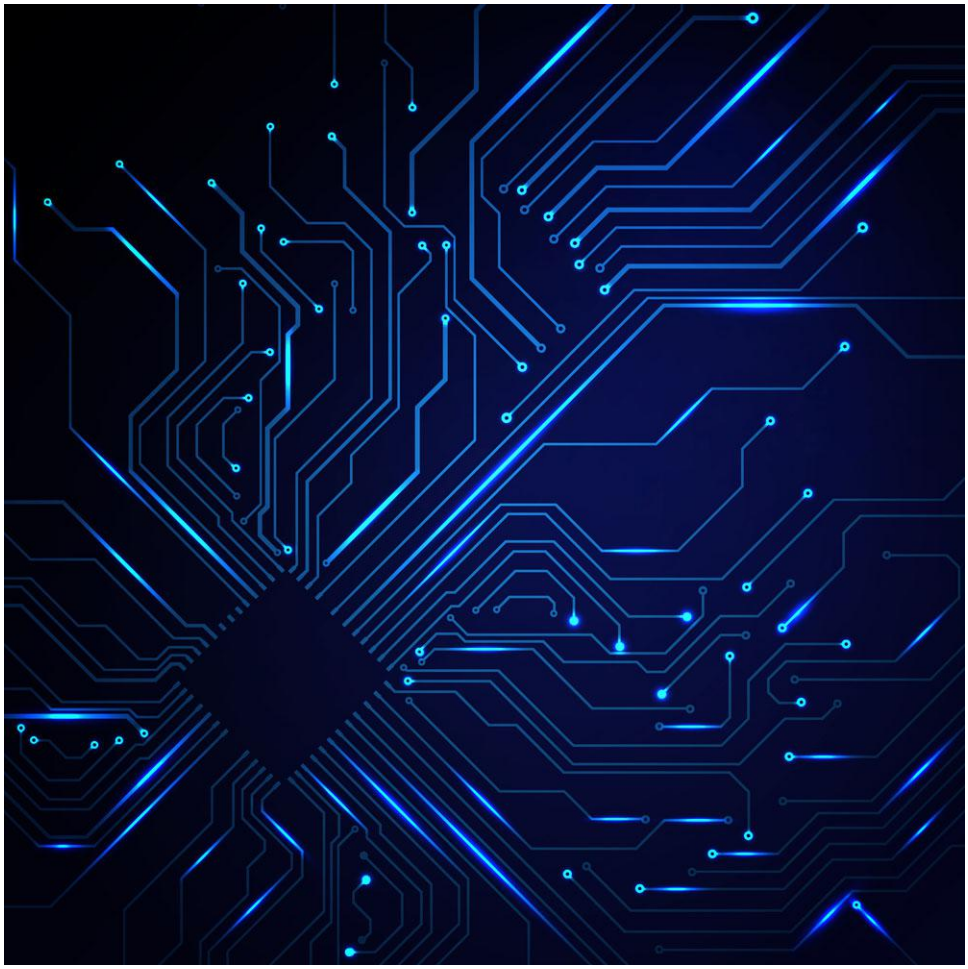


# 2022/09/28

# 實驗三

# 序向邏輯練習

姓名：王嘉羽　　學號：00957116

班級：資工 2b

E-mail：vayne20011125@gmail.com

## 注意

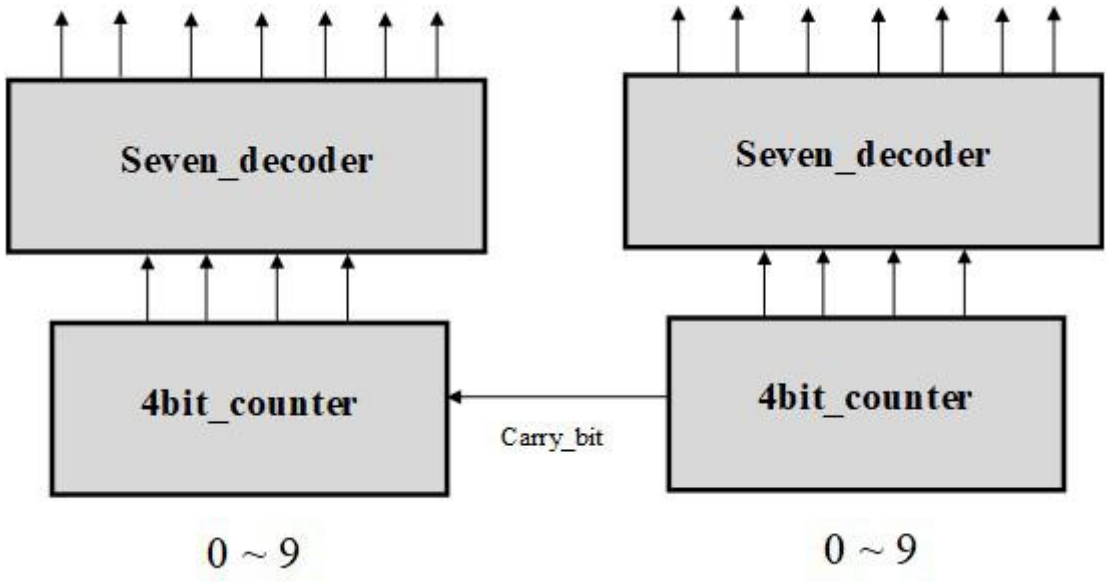
1. 繳交時一律轉 PDF 檔
2. 繳交期限為
   上完課後
   當週五晚上 12 點前
3. 一人繳交一份
4. 檔名：學號_HW?.pdf
   檔名請按照作業檔名格式進行填寫
   未依照格式不予批改

# 一、 BCD counter

## ● 實驗說明：

1. 使用兩個 4bit counter 接到七段顯示器，個位數數到 9 之後歸零並進位，十位數 counter+1。

## ● 系統硬體架構方塊圖（接線圖）：




## ● 系統架構程式碼、測試資料程式碼與程式碼說明

## 截圖請善用 win+shift+S

這是 0~9 的計數器

```
1  `timescale 1ns/10ps
2  module counter_4bit_0_9(
3      input reset,                //重置
4      input clk,                  //時脈
5      input carry_in,             //進位輸入
6      output logic [3:0] value,   //輸出
7      output logic carry_out      //進位輸出
8  );
9
10     always_ff @ (posedge clk)
11        begin
12           if(reset)                 //同步，如果reset 就把值歸0
13              value <= #1 0;
14           else if(value == 4'd9 && carry_in == 1) //數到9也把值歸0
15              value <= #1 0;
16           else if(carry_in == 1)
17              value <= #1 value + 1; //值加1
18        end
19
20     assign carry_out = (value == 4'd9 && carry_in == 1)?1:0; //當value = 9 且 carry_in = 1時，有進位
21
22 endmodule
```

這是七段顯示器(和上禮拜的作業一樣)

```
module seven_segment(
    input [3:0] A,          //輸入
    output logic [6:0] y  //輸出
);
 always_comb
begin
    //當A改變時，y會對應到七段顯示器，右邊的註解表示他在7段顯示器上的效果
    case(A)
        4'h0 : y = 7'b1000000; //0
        4'h1 : y = 7'b1111001; //1
        4'h2 : y = 7'b0100100; //2
        4'h3 : y = 7'b0110000; //3
        4'h4 : y = 7'b0011001; //4
        4'h5 : y = 7'b0010010; //5
        4'h6 : y = 7'b0000010; //6
        4'h7 : y = 7'b1111000; //7
        4'h8 : y = 7'b0000000; //8
        4'h9 : y = 7'b0010000; //9
        4'hA : y = 7'b0001000; //A
        4'hB : y = 7'b0000011; //b
        4'hC : y = 7'b1000110; //C
        4'hD : y = 7'b0100001; //d
        4'hE : y = 7'b0000110; //E
        4'hF : y = 7'b0001110; //F
    endcase
end
 endmodule
```

BCD 計數器，我是把前面 2 個東西組在一起，然後 s 是 16 進位，對應到七段顯示器，比較好 debug

0000,0000 -> 00

0000,0001 -> 01

0000,0002 -> 02

.....

0000,1001 -> 09

0001,0000 -> 10 大概是這樣的概念

```
`timescale 1ns/10ps
module counter_BCD(
    input reset,                //重置
    input clk,                  //時脈
    output logic [13:0] q,      //輸出
    output logic [7:0] s        //輸出16進位，方便debug <ex>1001,0001 -> 91
);

logic carry_out1,carry_out2;

//計數器，負責個位數，每次都要+1所以carry_in的位置放1，得到的值存在s[3:0]裡，進位放在carry_out1
counter_4bit_0_9 counter_4bit_0_9_1(reset, clk,          1, s[3:0], carry_out1);

//計數器，負責十位數，只有個位數有進位時才要+1,所以carry_in的位置放carry_out1，得到的值存在s[7:4]裡，進位放在carry_out2
counter_4bit_0_9 counter_4bit_0_9_2(reset, clk, carry_out1, s[7:4], carry_out2);

//個位數的七段顯示器，將s[3:0]轉成七段顯示
seven_segment seven_segment1(s[3:0],q[6:0]);

//個位數的七段顯示器，將s[7:4]轉成七段顯示
seven_segment seven_segment2(s[7:4],q[13:7]);

endmodule

```

測試程式碼:

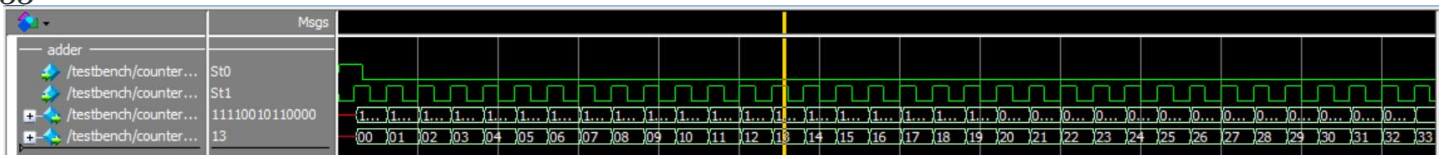
simulation > tb > testbench.sv

```
1   `timescale 1ns/10ps
2   module testbench;
3
4       logic reset;                    //重置
5       logic clk;                      //時脈
6       logic [13:0] q;                 //輸出
7       logic [7:0] s;                  //16進位數字 方便debug
8
9       counter_BCD counter_BCD1(
10          .reset(reset), //()內的變數為tb的變數，"."後面為counter_BCD1.sv的變數，將2者對應起來
11          .clk(clk),
12          .q(q),
13          .s(s)
14      );
15
16      always #10 clk = ~clk;
17
18      initial begin
19          reset = 1;clk = 0; //一開始先reset，將時脈歸0
20          #15 reset = 0;
21          #2500 $stop;
22      end
23  endmodule
```

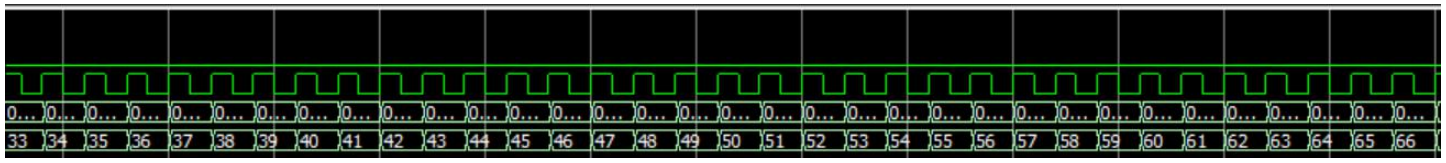
## 模擬結果與結果說明：

我放了 00~99 然後又變回 00 的 16 進位顯示，還有部分七段顯示器(七段顯示器全放可能會很長 qq)

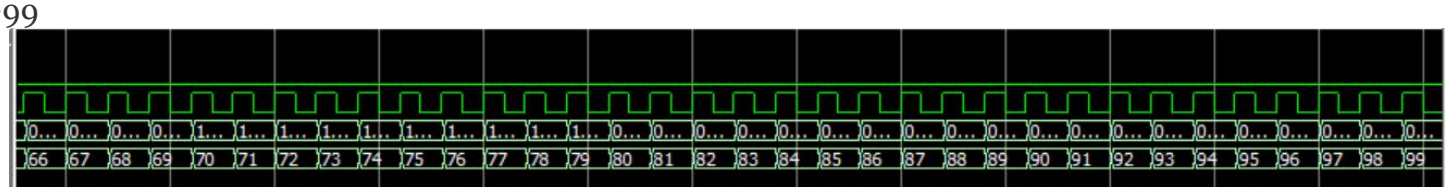
00~33



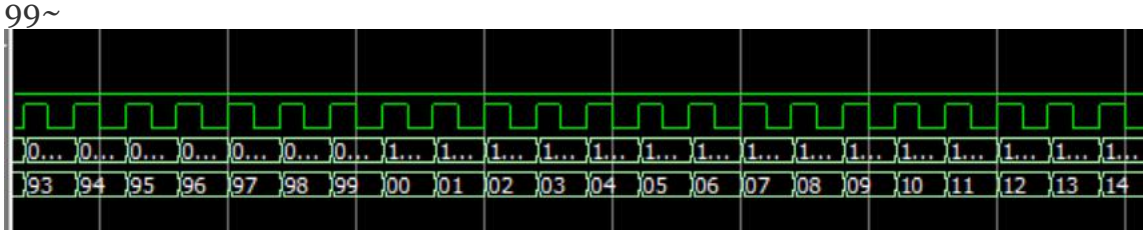

33~66



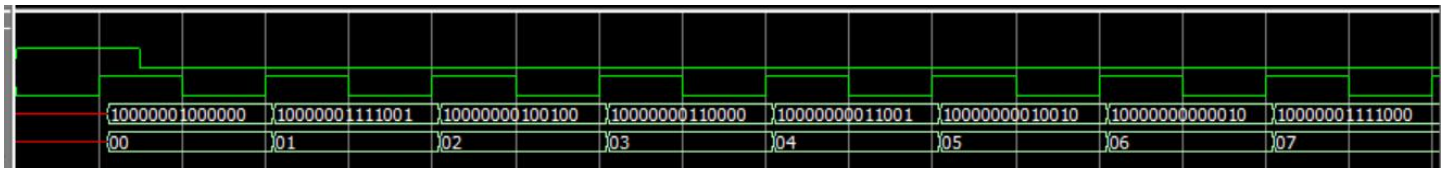
66~99



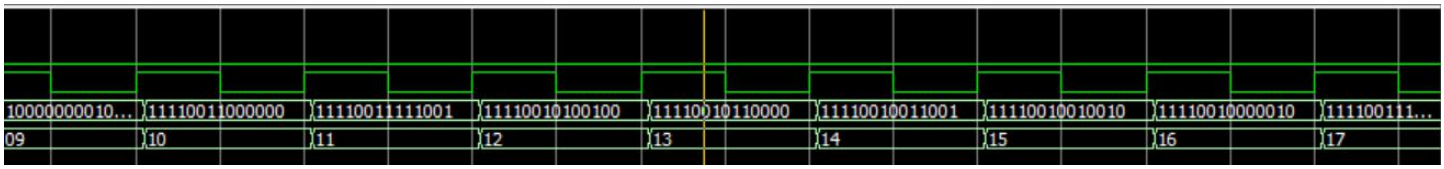
99~



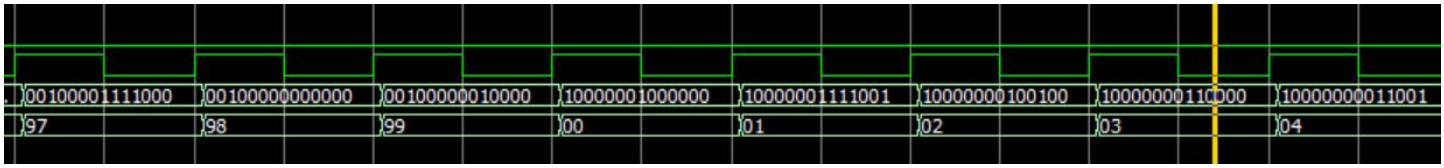
頭




進位的地方




尾

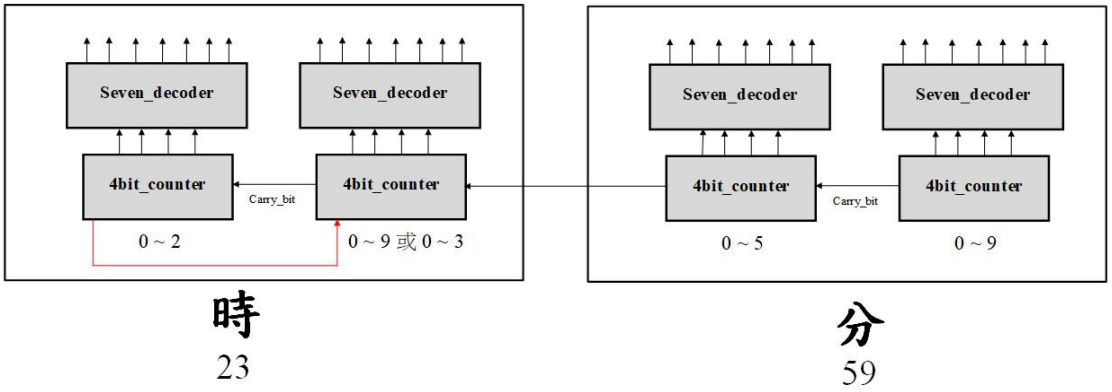



## 二、 時鐘

### ● 實驗說明：

1. 實作一個二十四時制的時鐘，分的部分數到 59 之後歸零進位。時的十位數部分數到 2 將紅色線變為 1，數到 23:59 之後歸零。

### ● 系統硬體架構方塊圖（接線圖）：




### ● 系統架構程式碼、測試資料程式碼與程式碼說明

### 截圖請善用 win+shift+S

總共有 7 個 sv 檔案

Clock (clock.sv)

- ✧ hour (counter_BCD_hour.sv)
  - ■ 0~9 and 0~3 計數器 (處理小時的個位數) (counter_4bit_0_9_0_3.sv)
  - ■ 0~2 計數器 (處理小時的十位數) (counter_4bit_0_2.sv)
- ✧ min (counter_BCD_min.sv)
  - ■ 0~9 計數器 (處理分鐘的個位數) (counter_4bit_0_9.sv)
  - ■ 0~5 計數器 (處理分鐘的十位數) (counter_4bit_0_5.sv)

由 2 個計數器合成分鐘或是小時，再將分鐘和小時合成時鐘。

clock.sv

```
`timescale 1ns/10ps
module clock(
    input reset,                //重置
    input clk,                  //時脈
    output logic [27:0] q,      //輸出
    output logic [15:0] s       //輸出16進位，方便debug
);

logic carry_out1,carry_out2;

//計數器，負責分鐘
counter_BCD_min counter_BCD_min1(reset, clk, q[13:0], s[7:0], carry_out1);

//計數器，負責小時
counter_BCD_hour counter_BCD_hour1(reset, clk, carry_out1, q[27:14], s[15:8]);

endmodule
```

counter_BCD_hour.sv

```
`timescale 1ns/10ps
module counter_BCD_hour(
    input reset,                //重置
    input clk,                  //時脈
    input carry_in,
    output logic [13:0] q,      //輸出
    output logic [7:0] s        //輸出16進位，方便debug
);

logic carry_out1,mode;

//計數器，負責個位數，得到的值存在s[3:0]裡，進位放在carry_out1，模式選擇控制是0~3還是0~9
counter_4bit_0_9_0_3 counter_4bit_0_9_0_3_1(reset, clk, carry_in, s[3:0], carry_out1,mode);

//計數器，負責十位數，只有個位數有進位時才要+1,所以carry_in的位置放carry_out1，得到的值存在s[7:4]裡，進位放在carry_out2
counter_4bit_0_2 counter_4bit_0_2_1(reset, clk, carry_out1, s[7:4], mode);

//個位數的七段顯示器，將s[3:0]轉成七段顯示
seven_segment seven_segment1(s[3:0],q[6:0]);

//個位數的七段顯示器，將s[7:4]轉成七段顯示
seven_segment seven_segment2(s[7:4],q[13:7]);

endmodule
```

counter_4bit_0_2.sv

```
`timescale 1ns/10ps
module counter_4bit_0_2(
    input reset,                //重置
    input clk,                  //時脈
    input carry_in,             //進位輸入
    output logic [3:0] value,   //輸出
    output logic mode           //切換模式 當vale == 2時，mode = 1
);

    always_ff @ (posedge clk)
    begin
        if(reset)               //同步，如果reset 就把值歸0
            value <= #1 0;
        else if(value == 4'd2 && carry_in == 1) //數到2也把值歸0
            value <= #1 0;
        else if(carry_in == 1)
            value <= #1 value + 1; //值加1
    end

    assign mode = (value == 4'd2)?1:0; //當value = 2 且 mode = 1時

endmodule
```

counter_4bit_0_9_0_3.sv

```sv
`timescale 1ns/10ps
module counter_4bit_0_9_0_3(
    input reset,                //重置
    input clk,                  //時脈
    input carry_in,             //進位輸入
    output logic [3:0] value,   //輸出
    output logic carry_out,     //進位輸出
    input mode                  //模式選擇 0:0~9，1:0~3
);

    always_ff @ (posedge clk)
        begin
            if(reset)               //同步，如果reset 就把值歸0
                value <= #1 0;
            else if(value == 4'd9 && carry_in == 1 && mode == 0) //mode = 0 ，數到9把值歸0
                value <= #1 0;
            else if(value == 4'd3 && carry_in == 1 && mode == 1) //mode = 1 ，數到3把值歸0
                value <= #1 0;
            else if(carry_in == 1)
                value <= #1 value + 1; //值加1
        end

    //當value = 9 && carry_in = 1 && mode = 0 時 或 當value = 3 && carry_in = 1 && mode = 1 時 ，有進位
    assign carry_out = ((value == 4'd9 && carry_in == 1 && mode == 0) || (value == 4'd3 && carry_in == 1 && mode == 1))?1:0;

endmodule
```

counter_BCD_min.sv

```sv
`timescale 1ns/10ps
module counter_BCD_min(
    input reset,                //重置
    input clk,                  //時脈
    output logic [13:0] q,      //輸出
    output logic [7:0] s,       //輸出16進位，方便debug <ex>1001,0001 -> 91
    output logic carry_out
);

logic carry_out1,carry_out2;

//計數器，負責個位數，每次都要+1所以carry_in的位置放1，得到的值存在s[3:0]裡，進位放在carry_out1
counter_4bit_0_9 counter_4bit_0_9_1(reset, clk,          1, s[3:0], carry_out1);

//計數器，負責十位數，只有個位數有進位時才要+1,所以carry_in的位置放carry_out1，得到的值存在s[7:4]裡，進位放在carry_out2
counter_4bit_0_5 counter_4bit_0_5_1(reset, clk, carry_out1, s[7:4], carry_out2);

//個位數的七段顯示器，將s[3:0]轉成七段顯示
seven_segment seven_segment1(s[3:0],q[6:0]);

//個位數的七段顯示器，將s[7:4]轉成七段顯示
seven_segment seven_segment2(s[7:4],q[13:7]);

assign carry_out = carry_out2;

endmodule
```

counter_4bit_0_5.sv

```sv
`timescale 1ns/10ps
module counter_4bit_0_5(
    input reset,                //重置
    input clk,                  //時脈
    input carry_in,             //進位輸入
    output logic [3:0] value,   //輸出
    output logic carry_out      //進位輸出
);

    always_ff @ (posedge clk)
        begin
            if(reset)               //同步，如果reset 就把值歸0
                value <= #1 0;
            else if(value == 4'd5 && carry_in == 1) //數到5也把值歸0
                value <= #1 0;
            else if(carry_in == 1)
                value <= #1 value + 1; //值加1
        end

    assign carry_out = (value == 4'd5 && carry_in == 1)?1:0; //當value = 5 且 carry_in = 1時，有進位

endmodule
```

counter_4bit_0_9.sv

```
`timescale 1ns/10ps
module counter_4bit_0_9(
    input reset,                //重置
    input clk,                  //時脈
    input carry_in,             //進位輸入
    output logic [3:0] value,   //輸出
    output logic carry_out      //進位輸出
);

    always_ff @ (posedge clk)
       begin
          if(reset)               //同步，如果reset 就把值歸0
             value <= #1 0;
          else if(value == 4'd9 && carry_in == 1) //數到9也把值歸0
             value <= #1 0;
          else if(carry_in == 1)
             value <= #1 value + 1; //值加1
       end

    assign carry_out = (value == 4'd9 && carry_in == 1)?1:0; //當value = 9 且 carry_in = 1時，有進位

endmodule
```

七段顯示器

```
module seven_segment(
    input [3:0] A,          //輸入
    output logic [6:0] y   //輸出
);
 always_comb
begin
    //當A改變時，y會對應到七段顯示器，右邊的註解表示他在7段顯示器上的效果
    case(A)
       4'h0 : y = 7'b1000000; //0
       4'h1 : y = 7'b1111001; //1
       4'h2 : y = 7'b0100100; //2
       4'h3 : y = 7'b0110000; //3
       4'h4 : y = 7'b0011001; //4
       4'h5 : y = 7'b0010010; //5
       4'h6 : y = 7'b0000010; //6
       4'h7 : y = 7'b1111000; //7
       4'h8 : y = 7'b0000000; //8
       4'h9 : y = 7'b0010000; //9
       4'hA : y = 7'b0001000; //A
       4'hB : y = 7'b0000011; //b
       4'hC : y = 7'b1000110; //C
       4'hD : y = 7'b0100001; //d
       4'hE : y = 7'b0000110; //E
       4'hF : y = 7'b0001110; //F
    endcase
end
endmodule
```

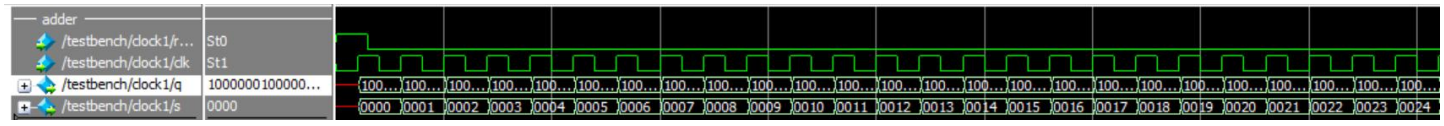
測試程式碼:

simulation > tb > testbench.sv

```
1   `timescale 1ns/10ps
2   module testbench;
3
4       logic reset;                    //重置
5       logic clk;                      //時脈
6       logic [27:0] q;                 //輸出
7       logic [14:0] s;                 //16進位數字 方便debug
8
9       clock clock1(
10          .reset(reset), //()內的變數為tb的變數，"."後面為clock.sv的變數，將2者對應起來
11          .clk(clk),
12          .q(q),
13          .s(s)
14      );
15
16      always #10 clk = ~clk;
17
18      initial begin
19          reset = 1;clk = 0; //一開始先reset，將時脈歸0
20          #15 reset = 0;
21          #40000 $stop;
22      end
23  endmodule
```

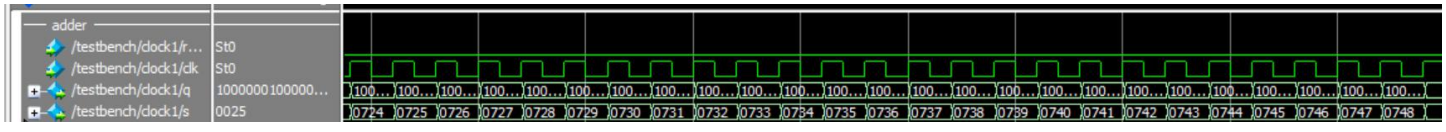
## ● 模擬結果與結果說明：

從頭:

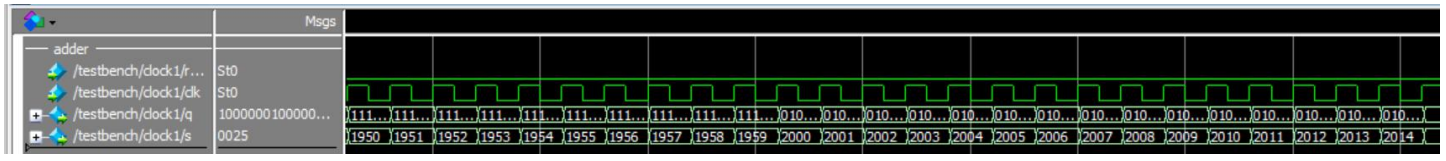



中間(59 進位):

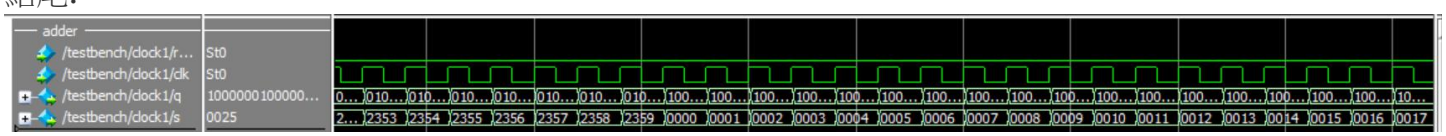

中間隨機截的-1:



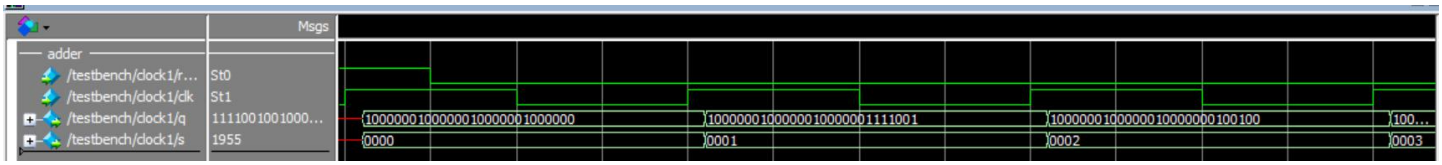

中間隨機截的-2:



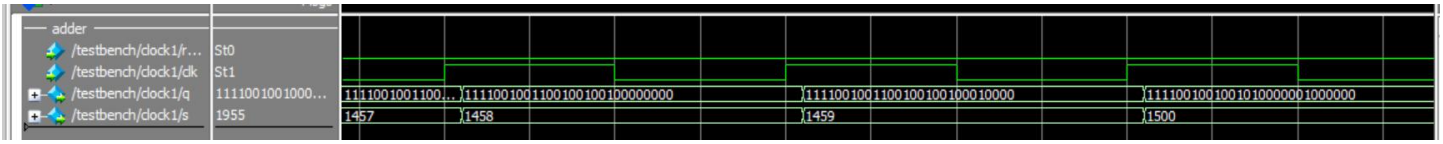

結尾:



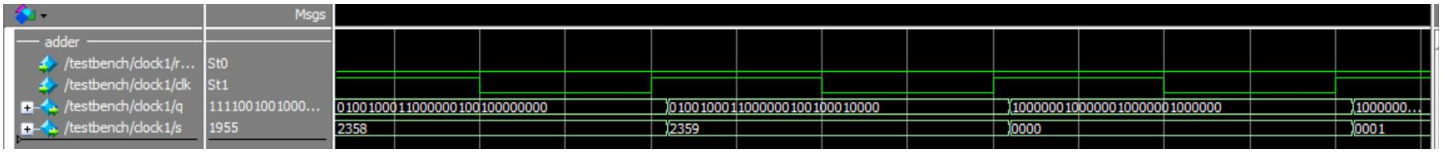
七段顯示器隨機截-1:




七段顯示器隨機截-2:




七段顯示器隨機截-3:




## ● 結論與心得：

這次作業有比較上手的感覺了，但還是有些地方沒有到很懂，我一開始遇到的 bug 是他會 09 -> 00 -> 11 -> 12 就是太晚進位了，之後我把進位判斷寫成 assign 而不是序向邏輯後就對了。我記得之前上數位邏輯時也有錯一樣的地方 qq，果然都忘光光了。

然後第二題我覺得我寫的好複雜，我在想有沒有更好的寫法，我寫好長 qqq，寫得好像軟體的感覺，就是一直呼叫函式的寫法...希望老師之後上課可以教我們硬體應該怎麼寫比較好，不然我寫的好奇怪歐...