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 03

實驗名稱 : Multiplexer

結報完成日期 : 2022.10.22

姓名 : 王語
系級 : 機械四
學號 : 0811127

## 一、 實驗目的

本次實驗藉由 LPM 的調用完成乘法以及加法的運算，並且練習負緣觸發的電路設計，也導入 pipeling 技巧，計算出延遲對電路的影響，並使電路能在特定頻率的 CLK 下正常運作。

## 二、 Verilog 程式碼

LAB03 > Lab03_0811127.v

```
// 0811127 王語
// LAB03

//import files
//`include "lpm_mult_my.v"
//`include "lpm_add16.v"

//top module
module Lab03_0811127 (  input [3:0] KEY, // press >> 0
                        input [9:0] SW,
                        output [9:9] LEDR,
                        output [6:0] HEX0 , HEX1 , HEX2 , HEX3 );
reg [7:0] A , B , C , D ;
reg [15:0] sum ;
wire [15:0] _sum ;
reg overflow ;
wire _overflow ;
reg [3:0] _hex0 , _hex1 , _hex2 , _hex3 ;
// wire [15:0] sum_ab , sum_cd ;
wire [15:0] A_mult_B , C_mult_D ;

// algorithm
lpm_mult_my AmultB (  .dataa(A),
                      .datab(B),
                      .result(A_mult_B) );

lpm_mult_my CmultD (  .dataa(C),
                      .datab(D),
                      .result(C_mult_D) );

lpm_add16_my add16 (   .dataa(A_mult_B),
                    .datab(C_mult_D),
                    .overflow(_overflow),
                    .result(_sum));

// display part
assign LEDR[9] = _overflow ;

always @(*) begin
    if (KEY[3]) begin           //ABCD
        if (SW[8]) begin        //CD
            _hex0 = D[3:0] ;
            _hex1 = D[7:4] ;
            _hex2 = C[3:0] ;
            _hex3 = C[7:4] ;
        end
        else begin              //AB
            _hex0 = B[3:0] ;
            _hex1 = B[7:4] ;
            _hex2 = A[3:0] ;
            _hex3 = A[7:4] ;
        end
    end
    else begin                  //S
            _hex0 = _sum[3:0] ;
            _hex1 = _sum[7:4] ;
            _hex2 = _sum[11:8] ;
```

```
    else begin                  //S
            _hex0 = _sum[3:0] ;
            _hex1 = _sum[7:4] ;
            _hex2 = _sum[11:8] ;
            _hex3 = _sum[15:12] ;
    end
end

ssd ssd_0(.Din(_hex0),.Dout(HEX0));
ssd ssd_1(.Din(_hex1),.Dout(HEX1));
ssd ssd_2(.Din(_hex2),.Dout(HEX2));
ssd ssd_3(.Din(_hex3),.Dout(HEX3));

//set part and algorithm
always @(negedge KEY[1] or negedge KEY[0]) begin
    if (!KEY[0]) begin:reset
        A <= 8'b0000_0000 ;
        B <= 8'b0000_0000 ;
        C <= 8'b0000_0000 ;
        D <= 8'b0000_0000 ;
        sum <= 16'b0000_0000_0000_0000 ;
        overflow <= 1'b0 ;
    end
    else begin:set_and_algorithm
        //algorithm part
        // sum <= _sum ;
        // overflow <= _overflow ;
        //set part
        if (SW[9]) begin:write_enable
            if (!SW [8]) begin       // AB
                if (KEY[2]) begin   //A
                    A <= SW[7:0];
                end
                else begin          //B
                    B <= SW[7:0];
                end
            end
            else begin              //CD
                if (KEY[2]) begin   //C
                    C <= SW[7:0];
                end
                else begin          //D
                    D <= SW[7:0];
                end
            end
        end
    end
end

endmodule
```

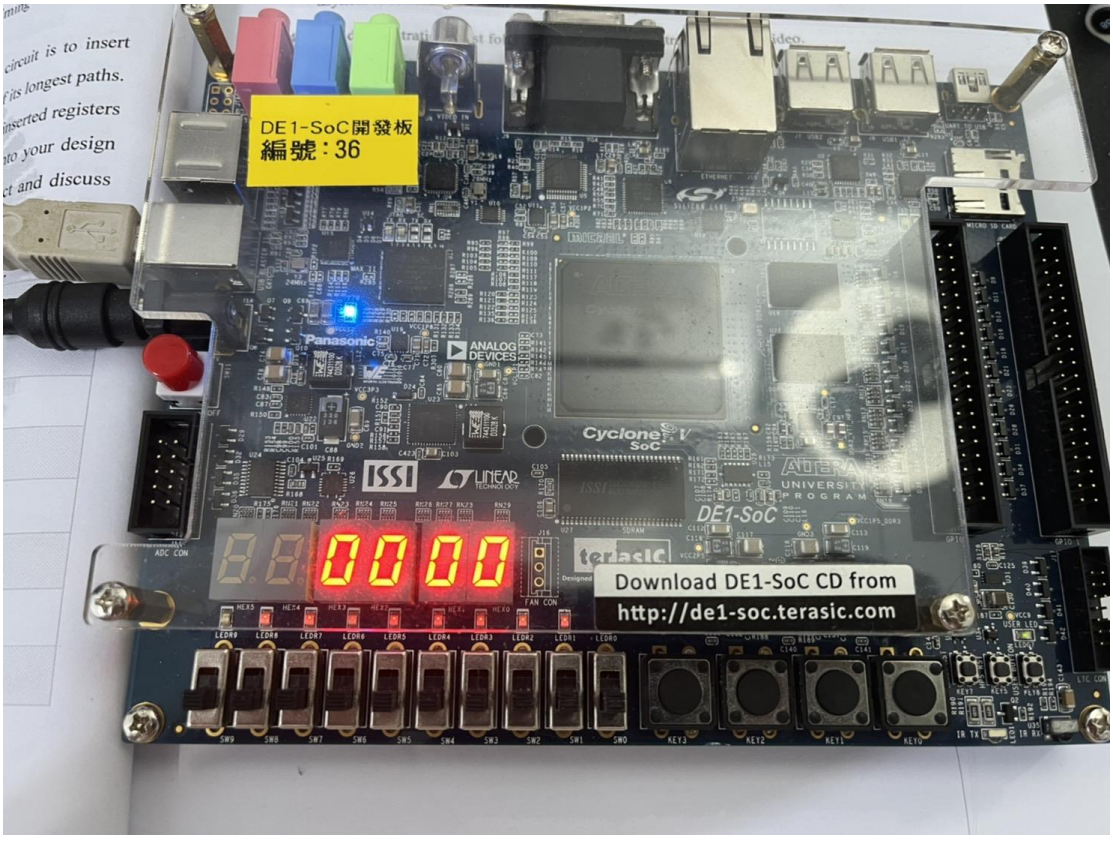
```
//seven segment display
module ssd (    input [3:0] Din,
                output [6:0] Dout );
assign Dout[0] =    ((!Din[3])&(!Din[2])&(!Din[1])&( Din[0]))|
                    ((!Din[3])&( Din[2])&(!Din[1])&(!Din[0]))|
                    (( Din[3])&( Din[2])&(!Din[1])&( Din[0]))|
                    (( 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

```

```
148 // sub modules from LPM
149 module lpm_add16_my (
150     dataa,
151     datab,
152     overflow,
153     result);
154 
155     input   [15:0]  dataa;
156     input   [15:0]  datab;
157     output    overflow;
158     output  [15:0]  result;
159 
160     wire  sub_wire0;
161     wire [15:0] sub_wire1;
162     wire  overflow = sub_wire0;
163     wire [15:0] result = sub_wire1[15:0];
164 
165     lpm_add_sub LPM_ADD_SUB_component (
166                 .dataa (dataa),
167                 .datab (datab),
168                 .overflow (sub_wire0),
169                 .result (sub_wire1)
170                 // synopsys translate_off
171                 ,
172                 .aclr (),
173                 .add_sub (),
174                 .cin (),
175                 .clken (),
176                 .clock (),
177                 .cout ()
178                 // synopsys translate_on
179                 );
180     defparam
181         LPM_ADD_SUB_component.lpm_direction = "ADD",
182         LPM_ADD_SUB_component.lpm_hint = "ONE_INPUT_IS_CONSTANT=NO,CIN_USED=NO",
183         LPM_ADD_SUB_component.lpm_representation = "UNSIGNED",
184         LPM_ADD_SUB_component.lpm_type = "LPM_ADD_SUB",
185         LPM_ADD_SUB_component.lpm_width = 16;
186 
187 
188 endmodule
189 
190 module lpm_mult_my (
```

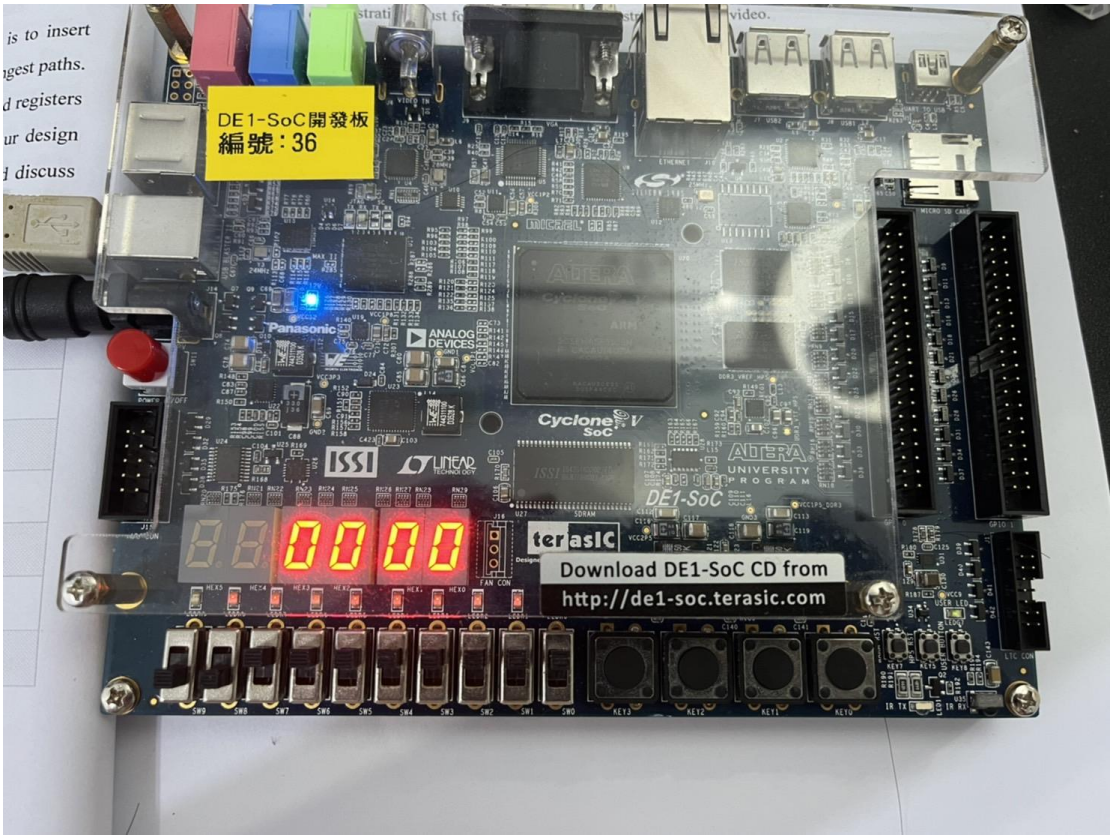
```
module lpm_mult_my (
    dataa,
    datab,
    result);

    input   [7:0]  dataa;
    input   [7:0]  datab;
    output  [15:0]  result;

    wire [15:0] sub_wire0;
    wire [15:0] result = sub_wire0[15:0];

    lpm_mult    lpm_mult_component (
                .dataa (dataa),
                .datab (datab),
                .result (sub_wire0),
                .aclr (1'b0),
                .clken (1'b1),
                .clock (1'b0),
                .sum (1'b0));
    defparam
        lpm_mult_component.lpm_hint = "MAXIMIZE_SPEED=9",
        lpm_mult_component.lpm_representation = "UNSIGNED",
        lpm_mult_component.lpm_type = "LPM_MULT",
        lpm_mult_component.lpm_widtha = 8,
        lpm_mult_component.lpm_widthb = 8,
        lpm_mult_component.lpm_widthp = 16;


endmodule

```

## 三、 實驗結果

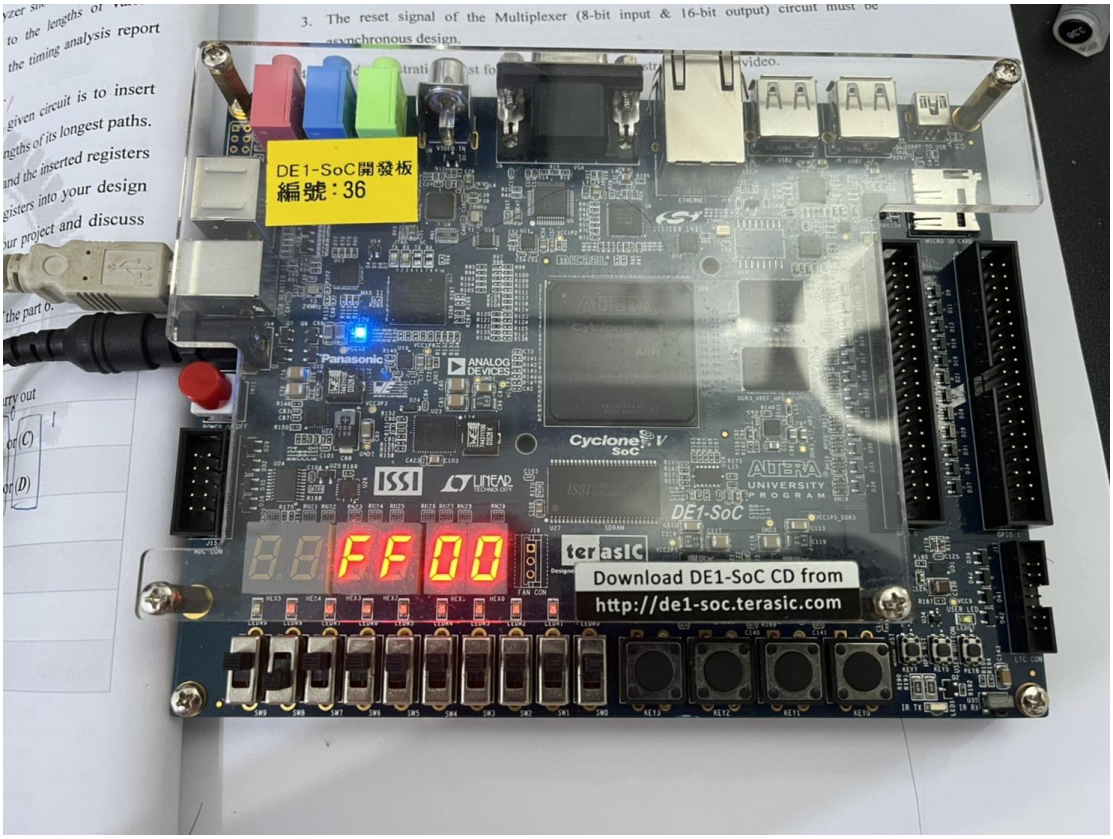
燒錄完成，初始狀態如下圖



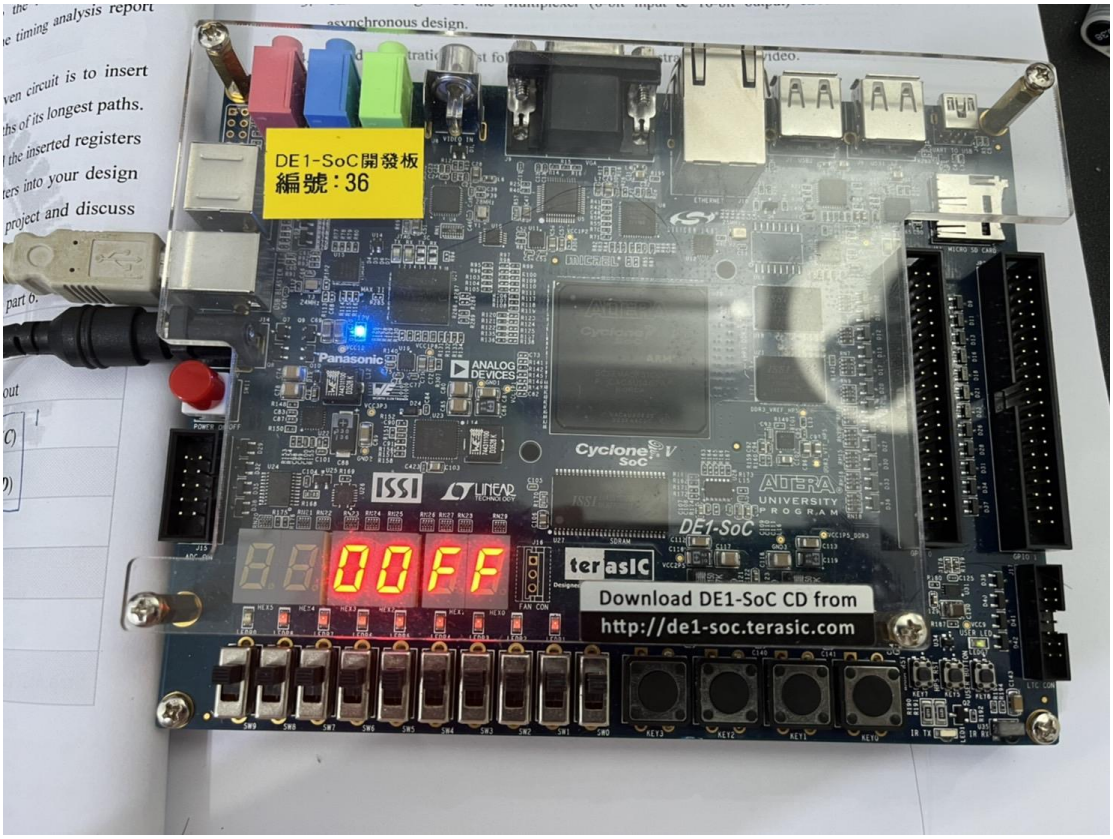
按下一次 KEY1



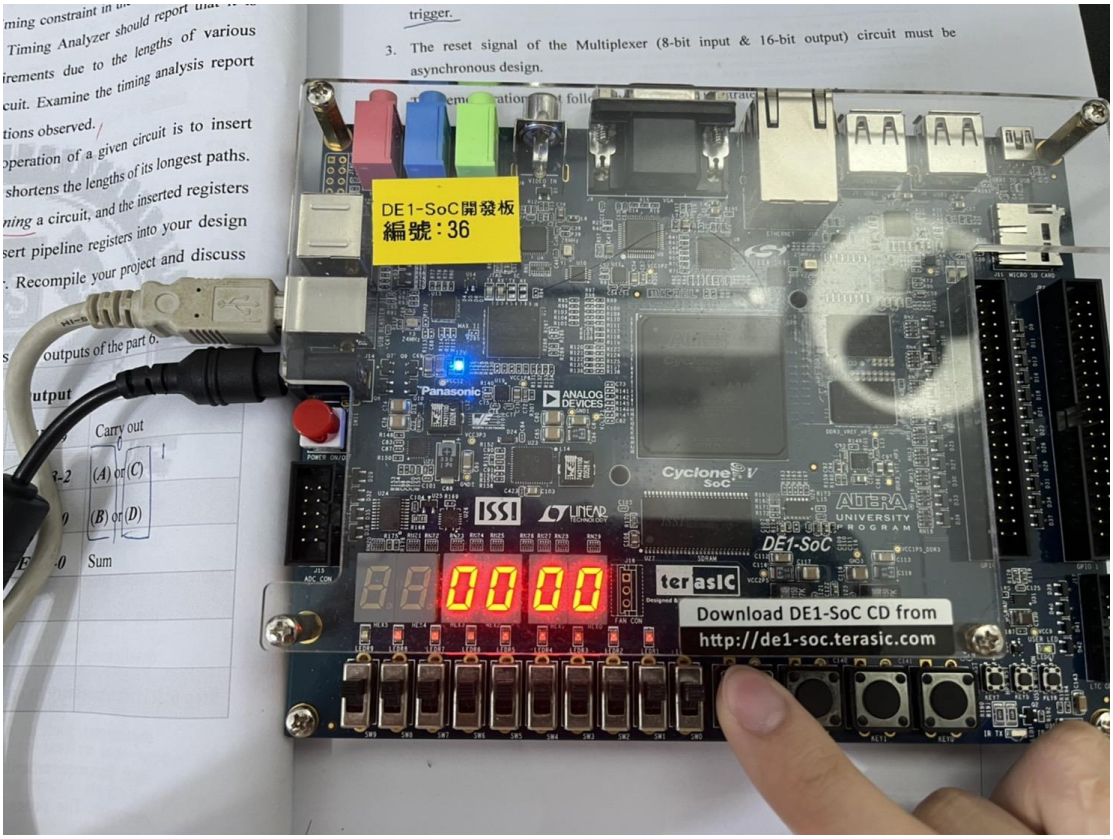

SW[9] = 1 按下一次 KEY1



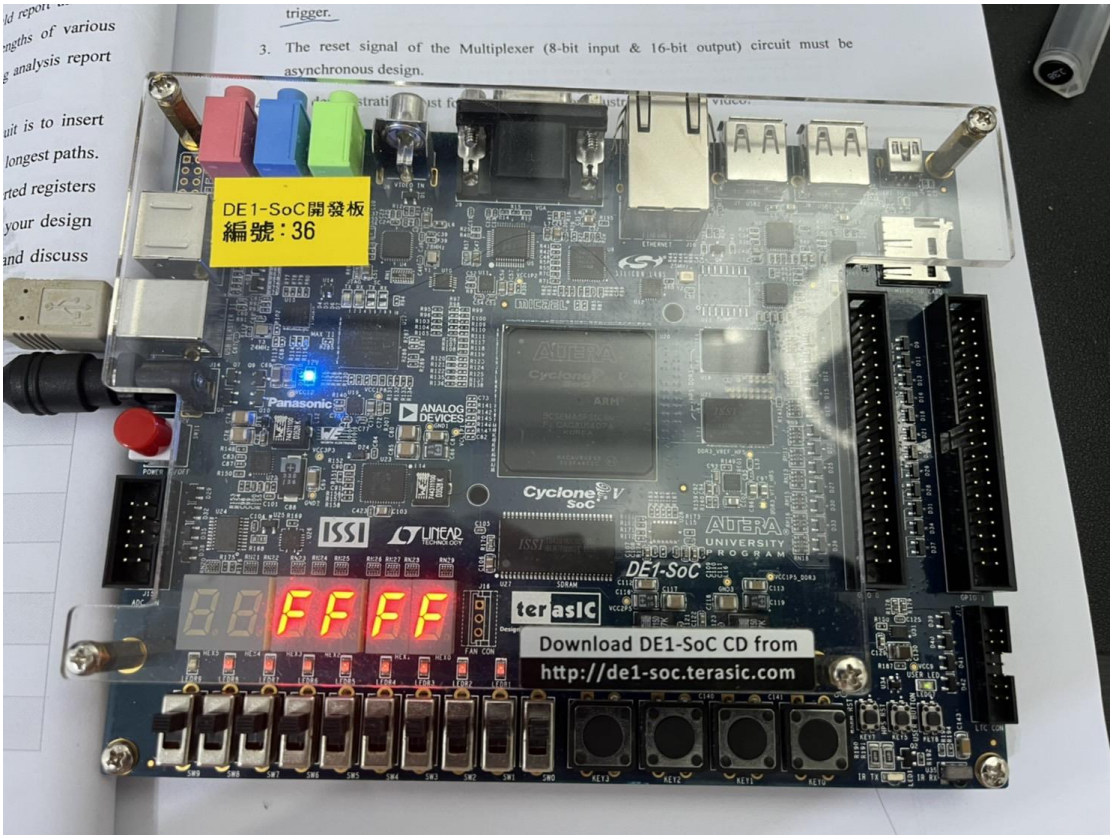
SW[9]=1，SW[8]=1，按住 KEY2 時按下一次 KEY1



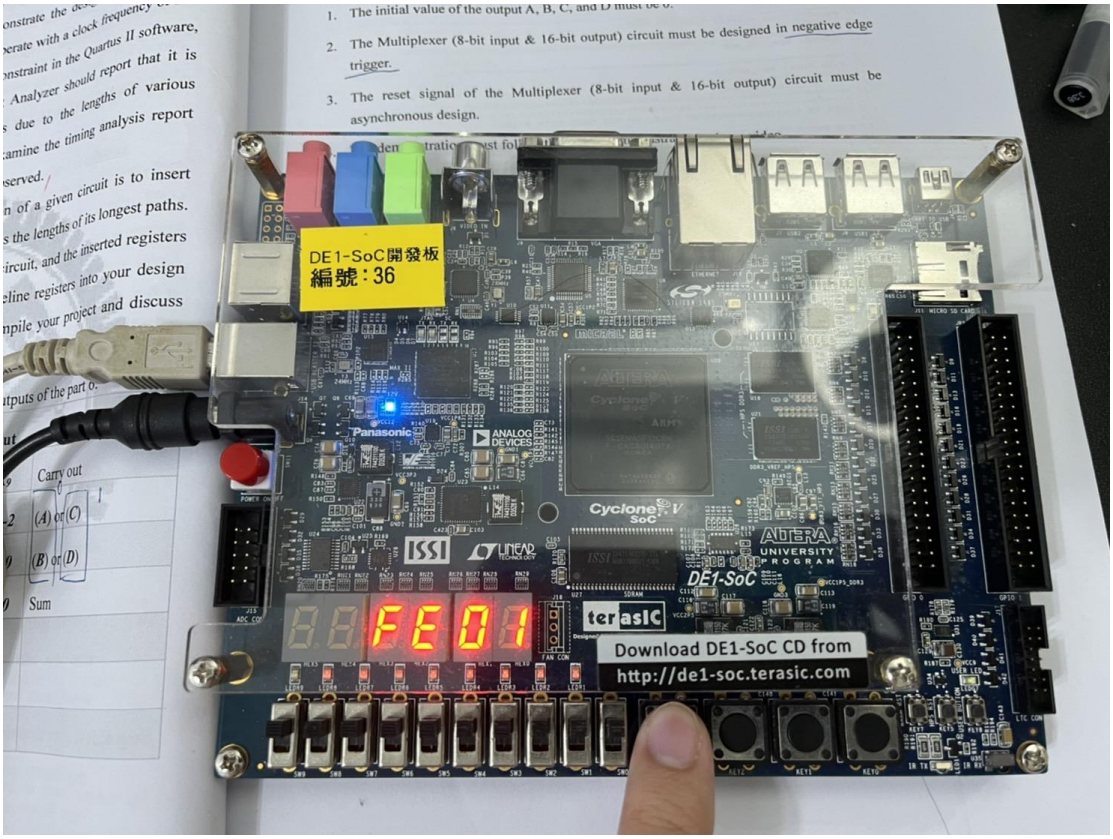

按住 KEY3 顯示當前暫存器計算結果



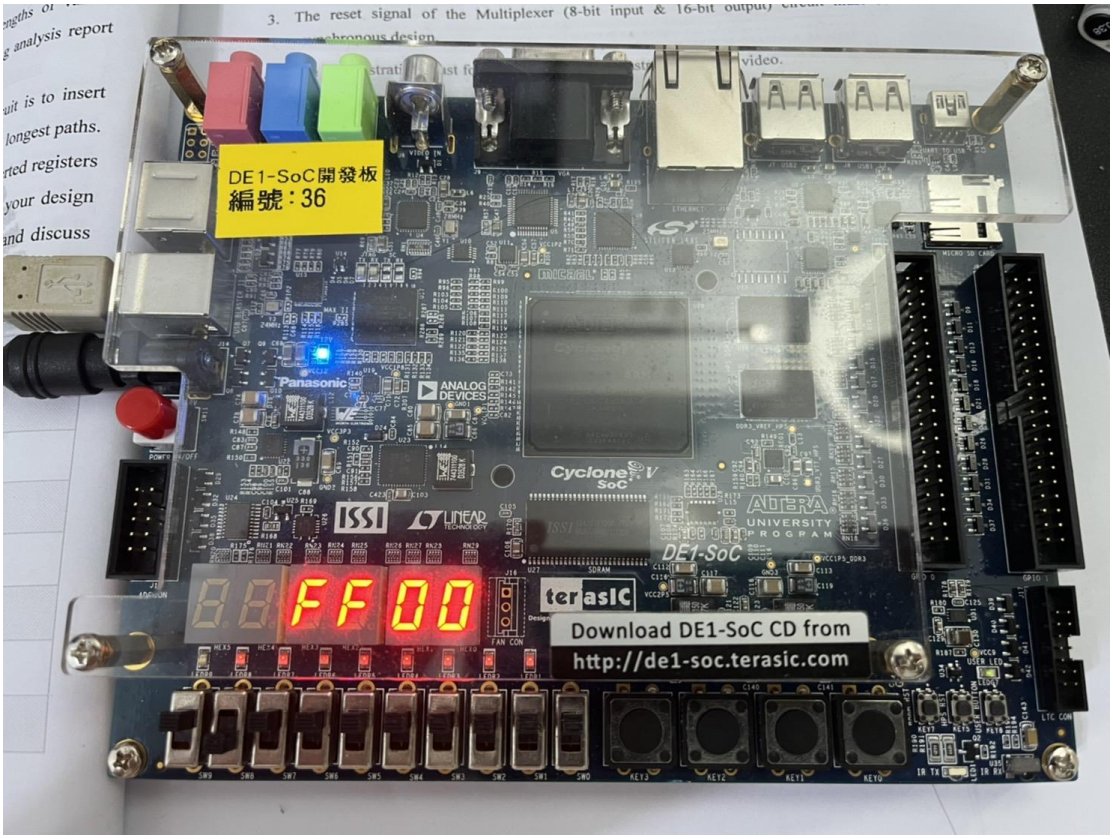
SW[9]=1，SW[8]=1，按下一次 KEY1



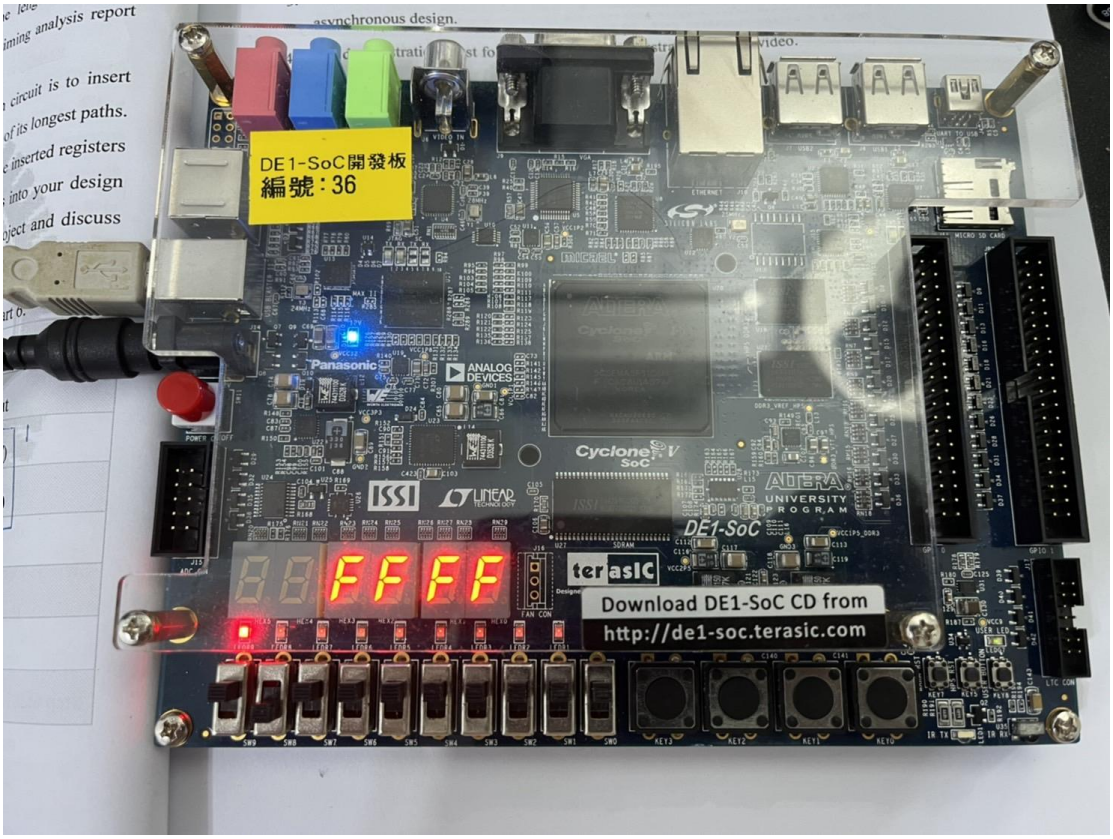

按住 KEY3 顯示當前暫存器計算結果



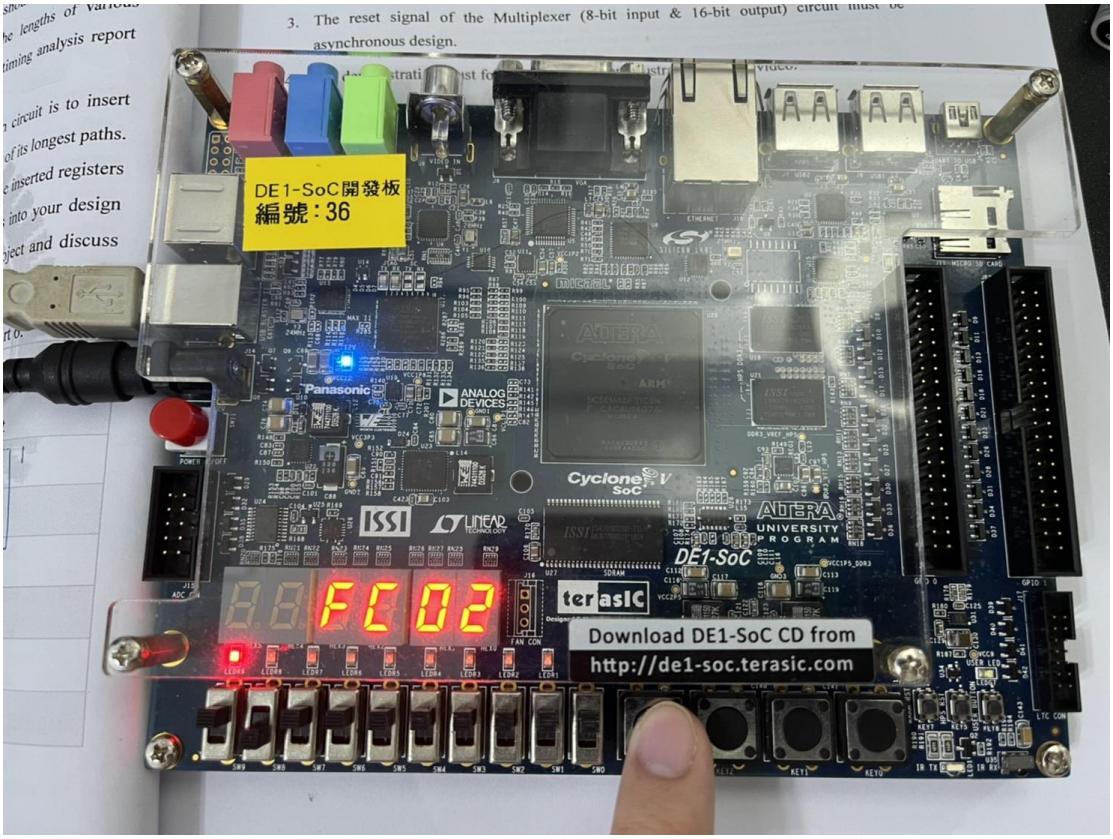
將 SW[8]切換回 0



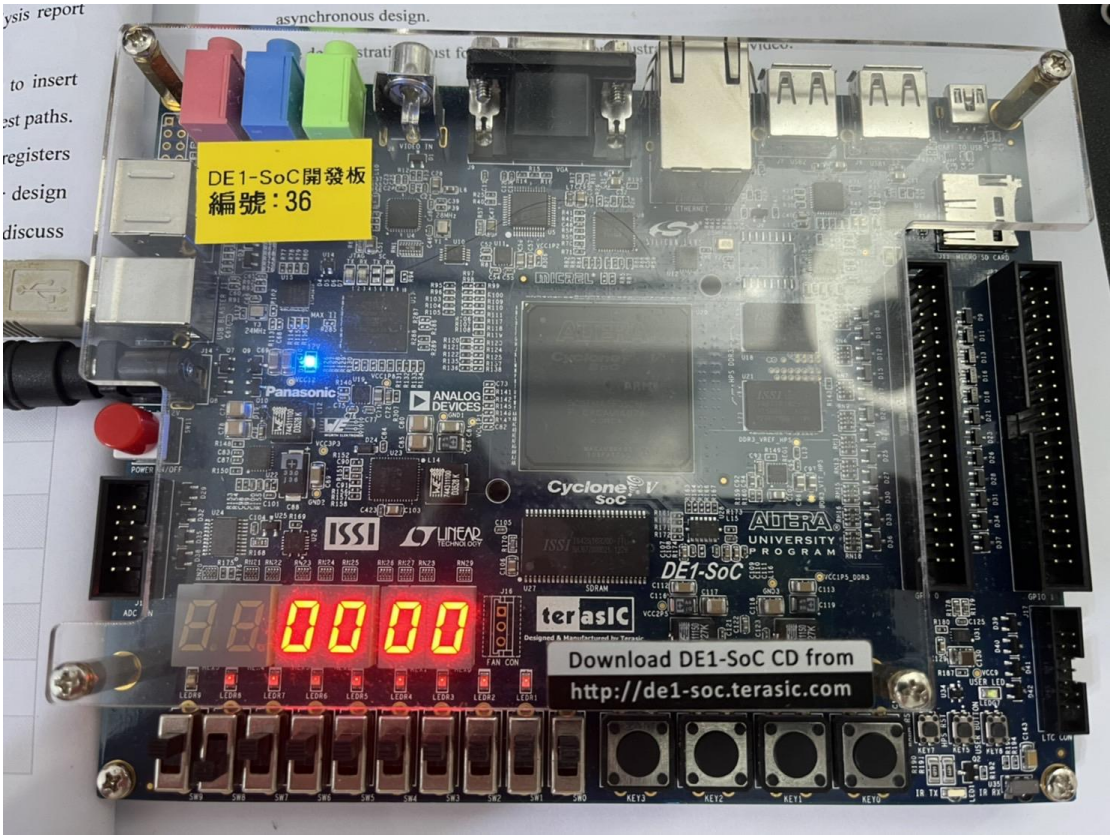

SW[9]=1，SW[8]=0，按住 KEY2 時按下一次 KEY1



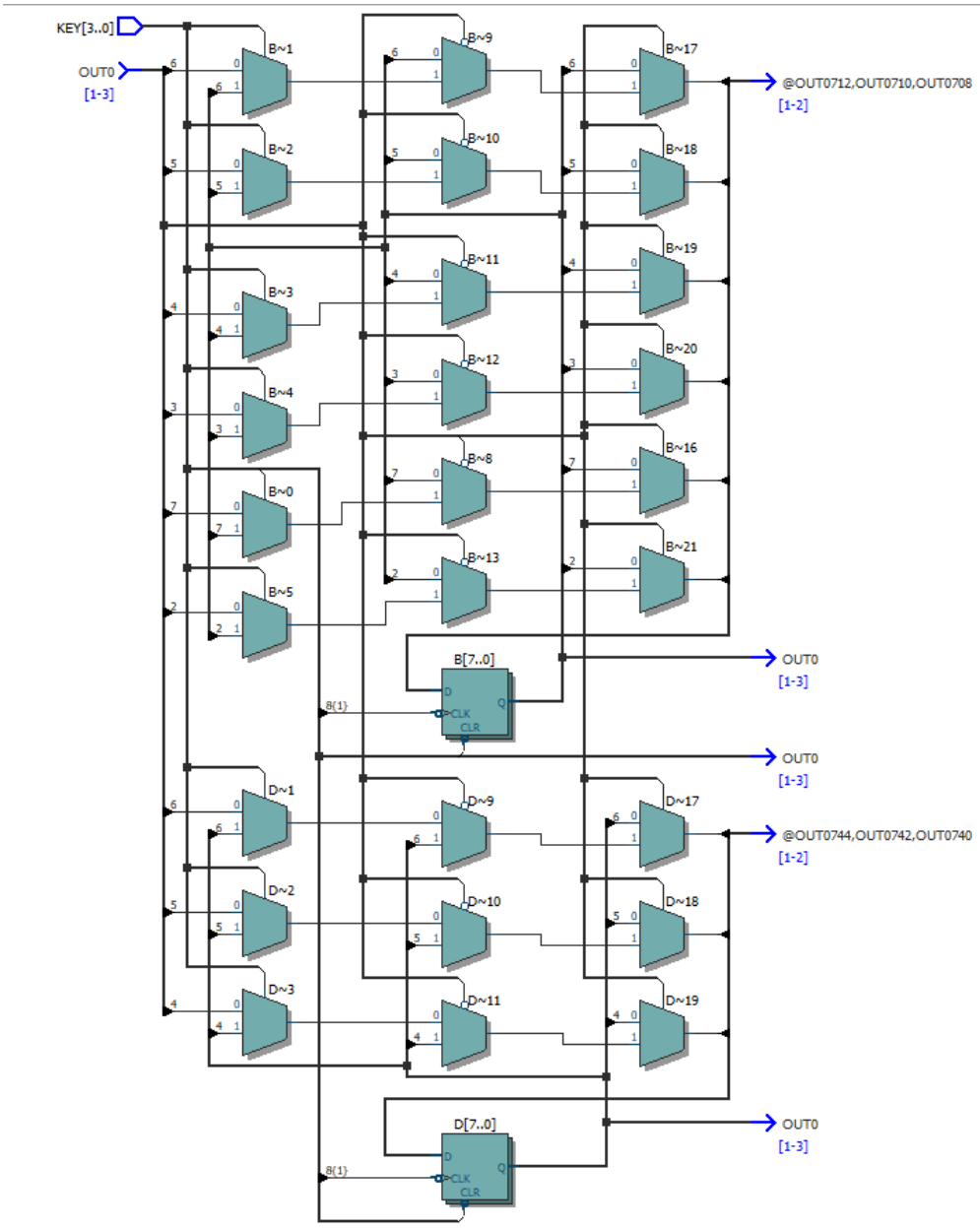
按住 KEY3 顯示當前暫存器計算結果

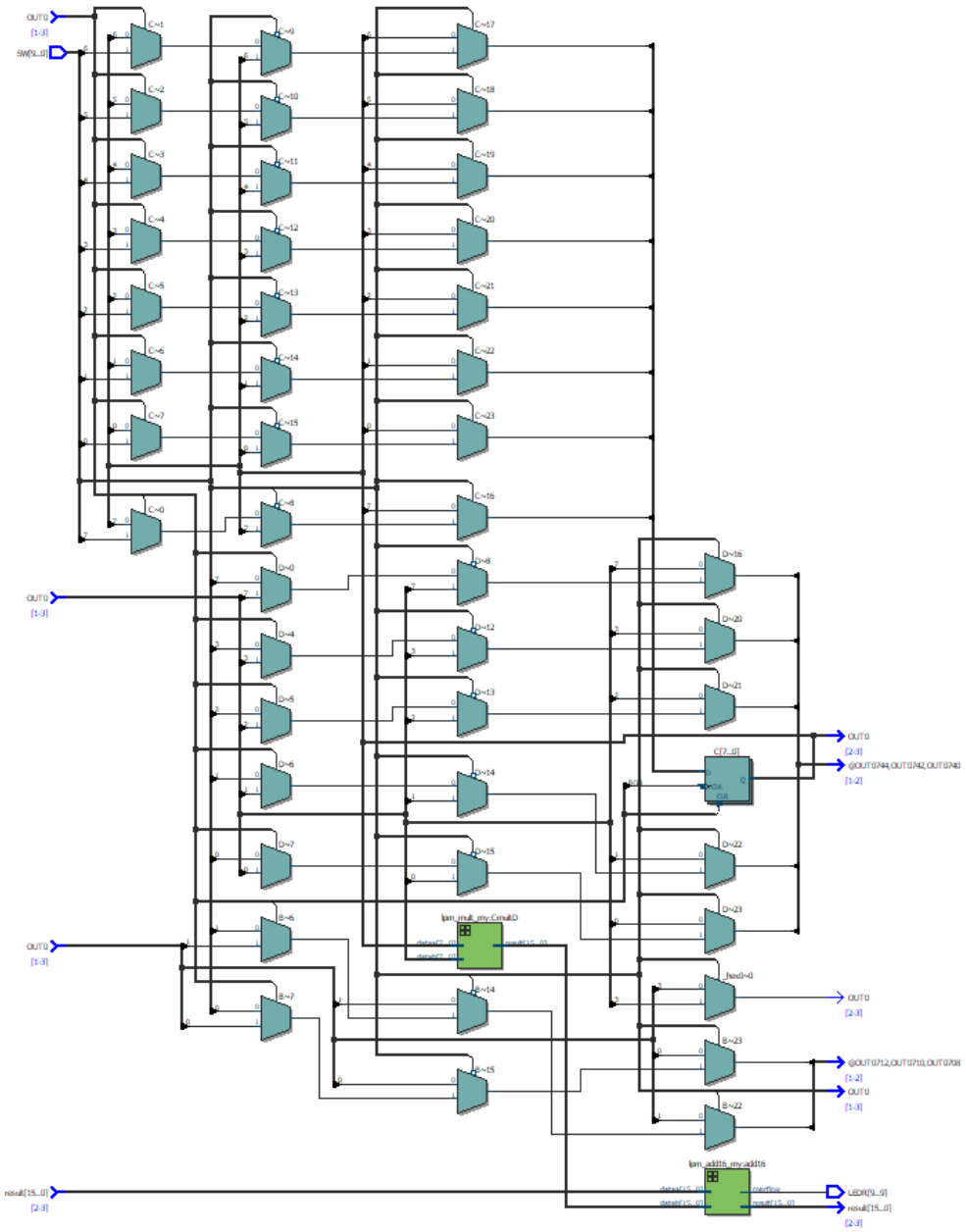


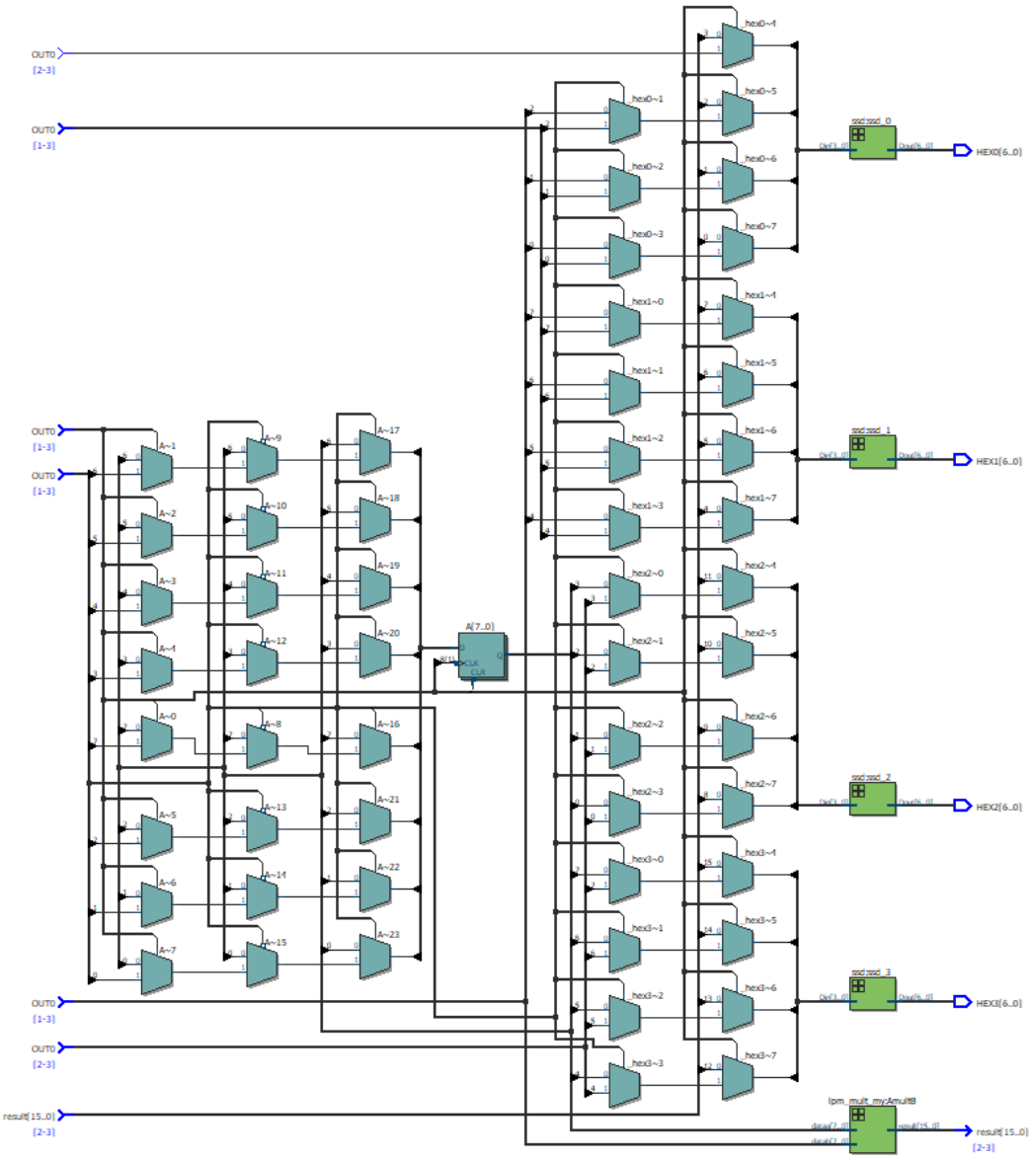

按下 KEY0，reset



四、 RTL


KEY[3..0]
OUT0
[1-3]
B~1
B~2
B~3
B~4
B~0
B~5
B~9
B~10
B~11
B~12
B~8
B~13
B~17
B~18
B~19
B~20
B~16
B~21
@OUT0712,OUT0710,OUT0708
[1-2]
B[7..0]
D
CLK
CLR
Q
8{1}
OUT0
[1-3]
OUT0
[1-3]
D~1
D~2
D~3
D~9
D~10
D~11
D~17
D~18
D~19
@OUT0744,OUT0742,OUT0740
[1-2]
D[7..0]
D
CLK
CLR
Q
8{1}
OUT0
[1-3]



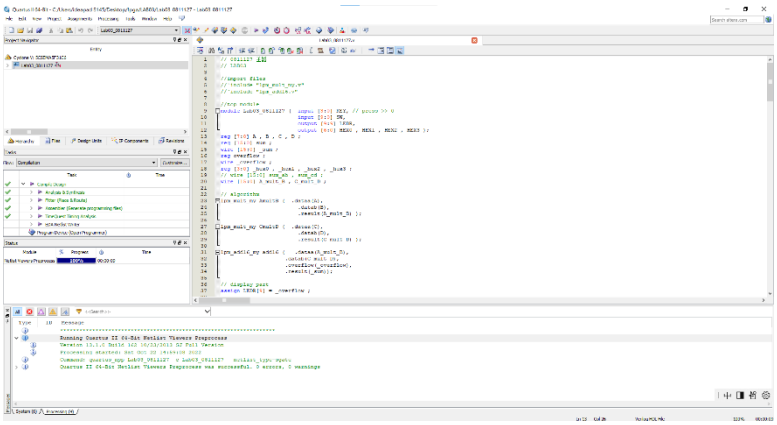


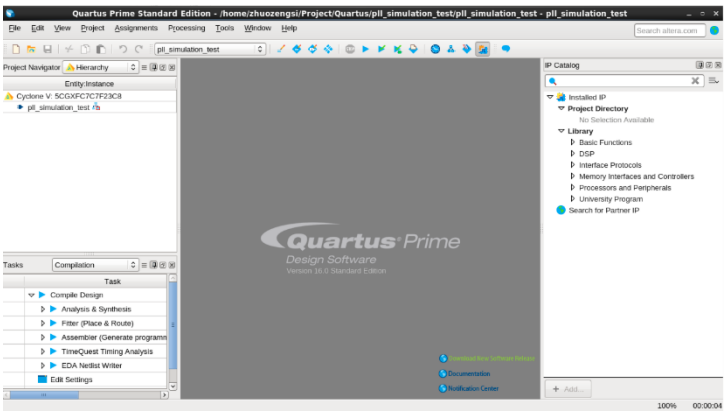


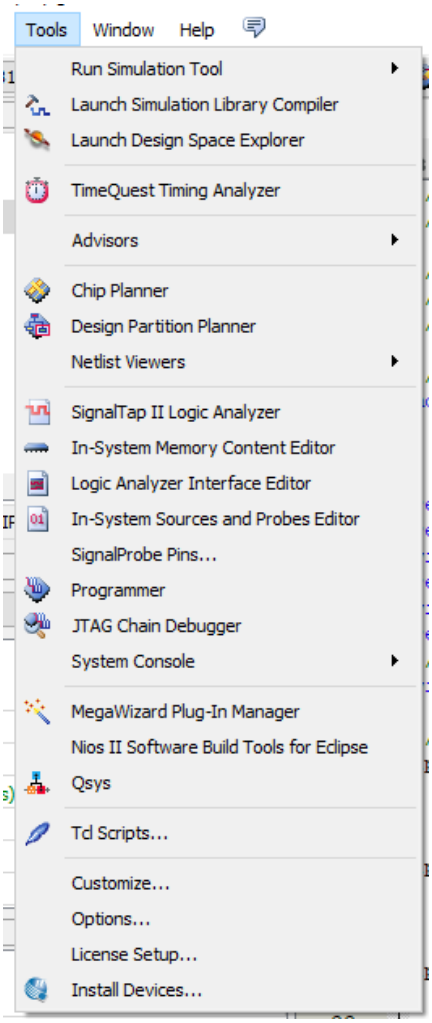
## 五、 問題與討論

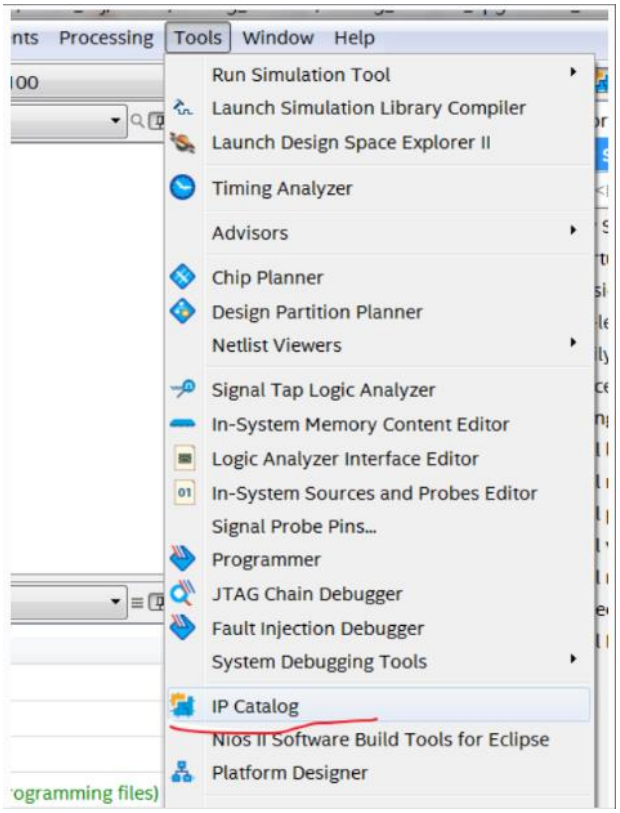
因為先前沒有導入 IP 的經驗，加上 Quartus II 介面和查詢到的網路資料有點對不上，因此這個步驟成了最困難的關卡。

我的 Quartus II 右方沒有 IP Catalog 搜尋框，在 tool 選單裡也沒有。左方是我的 Quartus II ，右方是網路資源中的 Quartus II。






好在最後發現可以用 MegaWizard Plug-In Manager 插入 LPM，才順利度過這項難關。

其餘的部分都很簡單，設定數值、七段顯示器等功能都和先前實驗類似，很直覺的就做出來了，其他像是 reset、WE 等功能也只是微調 if 函式，對我來說並不困難。

## 六、 心得

越來越難了，相較前幾次實驗只花一兩個小時就做完，這次實驗花了整整一個下午，大部分時間都花在釐清題目，其餘部分在查怎麼加入 IP，先前沒有用過以為要‘include，導致浪費很多時間卡在這個 BUG 上。

但是還是有不少收穫，至少現在知道怎麼使用與加入 LPM 了，希望日後的實驗能更順利。