

第三次實驗報告

題目:3-1~3-4

姓名:羅名志

學號:0813228

繳交日期:2022/3/16

一、3-1

(1)實驗程式碼:

```

1 module labthree1(SW,HEX0);
2   input [3:0]SW;
3   output [6:0]HEX0;
4
5   assign HEX0[0]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|
6   assign HEX0[1]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|
7   assign HEX0[2]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|
8   assign HEX0[3]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|
9   assign HEX0[4]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|
10  assign HEX0[5]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|
11  assign HEX0[6]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|
12
13 endmodule

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13

```

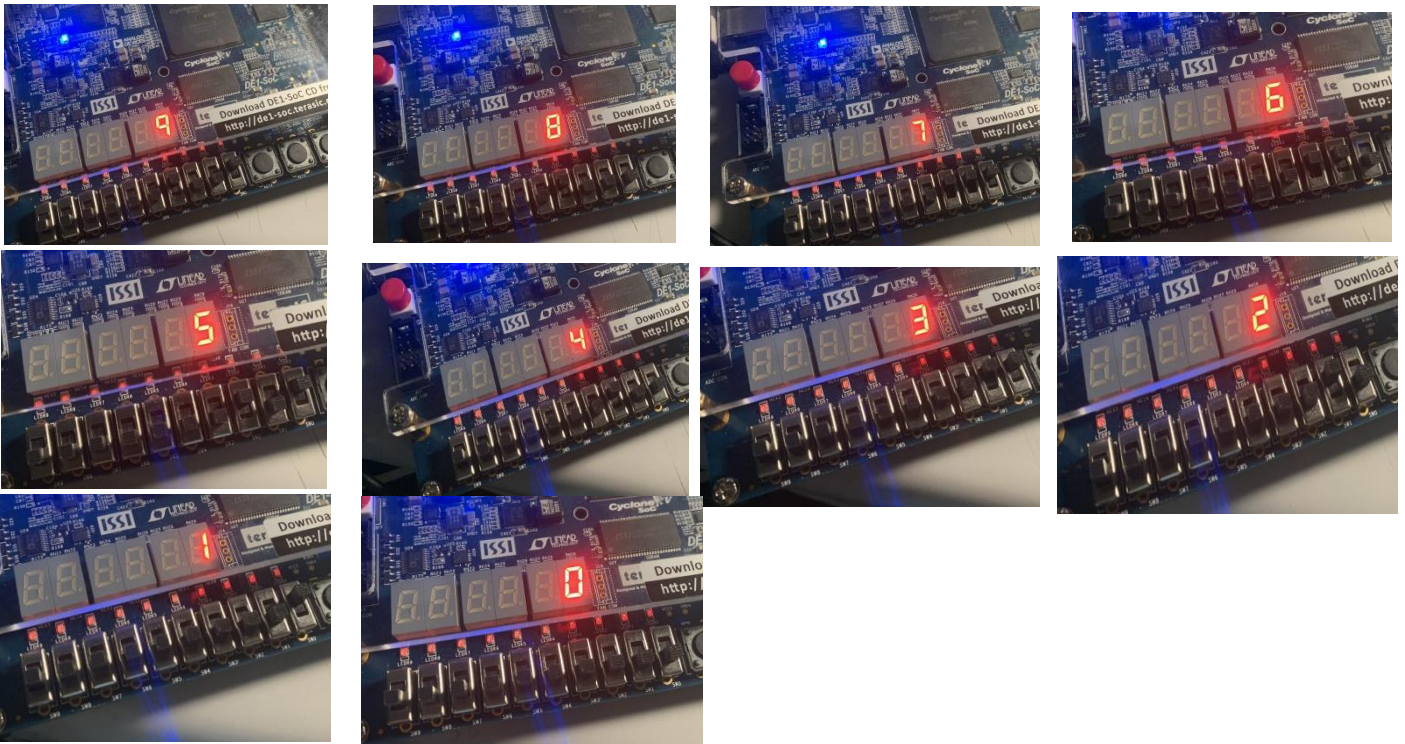
```

1
2
3
4
5
6
7
8
9
10
11
12
13

```

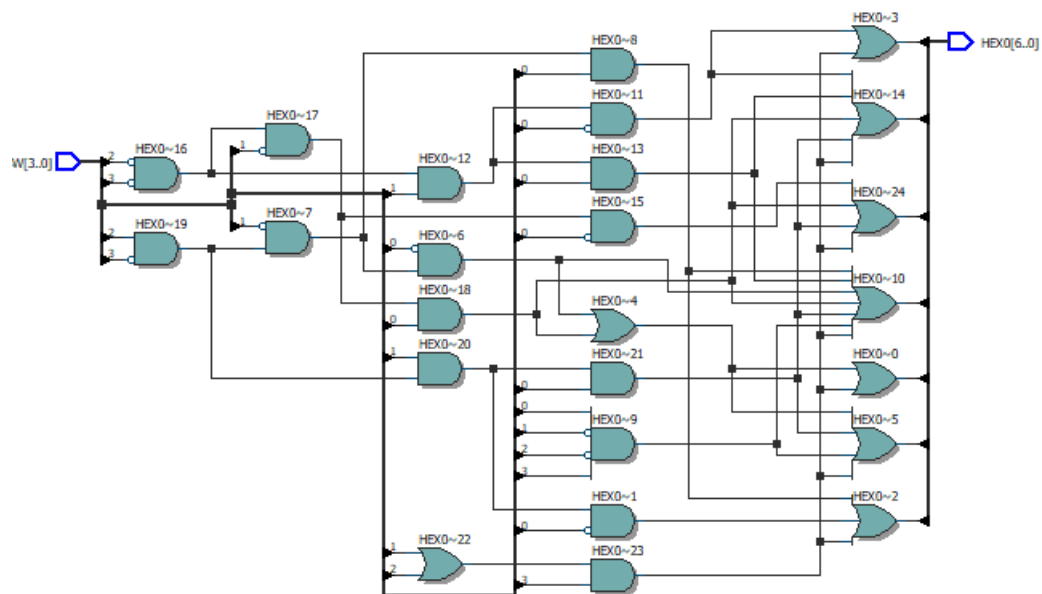
//由左至右分成 3 張圖，將每個組合對應到 hex0

(2)實驗結果:



//上排(由左至右，SW[3:0]):1001/1000/0111/0110
 //中排(由左至右，SW[3:0]):0101/0100/0011/0010
 //下排(由左至右，SW[3:0]):0001/0000

(3)RTL 布局:



(4)問題與討論:

此實驗較簡單，只要把不同組合分別對應到 HEX0 上就好，剛好可以複習上次的實驗內容。

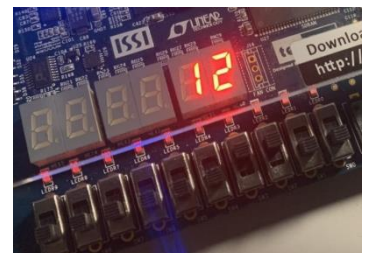
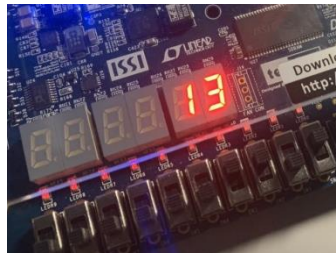
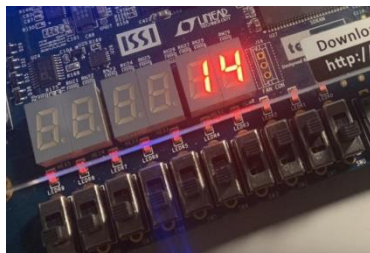
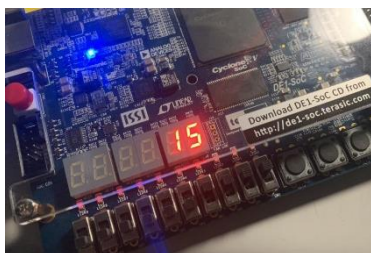
二、3-2

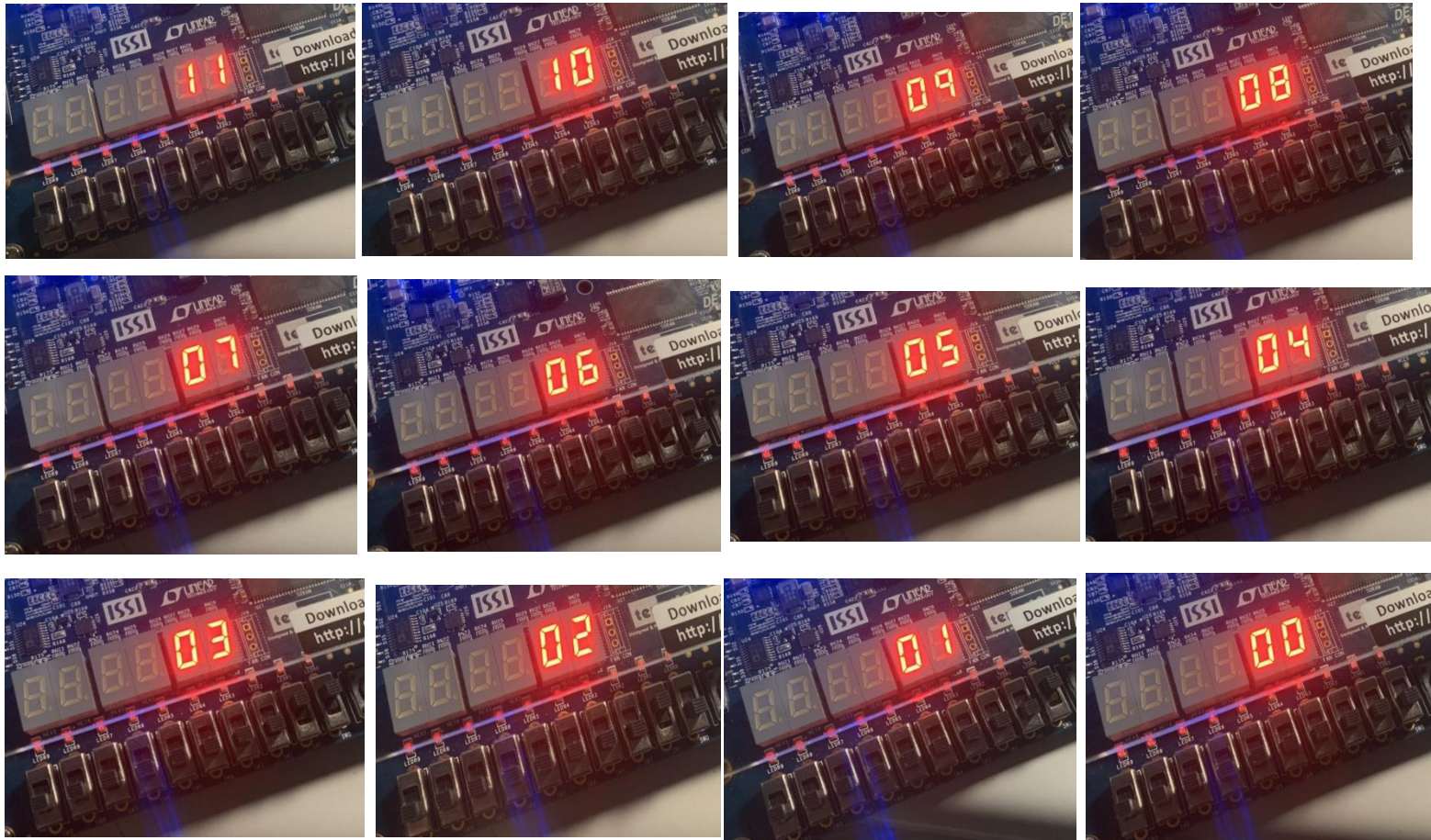
(1)實驗程式碼:

```
1 module labthree2(SW,HEX1,HEX0);
2 input [3:0]SW;
3 output [6:0]HEX1,HEX0;
4 wire sign ;
5
6 wire [3:0]y, v;
7
8
9
10 comparator comp(SW[3],SW[2],SW[1],SW[0],sign
11 circuitA a(SW[3:0],sign,v[3:0]);
12
13 circuitB b(sign,HEX1);
14 char_7seg c(v,HEX0 );
15
16 endmodule
17
18 module comparator(a,b,c,d,sign);
19 input a,b,c,d;
20 output sign ;
21 wire s5,s4,s3,s2,s1,s0;
22 assign sign = s5|s4|s3|s2|s1|s0;
23 assign s5 = a&~b&c&~d; //10
24 assign s4 = a&~b&c&d; //11
25 assign s3 = a&b&~c&~d; //12
26 assign s2 = a&b&~c&d; //13
27 assign s1 = a&b&c&~d; //14
28 assign s0 = a&b&c&d; //15
29 endmodule
30
31 module circuitA(SW,sign,v);
32 input [3:0]SW;
33 input sign;
34 wire [2:0]x;
35
36 output [3:0]v;
37 assign x[0]=(~SW[2]&SW[1]&~SW[0]&0)|(~SW[2]&SW[1]&SW[0]&1)|(SW[2]&~SW[1]&~SW[0]&0)|(SW[2]&~SW[1]&SW[0]&1)|(SW[2]&SW[1]&~SW[0]&0)|(SW[2]&SW[1]&SW[0]&1);
38 assign x[1]=(~SW[2]&SW[1]&~SW[0]&0)|(~SW[2]&SW[1]&SW[0]&1)|(SW[2]&~SW[1]&~SW[0]&1)|(SW[2]&~SW[1]&SW[0]&0)|(SW[2]&SW[1]&~SW[0]&1)|(SW[2]&SW[1]&SW[0]&0);
39 assign x[2]=(~SW[2]&SW[1]&~SW[0]&0)|(~SW[2]&SW[1]&SW[0]&1)|(SW[2]&~SW[1]&~SW[0]&1)|(SW[2]&~SW[1]&SW[0]&0)|(SW[2]&SW[1]&~SW[0]&1)|(SW[2]&SW[1]&SW[0]&0);
40
41 assign v[0]=(~sign&SW[0])|(sign&x[0]);
42 assign v[1]=(~sign&SW[1])|(sign&x[1]);
43 assign v[2]=(~sign&SW[2])|(sign&x[2]);
44 assign v[3]=(~sign&SW[3])|(sign&0);
45
46
47 endmodule
48
49 module circuitB(a,hex);
50 input a;
51 output [6:0]hex;
52
53
54 assign hex[0]=(a&1);
55 assign hex[1]=(a&0);
56 assign hex[2]=(a&0);
57 assign hex[3]=(a&1);
58 assign hex[4]=(a&1);
59 assign hex[5]=(a&1);
60 assign hex[6]=(a&1)|(~a&1);
61 endmodule
62
63
64 module char_7seg(SW,HEX);
65 input [3:0]SW;
66 output [6:0]HEX;
67
68 assign HEX[0]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|(~SW[3]&SW[2]&~SW[1]&~SW[0]&0)|
69 assign HEX[1]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|(~SW[3]&SW[2]&~SW[1]&~SW[0]&0)|
70 assign HEX[2]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|(~SW[3]&SW[2]&~SW[1]&~SW[0]&0)|
71 assign HEX[3]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|(~SW[3]&SW[2]&~SW[1]&~SW[0]&0)|
72 assign HEX[4]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|(~SW[3]&SW[2]&~SW[1]&~SW[0]&0)|
73 assign HEX[5]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|(~SW[3]&SW[2]&~SW[1]&~SW[0]&0)|
74 assign HEX[6]=(~SW[3]&~SW[2]&~SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&~SW[1]&SW[0]&1)|(~SW[3]&~SW[2]&SW[1]&~SW[0]&0)|(~SW[3]&~SW[2]&SW[1]&SW[0]&1)|(~SW[3]&SW[2]&~SW[1]&~SW[0]&0)|
75 endmodule
```

//68-75 行同 LAB 3-1，故部分省略

(2)實驗結果:





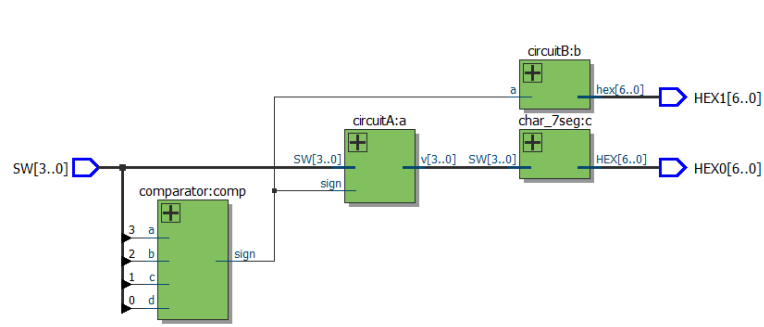
//1 排(由左至右, SW[3:0]):1111/1110/1101/1100

//2 排(由左至右, SW[3:0]):1011/1010/1001/1000

//3 排(由左至右, SW[3:0]):0111/0110/0101/0100

//4 排(由左至右, SW[3:0]):0011/0010/0001/0000

(3)RTL 布局:



(4)問題與討論:

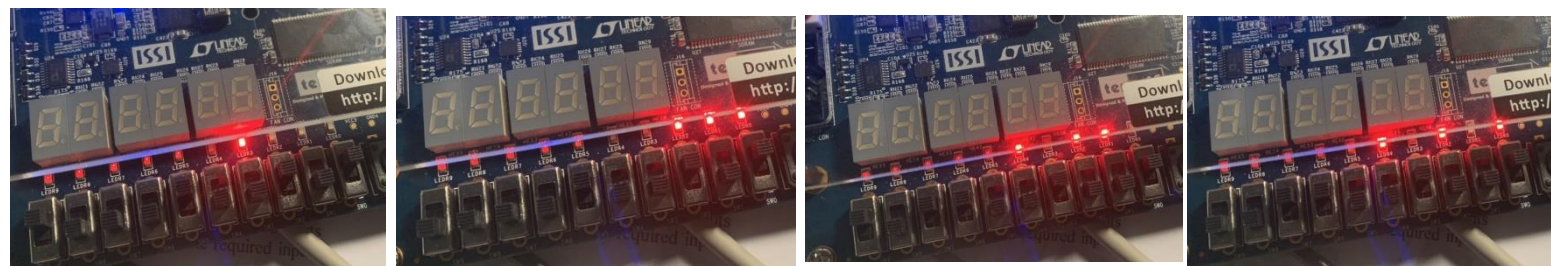
這邊開始有點複雜了起來, 想非常久, 尤其是 1010-1111 的部分如何轉換成 10-15, 同時學到了很多 AND OR 搭配 Wire 的用法, 算是這次實驗中最難的部分。

三、3-3

(1)實驗程式碼:

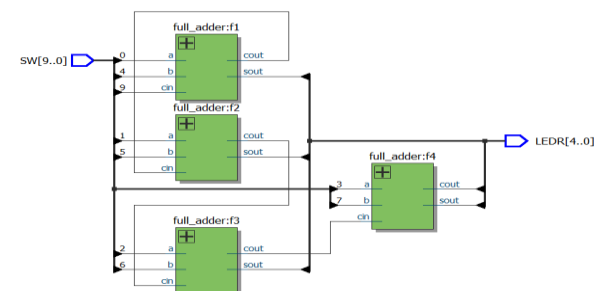
```
1 module labthree3(SW, LEDR);
2   input [9:0]SW;
3
4   wire [3:0]s;
5   wire [3:0]c;
6   output [4:0]LEDR;
7
8   full_adder f1(SW[0], SW[4], SW[9], c[0], s[0]);
9   full_adder f2(SW[1], SW[5], c[0], c[1], s[1]);
10  full_adder f3(SW[2], SW[6], c[1], c[2], s[2]);
11  full_adder f4(SW[3], SW[7], c[2], c[3], s[3]);
12
13  assign LEDR[0]=s[0];
14  assign LEDR[1]=s[1];
15  assign LEDR[2]=s[2];
16  assign LEDR[3]=s[3];
17  assign LEDR[4]=c[3];
18
19 endmodule
20
21 module full_adder(a,b,cin,cout,sout);
22   input a;
23   input b;
24   input cin;
25   output cout;
26   output sout;
27   wire C1, C2, S1;
28
29   process(a,b,C1,S1);
30   process(S1,cin,C2,sout);
31   assign cout=C1|C2;
32
33 endmodule
34
35 module process(x,y,C,S);
36   input x,y;
37   output C, S;
38
39   assign C=(x&y);
40   assign S=~(x&y) | (x&~y);
41
42 endmodule
```

(2)實驗結果:



//(由左至右，sw[9],sw[7:0]):1, 00100101/0, 00100101/1, 11001001/0, 11001001

(3)RTL 布局



(4)問題與討論:

是第一次學到多個加法器的運作，利用 XOR 和 AND 結合，產生出對應的 C、S，覺得是這次實驗中較簡單好懂的部分

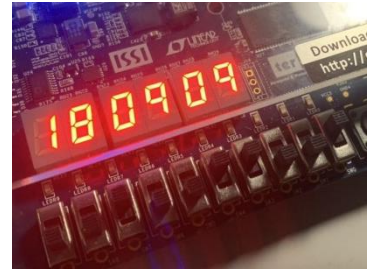
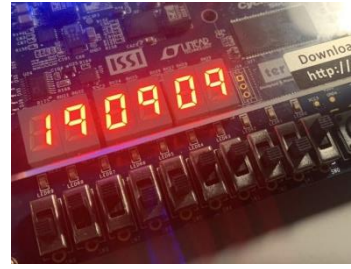
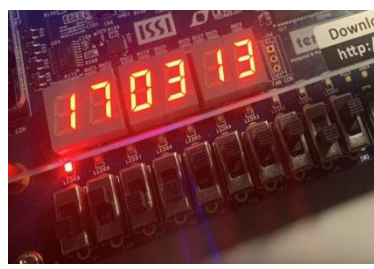
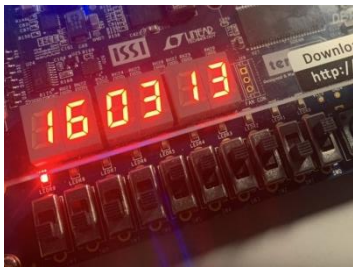
四、3-4

(1) 實驗程式碼:

```
1 module labthree4 (SW, HEX5, HEX4, HEX3, HEX2, HEX1, HEX0, LEDR);
2   input [8:0]SW;
3   output [6:0]HEX5, HEX4, HEX3, HEX2, HEX1, HEX0;
4   wire signa, signb, signc, signd, signe, signif;
5
6   wire [3:0]v, y, z;
7   output [6:0]LEDR;
8   comparator compa(0, SW[7], SW[6], SW[5], SW[4], signa, signb);
9   circuitA a(SW[7:4], signa, signb, v[3:0]);
10  circuitB b(signa, signb, HEX3);
11  char_7seg c(v, HEX2);
12
13
14
15  comparator compb(0, SW[3], SW[2], SW[1], SW[0], signc, signd);
16  circuitA d(SW[3:0], signc, signd, y[3:0]);
17
18  circuitB e(signc, signd, HEX1);
19  char_7seg f(y, HEX0);
20
21  assign LEDR[5] = (signa & signc);
22  wire [3:0]ss;
23  wire [3:0]cc;
24
25
26  full_adder f1(SW[0], SW[4], SW[8], cc[0], ss[0]);
27  full_adder f2(SW[1], SW[5], cc[0], cc[1], ss[1]);
28  full_adder f3(SW[2], SW[6], cc[1], cc[2], ss[2]);
29  full_adder f4(SW[3], SW[7], cc[2], cc[3], ss[3]);
30
31  comparator compc(cc[3], ss[3], ss[2], ss[1], ss[0], signe, signif);
32  circuitA g(ss[3:0], signe, signif, z[3:0]);
33
34  circuitB h(signa, signif, HEX5);
35
36  char_7seg i(z, HEX4);
37 endmodule
38
39 module comparator(top, a, b, c, d, sign, signn);
40   input a, b, c, d, top;
41   output sign, signn;
42   wire s5, s4, s3, s2, s1, s0;
43   assign sign = s5 & s4 & s3 & s2 & s1 & s0;
44   assign signn = top;
45   assign s5 = a < b & c < d;
46   assign s4 = a < b & c < d;
47   assign s3 = a < b & c < d;
48   assign s2 = a < b & c < d;
49   assign s1 = a < b & c < d;
50   assign s0 = a < b & c < d;
51 endmodule
52
53 module circuitA(SS, sign, signn, v);
54   input [3:0]SS;
55   input sign, signn;
56   wire [3:0]x, y;
57   output [3:0]v;
58
59   assign x[0] = (~SS[2] & SS[1] & ~SS[0] & 0) | (~SS[2] & SS[1] & SS[0] & 1) | (SS[2] & ~SS[1] & ~SS[0] & 0) | (SS[2] & ~SS[1] & SS[0] & 1) | (SS[2] & SS[1] & ~SS[0] & 0) | (SS[2] & SS[1] & SS[0] & 1) | (SS[2] & ~SS[1] & ~SS[0] & 0) | (SS[2] & ~SS[1] & SS[0] & 1);
60   assign x[1] = (~SS[2] & SS[1] & ~SS[0] & 0) | (~SS[2] & SS[1] & SS[0] & 1) | (SS[2] & ~SS[1] & ~SS[0] & 0) | (SS[2] & ~SS[1] & SS[0] & 1) | (SS[2] & SS[1] & ~SS[0] & 0) | (SS[2] & SS[1] & SS[0] & 1) | (SS[2] & ~SS[1] & ~SS[0] & 0) | (SS[2] & ~SS[1] & SS[0] & 1);
61   assign y[0] = (~SS[2] & ~SS[1] & ~SS[0] & 0) | (~SS[2] & ~SS[1] & SS[0] & 1) | (~SS[2] & SS[1] & ~SS[0] & 0) | (~SS[2] & SS[1] & SS[0] & 1) | (~SS[2] & ~SS[1] & ~SS[0] & 0) | (~SS[2] & ~SS[1] & SS[0] & 1) | (~SS[2] & SS[1] & ~SS[0] & 0) | (~SS[2] & SS[1] & SS[0] & 1);
62   assign y[1] = (~SS[2] & ~SS[1] & ~SS[0] & 0) | (~SS[2] & ~SS[1] & SS[0] & 1) | (~SS[2] & SS[1] & ~SS[0] & 0) | (~SS[2] & SS[1] & SS[0] & 1) | (~SS[2] & ~SS[1] & ~SS[0] & 0) | (~SS[2] & ~SS[1] & SS[0] & 1) | (~SS[2] & SS[1] & ~SS[0] & 0) | (~SS[2] & SS[1] & SS[0] & 1);
63   assign y[2] = (~SS[2] & ~SS[1] & ~SS[0] & 0) | (~SS[2] & ~SS[1] & SS[0] & 1) | (~SS[2] & SS[1] & ~SS[0] & 0) | (~SS[2] & SS[1] & SS[0] & 1) | (~SS[2] & ~SS[1] & ~SS[0] & 0) | (~SS[2] & ~SS[1] & SS[0] & 1) | (~SS[2] & SS[1] & ~SS[0] & 0) | (~SS[2] & SS[1] & SS[0] & 1);
64   assign y[3] = (~SS[2] & ~SS[1] & ~SS[0] & 0) | (~SS[2] & ~SS[1] & SS[0] & 1) | (~SS[2] & SS[1] & ~SS[0] & 0) | (~SS[2] & SS[1] & SS[0] & 1) | (~SS[2] & ~SS[1] & ~SS[0] & 0) | (~SS[2] & ~SS[1] & SS[0] & 1) | (~SS[2] & SS[1] & ~SS[0] & 0) | (~SS[2] & SS[1] & SS[0] & 1);
65   assign v[0] = (~sign & ~signn & SS[0]) | (sign & ~signn & x[0]) | (~sign & signn & y[0]);
66
67
68   assign v[1] = (~sign & ~signn & SS[1]) | (sign & ~signn & x[1]) | (~sign & signn & y[1]);
69   assign v[2] = (~sign & ~signn & SS[2]) | (sign & ~signn & x[2]) | (~sign & signn & y[2]);
70   assign v[3] = (~sign & ~signn & SS[3]) | (sign & ~signn & x[3]) | (~sign & signn & y[3]);
71
72
73 endmodule
74
75 module circuitB(a, b, hex);
76   input a, b;
77   output [6:0]hex;
78   wire c;
79   assign c = a < b;
80
81   assign hex[0] = (c & 1);
82   assign hex[1] = (c & 0);
83   assign hex[2] = (c & 0);
84   assign hex[3] = (c & 1);
85   assign hex[4] = (c & 1);
86   assign hex[5] = (c & 1);
87   assign hex[6] = (c & 1) | (~c & 1);
88 endmodule
89
90
91 module char_7seg(SW, HEX);
92   input [3:0]SW;
93   output [6:0]HEX;
94
95   assign HEX[0] = (~SW[3] & ~SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & ~SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & SW[1] & SW[0] & 1);
96   assign HEX[1] = (~SW[3] & ~SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & ~SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & SW[1] & SW[0] & 1);
97   assign HEX[2] = (~SW[3] & ~SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & ~SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & SW[1] & SW[0] & 1);
98   assign HEX[3] = (~SW[3] & ~SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & ~SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & SW[1] & SW[0] & 1);
99   assign HEX[4] = (~SW[3] & ~SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & ~SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & SW[1] & SW[0] & 1);
100  assign HEX[5] = (~SW[3] & ~SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & ~SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & SW[1] & SW[0] & 1);
101  assign HEX[6] = (~SW[3] & ~SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & ~SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & ~SW[2] & SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & ~SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & ~SW[1] & SW[0] & 1) | (~SW[3] & SW[2] & SW[1] & ~SW[0] & 0) | (~SW[3] & SW[2] & SW[1] & SW[0] & 1);
102 endmodule
103
104 module full_adder(a, b, cin, cout, sout);
105   input a;
106   input b;
107   input cin;
108   output cout;
109   output sout;
110   wire C1, C2, S1;
111
112   process(a, b, C1, S1);
113   process(S1, cin, C2, sout);
114   assign cout = C1 & C2;
115
116 endmodule
117
118 module process(x, y, C, S);
119   input x, y;
120   output C, S;
121
122   assign C = (x & y);
123   assign S = (~x & y) | (x & ~y);
124
125 endmodule
```

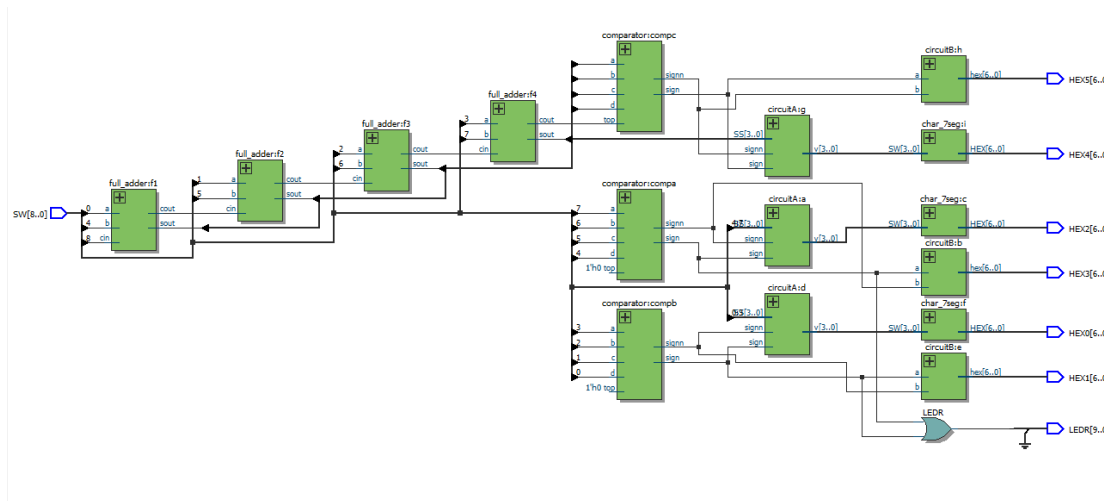
//95-101 同 LAB 3-1，故部分省略

(2)實驗結果:



//由左至右，SW[8:0]:000111101/100111101/110011001/010011001

(3)RTL 布局:



(4)問題與討論:

真的非常複雜，加上我的寫法用了許多 `module` 和參數，導致 `debug` 上花很多時間，學到了最重要的就是要把 `module` 的參數名稱好好定義，否則超容易混淆，另外也算是更熟悉 `call` 不同 `module` 的語法和細節了。