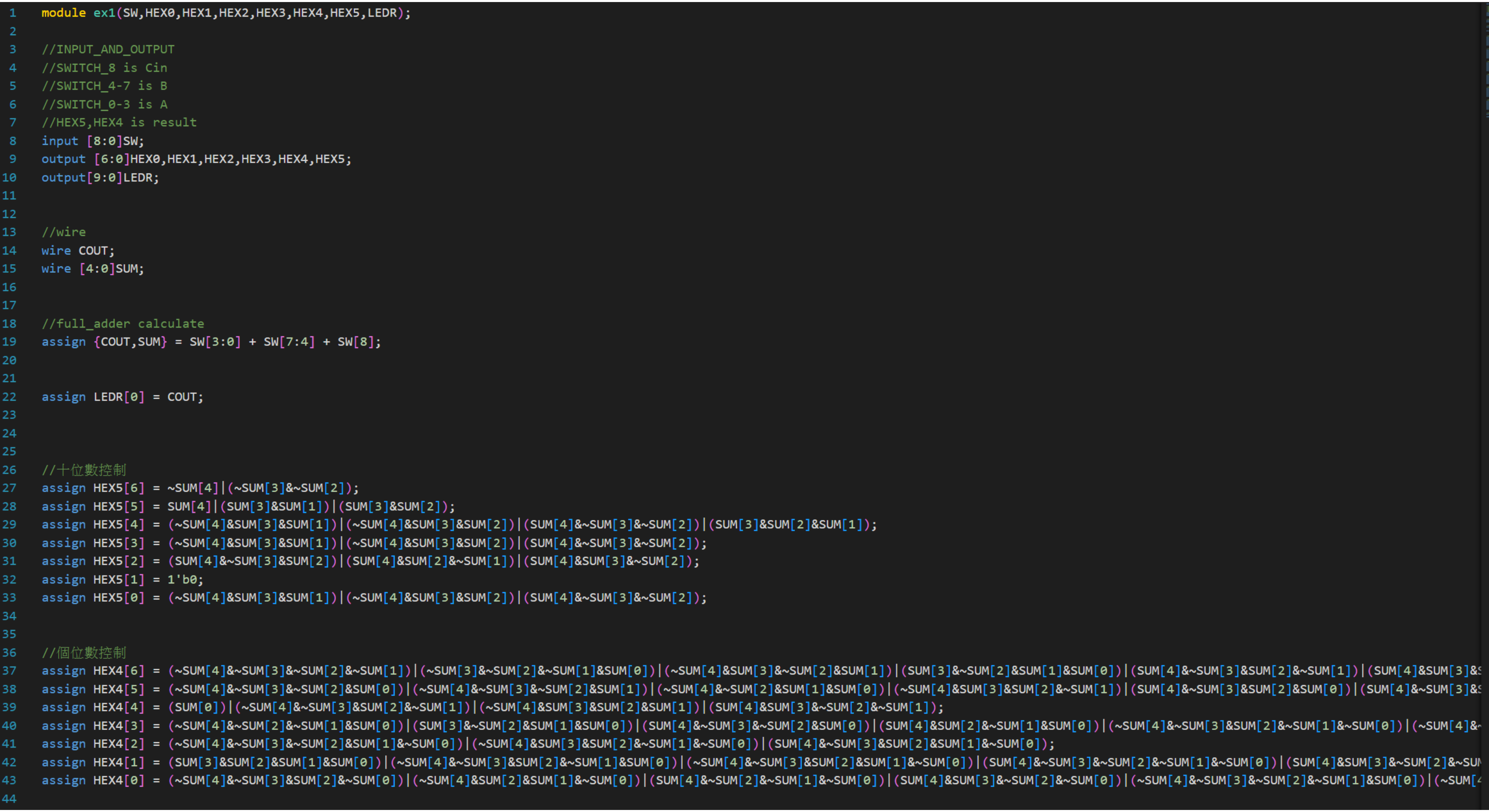
LAB01_312651057

吳鴻明

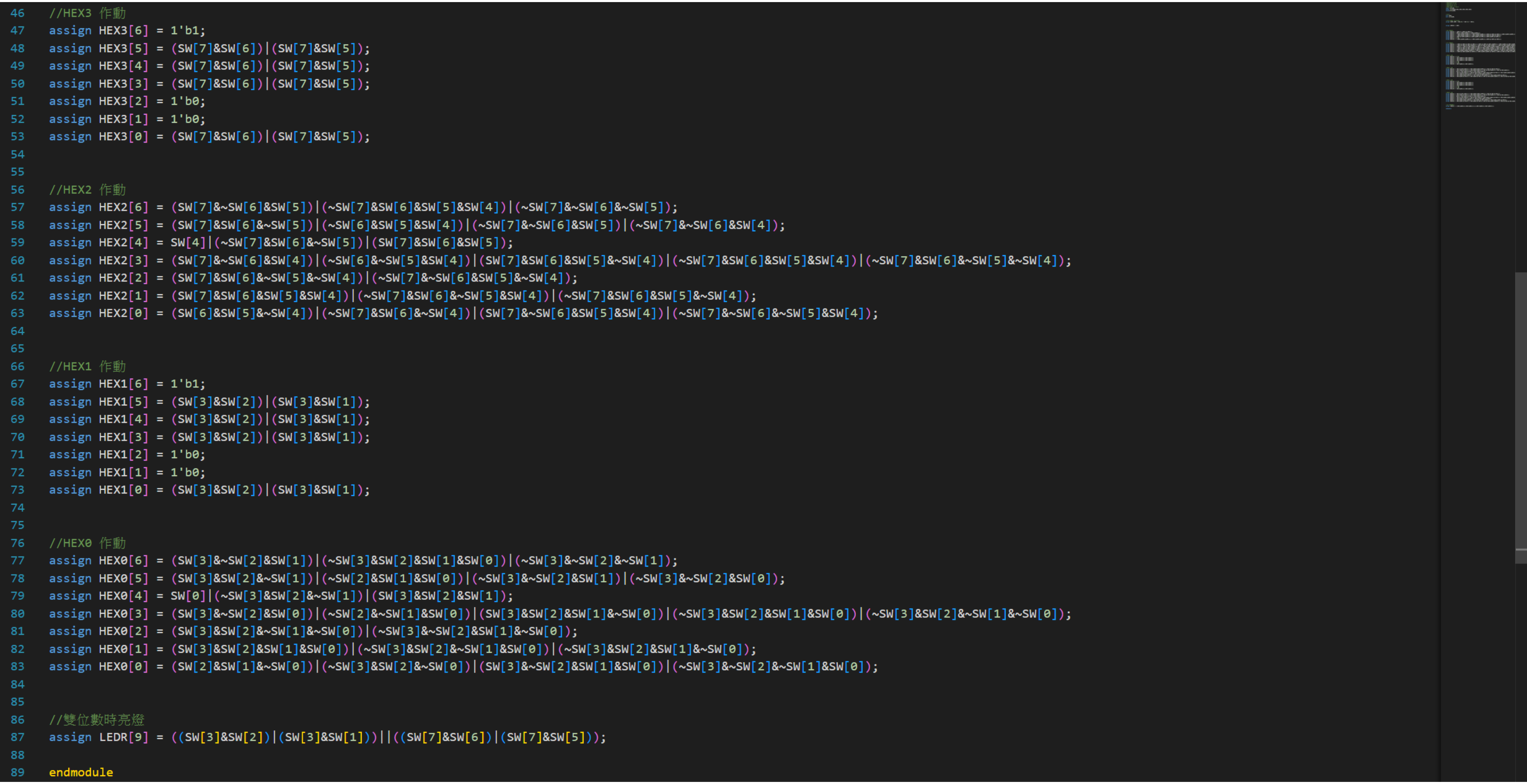
實驗目的

在未使用任何判斷語句的情況下，並且使用邏輯化簡技巧將功能實現。
將Full-Adder的Input、Output透過7段顯示起顯示出來。
透過以上初步熟悉Verilog的assign語法以及如何宣告port

實驗程式碼

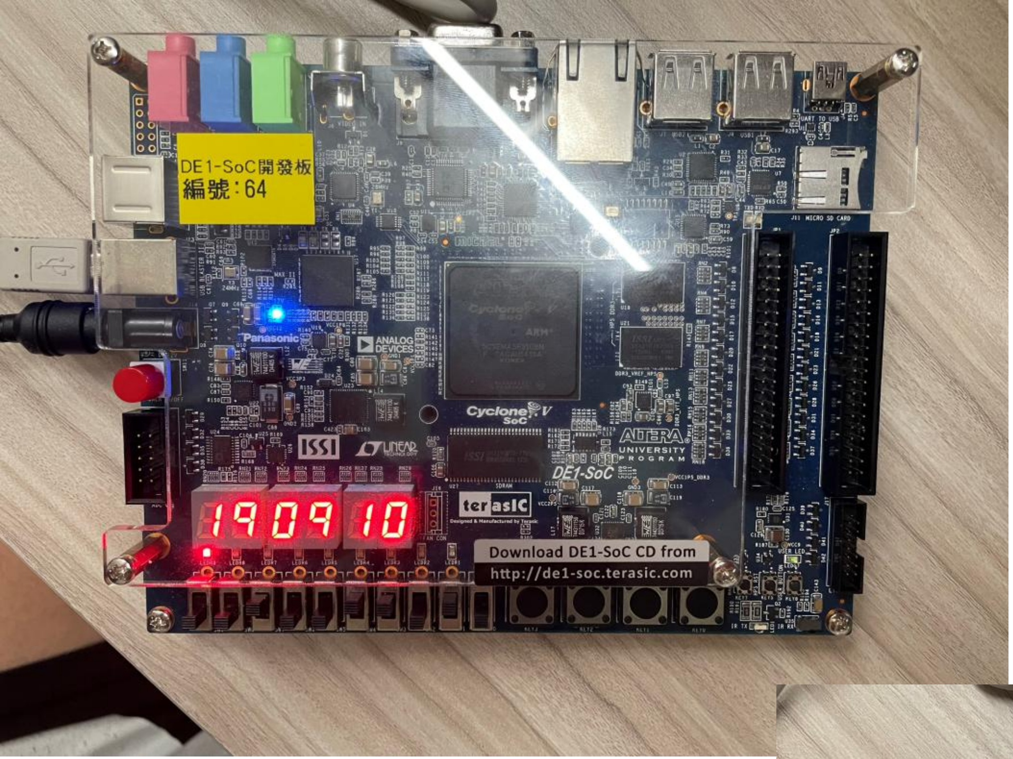
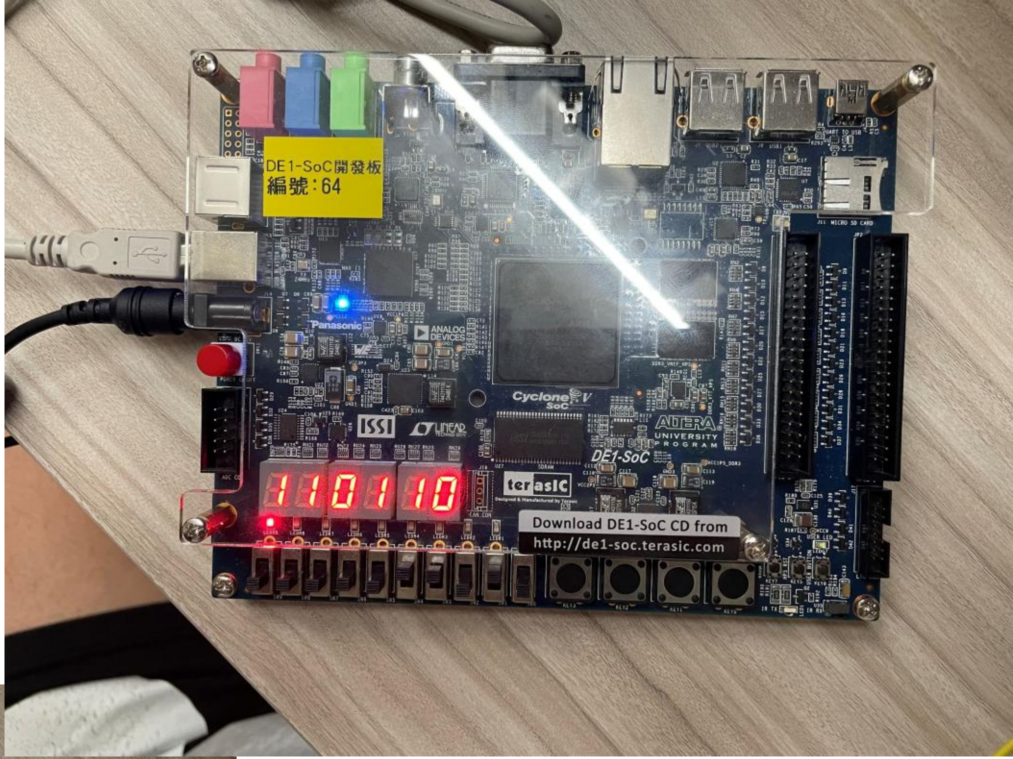
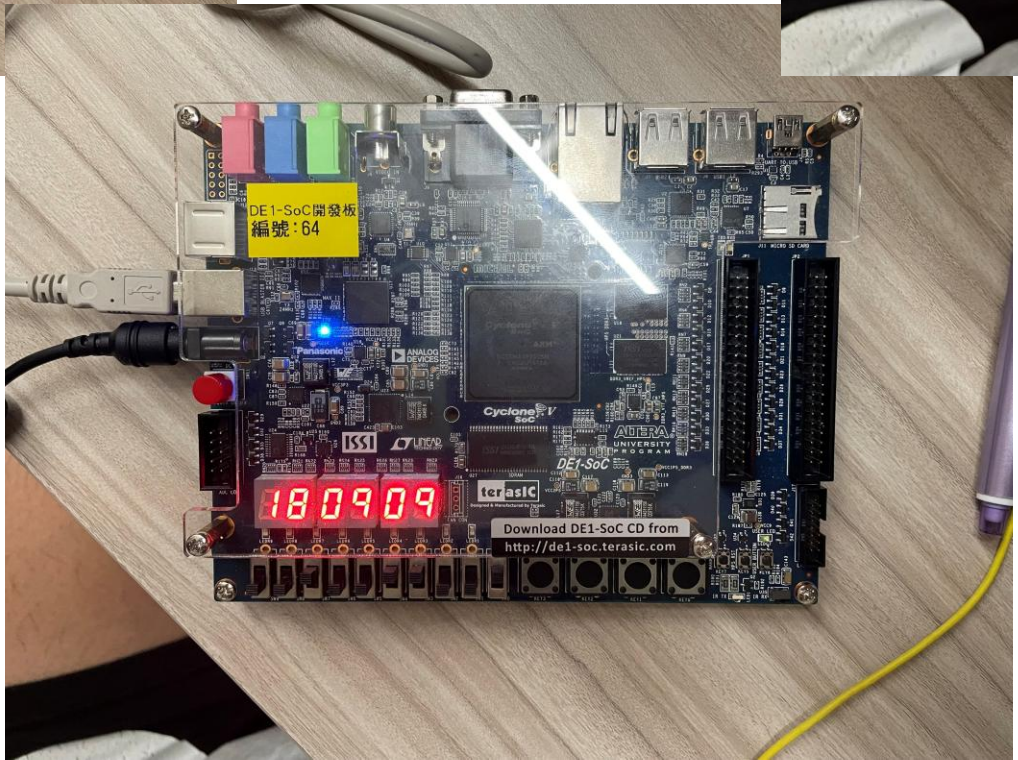


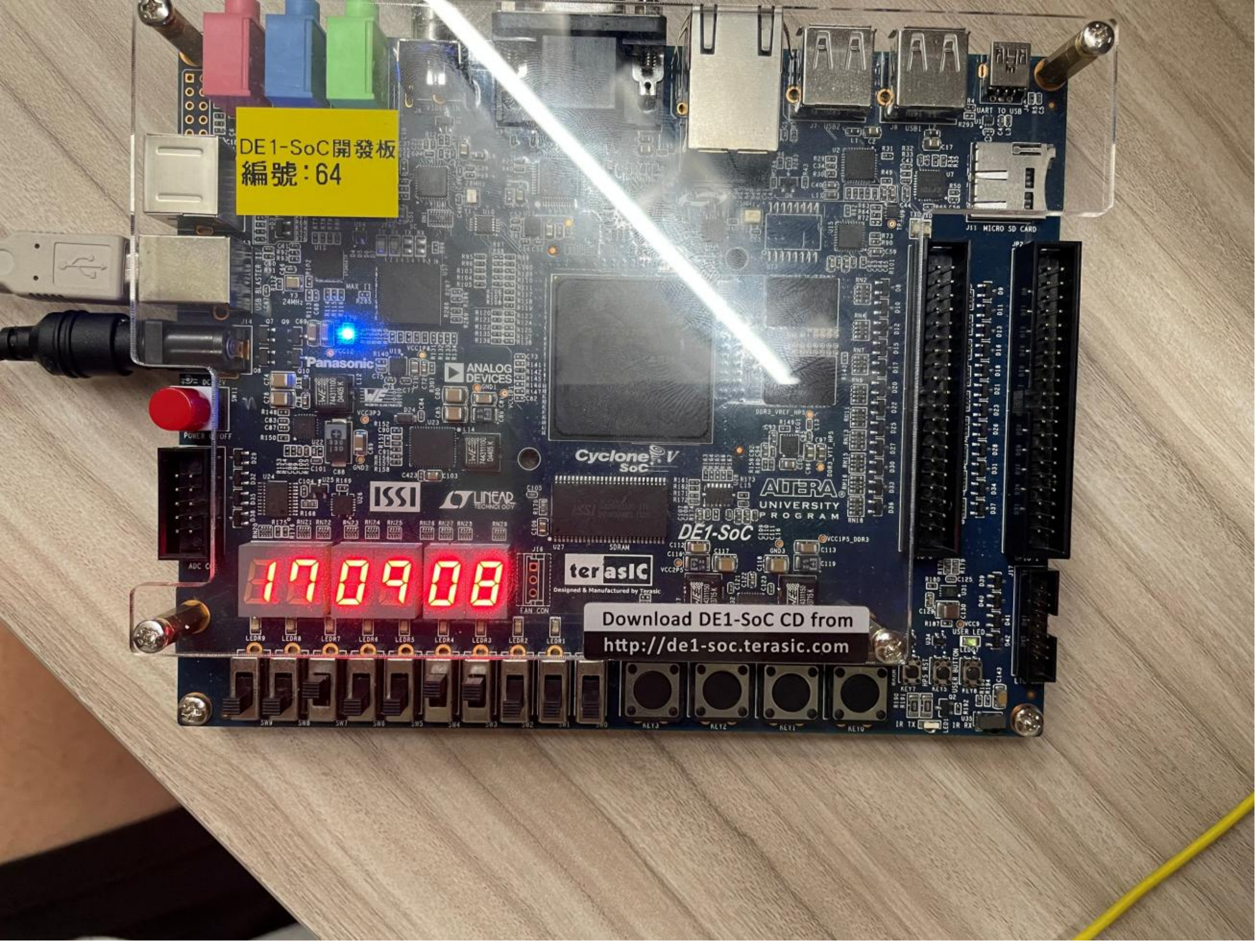
```verilog
module ex1(SW,HEX0,HEX1,HEX2,HEX3,HEX4,HEX5,LEDR);

//INPUT_AND_OUTPUT
//SWITCH_8 is Cin
//SWITCH_4-7 is B
//SWITCH_0-3 is A
//HEX5,HEX4 is result
input [8:0]SW;
output [6:0]HEX0,HEX1,HEX2,HEX3,HEX4,HEX5;
output[9:0]LEDR;


//wire
wire COUT;
wire [4:0]SUM;


//full_adder calculate
assign {COUT,SUM} = SW[3:0] + SW[7:4] + SW[8];


assign LEDR[0] = COUT;



//十位數控制
assign HEX5[6] = ~SUM[4]|(~SUM[3]&~SUM[2]);
assign HEX5[5] = SUM[4]|(SUM[3]&SUM[1])|(SUM[3]&SUM[2]);
assign HEX5[4] = (~SUM[4]&SUM[3]&SUM[1])|(~SUM[4]&SUM[3]&SUM[2])|(SUM[4]&~SUM[3]&~SUM[2])|(SUM[3]&SUM[2]&SUM[1]);
assign HEX5[3] = (~SUM[4]&SUM[3]&SUM[1])|(~SUM[4]&SUM[3]&SUM[2])|(SUM[4]&~SUM[3]&~SUM[2]);
assign HEX5[2] = (SUM[4]&~SUM[3]&SUM[2])|(SUM[4]&SUM[2]&~SUM[1])|(SUM[4]&SUM[3]&~SUM[2]);
assign HEX5[1] = 1'b0;
assign HEX5[0] = (~SUM[4]&SUM[3]&SUM[1])|(~SUM[4]&SUM[3]&SUM[2])|(SUM[4]&~SUM[3]&~SUM[2]);


//個位數控制
assign HEX4[6] = (~SUM[4]&~SUM[3]&~SUM[2]&~SUM[1])|(~SUM[3]&~SUM[2]&~SUM[1]&SUM[0])|(~SUM[4]&SUM[3]&~SUM[2]&SUM[1])|(SUM[3]&~SUM[2]&SUM[1]&SUM[0])|(SUM[4]&~SUM[3]&SUM[2]&~SUM[1])|(SUM[4]&SUM[3]&S
assign HEX4[5] = (~SUM[4]&~SUM[3]&~SUM[2]&SUM[0])|(~SUM[4]&~SUM[3]&~SUM[2]&SUM[1])|(~SUM[4]&~SUM[2]&SUM[1]&SUM[0])|(~SUM[4]&SUM[3]&SUM[2]&~SUM[1])|(SUM[4]&~SUM[3]&SUM[2]&SUM[0])|(SUM[4]&~SUM[3]&S
assign HEX4[4] = (SUM[0])|(~SUM[4]&~SUM[3]&SUM[2]&~SUM[1])|(~SUM[4]&SUM[3]&SUM[2]&SUM[1])|(SUM[4]&SUM[3]&~SUM[2]&~SUM[1]);
assign HEX4[3] = (~SUM[4]&~SUM[2]&~SUM[1]&SUM[0])|(SUM[3]&~SUM[2]&SUM[1]&SUM[0])|(SUM[4]&~SUM[3]&~SUM[2]&SUM[0])|(SUM[4]&SUM[2]&~SUM[1]&SUM[0])|(~SUM[4]&~SUM[3]&SUM[2]&~SUM[1]&~SUM[0])|(~SUM[4]&~
assign HEX4[2] = (~SUM[4]&~SUM[3]&~SUM[2]&SUM[1]&~SUM[0])|(~SUM[4]&SUM[3]&SUM[2]&~SUM[1]&~SUM[0])|(SUM[4]&~SUM[3]&SUM[2]&SUM[1]&~SUM[0]);
assign HEX4[1] = (SUM[3]&SUM[2]&SUM[1]&SUM[0])|(~SUM[4]&~SUM[3]&SUM[2]&~SUM[1]&SUM[0])|(~SUM[4]&~SUM[3]&SUM[2]&SUM[1]&~SUM[0])|(SUM[4]&~SUM[3]&~SUM[2]&~SUM[1]&~SUM[0])|(SUM[4]&SUM[3]&~SUM[2]&~SU
assign HEX4[0] = (~SUM[4]&~SUM[3]&SUM[2]&~SUM[0])|(~SUM[4]&SUM[2]&SUM[1]&~SUM[0])|(SUM[4]&~SUM[2]&~SUM[1]&~SUM[0])|(SUM[4]&SUM[3]&~SUM[2]&~SUM[0])|(~SUM[4]&~SUM[3]&~SUM[2]&~SUM[1]&SUM[0])|(~SUM[4
```

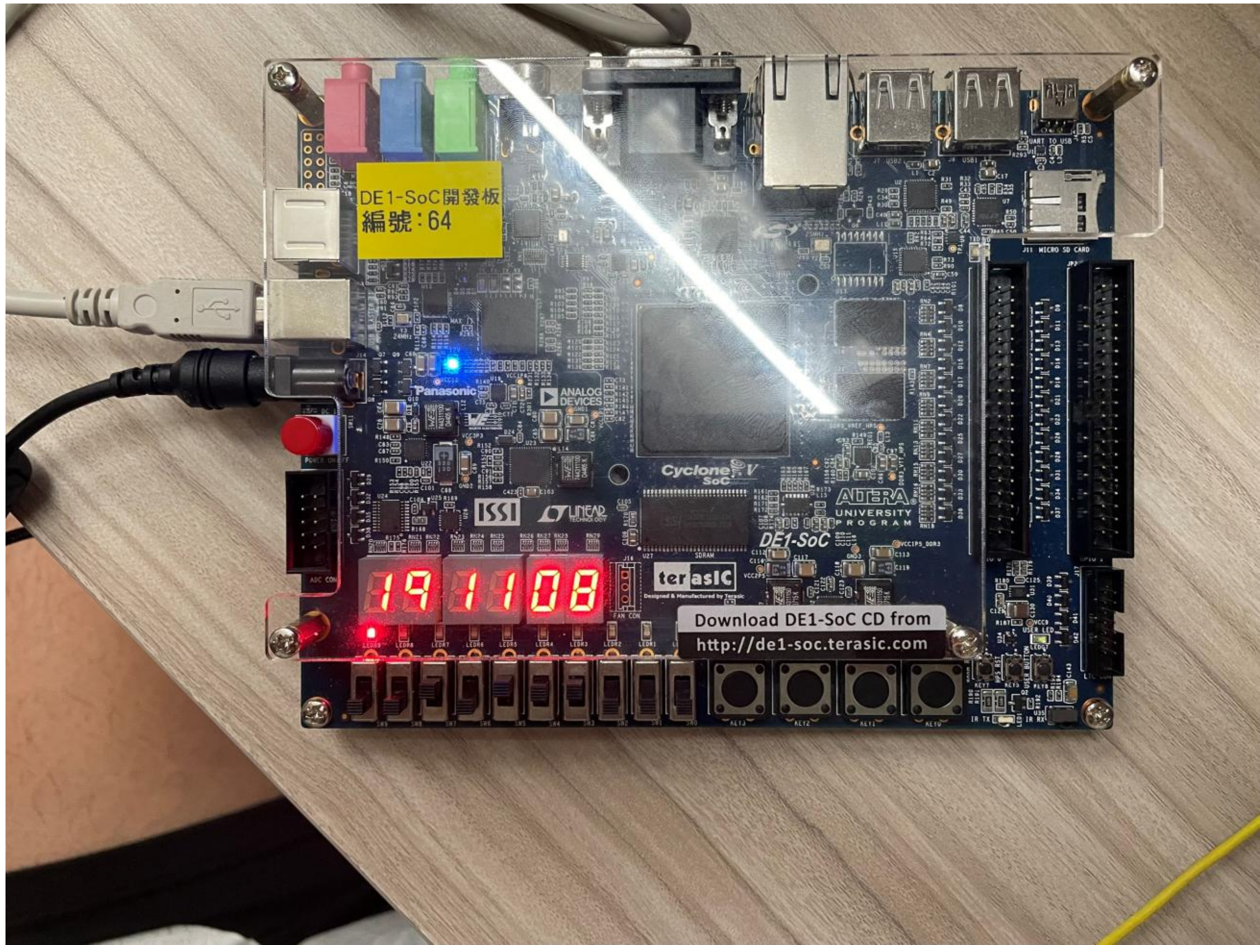


```verilog
//HEX3 作動
assign HEX3[6] = 1'b1;
assign HEX3[5] = (SW[7]&SW[6])|(SW[7]&SW[5]);
assign HEX3[4] = (SW[7]&SW[6])|(SW[7]&SW[5]);
assign HEX3[3] = (SW[7]&SW[6])|(SW[7]&SW[5]);
assign HEX3[2] = 1'b0;
assign HEX3[1] = 1'b0;
assign HEX3[0] = (SW[7]&SW[6])|(SW[7]&SW[5]);


//HEX2 作動
assign HEX2[6] = (SW[7]&~SW[6]&SW[5])|(~SW[7]&SW[6]&SW[5]&SW[4])|(~SW[7]&~SW[6]&~SW[5]);
assign HEX2[5] = (SW[7]&SW[6]&~SW[5])|(~SW[6]&SW[5]&SW[4])|(~SW[7]&~SW[6]&SW[5])|(~SW[7]&~SW[6]&SW[4]);
assign HEX2[4] = SW[4]|(~SW[7]&SW[6]&~SW[5])|(SW[7]&SW[6]&SW[5]);
assign HEX2[3] = (SW[7]&~SW[6]&SW[4])|(~SW[6]&~SW[5]&SW[4])|(SW[7]&SW[6]&SW[5]&~SW[4])|(~SW[7]&SW[6]&SW[5]&SW[4])|(~SW[7]&SW[6]&~SW[5]&~SW[4]);
assign HEX2[2] = (SW[7]&SW[6]&~SW[5]&~SW[4])|(~SW[7]&~SW[6]&SW[5]&~SW[4]);
assign HEX2[1] = (SW[7]&SW[6]&SW[5]&SW[4])|(~SW[7]&SW[6]&~SW[5]&SW[4])|(~SW[7]&SW[6]&SW[5]&~SW[4]);
assign HEX2[0] = (SW[6]&SW[5]&~SW[4])|(~SW[7]&SW[6]&~SW[4])|(SW[7]&~SW[6]&SW[5]&SW[4])|(~SW[7]&~SW[6]&~SW[5]&SW[4]);


//HEX1 作動
assign HEX1[6] = 1'b1;
assign HEX1[5] = (SW[3]&SW[2])|(SW[3]&SW[1]);
assign HEX1[4] = (SW[3]&SW[2])|(SW[3]&SW[1]);
assign HEX1[3] = (SW[3]&SW[2])|(SW[3]&SW[1]);
assign HEX1[2] = 1'b0;
assign HEX1[1] = 1'b0;
assign HEX1[0] = (SW[3]&SW[2])|(SW[3]&SW[1]);


//HEX0 作動
assign HEX0[6] = (SW[3]&~SW[2]&SW[1])|(~SW[3]&SW[2]&SW[1]&SW[0])|(~SW[3]&~SW[2]&~SW[1]);
assign HEX0[5] = (SW[3]&SW[2]&~SW[1])|(~SW[2]&SW[1]&SW[0])|(~SW[3]&~SW[2]&SW[1])|(~SW[3]&~SW[2]&SW[0]);
assign HEX0[4] = SW[0]|(~SW[3]&SW[2]&~SW[1])|(SW[3]&SW[2]&SW[1]);
assign HEX0[3] = (SW[3]&~SW[2]&SW[0])|(~SW[2]&~SW[1]&SW[0])|(SW[3]&SW[2]&SW[1]&~SW[0])|(~SW[3]&SW[2]&SW[1]&SW[0])|(~SW[3]&SW[2]&~SW[1]&~SW[0]);
assign HEX0[2] = (SW[3]&SW[2]&~SW[1]&~SW[0])|(~SW[3]&~SW[2]&SW[1]&~SW[0]);
assign HEX0[1] = (SW[3]&SW[2]&SW[1]&SW[0])|(~SW[3]&SW[2]&~SW[1]&SW[0])|(~SW[3]&SW[2]&SW[1]&~SW[0]);
assign HEX0[0] = (SW[2]&SW[1]&~SW[0])|(~SW[3]&SW[2]&~SW[0])|(SW[3]&~SW[2]&SW[1]&SW[0])|(~SW[3]&~SW[2]&~SW[1]&SW[0]);


//雙位數時亮燈
assign LEDR[9] = ((SW[3]&SW[2])|(SW[3]&SW[1]))||((SW[7]&SW[6])|(SW[7]&SW[5]));

endmodule
```

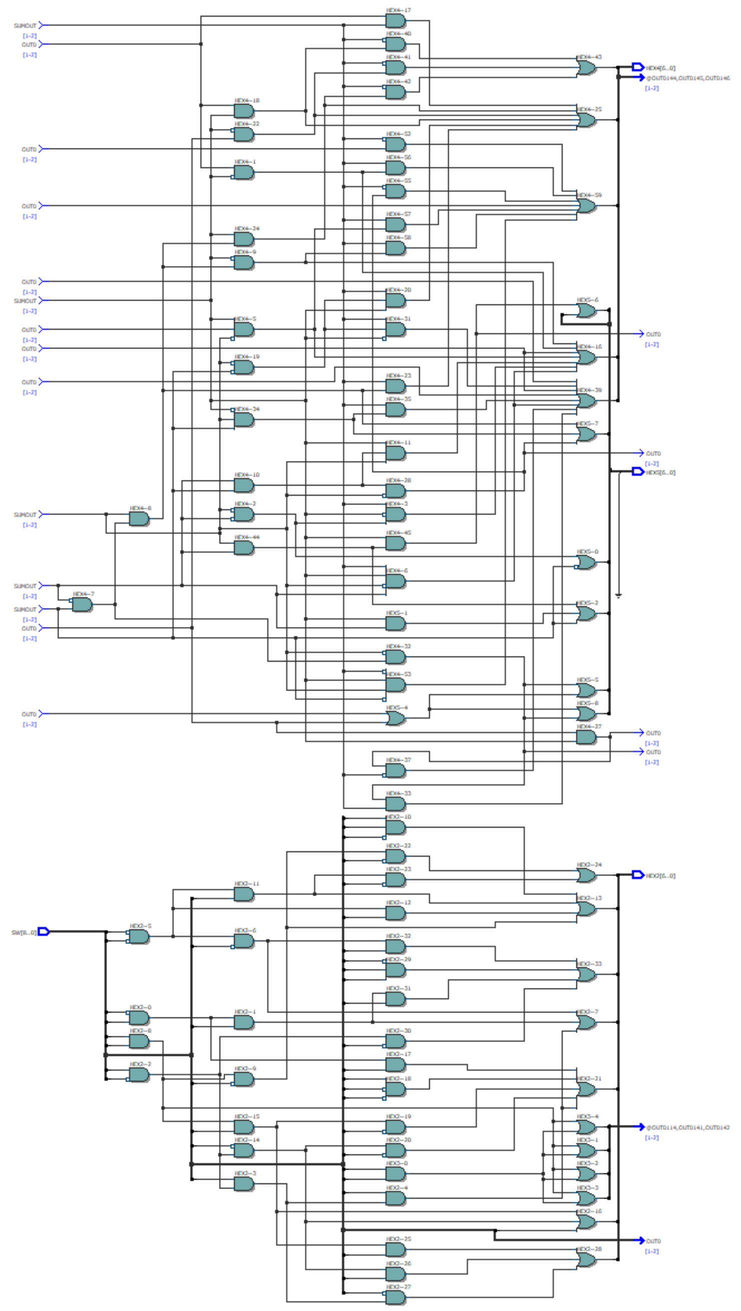
實驗結果照片

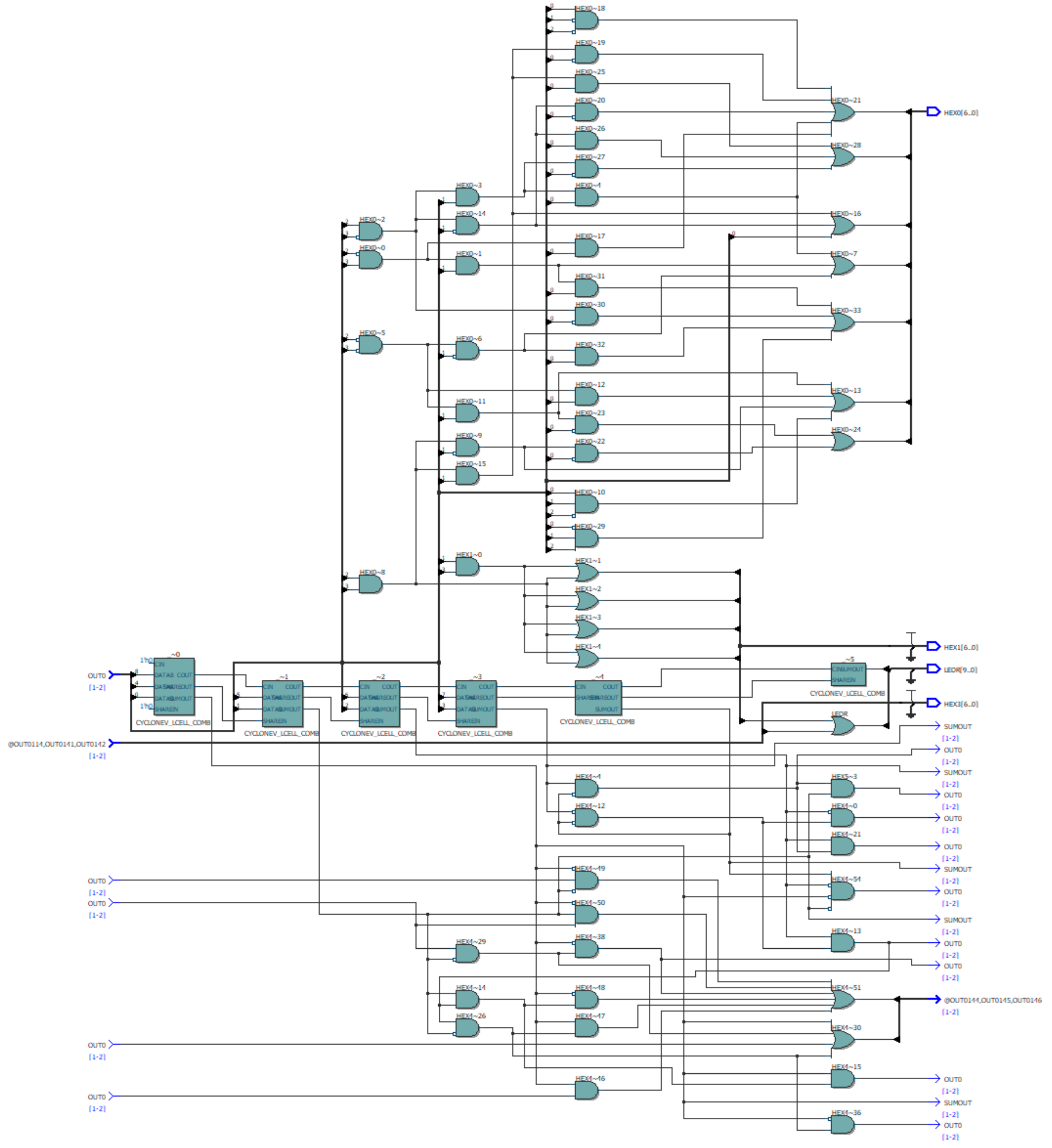

DE1-SoC開發板
編號：64
190910
Download DE1-SoC CD from
http://de1-soc.terasic.com

DE1-SoC開發板
編號：64
110110
Download DE1-SoC CD from
http://de1-soc.terasic.com

DE1-SoC開發板
編號：64
180909
Download DE1-SoC CD from
http://de1-soc.terasic.com


DE1-SoC開發板
編號：64
170908
Download DE1-SoC CD from
http://de1-soc.terasic.com


DE1-SoC開發板
編號：64
191108
Download DE1-SoC CD from
http://de1-soc.terasic.com

RTL佈局


HEX4[6..0]
HEX5[6..0]
HEX2[6..0]
SW[8..0]
SUMOUT
OUT0


HEX0[6..0]
HEX1[6..0]
LEDR[9..0]
HEX3[6..0]
CYCLONEV_LCELL_COMB
@OUT0114,OUT0141,OUT0142
@OUT0144,OUT0145,OUT0146
SUMOUT
OUT0

問題與討論

討論 ： 針對本次LAB，首先我是透過寫出真值表的方式，將七段顯示器的哪一個bit要亮先準備好，在一次用卡諾圖慢慢化簡，可稱為是土法煉鋼，因為此次LAB禁止使用判斷語句，所以只好一個一個慢慢做，一定有同學的辦法更好，但目前我能想到的只有這樣。

問題 ： 在燒進開發板時做驗證時，不乏會有一些小問題，如該顯示的未顯示，7段顯示器缺一腳等等問題，而我自己Debug的手段就是，針對出問題的Bit去看有沒有哪邊少加not等等，以及在重新化簡一次。