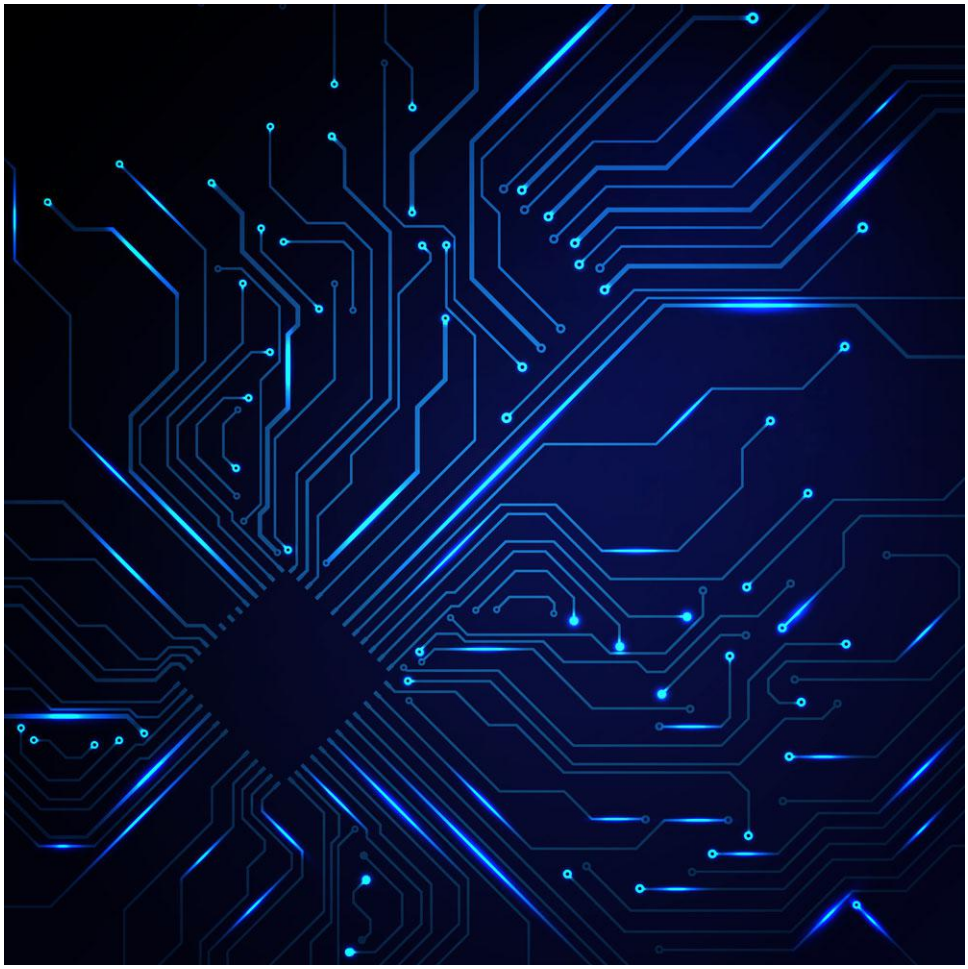


2022/09/22

實驗二

組合邏輯練習

姓名：王嘉羽　　學號：00957116

班級：資工 2B

E-mail：vayne20011125@gmail.com

注意

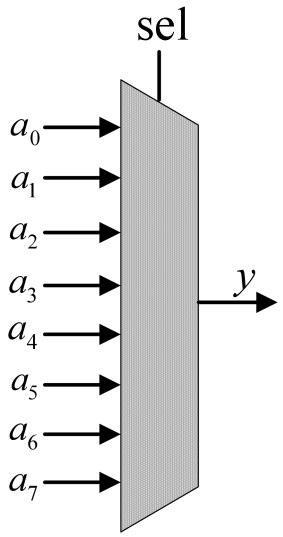
1. 繳交時一律轉 PDF 檔
2. 繳交期限為
   上完課後
   當週五晚上 12 點前
3. 一人繳交一份
4. 檔名：學號_HW?.pdf
   檔名請按照作業檔名格式進行填寫
   未依照格式不予批改

一、 多工器

● 實驗說明：

1. 實作多工器及 testbench。
2. 輸入資料 a_n [2:0]：a_0=0 、a_1=1、a_2=2 、a_3=3 、a_4=4 、a_5=5 、a_6=6 、a_7=7
3. 輸出： y [2:0]
4. Testbench 內容為 sel 由 0 到 7

● 系統硬體架構方塊圖（接線圖）：



● 系統架構程式碼、測試資料程式碼與程式碼說明

截圖請善用 win+shift+S

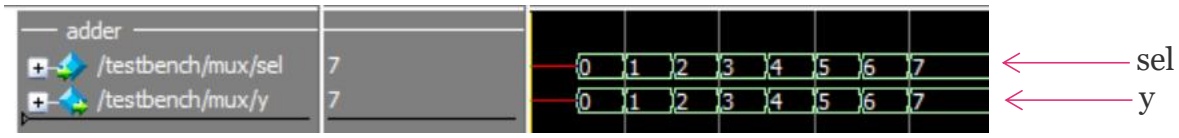
系統結構程式碼:

```
module mux_8to1(
    //變數宣告
    input [2:0] a0,
    input [2:0] a1,
    input [2:0] a2,
    input [2:0] a3,
    input [2:0] a4,
    input [2:0] a5,
    input [2:0] a6,
    input [2:0] a7,
    input [2:0] sel,      // sel為選擇線，選擇y的值(a0,a1....)
    output logic[2:0] y   // 要加logic
);
 always_comb
begin
    case(sel)
        3'b000: y = a0; //當sel = 0時，y = a0，以此類推
        3'b001: y = a1;
        3'b010: y = a2;
        3'b011: y = a3;
        3'b100: y = a4;
        3'b101: y = a5;
        3'b110: y = a6;
        3'b111: y = a7;
        default: y = 3'bx; //如果sel都不符合，y = 3'bx
    endcase
end
endmodule
```

測試資料程式碼:

```
module testbench;

    logic [2:0] a0, a1, a2, a3, a4, a5, a6, a7;
    logic [2:0] y;
    logic [2:0] sel;

    mux_8to1 mux(
        .a0(a0), //()內的變數為tb的變數，"."後面為mux_8to1.sv的變數，將2者對應起來
        .a1(a1),
        .a2(a2),
        .a3(a3),
        .a4(a4),
        .a5(a5),
        .a6(a6),
        .a7(a7),
        .sel(sel),
        .y(y)
    );

    initial begin
        a0 = 0; a1 = 1; a2 = 2; a3 = 3; a4 = 4; a5 = 5; a6 = 6; a7 = 7; //指定input的值
        #10 sel = 0; //延遲10ms，所以會從10後開始顯示結果
        #10 sel = 1; //每10ms改變一次sel，當sel改變y也會改變
        #10 sel = 2;
        #10 sel = 3;
        #10 sel = 4;
        #10 sel = 5;
        #10 sel = 6;
        #10 sel = 7;
        #1000 $stop;
    end
endmodule
```

● 模擬結果與結果說明：



根據題目規定:

a0 = 0, a1 = 1, a2 = 2.....

根據前面方塊圖知道:

這是一個 8to1 的多工器

當 sel = 0 時 y = a0 = 0;

當 sel = 1 時 y = a1 = 1;

當 sel = 2 時 y = a2= 2;

當 sel = 3 時 y = a3 = 3;

當 sel = 4 時 y = a4 = 4;

當 sel = 5 時 y = a5 = 5;

當 sel = 6 時 y = a6 = 6;

當 sel = 7 時 y = a7 = 7;

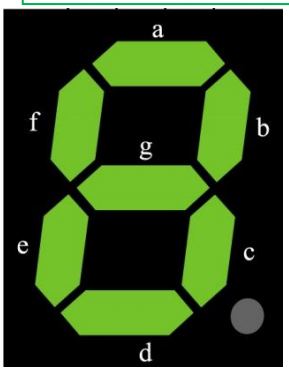
與結果圖吻合!

二、 七段顯示器 Decoder

● 實驗說明：

1. 實作共陽極七段顯示器的 Decoder 及 testbench，跑模擬波形。
2. 輸入資料 A[3:0] (二進制)：0000, 0001, 0010…依此類推

Input					Output						
a_3	a_2	a_1	a_0		g	f	e	d	c	b	a
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	0	0	1
0	0	1	0	2	0	1	0	0	1	0	0
0	0	1	1	3	0	1	1	0	0	0	0
0	1	0	0	4	0	0	1	1	0	0	1
0	1	0	1	5	0	0	1	0	0	1	0
0	1	1	0	6	0	0	0	0	0	1	0
0	1	1	1	7	1	1	1	1	0	0	0
1	0	0	0	8	0	0	0	0	0	0	0
1	0	0	1	9	0	0	1	0	0	0	0
1	0	1	0	A	0	0	0	1	0	0	0
1	0	1	1	b	0	0	0	0	0	1	1
1	1	0	0	C	1	0	0	0	1	1	0
1	1	0	1	d	0	1	0	0	0	0	1
1	1	1	0	E	0	0	0	0	1	1	0
1	1	1	1	F	0	0	0	1	1	1	0



● 系統架構程式碼、測試資料程式碼與程式碼說明

截圖請善用 win+shift+S

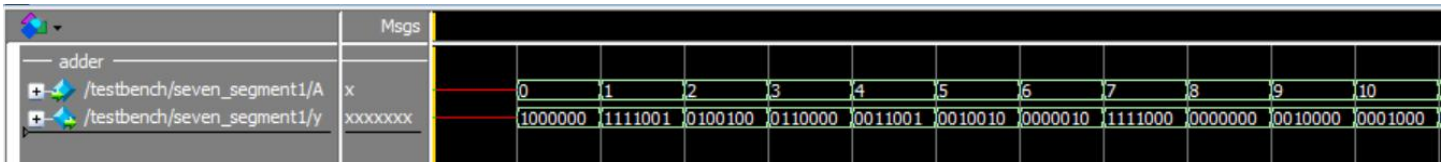
系統架構程式碼:

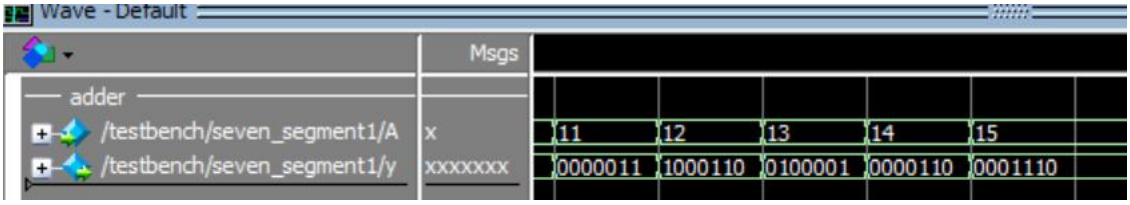
```
module seven_segment(
    input [3:0] A,          //輸入
    output logic [6:0] y   //輸出
);
 always_comb
begin
    //當A改變時，y會對應到七段顯示器，右邊的註解表示他在7段顯示器上的效果
    case(A)
        4'h0 : y = 7'b1000000; //0
        4'h1 : y = 7'b1111001; //1
        4'h2 : y = 7'b0100100; //2
        4'h3 : y = 7'b0110000; //3
        4'h4 : y = 7'b0011001; //4
        4'h5 : y = 7'b0010010; //5
        4'h6 : y = 7'b0000010; //6
        4'h7 : y = 7'b1111000; //7
        4'h8 : y = 7'b0000000; //8
        4'h9 : y = 7'b0010000; //9
        4'hA : y = 7'b0001000; //A
        4'hB : y = 7'b0000011; //b
        4'hC : y = 7'b1000110; //C
        4'hD : y = 7'b0100001; //d
        4'hE : y = 7'b0000110; //E
        4'hF : y = 7'b0001110; //F
    endcase
end
endmodule
```

測試資料程式碼:

```
module testbench;

    logic [3:0] A;
    logic [6:0] y;

    seven_segment seven_segment1(
        .A(A), //()內的變數為tb的變數，"."後面為seven_segment.sv的變數，將2者對應起來
        .y(y)
    );

    initial begin
        #10 A = 0; //延遲10ms，所以會從10後開始顯示結果
        #10 A = 1; //每10ms改變一次A，當A改變y也會改變
        #10 A = 2;
        #10 A = 3;
        #10 A = 4;
        #10 A = 5;
        #10 A = 6;
        #10 A = 7;
        #10 A = 8;
        #10 A = 9;
        #10 A = 10;
        #10 A = 11;
        #10 A = 12;
        #10 A = 13;
        #10 A = 14;
        #10 A = 15;
        #1000 $stop;
    end
endmodule
```

● 模擬結果與結果說明：





當 A = 0 時，y = 1000000，對應到七段顯示器上為數字 0；
當 A = 1 時，y = 1111001，對應到七段顯示器上為數字 1；
當 A = 2 時，y = 0100100，對應到七段顯示器上為數字 2；
當 A = 3 時，y = 0110000，對應到七段顯示器上為數字 3；
當 A = 4 時，y = 0011001，對應到七段顯示器上為數字 4；
當 A = 5 時，y = 0010010，對應到七段顯示器上為數字 5；
當 A = 6 時，y = 0000010，對應到七段顯示器上為數字 6；
當 A = 7 時，y = 1111000，對應到七段顯示器上為數字 7；
當 A = 8 時，y = 0000000，對應到七段顯示器上為數字 8；
當 A = 9 時，y = 0010000，對應到七段顯示器上為數字 9；
當 A = 10 時，y = 0001000，對應到七段顯示器上為數字 A；
當 A = 11 時，y = 0000011，對應到七段顯示器上為數字 b；
當 A = 12 時，y = 1000110，對應到七段顯示器上為數字 C；
當 A = 13 時，y = 0100001，對應到七段顯示器上為數字 d；
當 A = 14 時，y = 0000110，對應到七段顯示器上為數字 E；
當 A = 15 時，y = 0001110，對應到七段顯示器上為數字 F；

與模擬結果吻合。

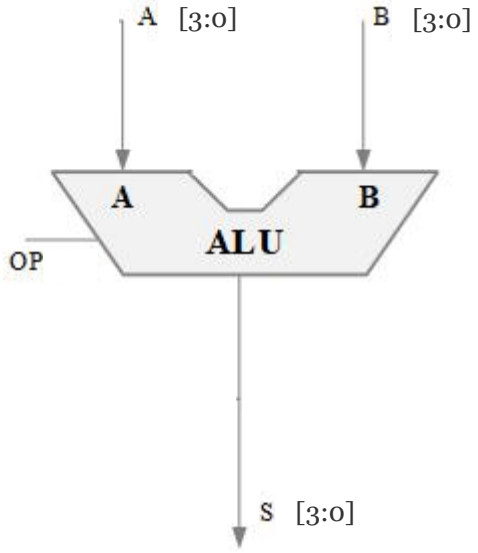
三、 算術邏輯單元(ALU)

● 實驗說明：

1. 實作 ALU 及 testbench，跑模擬波形。
2. S = 5 + B
3. S = D - 7
4. A[3:0]、B[3:0]、op 分別輸入資料(16 進位)：5, B, 0。
5. A[3:0]、B[3:0]、op 分別輸入資料(16 進位)：D, 7, 1。
6. 輸入：A[3:0], B[3:0], op
7. 輸出：S[3:0]

OP	ALU 運算	註解
0	S = A + B	加法
1	S = A - B	減法

● 系統硬體架構方塊圖（接線圖）：



● 系統架構程式碼、測試資料程式碼與程式碼說明

截圖請善用 win+shift+S

系統架構程式碼:

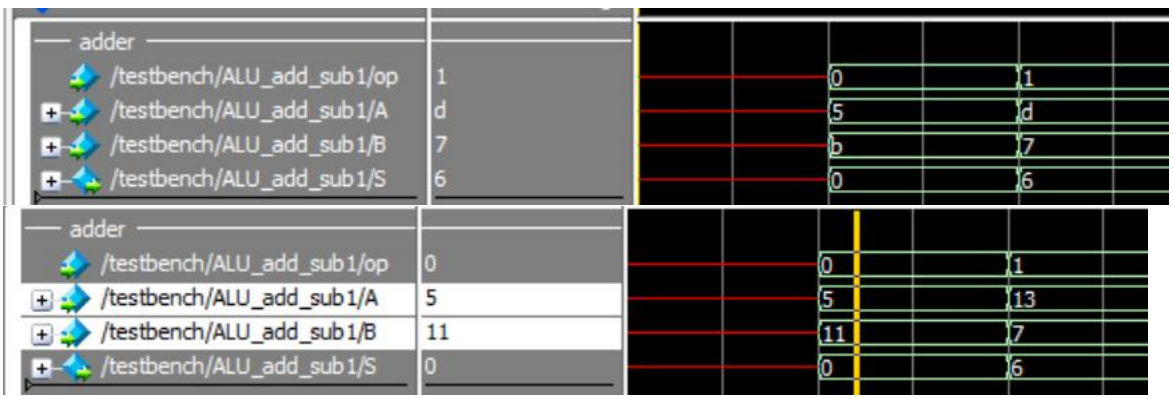
```
1  module ALU_add_sub(
2      input [3:0] A,        //輸入
3      input [3:0] B,
4      input op,             //控制+-
5      output logic [3:0] S //輸出
6  );
7
8   always_comb
9  begin
10     if(op == 0)     //op = 0，為加法
11        S = A + B;   //op = 1，為剪法
12     else
13        S = A - B;
14 end
15  endmodule
```

測試資料程式碼:

simulation > tb > testbench.sv

```
1   module testbench;
2
3       logic [3:0] A;
4       logic [3:0] B;
5       logic [3:0] S;
6       logic op;
7
8       ALU_add_sub ALU_add_sub1(
9           .A(A), //()內的變數為tb的變數，"."後面為ALU_add_sub.sv的變數，將2者對應起來
10          .B(B),
11          .S(S),
12          .op(op)
13      );
14
15      initial begin
16          #10 A = 4'h5; B = 4'hB; op = 0; //延遲10ms，所以會從10後開始顯示結果
17          #10 A = 4'hD; B = 4'h7; op = 1; //每10ms改變一次A，B，op，當他們改變S也會改變
18          #1000 $stop;
19      end
20  endmodule
```

● 模擬結果與結果說明：



上面那張為 16 進位的，為了方便驗證結果我也放了 unsigned 的(下面)。

當 op = 0 時，S = A + B => 16(0) = 5 + 11 (4bit 最大到 15 故 16 會 overflow 成 0)

當 op = 1 時，S = A - B => 6 = 13 - 7， 都與結果吻合!

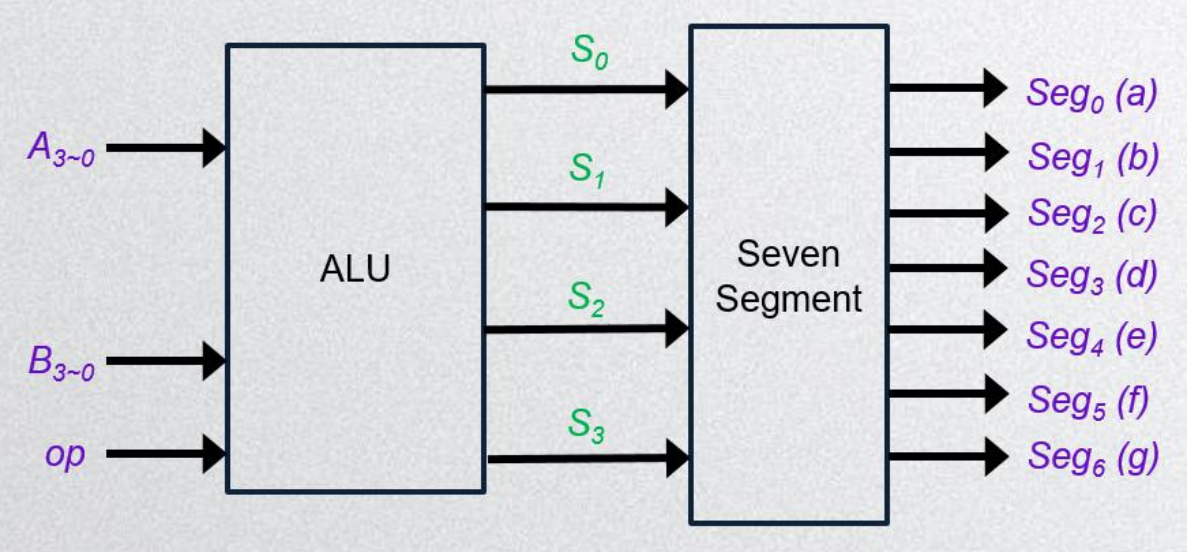
四、 ALU+七段顯示器

● 實驗說明：

1. 將 ALU 接上七段顯示器
2. A[3:0]、B[3:0]、op 分別輸入資料(16 進位)：5, B, 0。
3. A[3:0]、B[3:0]、op 分別輸入資料(16 進位)：C, 2, 0。
4. A[3:0]、B[3:0]、op 分別輸入資料(16 進位)：9, 4, 1。
5. A[3:0]、B[3:0]、op 分別輸入資料(16 進位)：3, B, 1。
6. A[3:0]、B[3:0]、op 分別輸入資料(16 進位)：D, 7, 1。
7. 輸入：A[3:0], B[3:0], op
8. 輸出：7_seg[6:0]

OP	ALU 運算	註解
0	S = A + B	加法
1	S = A - B	減法

● 系統硬體架構方塊圖（接線圖）：



● 系統架構程式碼、測試資料程式碼與程式碼說明

截圖請善用 win+shift+S

系統架構程式碼:

```
module ALU_seven_segment(
    input [3:0] A,          //輸入
    input [3:0] B,
    input op,               //控制+-
    output logic [6:0] seven_seg //輸出
);

logic [3:0] S; //宣告變數S
//直接呼叫第2.3小題做的元件
ALU_add_sub alu_add_sub1(A,B,op,S);          //如果op=0，將A,B相加，反則相減 。將結果放入S
seven_segment seven_segment1(S,seven_seg);   //將S以七段顯示器的顯示方式存入seven_seg

endmodule
```

測試資料程式碼:

simulation > tb > testbench.sv

```
module testbench;

    logic [3:0] A;
    logic [3:0] B;
    logic [3:0] seven_seg;
    logic op;

    ALU_seven_segment ALU_seven_segment1(
        .A(A), //()內的變數為tb的變數，"."後面為ALU_add_sub.sv的變數，將2者對應起來
        .B(B),
        .seven_seg(seven_seg),
        .op(op)
    );

    initial begin
        #10 A = 4'h5; B = 4'hB; op = 0; //延遲10ms，所以會從10後開始顯示結果
        #10 A = 4'hC; B = 4'h2; op = 0; //每10ms改變一次A，B，op，當他們改變seven_seg也會改變
        #10 A = 4'h9; B = 4'h4; op = 1;
        #10 A = 4'h3; B = 4'hB; op = 1;
        #10 A = 4'hD; B = 4'h7; op = 1;
        #1000 $stop;
    end
endmodule
```

● 模擬結果與結果說明：

這次是結合 2.3 小題，將 A,B 做運算後，用七段顯示器顯示。

5+B = 16(0) -> 100000

C+2 = E -> 0000110

9 - 4 = 5 -> 0010010

3 - B = 3 - 11 = -8(+16) = 8 -> 0000000

D - 7 = 13 - 7 = 6 -> 0000010　與結果相吻合!

● 結論與心得：

這是我第二次上硬體相關課程，第一次是上 VHDL，然而我發現我都忘光光了...或是其實這兩種語言很不一樣，當初想選這堂課就是因為對硬體蠻有興趣的。結果這次作業遇到大大大問題，一開始在測試環境時，他竟然報錯了: “找不到 path” ，一查發現不能中文路徑....。但我的使用者名稱是中文...，經過一番折騰後就解決了。

之後開始寫第一題時，發生各式各樣的問題....好不習慣這個語言，因為實在是一直不知道為甚麼錯，就去問好朋友，他就好好的解釋 vs 和 test bench 的寫法以及是如何對應的(他上過 veilog)，弄懂後，就比較上手了。硬體語言真的蠻好玩的，很期待燒錄到機器裡!