# 第二次實驗報告

題目：

**Time-of-day clock**

姓名：黃子軒

學號：**310581003**

繳交日期：**111** 年 **10** 月 **24** 日

## 1. 實驗目的

透過本次實驗，除了熟悉如何撰寫 verilog 並實現於 FPGA 板，更學習一些常見 Sequential circuits (latches, flip-flops, registers) 與 Counters 的架構及運作原理，並嘗試結合其電路實現一個具有小時、分鐘、秒的計時時鐘。

## 2. 實驗程式碼

```
module Lab2_part7(CLOCK_50, HEX0, HEX1, HEX2, HEX3
                                    , HEX4, HEX5, SW[9:0]);
    input CLOCK_50;
    input [9:0]SW;
    output [6:0]HEX0;
    output [6:0]HEX1;
    output [6:0]HEX2;
    output [6:0]HEX3;
    output [6:0]HEX4;
    output [6:0]HEX5;
    wire clock_out;
    wire [3:0] Q0;
    wire [3:0] Q1;
    wire [3:0] Q2;
    wire [3:0] Q3;
    wire [3:0] Q4;
    wire [3:0] Q5;
    div clock_div(clock_out, CLOCK_50);

    counter0 CTR0(clock_out, carry01, Q0, Q1);
    seven_segments s0(clock_out, Q0, HEX0);
    seven_segments s1(clock_out, Q1, HEX1);
```

```
    counter1 CTR1(clock_out, carry01, SW[8], SW[9], SW[7:0], carry12, Q2, Q3);
    seven_segment s2(clock_out, Q2, HEX2,SW[8]);
    seven_segment s3(clock_out, Q3, HEX3,SW[8]);

    counter2 CTR2(clock_out, carry01, SW[8], SW[9], SW[7:0], carry12, Q4, Q5);
    seven_segment s4(clock_out, Q4, HEX4,SW[8]);
    seven_segment s5(clock_out, Q5, HEX5,SW[8]);
endmodule

// Second //
module counter0(Clk, carry01, Q0, Q1);
    input Clk;
    output reg carry01;
    output reg [3:0] Q0;
    output reg [3:0] Q1;
    always @(posedge Clk)
    begin
        if (Q0==4'd9 & Q1!=4'd5)
            begin
            Q0<=0;
            Q1<=Q1+1;
            carry01<=0;
            end
        else if (Q0==4'd8& Q1!=4'd5)
            begin
```

```
            Q0<=Q0+1;
            carry01<=0;
            end
        else if (Q0==4'd9& Q1==4'd5)
            begin
            Q0<=0;
            Q1<=0;
            carry01<=1;
            end
        else if (Q0==4'd8& Q1==4'd5)
            begin
            Q0<=Q0+1;
            carry01<=0;
            end
        else
        begin
            Q0<=Q0+1;
            carry01<=0;
            end
    end
endmodule

// Minute //
module counter1(Clk, carry01, set, select, SW[7:0], carry12, Q2, Q3);
    input Clk, carry01, set, select;
    input [7:0]SW;
```

```
reg en;
output reg carry12;
output reg [3:0] Q2;
output reg [3:0] Q3;

always @(posedge set or posedge carry01)
begin
    if (carry01==1)
        en<=0;
    else if (set==1)
        en<=1&~select;
    else
        en<=0;
end

always @(posedge Clk)
begin
if(set)
begin
    if (Q2==4'd9 & Q3!=4'd5 & carry01==1)
        begin
        Q2<=0;
        Q3<=Q3+1;
        carry12<=0;
        end
    else if (Q2==4'd8& Q3!=4'd5 & carry01==1)
```

```
        begin
        Q2<=Q2+1;
        carry12<=0;
        end
else if (Q2==4'd9& Q3==4'd5 & carry01==1)
        begin
        Q2<=0;
        Q3<=0;
        carry12<=0;
        end
else if (Q2==4'd9& Q3==4'd5 & carry01==0)
        begin
        carry12<=1;
        end
else if (Q2==4'd8& Q3==4'd5)
        begin
        Q2<=Q2+1;
        carry12<=0;
        end
else if (carry01==1)
begin
        Q2<=Q2+1;
        carry12<=0;
end
else if(en==1)
begin
```

```
            Q2<=SW[3:0];
            Q3<=SW[7:4];
        end
end
else
begin
        if (Q2==4'd9 & Q3!=4'd5 & carry01==1)
            begin
            Q2<=0;
            Q3<=Q3+1;
            carry12<=0;
            end
        else if (Q2==4'd8& Q3!=4'd5 & carry01==1)
            begin
            Q2<=Q2+1;
            carry12<=0;
            end
        else if (Q2==4'd9& Q3==4'd5 & carry01==1)
            begin
            Q2<=0;
            Q3<=0;
            carry12<=0;
            end
        else if (Q2==4'd9& Q3==4'd5 & carry01==0)
            begin
            carry12<=1;
```

```
            end
        else if (Q2==4'd8& Q3==4'd5)
            begin
            Q2<=Q2+1;
            carry12<=0;
            end
        else if (carry01==1)
        begin
            Q2<=Q2+1;
            carry12<=0;
        end
        end
    end
endmodule

// Hour //
module counter2(Clk, carry01, set, select, SW[7:0], carry12, Q4, Q5);
    input Clk, carry01, carry12,set,select;
    input [7:0] SW;
    output reg [3:0] Q4;
    output reg [3:0] Q5;
    reg en;

    always @(posedge set or posedge carry12)
    begin
        if (carry12==1)
```

```
        en<=0;
    else if (set==1)
        en<=1&select;
    else
        en<=0;
end

always @(posedge Clk)
begin
if(set)
begin
    if (Q4==4'd3 & Q5==4'd2 & carry12==1 & carry01==1)
        begin
        Q4<=0;
        Q5<=0;
        end
    else if (Q4==4'd9 & carry12==1    & carry01==1)
        begin
        Q4<=0;
        Q5<=Q5+1;
        end
    else if (Q4==4'd8 & carry12==1 & carry01==1)
        begin
        Q4<=Q4+1;
        end
    else if (carry12==1 & carry01==1)
```

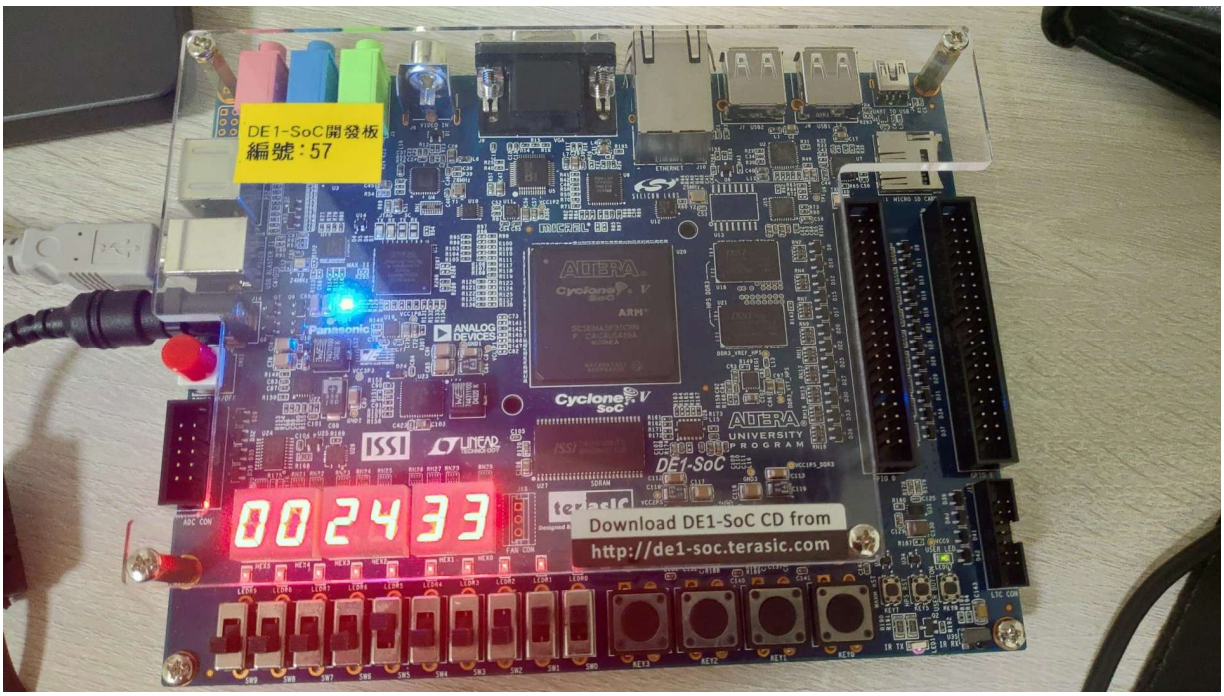
```
        begin
            Q4<=Q4+1;
        end
        else if(en==1)
        begin
            Q4<=SW[3:0];
            Q5<=SW[7:4];
        end
end
else
begin
            if (Q4==4'd3 & Q5==4'd2 & carry12==1 & carry01==1)
            begin
            Q4<=0;
            Q5<=0;
            end
        else if (Q4==4'd9 & carry12==1     & carry01==1)
            begin
            Q4<=0;
            Q5<=Q5+1;
            end
        else if (Q4==4'd8 & carry12==1 & carry01==1)
            begin
            Q4<=Q4+1;
            end
        else if (carry12==1 & carry01==1)
```

```
            begin
                Q4<=Q4+1;
            end
            end
        end
endmodule

// Seven segment display //
module seven_segment(Clk, in, HEX0,set);
        output reg [6:0] HEX0;
        input [3:0] in;
        input Clk,set;
        always @(posedge Clk, posedge set)
        begin
          HEX0[0] <= (~in[3]&~in[2]&~in[1]&in[0])|(in[2]&~in[1]&~in[0])|(in[3]&in[2])|(in[3]&in[1]);
          HEX0[1] <= (in[2]&~in[1]&in[0])|(in[2]&in[1]&~in[0])|(in[3]&in[2])|(in[3]&in[1]);
          HEX0[2] <= (in[3]&in[2])|(in[3]&in[1])|(~in[2]&in[1]&~in[0]);
          HEX0[3] <=
(in[2]&~in[1]&~in[0])|(in[2]&in[1]&in[0])|(in[3]&in[1])|(~in[3]&~in[2]&~in[1]&in[0])|(in[3]&in[2])|(in[3]
&in[0]);
          HEX0[4] <= (in[2]&~in[1])|(in[3]&in[1])|(~in[1]&in[0])|(in[1]&in[0]);
          HEX0[5] <= (in[3]&in[2])|(~in[2]&in[1])|(~in[3]&~in[2]&in[0])|(in[1]&in[0]);
          HEX0[6] <= (~in[3]&~in[2]&~in[1])|(in[3]&in[2])|(in[2]&in[1]&in[0])|(in[3]&in[1]);
        end
endmodule

module seven_segments(Clk, in, HEX0);
```

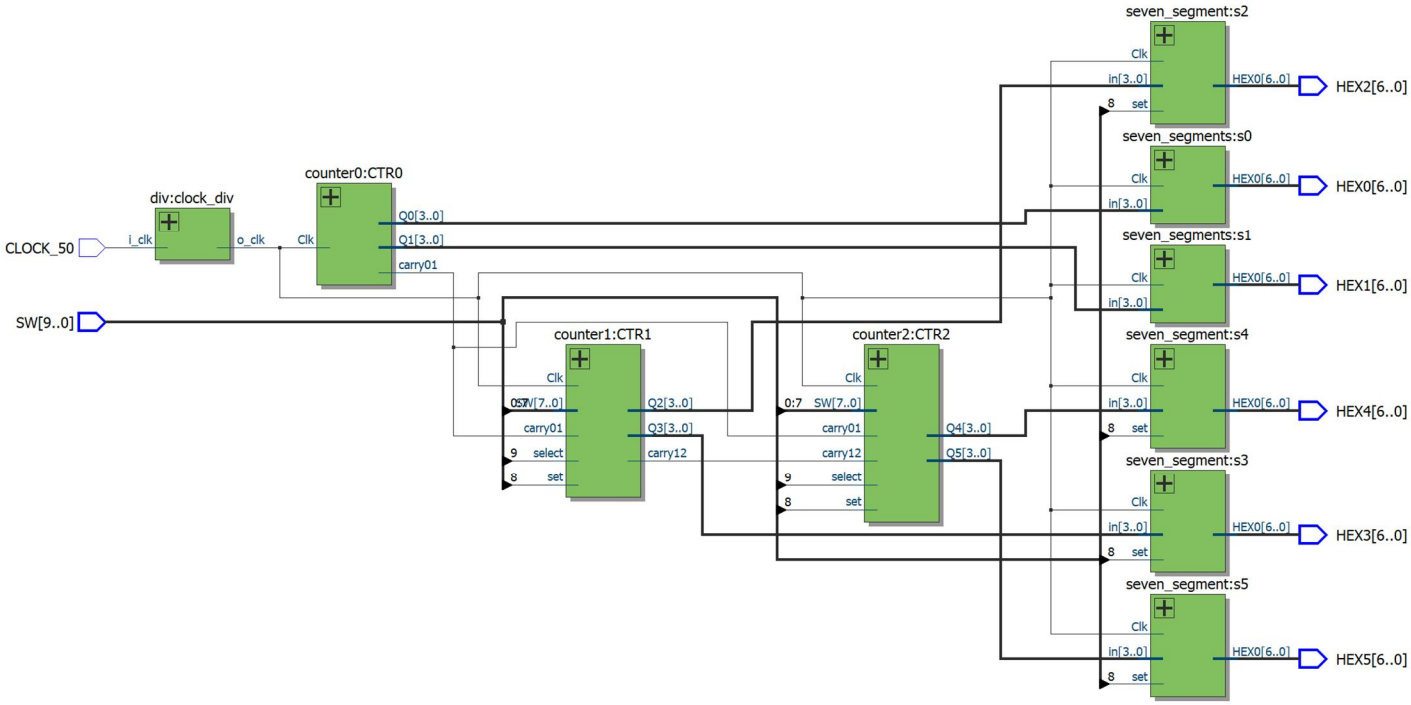
```
        output reg [6:0] HEX0;
        input [3:0] in;
        input Clk;
        always @(posedge Clk)
        begin
          HEX0[0] <= (~in[3]&~in[2]&~in[1]&in[0])|(in[2]&~in[1]&~in[0])|(in[3]&in[2])|(in[3]&in[1]);
          HEX0[1] <= (in[2]&~in[1]&in[0])|(in[2]&in[1]&~in[0])|(in[3]&in[2])|(in[3]&in[1]);
          HEX0[2] <= (in[3]&in[2])|(in[3]&in[1])|(~in[2]&in[1]&~in[0]);
          HEX0[3] <=
(in[2]&~in[1]&~in[0])|(in[2]&in[1]&in[0])|(in[3]&in[1])|(~in[3]&~in[2]&~in[1]&in[0])|(in[3]&in[2])|(in[3]
&in[0]);
          HEX0[4] <= (in[2]&~in[1])|(in[3]&in[1])|(~in[1]&in[0])|(in[1]&in[0]);
          HEX0[5] <= (in[3]&in[2])|(~in[2]&in[1])|(~in[3]&~in[2]&in[0])|(in[1]&in[0]);
          HEX0[6] <= (~in[3]&~in[2]&~in[1])|(in[3]&in[2])|(in[2]&in[1]&in[0])|(in[3]&in[1]);
        end
endmodule

// Divider //
module div(
output reg o_clk,
input i_clk
  );

        parameter N=50_000_000;
        parameter M=24_999_999;

        reg [25:0]cnt;
```

```
always @(posedge i_clk)
begin
if(cnt==N-1)
cnt<=26'b0;
else
cnt<=cnt+26'b1;
end
always @(posedge i_clk)
begin
if(cnt<=M)
o_clk<=1;
else
o_clk<=0;
end
endmodule
```

### 3. 實驗結果照片 (optional)



## 4. RTL 佈局(optional)




## 5.問題與討論

本次實驗開始學習如何使用 always 語法，不僅從中了解 Blocking 和 Non-blocking 的差別，並了解到許多硬體描述語法的細節與特別注意的地方，例如：中間節點要宣告為 wire 還是 reg。另外，本次實驗更學習到 sequential circuit 與 clock 的實現。