

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

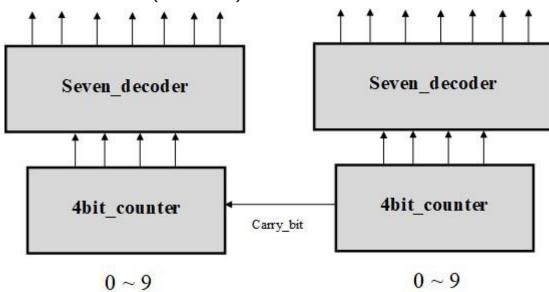
- 1. 繳交時一律轉 PDF 檔
- 2. 繳交期限為 上完課後 當週五晚上 12 點前
- 3. 一人繳交一份
- 4. 檔名:學號_HW?.pdf 檔名請按照作業檔名格 式進行填寫

— \ 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 \( \text{module counter_4bit_0_9} \)
 3
        input reset,
        input clk,
 4
                                      // 時脈
                                     //進位輸入
 5
        input carry_in,
 6
        output logic [3:0] value,
                                     //輸出
        output logic carry_out
 7
                                     //進位輸出
    );
 8
 9
        always_ff @ (posedge clk)
10
           begin
11 ⊟
                                       //同步,如果reset 就把值歸0
12
               if (reset)
                  value <= #1 0;</pre>
13
14
               else if(value == 4'd9 && carry_in == 1) //數到9也把值歸0
                  value <= #1 0;</pre>
15
              else if(carry_in == 1)
    value <= #1 value + 1; //值加1
16
17
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
     endmodule
```

這是七段顯示器(和上禮拜的作業一樣)

```
Emodule seven segment (
 1
 2
                              //輸入
        input [3:0] A,
 3
        output logic [6:0] y //輸出
    L);
 4
 5
    always_comb
 6
   ⊟begin
 7
        //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
 8
        case (A)
           4'h0 : y = 7'b10000000; //0
 9
           4'h1 : y = 7'b1111001; //1
10
           4'h2 : y = 7'b0100100; //2
11
           4'h3 : y = 7'b0110000; //3
12
           4'h4 : y = 7'b0011001; //4
13
           4'h5 : y = 7'b0010010; //5
14
           4'h6 : y = 7'b0000010; //6
15
           4'h7 : y = 7'b1111000; //7
16
17
           4'h8 : y = 7'b00000000; //8
           4'h9 : y = 7'b0010000; //9
18
           4'hA : y = 7'b0001000; //A
19
           4'hB : y = 7'b0000011; //b
20
           4'hC : y = 7'b1000110; //C
21
22
           4'hD : y = 7'b0100001; //d
23
           4'hE : y = 7'b0000110; //E
24
           4'hF : y = 7'b0001110; //F
25
        endcase
    end
26
27
    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 lns/10ps
 2
   Emodule counter_BCD(
 3
        input reset,
                                    //重置
                                    //時脈
        input clk,
 5
        output logic [13:0] q,
                                    //輸出
        output logic [7:0] s
                                    //輸出16進位・方便debug <ex>1001,0001 -> 91
 6
 7
    );
8
9
     logic carry out1, carry out2;
10
11
     //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
12
     counter 4bit 0 9 counter 4bit 0 9 1 (reset, clk,
                                                                1, s[3:0], carry out1);
13
    //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);
14
15
16
     //個位數的七段顯示器,將s[3:0]轉成七段顯示
17
18
     seven segment seven segment1(s[3:0],q[6:0]);
19
20
     //個位數的七段顯示器,將s[7:4]轉成七段顯示
21
     seven segment seven segment2(s[7:4],q[13:7]);
22
23
     endmodule
24
```

測試程式碼:

```
simulation > tb > ≡ testbench.sv
      `timescale 1ns/10ps
      module testbench;
                                         //重置
          logic reset;
          logic clk;
          logic [13:0] q;
                                         //輸出
  7
          logic [7:0] s;
                                         //16進位數字 方便debug
          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
              .clk(clk),
 11
              .q(q),
 12
 13
              .s(s)
          );
          always #10 clk = ~clk;
 17
          initial begin
 19
              reset = 1;clk = 0; //一開始先reset, 將時脈歸0
              #15 reset = 0;
 20
 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)



| <i>ν</i> Ε | | | | | | | | | | | | | | | | |
|------------|-----------|---------|----------|---------|-----------|--------|-----------|--------|-----------|--------|-----------|--------|-----------|---------|-----------|--------|
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | (1000000) | 1000000 | (1000000 | 1111001 | (10000000 | 100100 | (10000000 | 110000 | (10000000 | 011001 | (10000000 | 010010 | (10000000 | 0000010 | (10000001 | 111000 |
| | (00 | | 01 | | 02 | | 03 | | 04 | | 05 | | 06 | | (07 | |
| | | | | | | | | | | | | | | | | |

進位的地方

| | V | Y11110011111001 | V | V | V11110010011001 | V | V | Y111100111 |
|--------------------|-------------------------|--------------------------|------------------------|-----------------|------------------------|------------------------|------------------------|----------------------|
| 10000000010. 09 | (111110011000000 (10 |))11110011111001)11 | (11110010100100 (12 | (11110010110000 | X11110010011001 X14 |)11110010010010 115 | X11110010000010 X16 |),1111100111),17 |
| | | | | | 1,000 | | | ,, |

尾

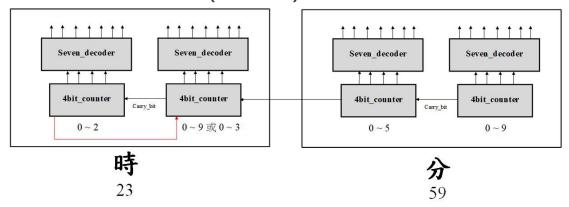
| 0010000 | 1111000 | 0010000 | 000000 | 0010000 | 010000 | (1000000 | 1000000 | 1000000 | 1111001 | (10000000 | 100100 | (10000000 | 110 | 000 | (10000000 | 011001 |
|---------|---------|---------|--------|---------|--------|----------|---------|---------|---------|-----------|--------|-----------|-----|-----|-----------|--------|
| 97 | | (98 | |)99 | |)00 | | 01 | | 02 | | 03 | | | 04 | |
| | | | | ** | | | | | | | | | | | | |

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)
- - 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)

clock.sv

```
timescale lns/10ps
 2
    ⊟module clock(
 3
        input reset,
                                   //重置
 4
                                   // 時脈
        input clk,
 5
        output logic [27:0] q,
                                   //輸出
 6
        output logic [15:0] s
                                   //輸出16進位,方便debug
 7
     );
 8
 9
     logic carry_out1, carry_out2;
10
11
     //計數器,負責分鐘
12
     counter BCD min counter BCD min1(reset, clk, q[13:0], s[7:0], carry out1);
13
14
     //計數器,負責小時
15
     counter BCD hour counter BCD hour1 (reset, clk, carry out1, q[27:14], s[15:8]);
16
17
     endmodule
18
```

counter BCD hour.sv

```
timescale <a href="mailto:lns.right">lns./10ps</a>
   ⊟module counter_BCD_hour(
       input reset,
                                  // 重署
 4
       input clk,
                                 //時脈
       input carry_in,
 5
 6
       output logic [13:0] q,
                                 //輸出
       output logic [7:0] s
                                 //輸出16進位,方便debug
   );
8
 9
10
    logic carry out1, mode;
11
12
    //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);
13
14
15
    //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
16
    counter_4bit_0_2 counter_4bit_0_2_1(reset, clk, carry_out1, s[7:4], mode);
17
18
    //個位數的七段顯示器,將s[3:0]轉成七段顯示
19
    seven segment seven segment1(s[3:0],q[6:0]);
20
21
    //個位數的七段顯示器,將s[7:4]轉成七段顯示
22
    seven_segment seven_segment2(s[7:4],q[13:7]);
23
24
    endmodule
```

counter 4bit 0 2.sv

```
`timescale 1ns/10ps
   ⊟module counter 4bit 0 2(
 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
                                   //切換模式 當vale == 2時, mode = 1
        output logic mode
 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;
              else if(value == 4'd2 && carry_in == 1) //數到2也把值歸0
14
15
                 value <= #1 0;
16
              else if(carry_in == 1)
17
                 value <= #1 value + 1; //值加1
18
19
20
        assign mode = (value == 4'd2)?1:0; //當value = 2 且 mode = 1時
21
22
     endmodule
```

```
counter 4bit 0 9 0 3.sv
      `timescale 1ns/10ps
     Emodule counter 4bit 0 9 0 3(
         input reset,
                                    // 時脈
// 進位輸入
  4
         input clk,
         input carry in,
  5
         output logic [3:0] value,
                                    //進位輸出
         output logic carry_out,
                                    //模式選擇 0:0~9・1:0~3
  8
         input mode
 10
         always_ff @ (posedge clk)
 11
 12
            begin
 13
               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
 14
 15
 16
                  value <= #1 0;
 17
               else if(value == 4'd3 && carry_in == 1 && mode == 1) //mode = 1 ,數到3把值歸0
                  value <= #1 0;
 18
 19
               else if(carry in == 1)
 20
                  value <= #1 value + 1; //值加1
 21
            end
 22
         //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;
 23
 24
 25
 26
counter BCD min.sv
       `timescale 1ns/10ps
     Emodule counter BCD min (
          input reset,
                                          //重置
          input clk,
  4
                                          //時脈
          output logic [13:0] q, output logic [7:0] s,
  5
                                         //輸出
                                           //輸出16進位·方便debug <ex>1001,0001 -> 91
  6
          output logic carry_out
  8
  9
 10
      logic carry out1, carry out2;
 11
      //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);
 12
 13
 14
 15
      //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
 16
      counter 4bit 0 5 counter 4bit 0 5 1(reset, clk, carry out1, s[7:4], carry out2);
 17
 18
      //個位數的七段顯示器,將s[3:0]轉成七段顯示
 19
      seven_segment seven_segment1(s[3:0],q[6:0]);
 20
 21
      //個位數的七段顯示器,將s[7:4]轉成七段顯示
 22
      seven_segment seven_segment2(s[7:4],q[13:7]);
 23
 24
      assign carry out = carry out2;
 25
 26
      endmodule
counter 4bit 0 5.sv
        timescale lns/10ps
  1
     Emodule counter 4bit 0 5(
```

```
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
                                     //同步,如果reset 就把值歸0
12
             if (reset)
13
                 value <= #1 0;</pre>
14
              else if (value == 4'd5 && carry in == 1) //數到5也把值歸0
15
                value <= #1 0;</pre>
16
              else if(carry in == 1)
17
                 value <= #1 value + 1; //值加1
18
           end
19
20
        assign carry out = (value == 4'd5 && carry in == 1)?1:0; //當value = 5 且 carry in = 1時, 有進位
21
    endmodule
```

counter 4bit 0 9.sv

```
`timescale lns/10ps
    Emodule counter 4bit 0 9(
 3
        input reset,
                                   //重置
                                   // 蒔脈
        input clk,
 4
 5
        input carry_in,
                                   //進位輸入
 6
        output logic [3:0] value,
                                   //輸出
 7
                                   //進位輸出
        output logic carry_out
 8
     );
 9
10
        always_ff @ (posedge clk)
11
    begin
12
              if(reset)
                                     //同步·如果reset 就把值歸0
                 value <= #1 0;</pre>
13
              else if(value == 4'd9 && carry_in == 1) //數到9也把值歸0
14
15
                 value <= #1 0;
              else if(carry_in == 1)
16
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
    endmodule
22
```

七段顯示器

```
1
   ⊟module seven_segment(
 2
       input [3:0] A,
                              //輸入
 3
        output logic [6:0] y //輸出
    L);
 4
 5
    always_comb
 6
   ⊟begin
 7
        //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
 8
   case (A)
          4'h0 : y = 7'b1000000; //0
 9
          4'h1 : y = 7'b1111001; //1
10
          4'h2 : y = 7'b0100100; //2
11
          4'h3 : y = 7'b0110000; //3
12
          4'h4 : y = 7'b0011001; //4
13
          4'h5 : y = 7'b0010010; //5
14
          4'h6: y = 7'b0000010; //6
15
16
          4'h7 : y = 7'b1111000; //7
17
          4'h8 : y = 7'b00000000; //8
          4'h9 : y = 7'b0010000; //9
18
19
          4'hA : y = 7'b0001000; //A
          4'hB : y = 7'b00000011; //b
20
          4'hC : y = 7'b1000110; //C
21
          4'hD : y = 7'b0100001; //d
22
23
          4'hE : y = 7'b0000110; //E
          4'hF : y = 7'b0001110; //F
24
25
        endcase
26
     end
27
     endmodule
```

測試程式碼:

```
simulation > tb > 

    testbench.sv
       `timescale 1ns/10ps
      module testbench;
                                          //重置
          logic reset;
                                          // 時脈
          logic clk;
          logic [27:0] q;
                                          //輸出
          logic [14:0] s;
                                          //16進位數字 方便debug
          clock clock1(
              .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
 10
 11
 12
              .q(q),
              .s(s)
 13
 14
          );
 15
          always #10 clk = ~clk;
 16
 17
          initial begin
 18
              reset = 1;clk = 0; //一開始先reset, 將時脈歸0
 19
              #15 reset = 0;
 20
 21
              #40000 $stop;
 22
          end
      endmodule
 23
```

● 模擬結果與結果說明:

從頭:



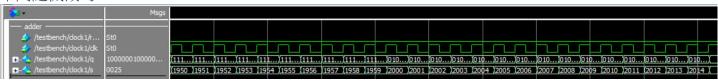
中間(59 進位):



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:

| ≨ i • | Msgs | | | | | | | | | | |
|------------------------------------|------|------------|-------------|-----------|------------|-------------|----------|------------|-------------|----------|-------|
| — adder — // /testbench/dock1/r | St0 | | | | | | | | | | |
| /testbench/dock1/dk | | (100000010 | 00000100000 | 0.1000000 | Y100000010 | 00000100000 | 01111001 | Y100000010 | 00000100000 | 00100100 |),100 |
| | 1955 | 0000 | 00000100000 | 01000000 | 0001 | 00000100000 | 01111001 | (0002 | 00000100000 | 00100100 | 0003 |

七段顯示器隨機截-2:

| 12 411 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | | | | | | | | | | | |
|--|----------------|---------------|------------|----------|-----------|-------------|----------|-----------|-------------|---------|--|
| — adder — | | | | | | | | | | | |
| /testbench/dock1/dk St1 | | | | | | | | | | | |
| | 1111001001100. | . 11110010011 | .001001001 | 00000000 | 111100100 | 11001001001 | 00010000 | 111100100 | 10010100000 | 1000000 | |
| | 1457 | (1458 | | | 1459 | | | (1500 | | | |
| | | | | | | | | | | | |

七段顯示器隨機截-3:

| \$1 • | Msgs | | | | | | | | | | |
|--------------------------------|---------------|---------|--------------|-----------|-----------|-------------|----------|-----------|-------------|----------|----------|
| — adder — | | | | | | | | | | | |
| /testbench/dock1/r | St0 | | | | | | | | | | |
| /testbench/dock1/dk | St1 | | | | | | | | | | |
| = -4 /testbench/dock1/q | 1111001001000 | 0100100 | 011000000100 | 100000000 | 010010001 | 10000001001 | 00010000 | 100000010 | 00000100000 | 01000000 | (1000000 |
| +-4 /testbench/clock1/s | 1955 | 2358 | | | 2359 | | | (0000) | | | 0001 |
| | | | | | | | | | | | |

● 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...