12/5

1. 實驗目的

了解 Decoder、 Multiplexer之操作原理，使用 Verilog 語言設計電路，並應用 CAD 軟體來輔助驗證所設計的電路。

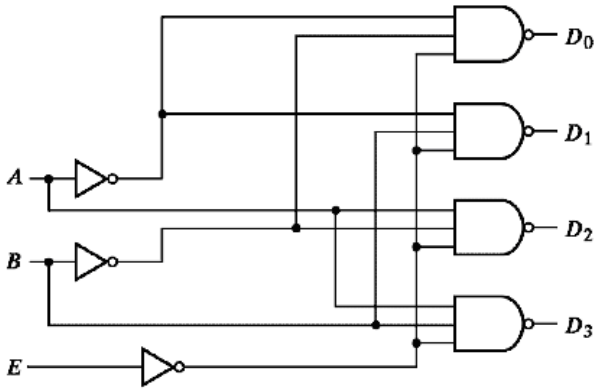
2. 實驗器材 Quartus Ⅱ(CAD tools)、DE2

3. 實驗內容請使用 Verilog HDL 描寫出：

(1)具有 Enable 功能之 3 對 8 解碼器。 (2) 8 對 1 的多工器。並用波型圖驗證後，再利用 DE2 實驗板上的 FPGA晶片 CycloneII 系列 EP2C35F672C6，將其指撥開關作為輸入，並將結果輸出於七段顯示器上加以驗證結果。

4. 實驗原理

(a) 解碼器 (Decoder) 解碼是將某一數碼轉換成其所對應值至單一輸出端的過程，將 n 位元二進數字轉變成 m 個 ($m \leqq 2^n$) 單一輸出的組合邏輯電路。任一輸入組合僅能對應 m 條輸出線中的一條，使此對應之輸出線成為高電位 (低電位)，其他 m-1 條輸出線成為低電位 (高電位)。下圖6-1 為 2 對 4 解碼器之電路圖， A、B 為輸入，E 為 enable，D_0、D_1、D_2、D_3 為輸出。



Truth table

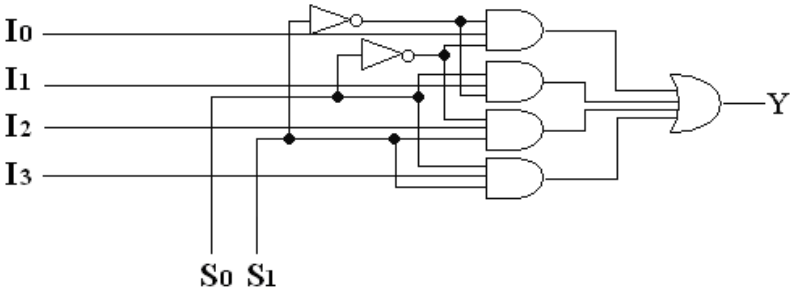
E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

圖6-1

Source: Mano, “Digital Design,” Prentice Hall.

(b) 多工器（Multiplexer）

多工器又稱為資料選擇器(Data Selector)，它可以接數個資料輸入，而選擇其中一個輸入資料送至單一輸出端，至於要將那一個輸入資料送至輸出端，是由選擇線來決定。圖6-3 為一4 對1 多工器，四條輸入線 (I_0，I_1，I_2，I_3) 及兩條選擇線 (S_0，S_1)，而其輸出為Y。



Function table

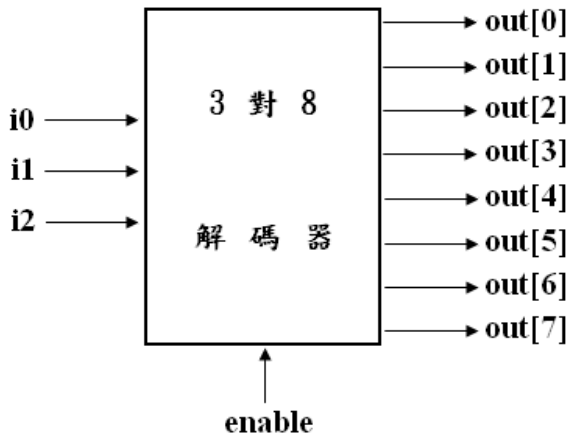
s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

圖6-2

Source: Mano, “Digital Design,” Prentice Hall.

5. 實驗範例與結果

(1) 具有 Enable 功能之 3 對 8 解碼器



Truth Table

Input			Output	
i0	i1	i2	enable	out[0:7]
X	X	X	0	00000000
0	0	0	1	00000001
0	0	1	1	00000010
0	1	0	1	00000100
0	1	1	1	00001000
1	0	0	1	00010000
1	0	1	1	00100000
1	1	0	1	01000000
1	1	1	1	10000000

程式碼

```
//to de2
module test(seg7_out1,seg7_out2, i0, i1, i2, enable);
input      i0, i1, i2, enable;
output [6:0]seg7_out1,seg7_out2;
wire      [7:0]out;
DEC_3X8    test1(i0, i1, i2, enable,out);
binary_to_7seg    test2(out[3:0],seg7_out1);
binary_to_7seg    test3(out[7:4],seg7_out2);
endmodule

//Description of Decoder_3X8
module DEC_3X8(i0, i1, i2, enable, out);
input i0, i1, i2, enable;
output [7:0]out;
reg [7:0]out;
always @(enable or i0 or i1 or i2)
begin
if (enable == 1'b1)
begin
case ({i0, i1, i2})
3'b000: out = 8'b00000001;
3'b001: out = 8'b00000010;
3'b010: out = 8'b00000100;
```

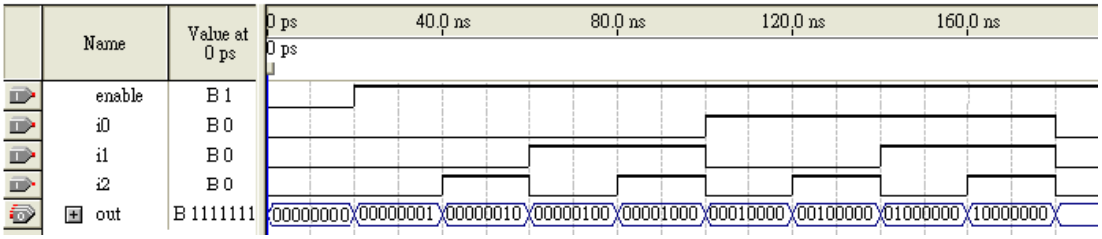
```
3'b011: out = 8'b00001000;
3'b100: out = 8'b00010000;
3'b101: out = 8'b00100000;
3'b110: out = 8'b01000000;
3'b111: out = 8'b10000000;
endcase
end
else
out = 8'b00000000;
end
endmodule

//DE2_7seg
module binary_to_7seg(binary_in,seg7_out);
        input    [3:0] binary_in;
        output [6:0] seg7_out;
        reg        [6:0] seg7_out;
        always @ (binary_in)
          begin
            case(binary_in)
            4'h0: seg7_out = 7'b1000000;
            4'h1: seg7_out = 7'b1111001;
            4'h2: seg7_out = 7'b0100100;
            4'h3: seg7_out = 7'b0110000;
            4'h4: seg7_out = 7'b0011001;
            4'h5: seg7_out = 7'b0010010;
            4'h6: seg7_out = 7'b0000010;
            4'h7: seg7_out = 7'b1111000;
            4'h8: seg7_out = 7'b0000000;
            4'h9: seg7_out = 7'b0011000;
            4'hA: seg7_out = 7'b0001000;
            4'hB: seg7_out = 7'b0000011;
            4'hC: seg7_out = 7'b1000110;
            4'hD: seg7_out = 7'b0100001;
            4'hE: seg7_out = 7'b0000110;
            4'hF: seg7_out = 7'b0001110;
            default: seg7_out = 7'b1111111;
            endcase
```

```
        end
endmodule
```

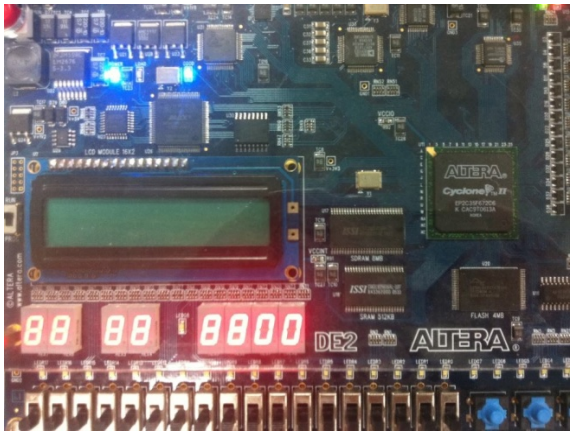
模擬結果

首先當enable 為0 時，out 值為8'b00000000，當enable 為1 時，判斷所輸入i0、i1、i2 之值，下圖之out 如Truth Table 所示相同。



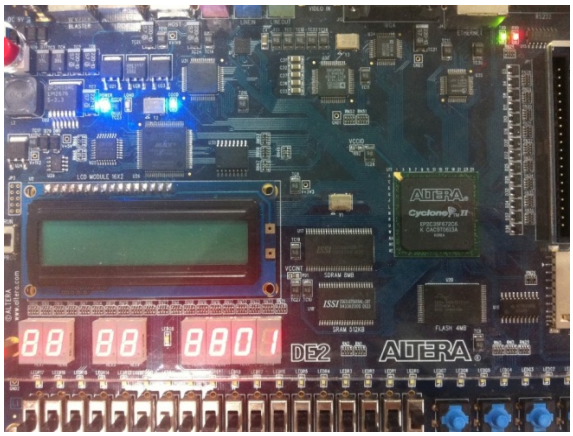
參考上面真值表

4個開關依序為i2, i1, i0, enable



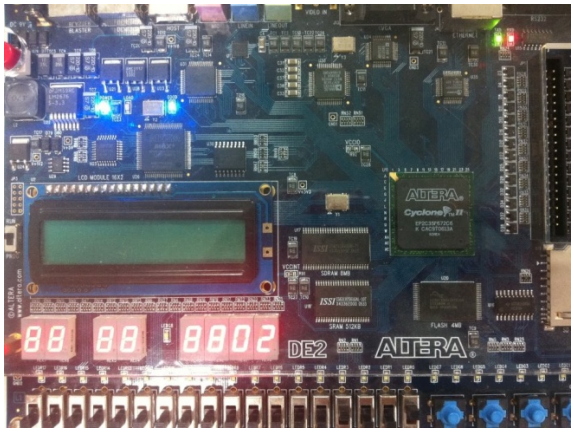
開關撥0000或1110時→00000000

顯示00



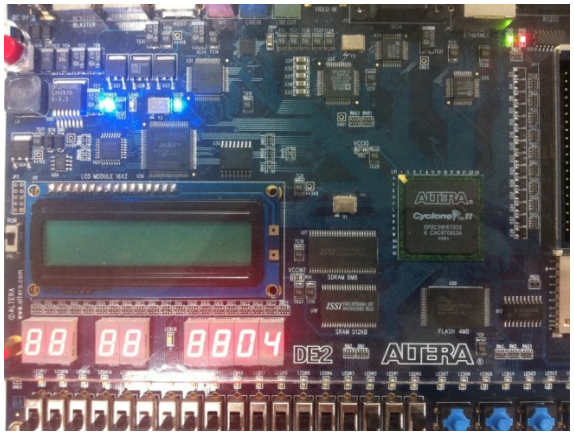
開關撥0001時→00000001

顯示01



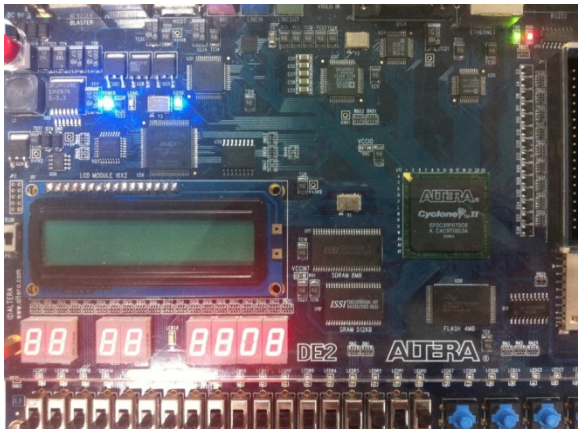
開關撥1001時→00000010

顯示02



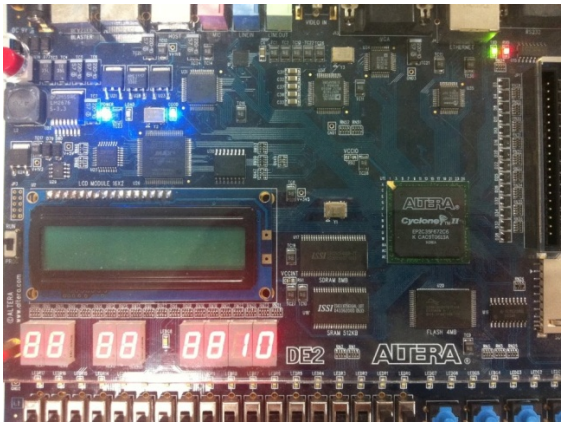
開關撥0101時→00000100

顯示04



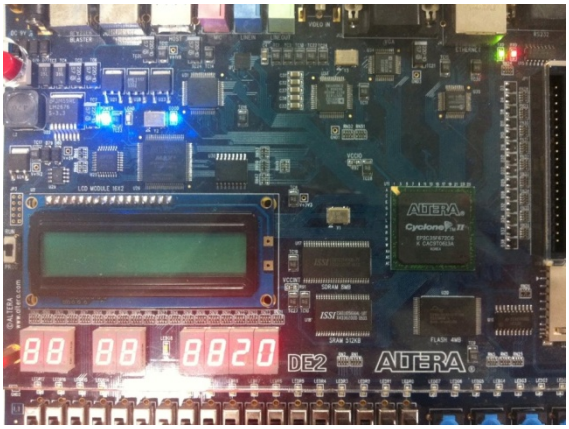
開關撥1101時→00001000

顯示08



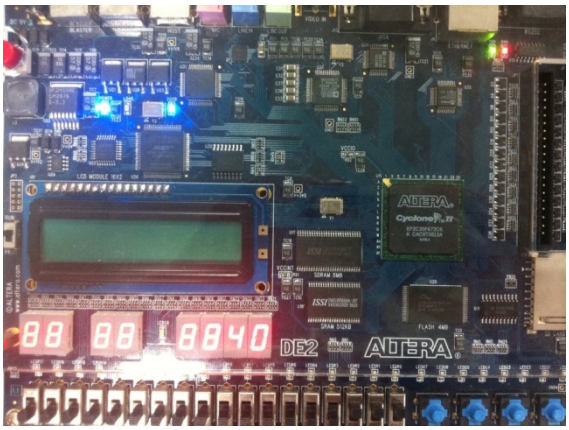
開關撥0011時→00010000

顯示10



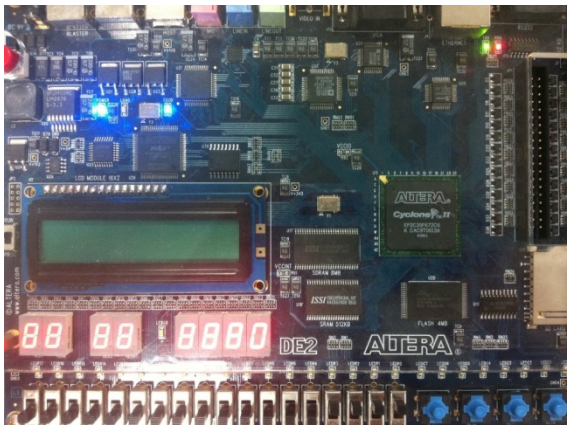
開關撥1011時→00100000

顯示20



開關撥0111時→01000000

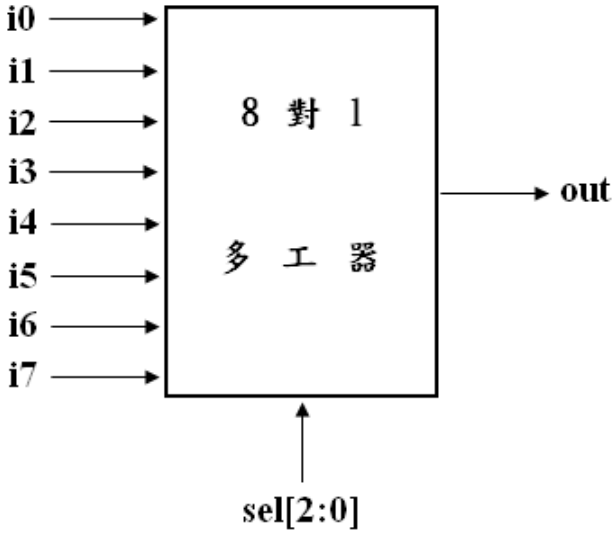
顯示40



開關撥1111時→10000000

顯示80

(2) 8對1多工器



Truth Table

Input			output
sel[2]	sel[1]	sel[0]	out
0	0	0	i0
0	0	1	i1
0	1	0	i2
0	1	1	i3
1	0	0	i4
1	0	1	i5
1	1	0	i6
1	1	1	i7

程式碼

```
//to de2
module mm(seg7_out,sel);
input    [2:0] sel;
output [6:0] seg7_out;
wire      [3:0] out;
MUX_8X1 mm1(sel,out);
binary_to_7seg    test2(out,seg7_out);
endmodule

// Description of    MUX_8X1
module MUX_8X1(sel, out);
```

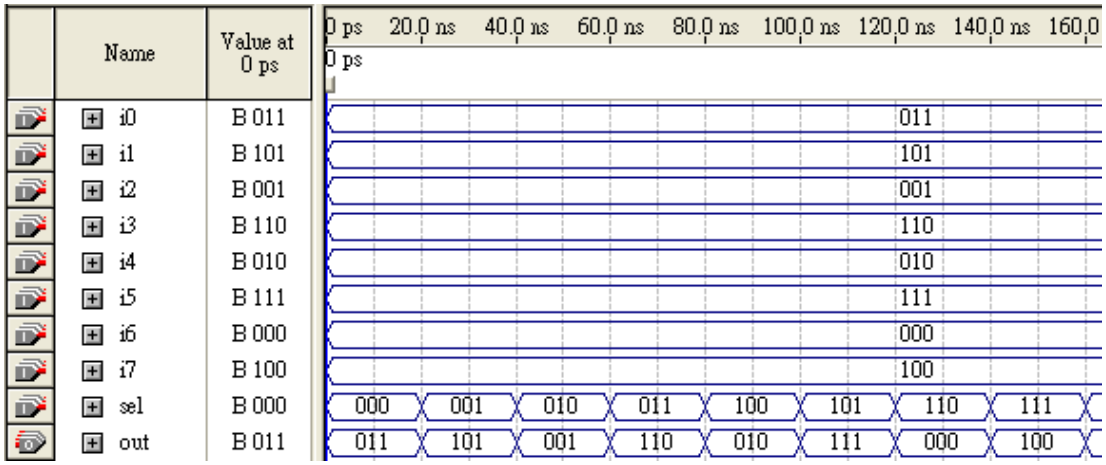
```
input [2:0]sel;
output [3:0]out;
reg [3:0]out;
always @(sel)
case(sel)
3'b000: out = 4'h0;
3'b001: out = 4'h1;
3'b010: out = 4'h2;
3'b011: out = 4'h3;
3'b100: out = 4'h4;
3'b101: out = 4'h5;
3'b110: out = 4'h6;
3'b111: out = 4'h7;
endcase
endmodule

//DE2_7seg
module binary_to_7seg(binary_in,seg7_out);
        input    [3:0] binary_in;
        output [6:0] seg7_out;
        reg        [6:0] seg7_out;
        always @ (binary_in)
          begin
            case(binary_in)
            4'h0: seg7_out = 7'b1000000;
            4'h1: seg7_out = 7'b1111001;
            4'h2: seg7_out = 7'b0100100;
            4'h3: seg7_out = 7'b0110000;
            4'h4: seg7_out = 7'b0011001;
            4'h5: seg7_out = 7'b0010010;
            4'h6: seg7_out = 7'b0000010;
            4'h7: seg7_out = 7'b1111000;
            4'h8: seg7_out = 7'b0000000;
            4'h9: seg7_out = 7'b0011000;
            4'hA: seg7_out = 7'b0001000;
            4'hB: seg7_out = 7'b0000011;
            4'hC: seg7_out = 7'b1000110;
            4'hD: seg7_out = 7'b0100001;
```

```
            4'hE: seg7_out = 7'b0000110;
            4'hF: seg7_out = 7'b0001110;
            default: seg7_out = 7'b1111111;
            endcase
          end
endmodule
```

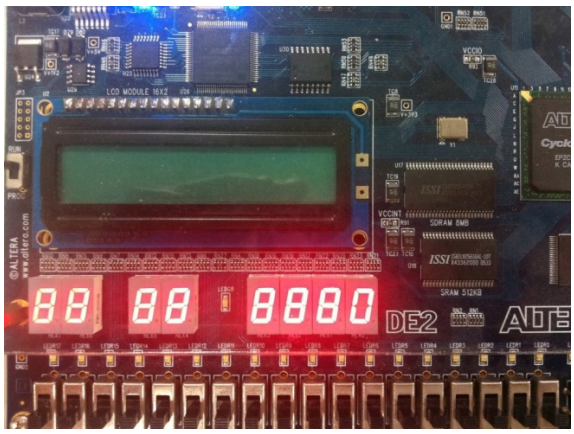
模擬結果

如下圖所示，當sel 分別為000、001、010、.....111 時，out 如Truth Table 所表示分別為i0 值為011、i1 值為101、i2 值為010、……i7 值為100。

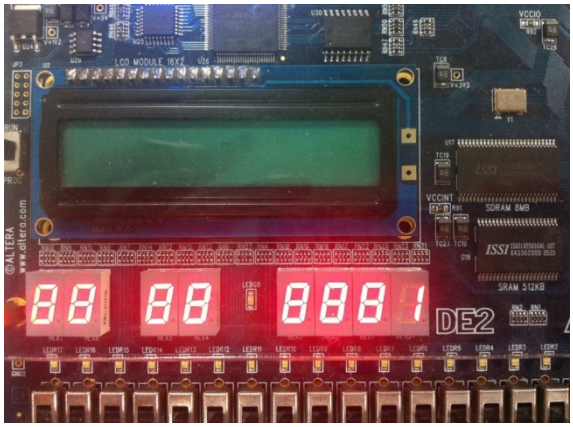


參考上面真值表

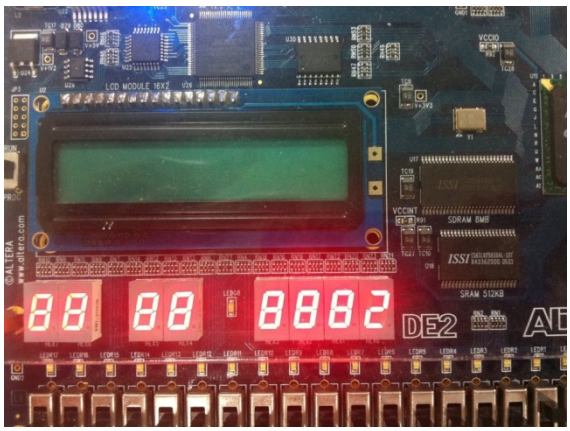
3個開關依序為sel[2], sel[1], sel[0]



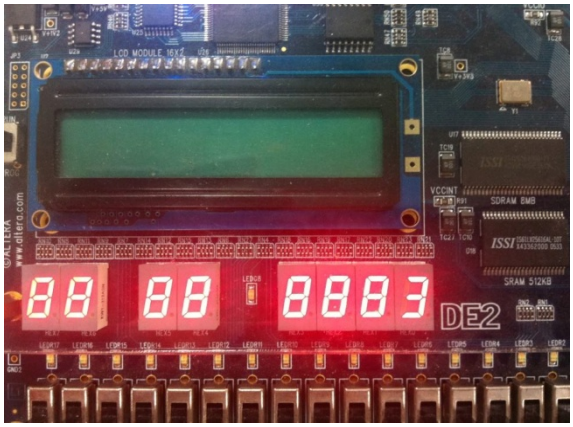
開關撥000→顯示0



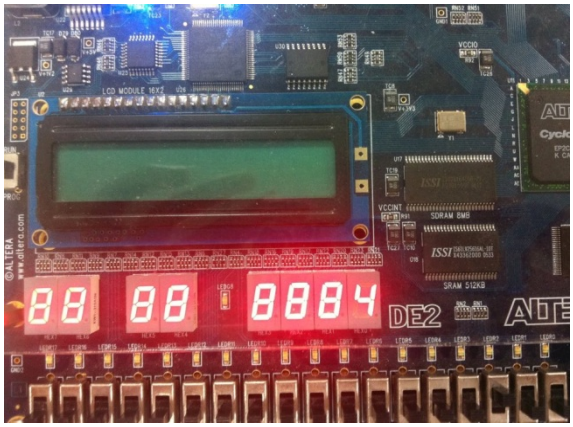
開關撥001→顯示1



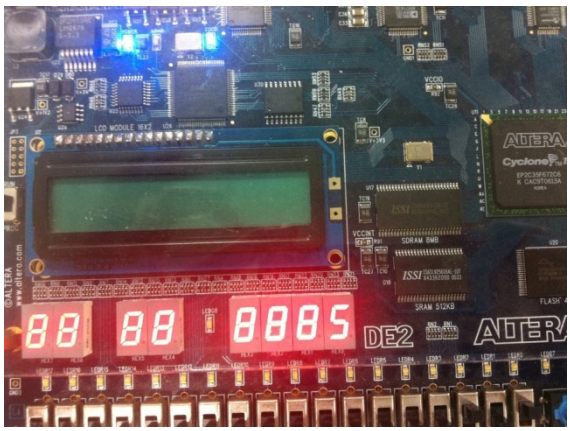
開關撥010→顯示2



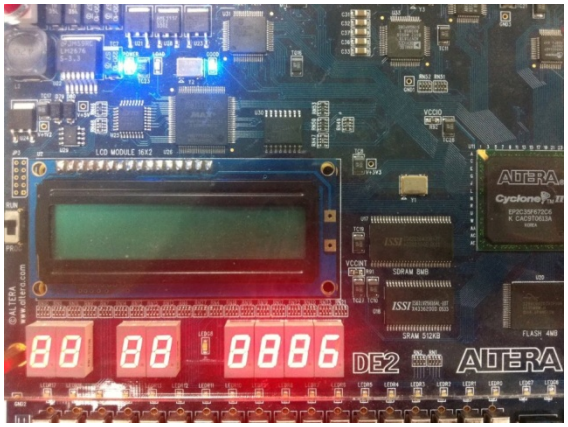
開關撥011→顯示3



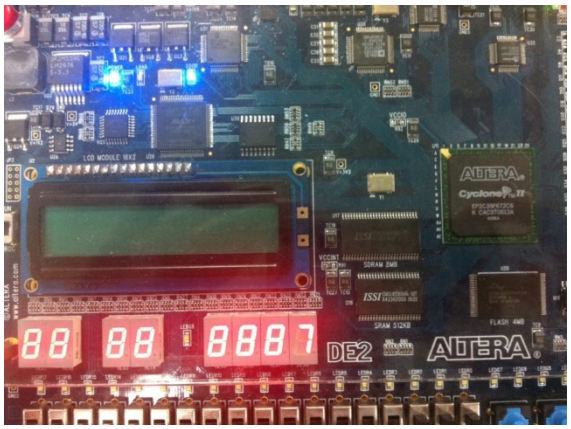
開關撥100→顯示4



開關撥101→顯示5



開關撥110→顯示6



開關撥111→顯示7

6. 問題與討論

(1) 在3x8 decoder 範例中， (a)若在 always 敘述中的sensitive list 漏掉i1，電路有何不同？(b)另外若漏寫else out = 8‘b00000000 ，電路有何不同？

(2) 設每個輸入是4位元，請利用五個 4x1 多工器完成一個16x1 的多工器 (只要有模擬結果即可，建議這16個輸入恰好給0000~1111 (0h~Fh)，這樣模擬可清楚看出電路是否正確)。