題目：Time-of-day clock

1. 實驗程式碼

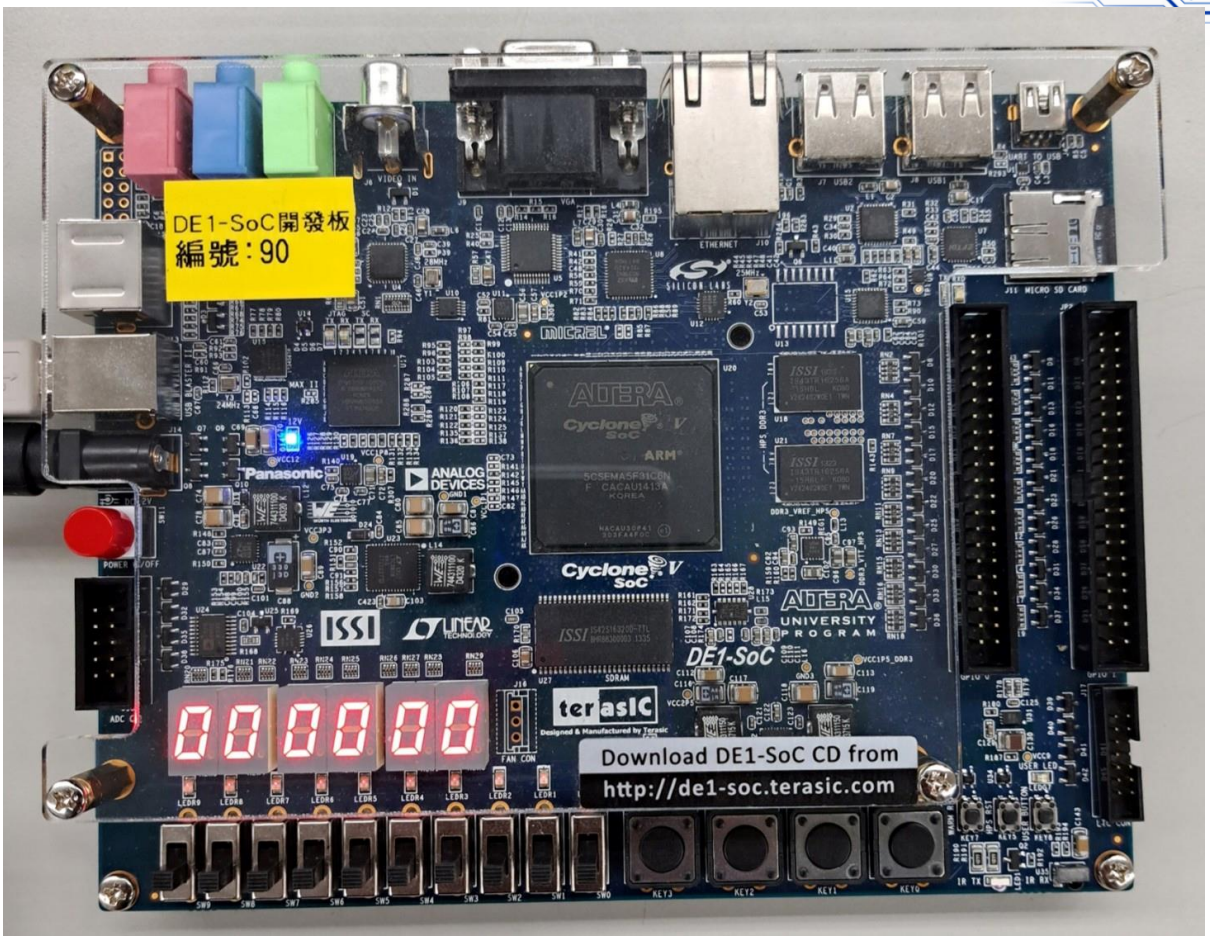
```
module LAB2 (SW, CLOCK_50, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5);
   input [9:0] SW;
   input CLOCK_50;
   output [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
   wire [5:0] sec;
   wire [7:0] hour, min;

   timer Timer (CLOCK_50, SW[8], SW[9], SW[7:0], hour, min, sec);

   two_digit_seg sec_display(sec, HEX1, HEX0);
   two_digit_seg min_display(min, HEX3, HEX2);
   two_digit_seg hour_display(hour, HEX5, HEX4);
endmodule

module timer (clk, set, select, setting_time, hour, min, sec);
   input clk, set, select;
   input [7:0] setting_time;
   output [7:0] hour, min;
   output [5:0] sec;
   reg [7:0] hour, min;
   reg [5:0] sec;
   reg [25:0] sec_count;
   integer flag = 0; //for the purpose of posedge of set
```

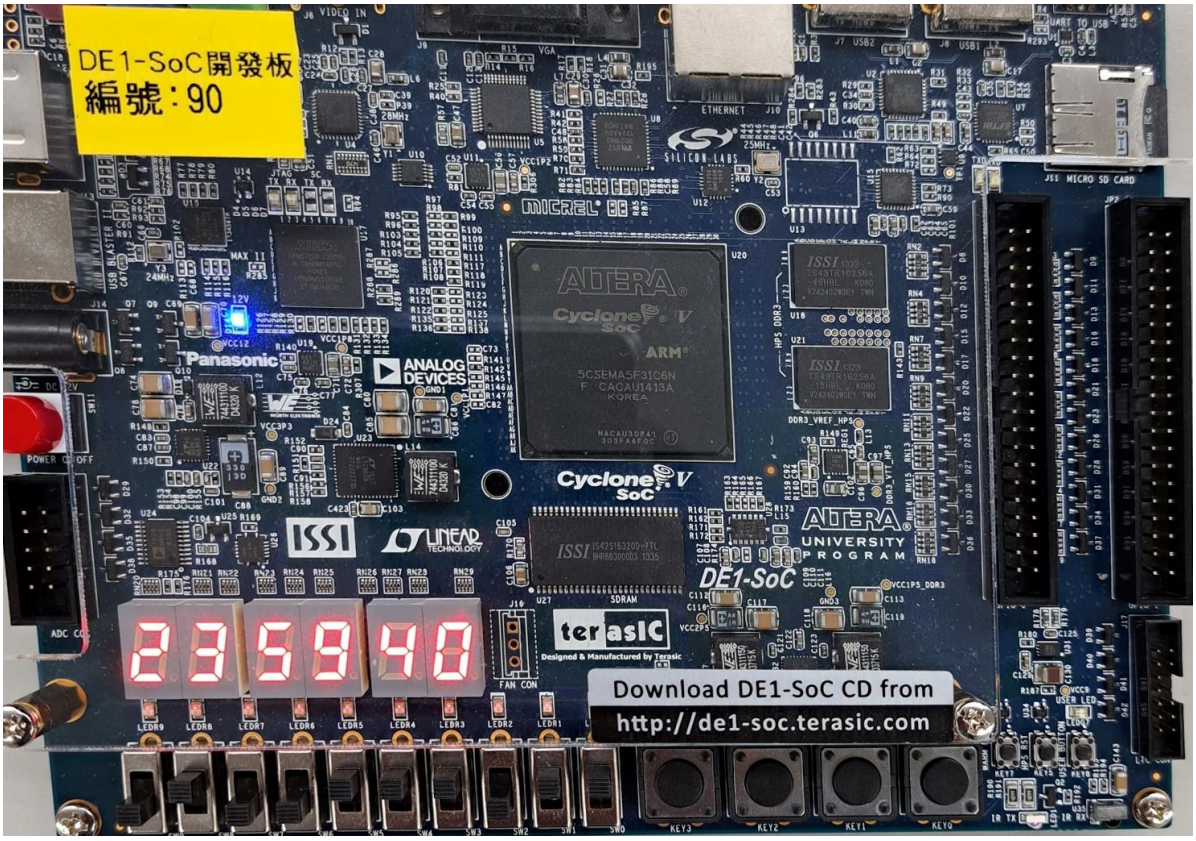
```
always @(posedge clk) begin
   //clock 00:00:00 ~ 23:59:59
   if(sec_count >= 50000000) begin
      sec_count = 0;
      sec = sec + 1;
      if(sec >= 60) begin
         sec = 0;
         min = min + 1;
         if(min >= 60) begin
            min = 0;
            hour = hour + 1;
            if(hour >= 24) begin
               hour = 0;
               min = 0;
               sec = 0;
            end
         end
      end
   end
   else begin
      sec_count = sec_count + 1;
   end
```

```
        //set hour or min
        if(set == 1) begin
            if(flag == 0) begin
                flag = 1;
                if(select == 1) begin
                    //setting hour time must be less than 24
                    if(setting_time > 23)
                        hour = 23;
                    else begin
                        hour = setting_time;
                    end
                end
                else begin
                    //setting minute time must be less than 60
                    if(setting_time > 59)
                        min = 59;
                    else begin
                        min = setting_time;
                    end
                end
            end
        end
        else begin
            flag = 0;
        end
    end
endmodule
```

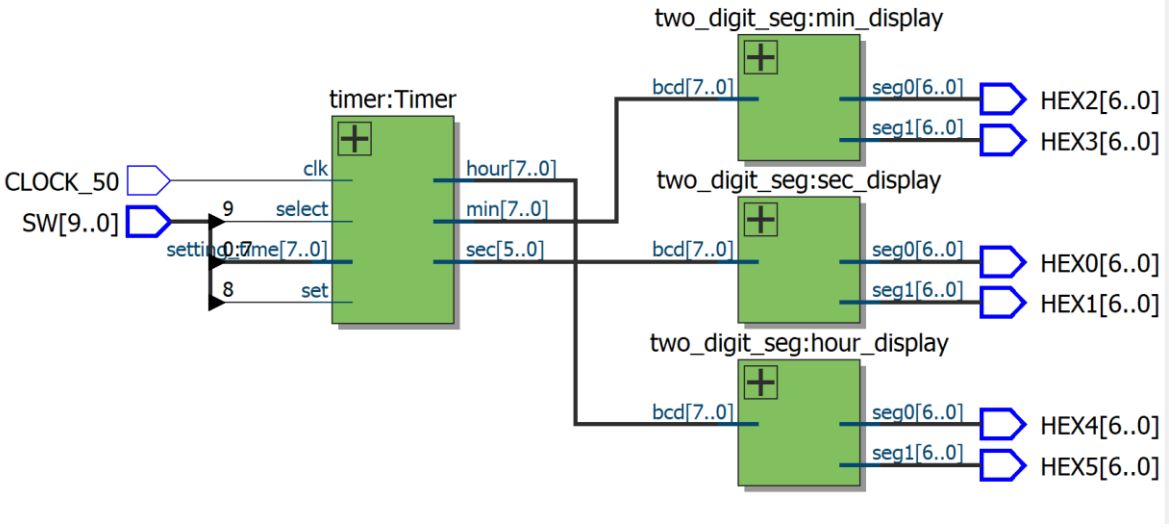
```
//bcd to seg
module seg_decoder (bcd, seg);
   input [7:0] bcd;
   output [6:0] seg;
   reg [6:0] seg;

   always @(bcd) begin
      case (bcd)
         0 : seg = 7'b1000000;
         1 : seg = 7'b1111001;
         2 : seg = 7'b0100100;
         3 : seg = 7'b0110000;
         4 : seg = 7'b0011001;
         5 : seg = 7'b0010010;
         6 : seg = 7'b0000010;
         7 : seg = 7'b1111000;
         8 : seg = 7'b0000000;
         9 : seg = 7'b0010000;
         //lights out
         default : seg = 7'b1111111;
      endcase
   end
endmodule

//display two digit in seg
module two_digit_seg (bcd, seg1, seg0);
   input [7:0] bcd;
   output [6:0] seg1, seg0;

   seg_decoder de1(bcd/10, seg1);
   seg_decoder de0(bcd%10, seg0);
endmodule
```

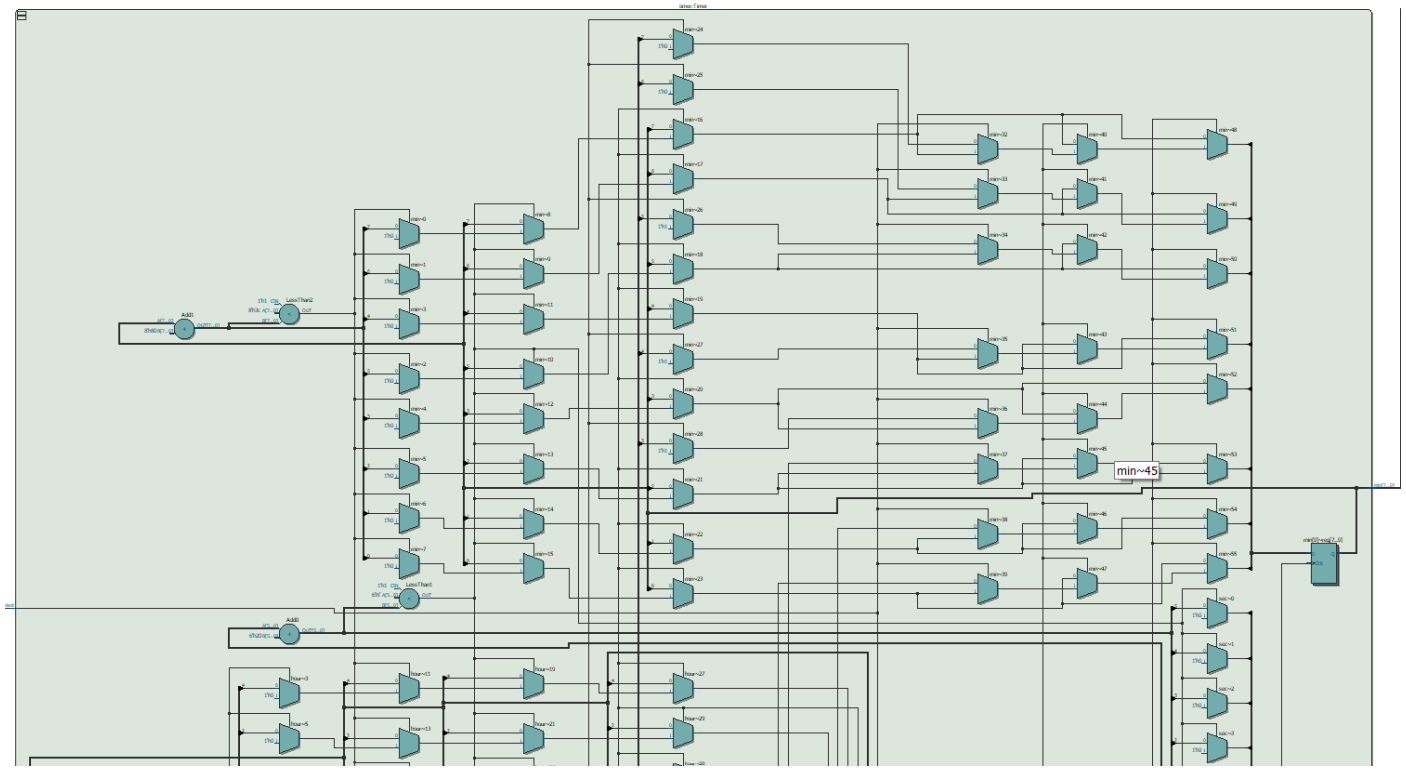
2. 實驗結果

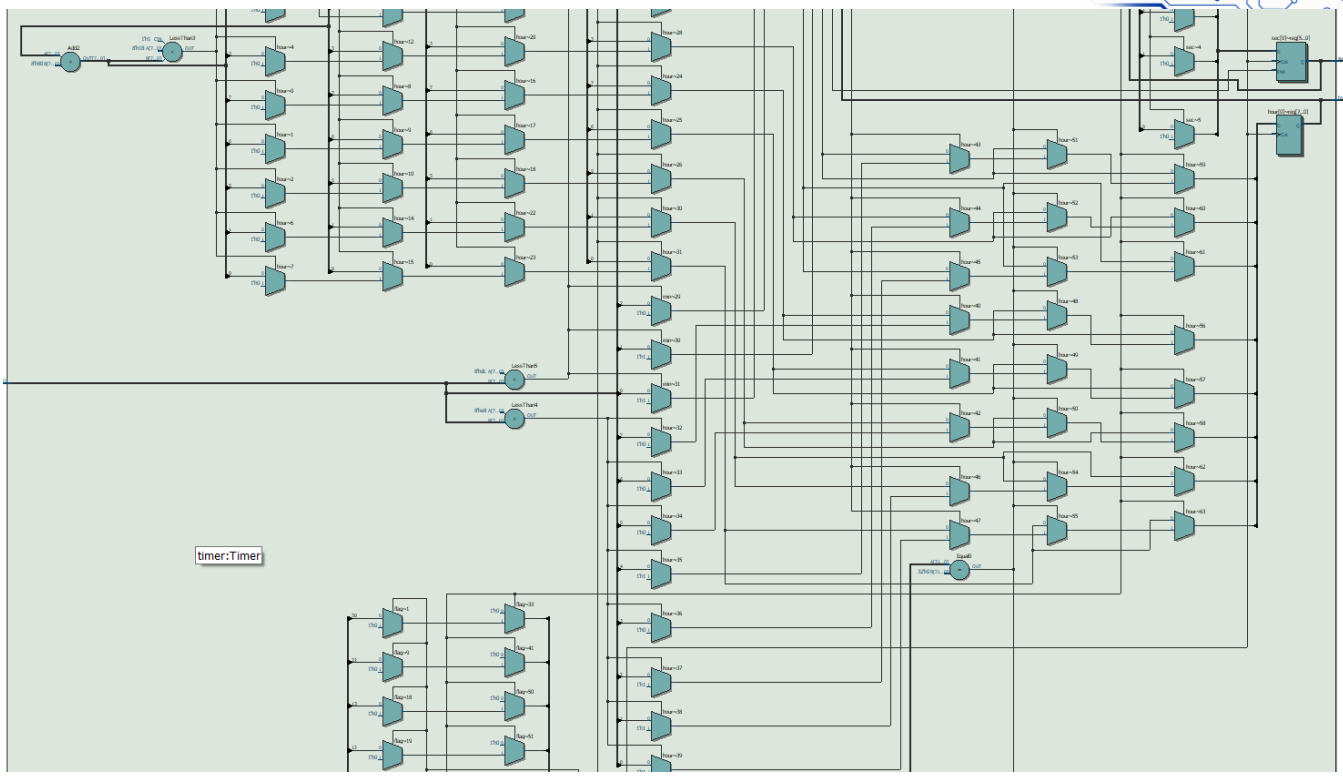

DE1-SoC開發板
編號：90
Download DE1-SoC CD from
http://de1-soc.terasic.com

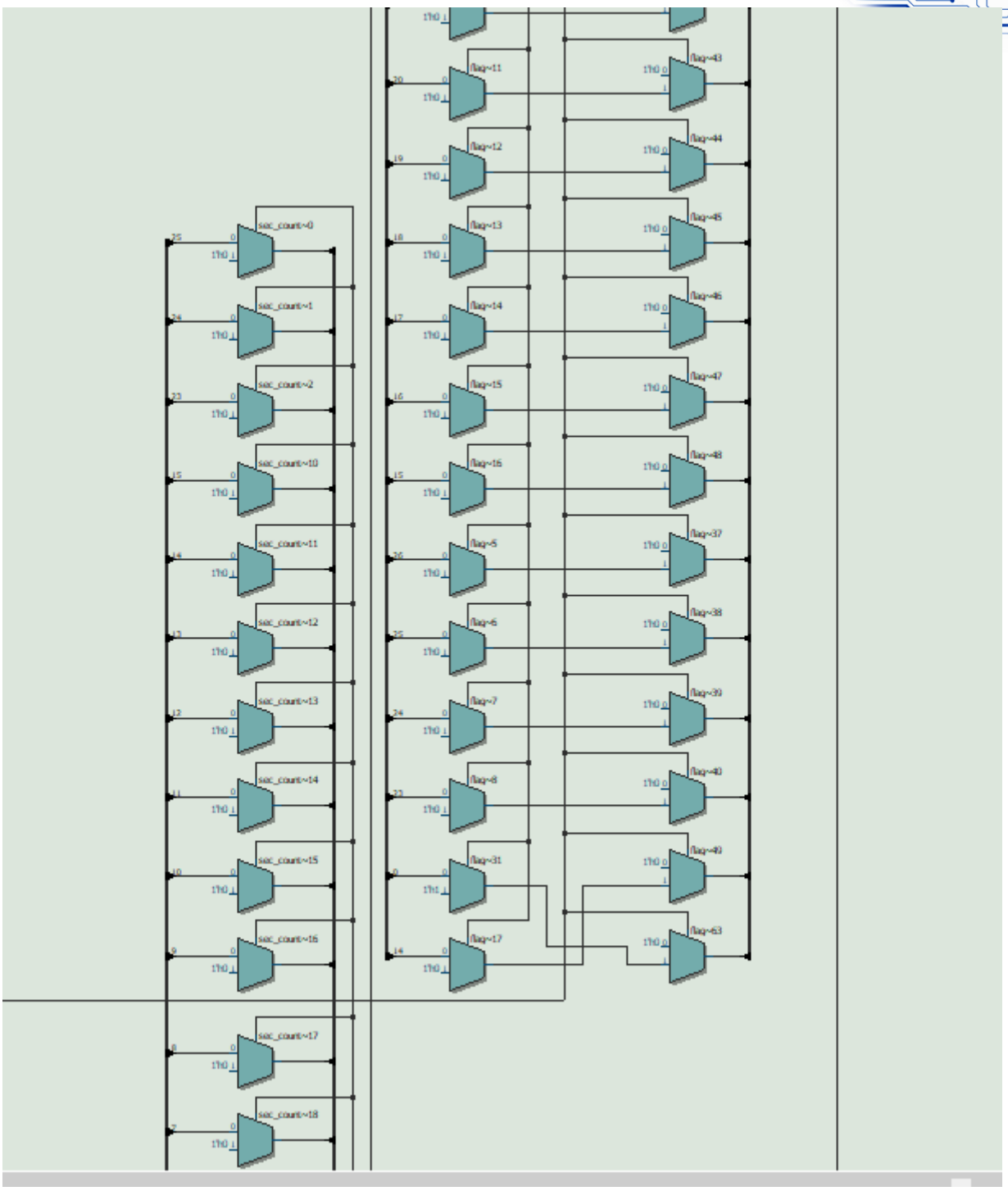

DE1-SoC開發板
編號：90
Download DE1-SoC CD from
http://de1-soc.terasic.com

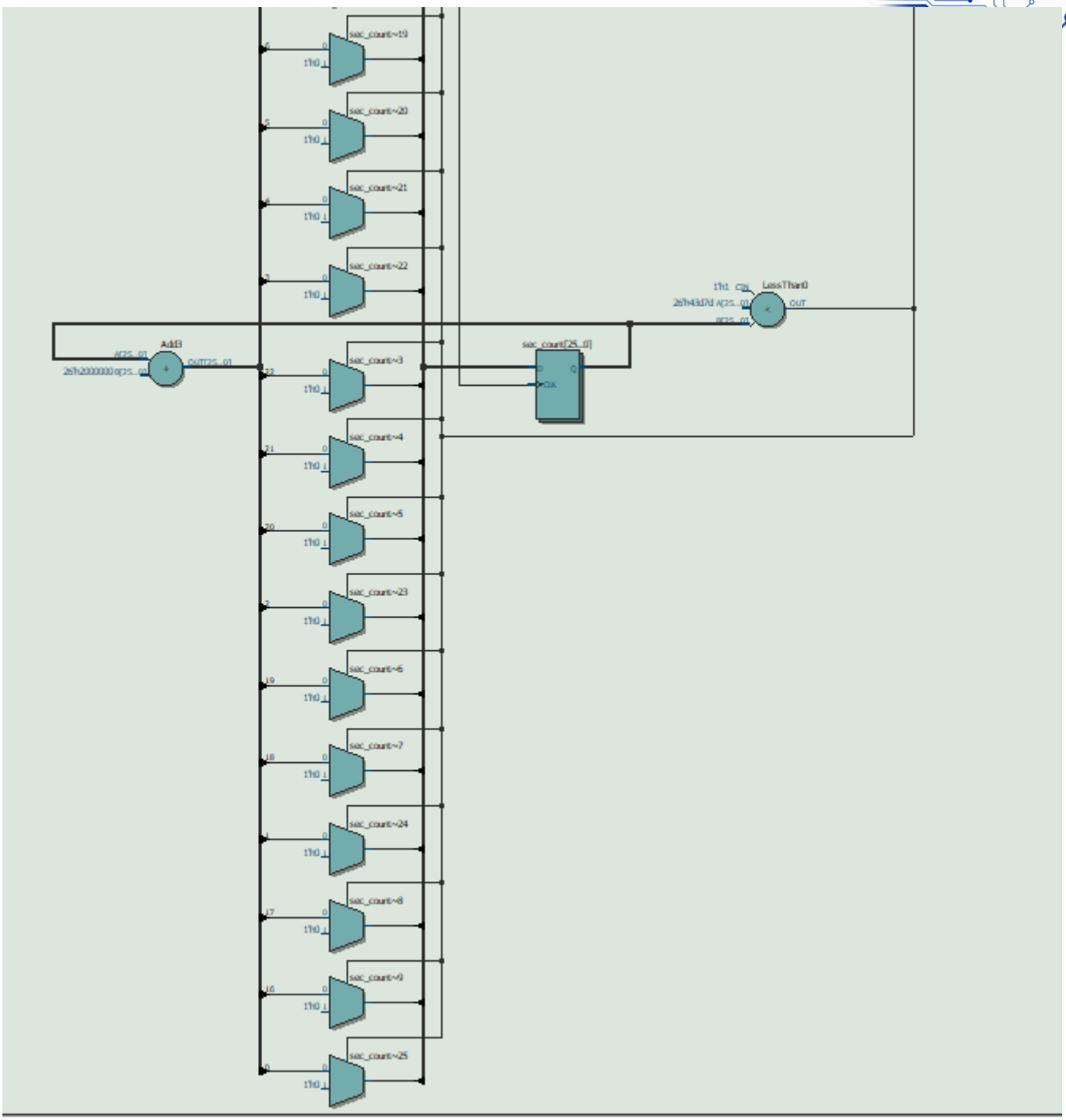
3. **RTL 佈局**


two_digit_seg:min_display
timer:Timer
bcd[7..0]
seg0[6..0]
seg1[6..0]
HEX2[6..0]
HEX3[6..0]
CLOCK_50
SW[9..0]
clk
9
select
setting_time[7..0]
8
set
hour[7..0]
min[7..0]
sec[5..0]
two_digit_seg:sec_display
HEX0[6..0]
HEX1[6..0]
two_digit_seg:hour_display
HEX4[6..0]
HEX5[6..0]

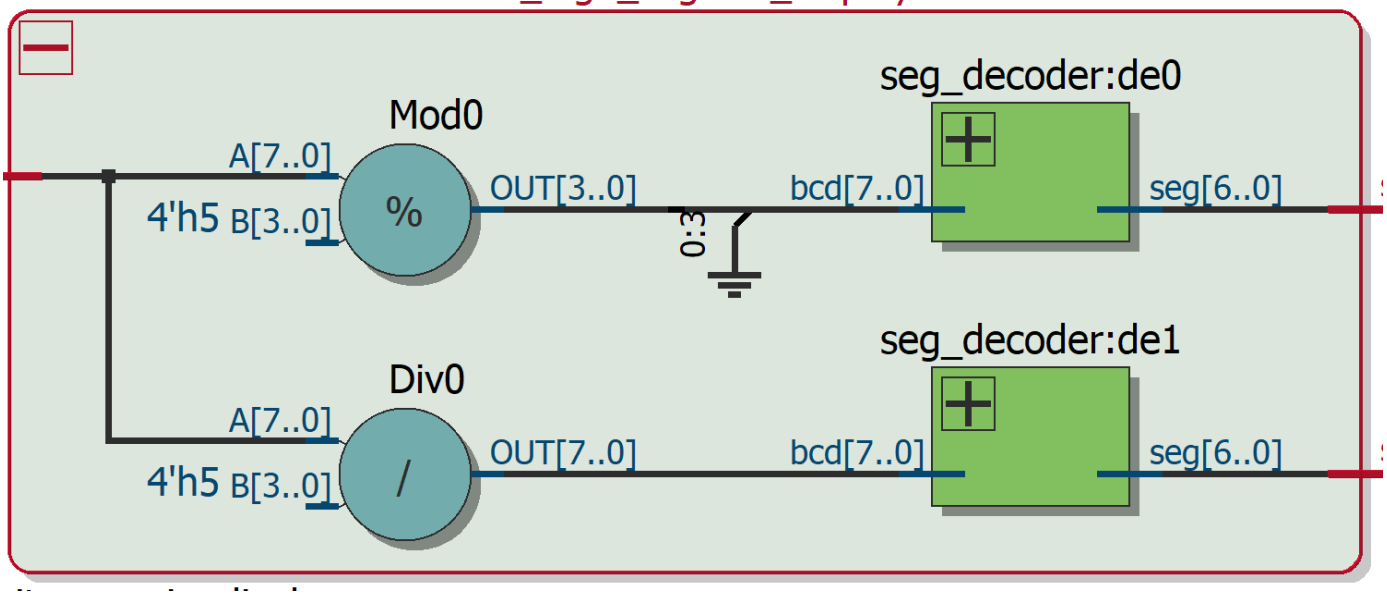

min~45

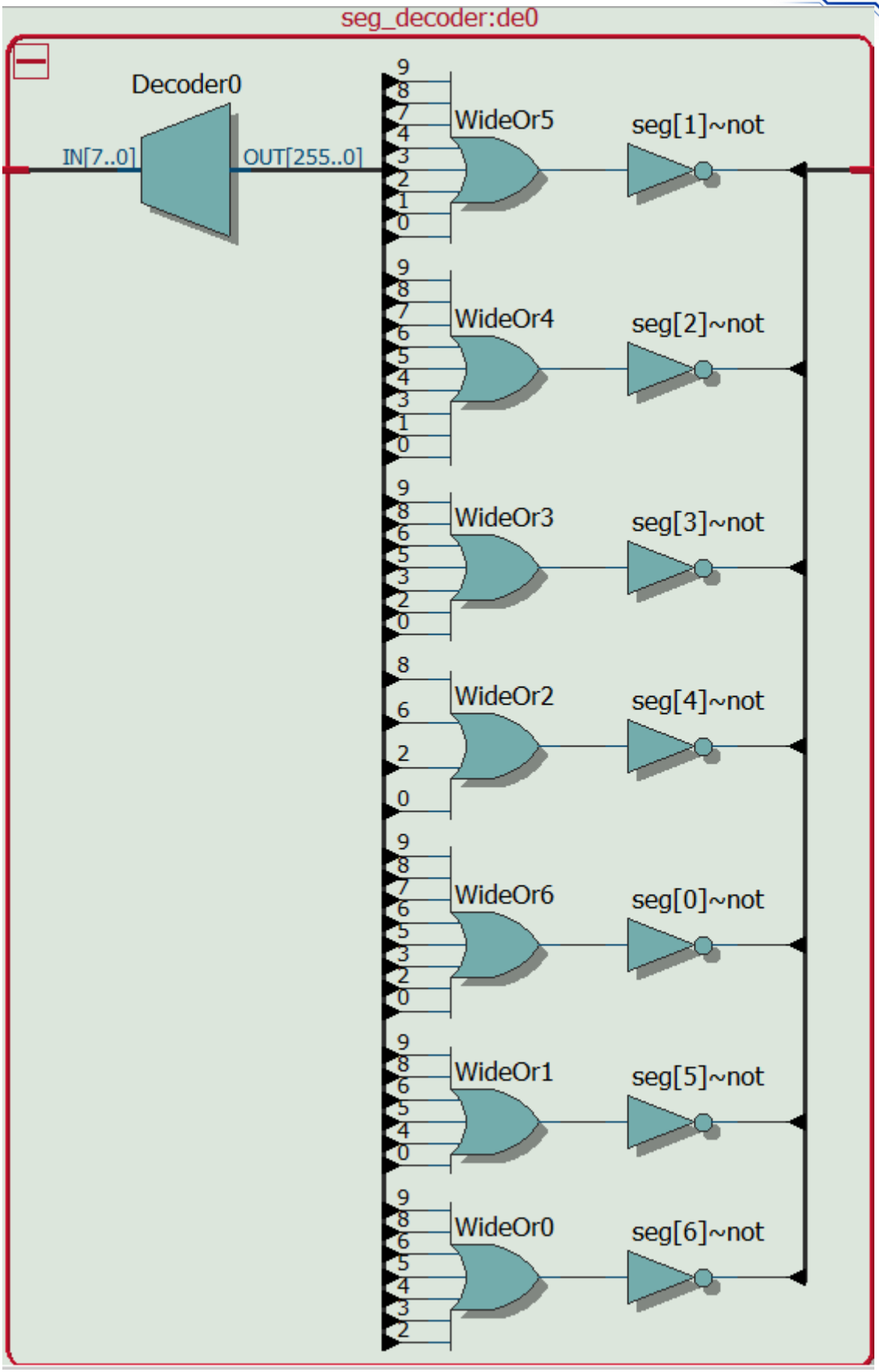

timer:Timer


sac_count~0
sac_count~1
sac_count~2
sac_count~10
sac_count~11
sac_count~12
sac_count~13
sac_count~14
sac_count~15
sac_count~16
sac_count~17
sac_count~18
flag~11
flag~12
flag~13
flag~14
flag~15
flag~16
flag~5
flag~6
flag~7
flag~8
flag~31
flag~17
flag~43
flag~44
flag~45
flag~46
flag~47
flag~48
flag~37
flag~38
flag~39
flag~40
flag~49
flag~63


sec_count~19
sec_count~20
sec_count~21
sec_count~22
sec_count~3
sec_count~4
sec_count~5
sec_count~23
sec_count~6
sec_count~7
sec_count~24
sec_count~8
sec_count~9
sec_count~25
Add3
LessThan0
sec_count[25..0]
CLK
OUT

two_digit_seg:sec_display





4. **問題與討論**

- 不能在不同的 always block 對相同變數賦值。

- always block 的敏感列表如果有兩個 edge 訊號，其中一個必須寫在 if 判斷式裡面，而且必須針對 pos 或是 neg 去做匹配(neg 要加~，pos 不用加)。如果是三個依此類推，並用 if else_if else。沒寫在條件判斷式的訊號就是 clock，剩餘的都是非同步訊號。
- 當有兩個非同步訊號輸入都有效時，就會有優先級的問題，擺在 if 的那個最優先。
- 通常非同步訊號都用作 set 或 reset 用途，reset 的功能為讓電路訊號通電後進入一個穩定狀態，以免進入隨機狀態而無法正常工作。
- integer 為 32 位元的有號數，用來當作 for loop 計數，for loop 可以複製電路以精簡程式碼，例如 32 位元加法器可以用一行加法器程式碼配合迴圈 32 次完成，效果是等價。除此之外，不要使用 integer，要用 wire 或 reg。