
	input D;
	output reg Q;

	always@(posedge clk ) begin
		if(~reset) begin
			Q<=0;
		end
		else begin
			Q<=D;
		end
	end

endmodule
```

(2)實驗結果:

// 從上至下，左至右: A(resset) / B / C / D / E / I

(3)RTL 布局:



(4)問題與討論:

這次實驗內容主要跟狀態機有關，透過 flip-flop 的設計根據上個狀態來運算下個狀態的位置，而這部分較為複雜的部份是要設計 2 種 flip-flop，才能區隔開 A 狀態和其他狀態，整體來說算是這次實驗較為簡單的部分。

二、8-2

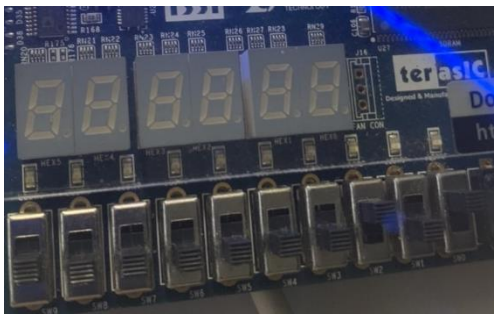
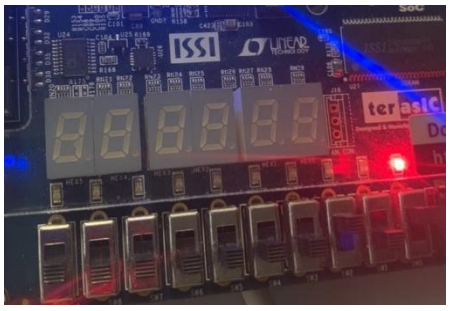
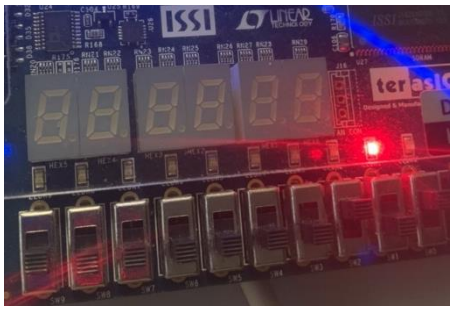
(1)實驗程式碼:

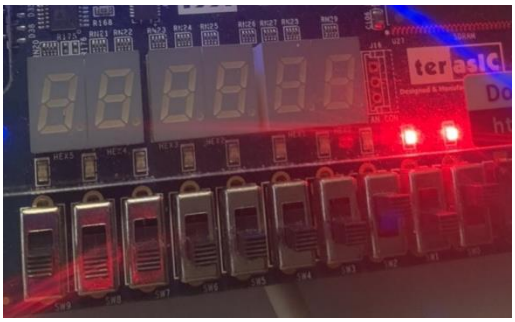
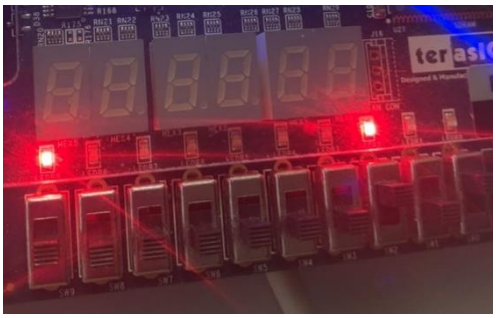
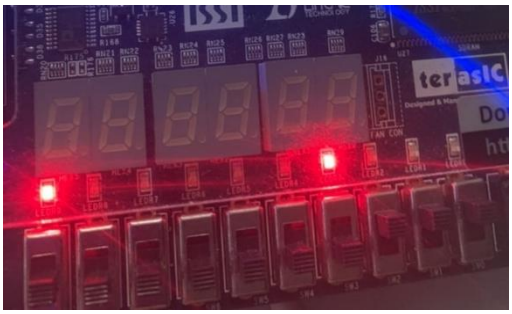
```
module labeight2(SW,KEY,LEDR);
	input [1:0]SW;
	input [0:0]KEY;
	output [9:0]LEDR;
	reg [3:0] Y;
	reg r;
	parameter
A=4'b0000,B=4'b0001,C=4'b0010,D=4'b0011,E=4'b0100,F=4'b0101,G=4'b0110,H=4'b
0111,I=4'b1000;

	always @(posedge KEY[0]) begin
		if(KEY[0]==1) begin
		case(Y)
			A: if(SW[1]==0) begin
					Y=B;
					r=0;
				end
				else begin
					Y=F;
					r=0;
				end
			B: if(SW[1]==0) begin
					Y=C;
					r=0;
				end
				else begin
					Y=F;
					r=0;
				end
			C: if(SW[1]==0) begin
					Y=D;
					r=0;
				end
				else begin
					Y=F;
					r=0;
```

```
        end
D: if(SW[1]==0) begin
            Y=E;
            r=1;
        end
        else begin
            Y=F;
            r=0;
        end
E: if(SW[1]==0) begin
            Y=E;
            r=1;
        end
        else begin
            Y=F;
            r=0;
        end
F: if(SW[1]==0) begin
            Y=B;
            r=0;
        end
        else begin
            Y=G;
            r=0;
        end
G: if(SW[1]==0) begin
            Y=B;
            r=0;
        end
        else begin
            Y=H;
            r=0;
        end
H: if(SW[1]==0) begin
            Y=B;
            r=0;
        end
        else begin
```

```
                    Y=I;
                    r=1;
                end
            I: if(SW[1]==0) begin
                    Y=B;
                    r=0;
                end
                else begin
                    Y=I;
                    r=1;
                end
            default: begin
                Y=A;
                r=0;
            end
        endcase
        end

    if(SW[0]==0) begin
        Y=A;
        r=0;
    end

    end

    assign LEDR[3:0]=Y;
    assign LEDR[9]=r;

endmodule
```

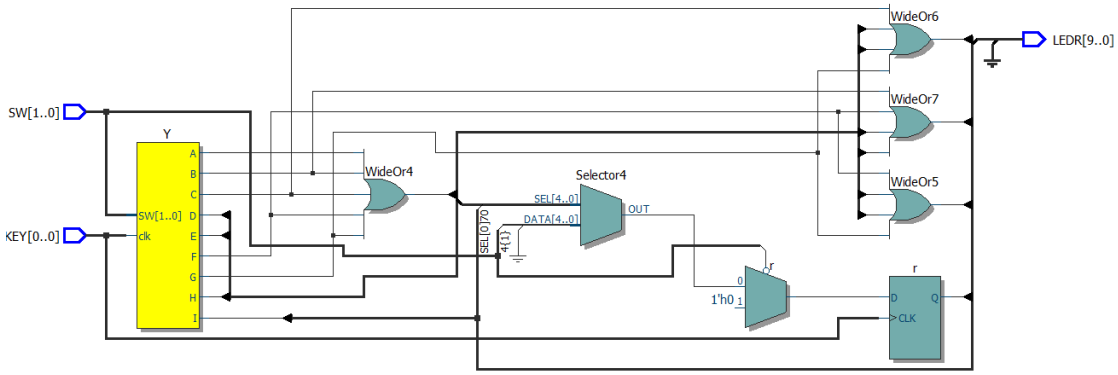
(2)實驗結果:

// 從上至下，左至右: A(resset) / B / C / D / E / I

(3)RTL 布局:



(4)問題與討論:

是第一次學到 case，發現透過 case 和 if-else 的幫助可以大幅降低程式的複雜度，這 part 與第一部分非常像，不同的地方只有 state code 為 4 bit，所以透過簡單的狀態描述以及對應到的 assignment 即可完成，算是此次學到最多的部分。

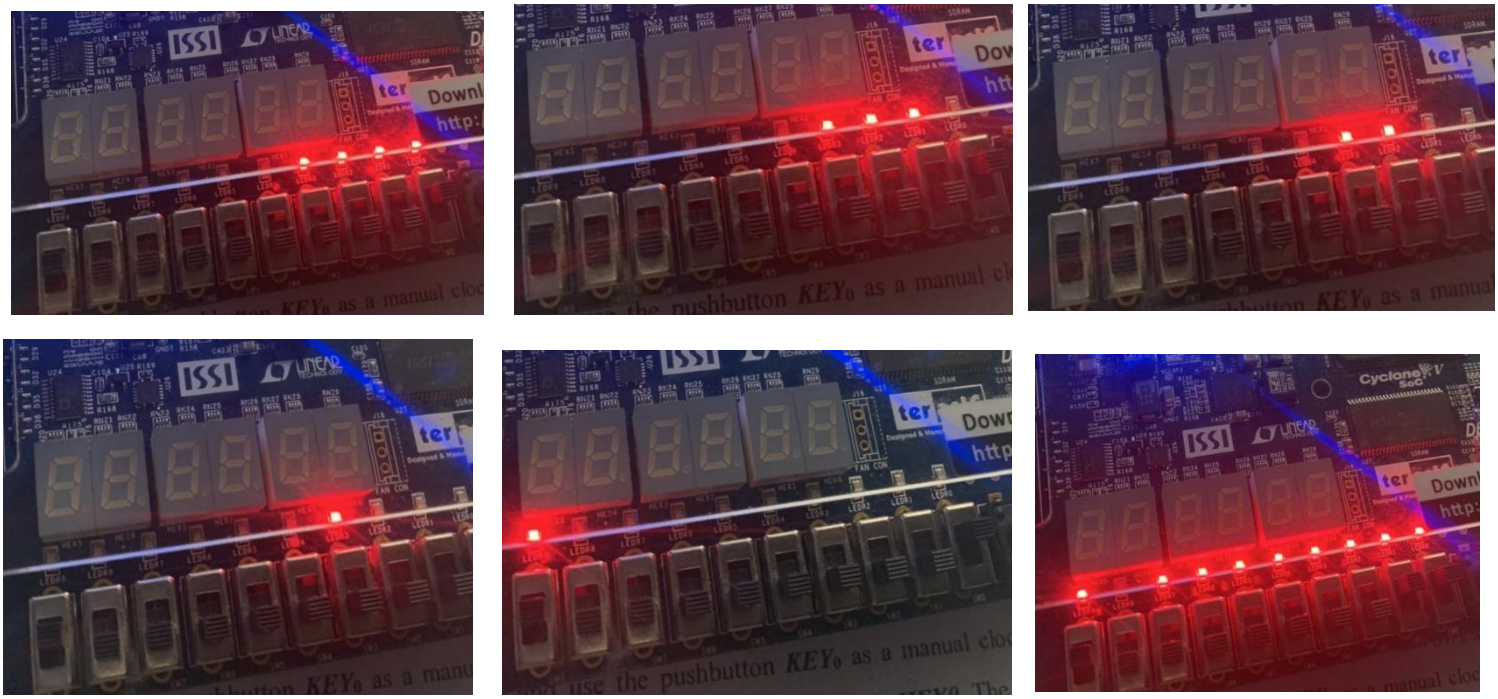
三、8-3

(1)實驗程式碼:

```
module labeight3(SW, KEY, LEDR);
	input [1:0]SW;
	input [1:0]KEY;
	output [9:0]LEDR;
	reg [3:0]shift1;
	reg [3:0]shift2;
	reg pre,now;
	reg z;

	always@(posedge KEY[0]) begin
		if(~SW[0]) begin
			shift1 = 4'b1111;
			shift2 = 0;
			pre = 0;
			now = 0;
			z = 0;
		end
		else begin
			pre = now;
			now = SW[1];
			if(pre==now) begin
				if(SW[1]) begin
					shift2=shift2<<1;
					shift2[0] = 1;
				end
				else begin
					shift1 = shift1<<1;
					shift1[0] = 0;
				end
			end

			else begin
				if(now) begin
					shift1 = 4'b1111;
					shift2 = 4'b0001;
				end
```

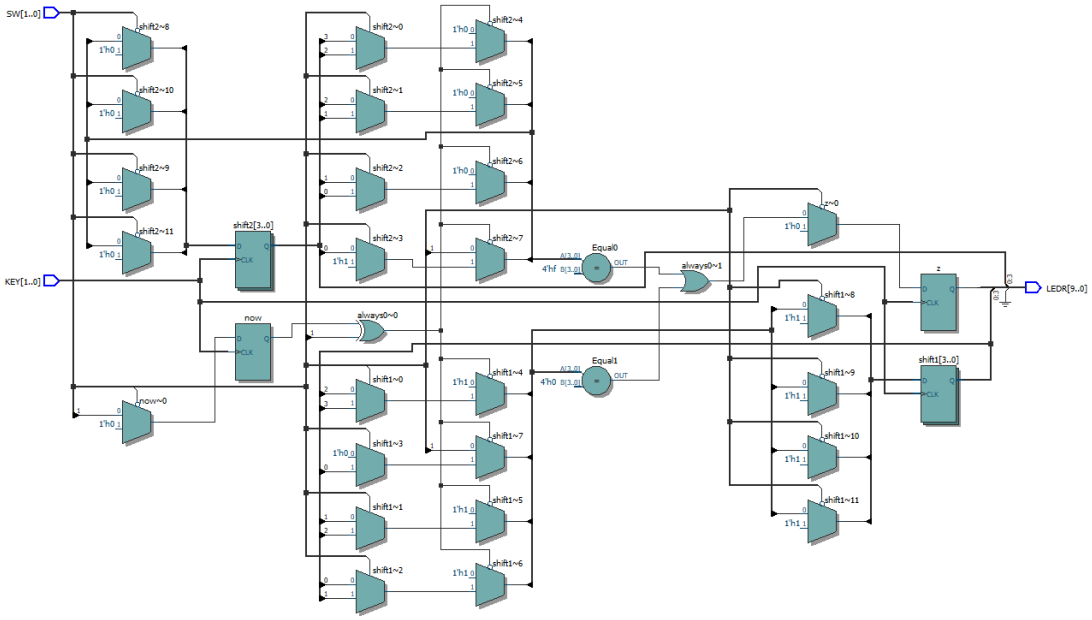
```
                else begin
                        shift2 = 0;
                shift1 = 4'b1110;
                end
        end

        if((shift2==4'b1111)|(shift1==4'b0000)) begin
                z = 1;
        end
        else begin
                z = 0;
        end
    end
end

assign LEDR[9] = z;
assign LEDR[7:4] = shift2[3:0];
assign LEDR[3:0] = shift1[3:0];

endmodule
```

(2)實驗結果:



// 從上至下，左至右: A(resset) / F / G / H / I / E

(3)RTL 布局:



(4)問題與討論:

第一次利用 shift 直接做乘除，透過兩個 shift，分別是前 4 bit 進位和後 4 bit 的近位，因此當連續狀態發生時(pre=now)，即將對應的部分進行進位並調整最後一項的值，算是這次實驗最複雜的部份，需要再更熟悉 shift 的使用

四、8-4

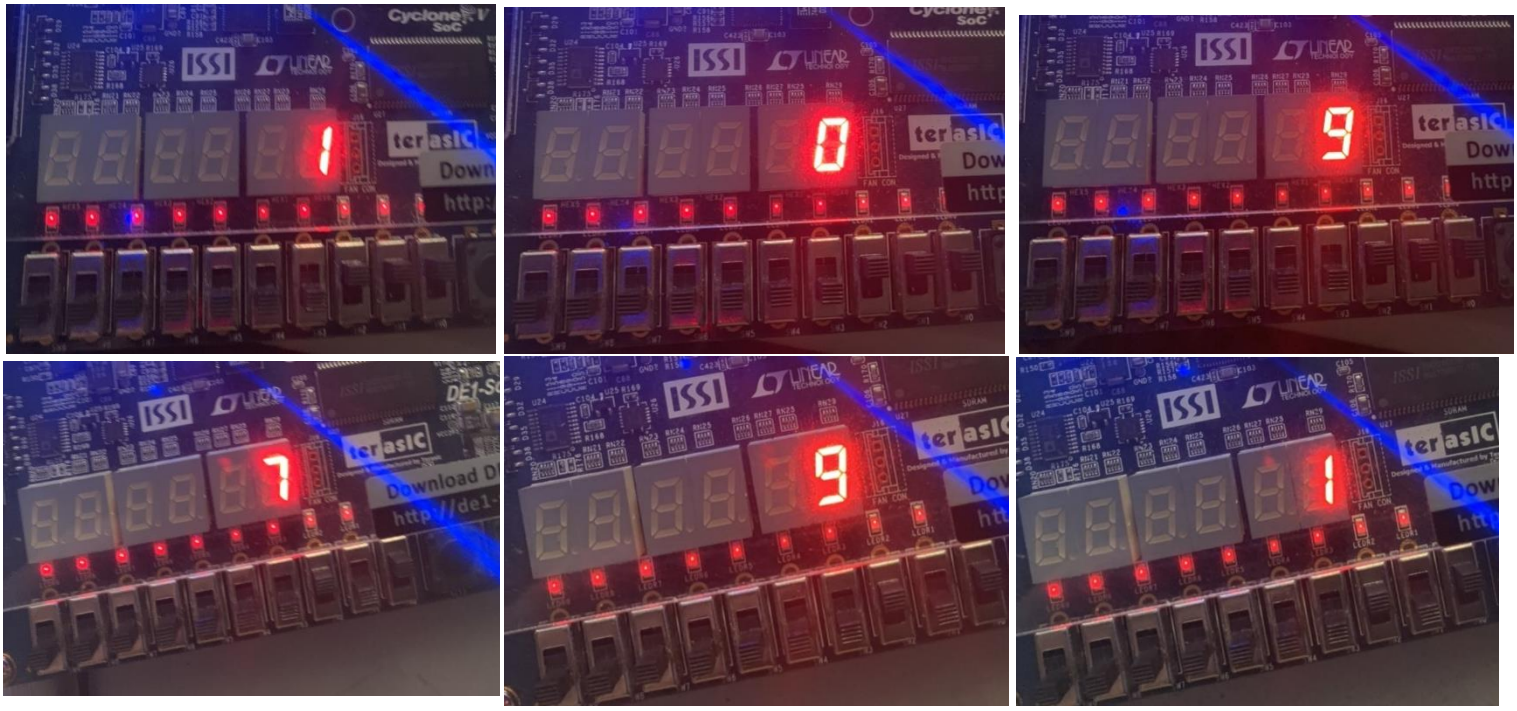
(1)實驗程式碼:

```
module labeight4(SW,KEY,HEX0);
	input [2:0]SW;
	input [1:0]KEY;
	output [6:0]HEX0;
	reg [3:0] num;
	reg w0,w1;
	parameter c1=2'b00,c2=2'b01,c3=2'b10,c4=2'b11;
	integer flag=0;

	always @(negedge KEY[0]) begin
		w0=SW[1];
		w1=SW[2];
		if(~SW[0]) begin
			num=0;
			w0=0;
			w1=0;
		end
		else begin
			case({w0,w1})
				c1: begin
					num=num;
				end
				c2: begin
					num=num+1;
					if(num>=10) begin
						num=0;
					end
				end
				c3: begin
					num=num+2;
					if(num==10) begin
						num=0;
					end
					else if(num==11) begin
						num=1;
					end
```

```
                end

                c4: begin
                    if(flag==1) begin
                        num=10;
                        flag=0;
                    end
                    if(num>=1)
                        if(flag==0)
                            num=num-1;
                                if(num==0)
                                    flag=1;
                    end

                default: begin
                    num=0;
                end
            endcase
        end
    end

    display7seg seg1(num,HEX0);

endmodule

module display7seg(num,HEX);
    input [3:0]num;
    output reg[6:0]HEX;
    always@(num) begin
        case(num)
            0:HEX=7'b1000000;
            1:HEX=7'b1111001;
            2:HEX=7'b0100100;
            3:HEX=7'b0110000;
            4:HEX=7'b0011001;
            5:HEX=7'b0010010;
            6:HEX=7'b0000010;
            7:HEX=7'b1111000;
```

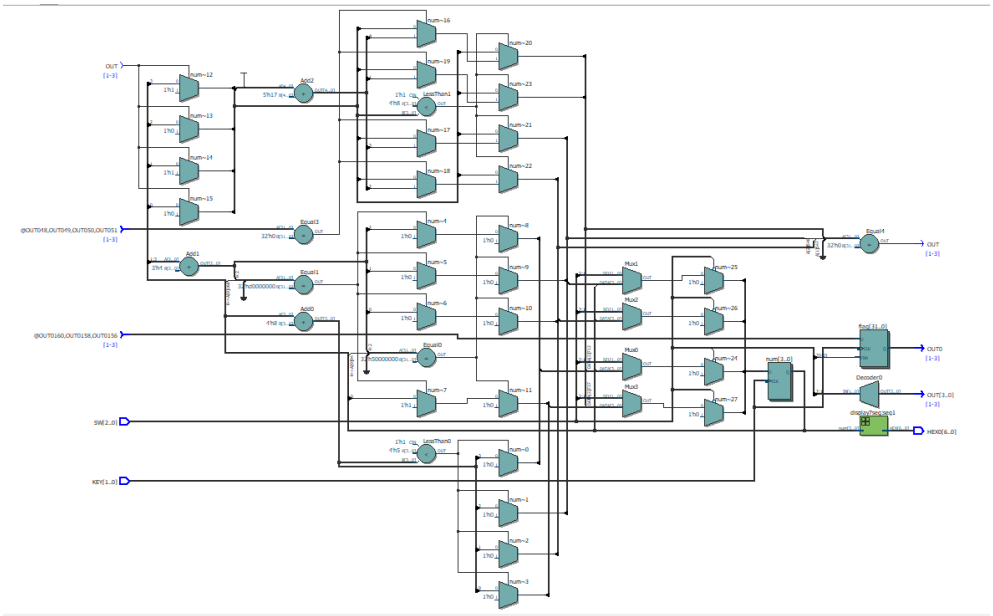
```
                8:HEX=7'b0000000;
                9:HEX=7'b0010000;
                10:HEX=7'b0001000;
                11:HEX=7'b0000011;
                12:HEX=7'b1000110;
                13:HEX=7'b0100001;
                14:HEX=7'b0000110;
                15:HEX=7'b0001110;
            endcase
        end
endmodule
```

(2)實驗結果:



//上: w0=1 / w1=1, 下: w0=0 / w1=1

(3)RTL 布局:



(4)問題與討論:

這部分我幾乎是完全不會，不過與同學討論後發現只要用 case 同時考慮 w1、w0 兩個變數即可，另外再把超出 9 的值歸零開始算，而減法的部分則是利用 assign 值等於 10，再從 10 開始扣，算是需要一點技巧的部分，總結來說，這次實驗收穫很多，熟悉了 case、shift 的應用。