106061151 劉安得

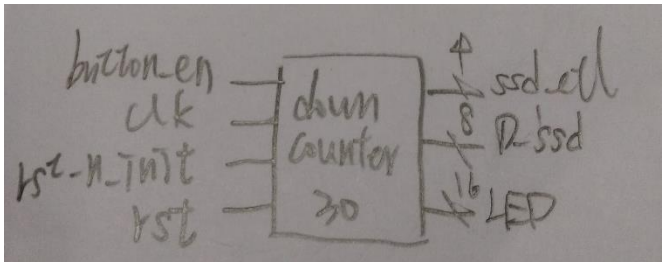
Prelab5 & Lab5-1.

1. **Design Specification**

   30 down counter with button

   Input : button_en, clk, rst, rst_n_init

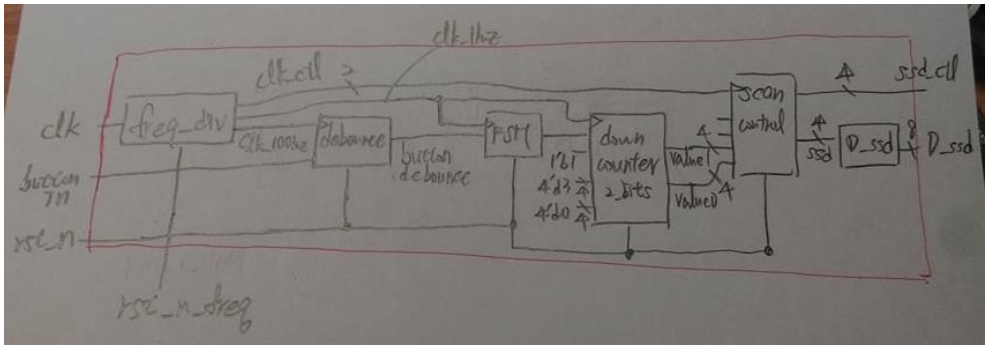
   Output : [3:0] ssd_ctl, [7:0] D_ssd, [16:0] LED



2. **Design implementation**

   使用 prelab5 所寫的 finite state machine 來 implement

   以下為各 module 間 wire 的連結，在 top module 我有用兩個 MUX 來判斷 LED 顯示和是否已經數到零。我還有寫 rst_n_init 來用 DIP reset 那些一開始就要 reset 的 module



I/O	D_ssd[7]	D_ssd[6]	D_ssd[5]	D_ssd[4]	D_ssd[3]	D_ssd[2]	D_ssd[1]	D_ssd[0]
Pin	W7	W6	U8	V8	U5	V5	U7	V7

I/O	Ssd_ctl[3]	Ssd_ctl[2]	Ssd_ctl[1]	Ssd_ctl[0]
Pin	W4	V4	U4	U2

I/O	LED[15]	LED[14]	LED[13]	LED[12]	LED[11]	LED[10]	LED[9]	LED[8]
Pin	L1	P1	N3	P3	U3	W3	V3	V13

I/O	D_ssd[7]	D_ssd[6]	D_ssd[5]	D_ssd[4]	D_ssd[3]	D_ssd[2]	D_ssd[1]	D_ssd[0]
Pin	V14	U14	U15	W18	V19	U19	E19	U16

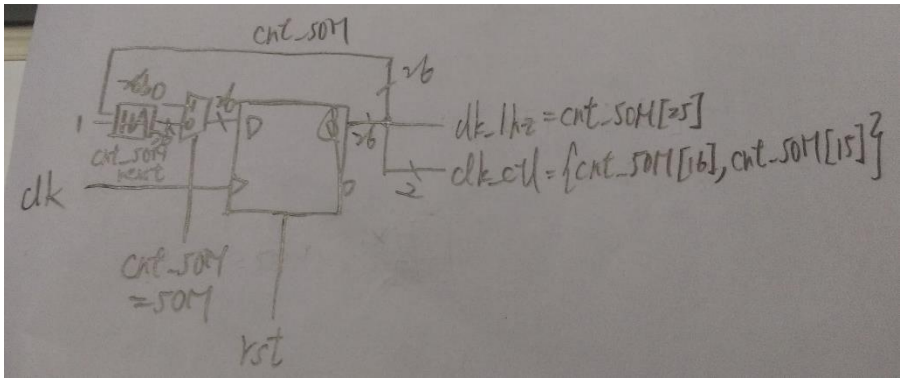
I/O	button_en	clk	rst	Rst_n_init
Pin	U18	W5	U17	V16

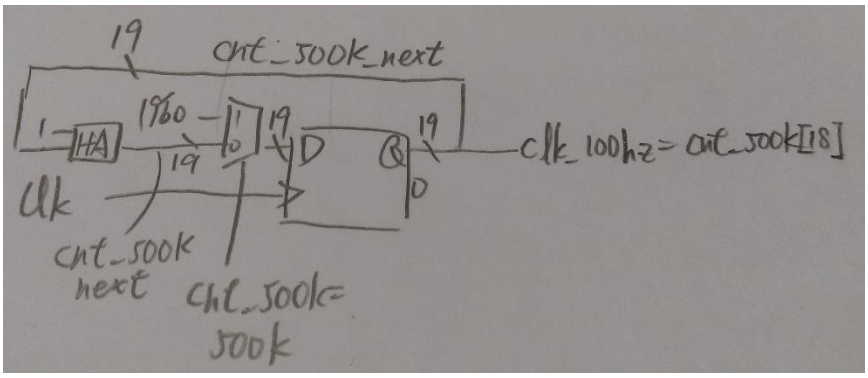
Frequency divider

和之前不同的是，我增加了 100HZ

If(cnt_50M = 50000000)
  Cnt_50M_next = 0
  Cnt_1hz_next = ~clk_1hz
Else
  Cnt_50M_next = cnt_50M + 1
  Cnt_1hz_next = clk_1hz

If(cnt_500K = 500000)
  Cnt_500K_next = 0
  Cnt_100hz_next = ~clk_100hz
Else
  Cnt_500K_next = cnt_500K + 1
  Cnt_100hz_next = clk_100hz






```
always @*
    begin
        if(cnt_50M == 26'd50000000)
        begin
            cnt_50M_next = 26'd0;
            clk_1hz_next = ~ clk_1hz;
        end
        else
        begin
            cnt_50M_next = cnt_50M + 1'b1;
            clk_1hz_next = clk_1hz;
        end

        if(cnt_500K == 19'd500000)
        begin
            cnt_500K_next = 19'd0;
            clk_100hz_next = ~ clk_100hz;
        end
        else
        begin
            cnt_500K_next = cnt_500K + 1'b1;
            clk_100hz_next = clk_100hz;
        end
    end
```

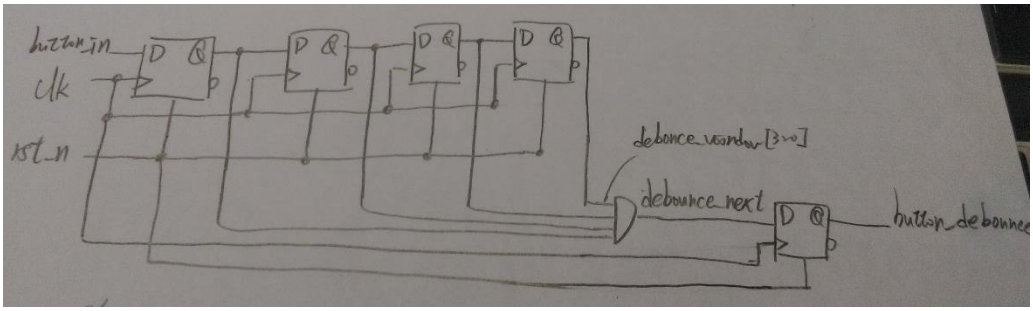
Debounce

因為 button 會有 signal bounce 當切換的時候，所以要寫 debounce 來整流
這裡是使用 4-bits shifter，當 4 個 register 都讀到 1 時，才 output1。

Window_next = {debounce_window[2:0], button_in}

If(debounce_window = 1111), debounce_next = 1

Else, debounce_next = 0

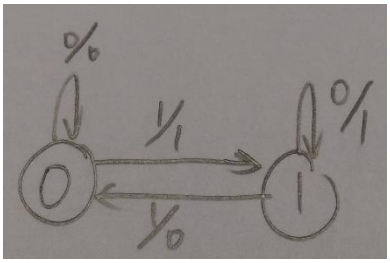



```
assign window_next = {debounce_window[2:0], button_in};

always @*
      begin
            if(debounce_window == 4'b1111)
                  debounce_next = 1'b1;
            else
                  debounce_next = 1'b0;
     end
```

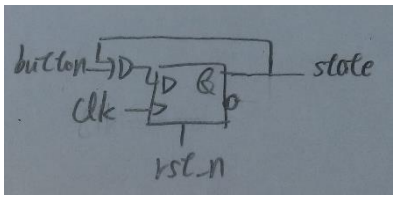
FSM button

State 1 is enabled, state 0 is disabled



state	in	Next state
0	0	0
0	1	1
1	0	1
1	1	0

Next state = Button ^ state

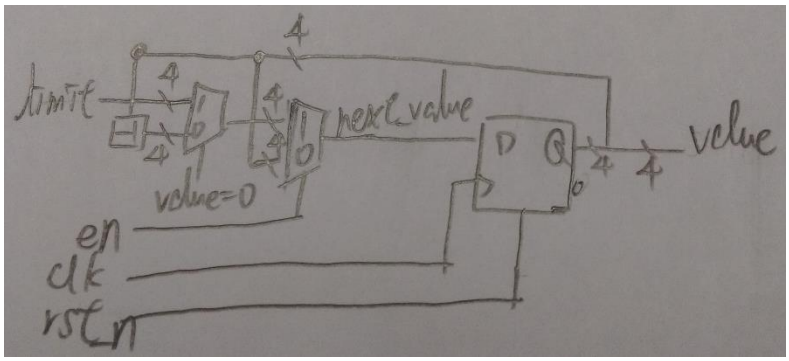


assign next_state = state ^ button;

down counter

之前的 down counter 只有特地為倒數 30 秒設計，我現在寫得比較有泛用性，他有 initial value, limit, borrow and enable

```
If(en = 0)
  next_value = value
  borrow = 0
else if(value = 0)
  next_value = limit
  borrow = 1
else
  next_value = value - 1
  borrow = 0
```



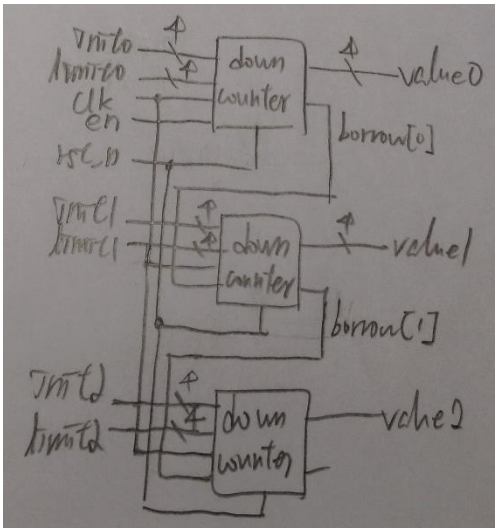

```
always @*
        begin
                if(en == 0)
                begin
                        next_value = value;
                        borrow = 1'b0;
                end
                else if(value == 4'b0)
                begin
```

```
            next_value = limit;
            borrow = 1'b1;
        end
        else
        begin
            next_value = value - 1'b1;
            borrow = 1'b0;
        end
    end
```

downcounter 3-bit

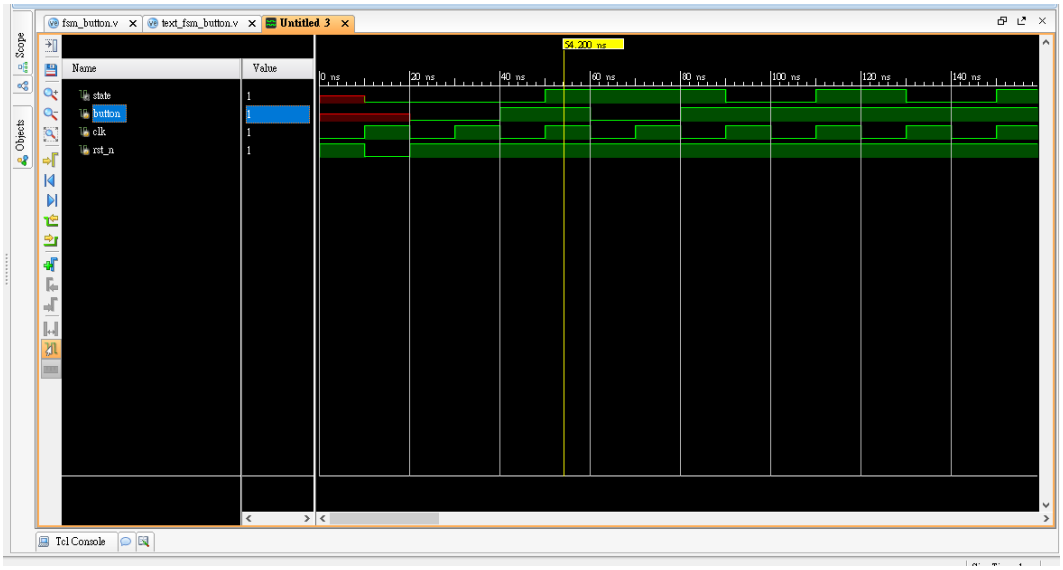
因為 lab5-3 要倒數 3bits，所以一次就寫 3bits down counter

以下為三個 downcounter 間的 wire 的連結



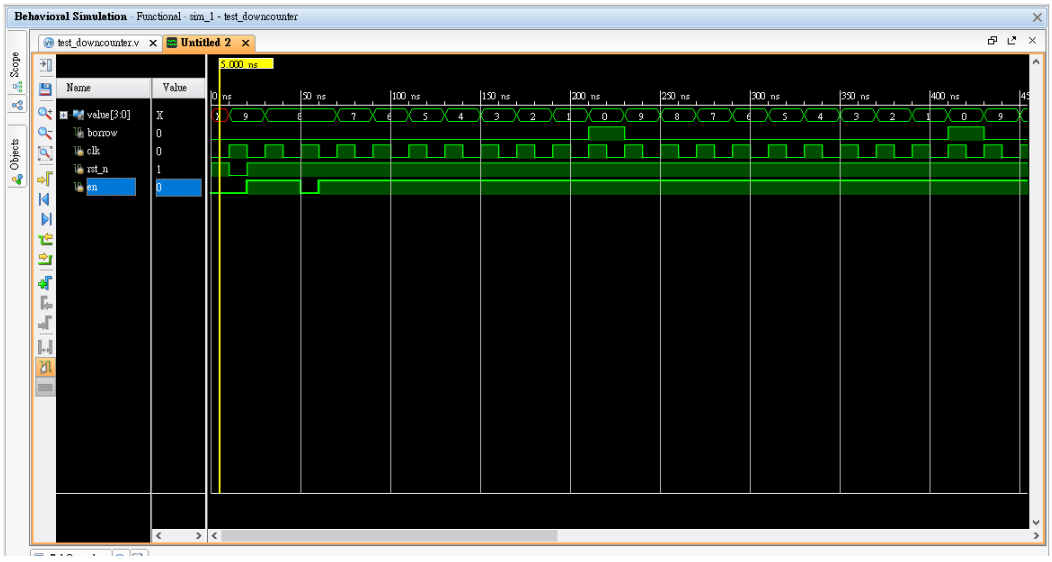
3. Simutation

FSM button



State	button	Next state
0	0	0
0	1	1
1	0	1
1	1	0

Downcounter



Value[3:0]	clk	Rst_n	En
X	0	1	0
9(1001)	1	0	0

9(1001)	0	1	1
8(1000)	1	1	1
8(1000)	0	1	1
8(1000)	1	1	0
8(1000)	0	1	1
7(0111)	1	1	1
7(0111)	0	1	1
6(0101)	1	1	1
6(0101)	0	1	1
5(0101)	1	1	1
5(0101)	0	1	1
4(0100)	1	1	1
4(0100)	0	1	1
3(0011)	1	1	1
3(0011)	0	1	1
2(0010)	1	1	1
2(0010)	0	1	1
1(0001)	1	1	1
1(0001)	0	1	1
0(0000)	1	1	1
0(0000)	0	1	1
9(1001)	1	1	1

4. Discussion

這次很多 module 因為之前沒有寫好或是要新增功能，所以都要重寫，例如 frequency dicider, down counter, display。所以最好第一次就寫得比較有泛用性，這樣才能一勞永逸。

Lab5 有很多新概念，例如 debounce, FSM 等等，再加上不熟悉 button，所以第一個子實驗我做很久

button 按的時候為 1，因此如果要用 button 來 reset 的話，要做 inverter，才能當作 rst_n 使用。然後我在 top module 還有寫一個 if else 判斷 zero，如果數到 0 的話，要讓 en 關掉，這樣才會停止。我 LED 的賦值也在 top module 寫。最後，我在 display 新增了 input 1111 時，都不要亮，這樣就會讓只有三個七段顯示器亮，而不用顯示 0

除此之外，我不像範例那樣使用 case 和 if else，我直接用 combinational circuit，來決定 next state 和 state 的關係。我認為這樣會比較省元件。

最後我發現，D flip-flop 最好獨立寫，雖然跟其他 code 寫一起會比較直觀，但這樣也會讓 code 不容易找到錯誤。我認為這點蠻重要的，打邏輯設

計的時候，打對跟打得好是兩回事，要如何簡潔又有效率地達到目的很重要

5. Conclusion

做完這次 prelab，讓我學會如何應用 FSM 和使用 button。

6. Reference

05_Push_Buttons P. 3, P. 4

Debounce

Debounce 我和講義上寫法一樣，使用 shifter 來 debounce

05_Push_Buttons P. 14

The schematic of top module

實際上我又多加了 debounce module

05_Push_Buttons P.19, P.20

和範例不同的是，我沒有使用 case, if else

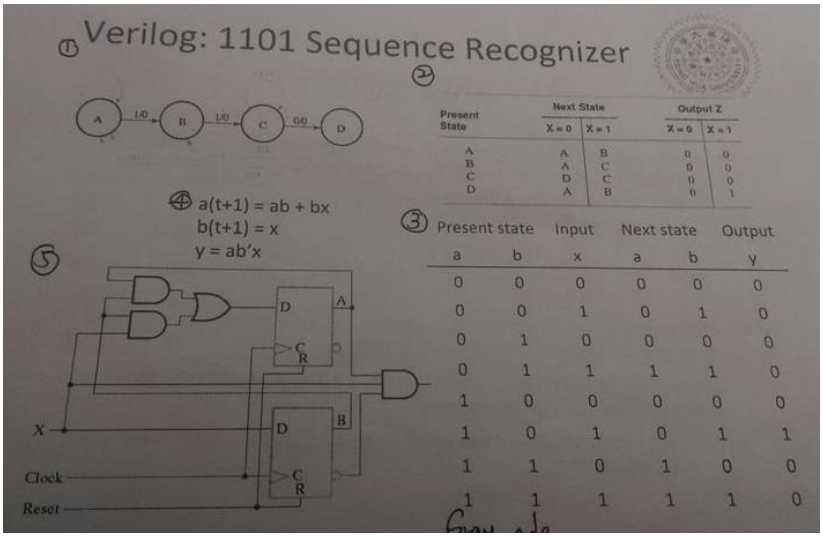
05_Push_Buttons P. 21, P. 22

The Verilog of top module

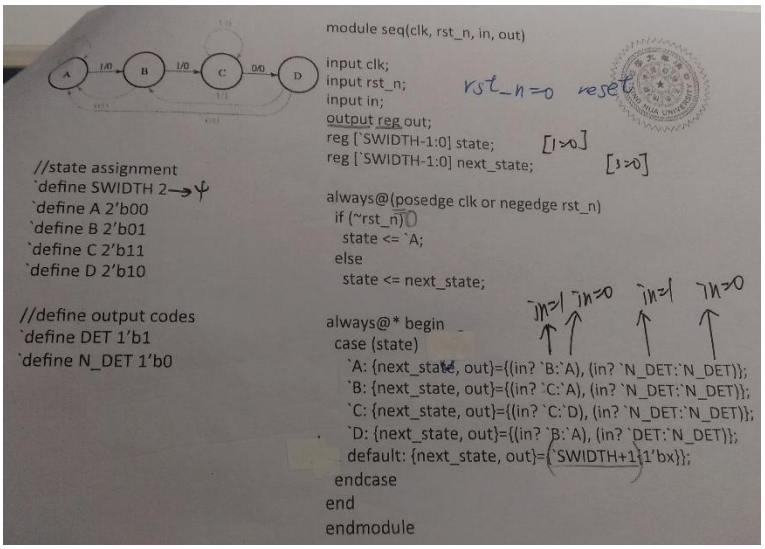
一樣，實際上我多加了 debounce module

Logic Design Lecture 7 Sequential Circuits

FSM 應用範例



FSM verilog


A
B
C
D
1/0
1/0
0/0
//state assignment
`define SWIDTH 2
`define A 2'b00
`define B 2'b01
`define C 2'b11
`define D 2'b10
//define output codes
`define DET 1'b1
`define N_DET 1'b0
module seq(clk, rst_n, in, out)
input clk;
input rst_n;
rst_n=0 reset
input in;
output reg out;
reg [`SWIDTH-1:0] state;
[1:0]
reg [`SWIDTH-1:0] next_state;
[1:0]
always@(posedge clk or negedge rst_n)
if (~rst_n)
state <= `A;
else
state <= next_state;
always@* begin
case (state)
`A: {next_state, out}={(in? `B:`A), (in? `N_DET:`N_DET)};
`B: {next_state, out}={(in? `C:`A), (in? `N_DET:`N_DET)};
`C: {next_state, out}={(in? `C:`D), (in? `N_DET:`N_DET)};
`D: {next_state, out}={(in? `B:`A), (in? `DET:`N_DET)};
default: {next_state, out}={(`SWIDTH+1){1'bx}};
endcase
end
endmodule

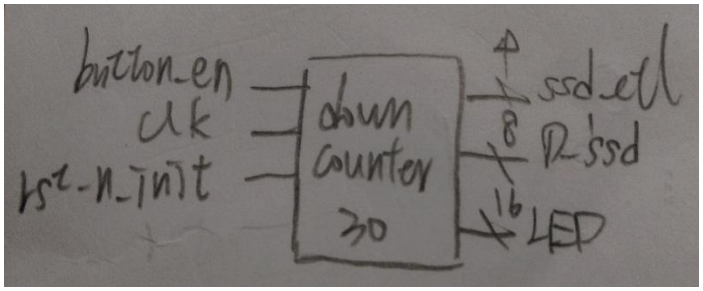
Lab5-2.

1. **Specification**

   和 lab5-1 相似，只是 reset 跟 start/pause 從 two button 變成 one button，長按為 reset，短按為 start/pause

   Input : button_en, clk, rst_n_init

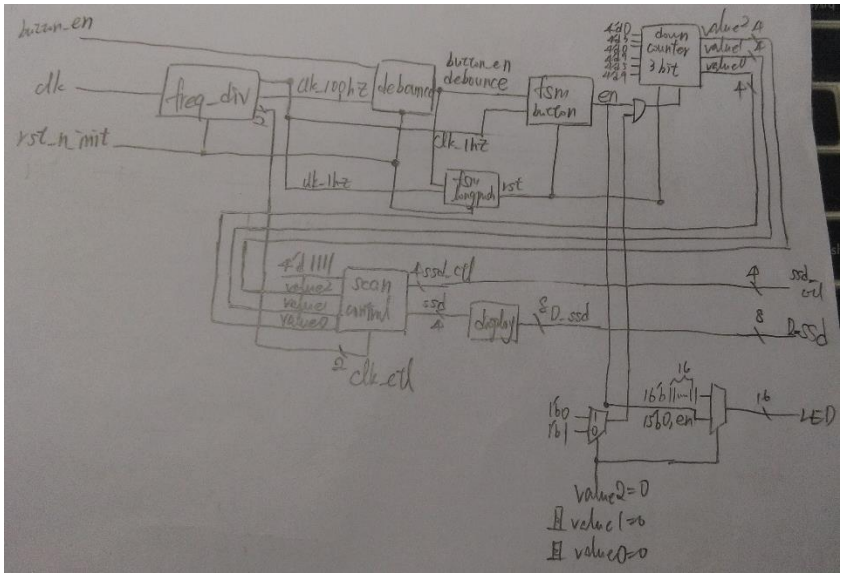
   Output : [3:0] ssd_ctl, [7:0] D_ssd, [16:0] LED



2. **Design implementation**

   使用 lab5-1 所寫的各 module 來 implement，lab5-2 我只多寫 fsm_longpush module 來判斷長按

   以下為各 module 間 wire 的連結，在 top module 我有用兩個 MUX 來判斷 LED 顯示和是否已經數到零。我還有寫 rst_n_init 來用 DIP reset 那些一開始就要 reset 的 module



I/O	D_ssd[7]	D_ssd[6]	D_ssd[5]	D_ssd[4]	D_ssd[3]	D_ssd[2]	D_ssd[1]	D_ssd[0]
Pin	W7	W6	U8	V8	U5	V5	U7	V7

I/O	Ssd_ctl[3]	Ssd_ctl[2]	Ssd_ctl[1]	Ssd_ctl[0]
Pin	W4	V4	U4	U2

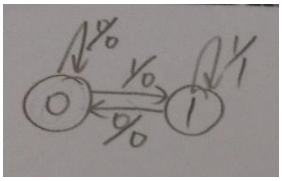
I/O	LED[15]	LED[14]	LED[13]	LED[12]	LED[11]	LED[10]	LED[9]	LED[8]
Pin	L1	P1	N3	P3	U3	W3	V3	V13

I/O	D_ssd[7]	D_ssd[6]	D_ssd[5]	D_ssd[4]	D_ssd[3]	D_ssd[2]	D_ssd[1]	D_ssd[0]
Pin	V14	U14	U15	W18	V19	U19	E19	U16

I/O	button_en	clk	Rst_n_init
Pin	U18	W5	V16

Fsm_longpush

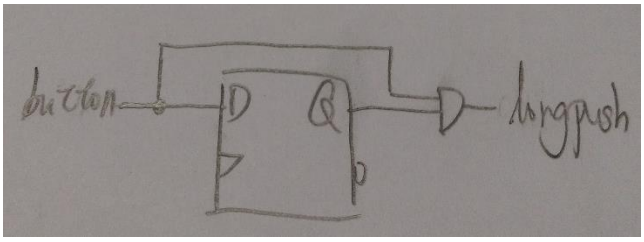
State 0 為前一個不為 0, state 1 為前一個為 1



state	in	nextstate	out
0	0	0	0
0	1	1	0
1	0	0	0
1	1	1	1

longpush = state & in

next_state = in

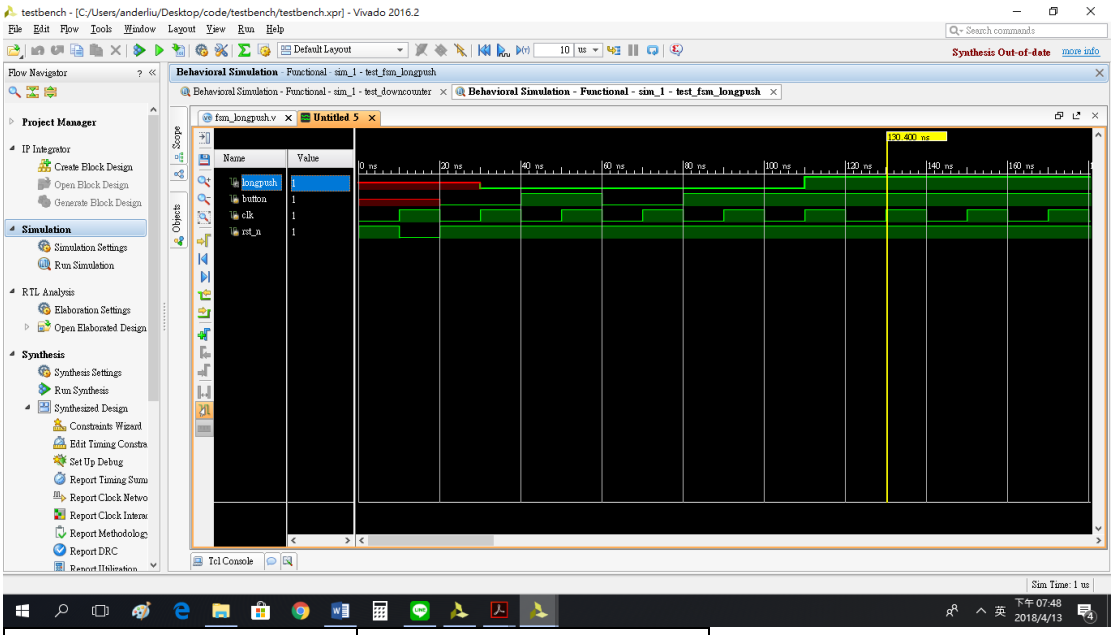


```
assign next_state = button;

    always @(posedge clk or negedge rst_n)
    begin
        if(rst_n == 0)
            state <= 1'b0;
        else
            state <= next_state;
            longpush = state & button;
    end
```

3. Simutation

Fsm_longpush



button	Longpush
0	0
1	0
0	0
1	0
1	1

4. Discussion

我在寫長按的 FSM 時，同樣不像範例那樣使用 case 和 if else，我直接用 combinational circuit。但我發現如果 out 直接和 state 連接會出問題，因為 state 會瞬間改變，但 state table 上的 out 是和 next state 同時產生的，因此連接 out 和 state 要寫在 always 裡面。

除此之外我一開始 FSM 的 state 寫三個，但我最後發現只要兩個就好

5. Conclusion

做完這次實驗，讓我學會如何用長按來 reset

6. Reference

因為 lab5-2 和 lab5-1 很像，所以同 lab5-1

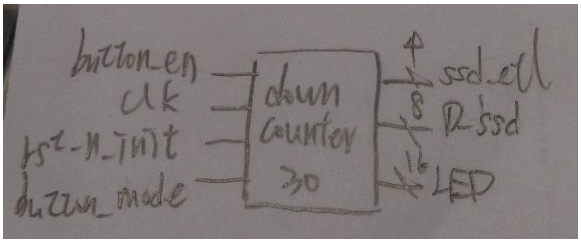
Lab5-3.

1. **Specification**

   和 lab5-2 相似，只是多一個 mode 控制倒數 30 秒或一分鐘

   Input : button_en, button_mode, clk, rst_n_init

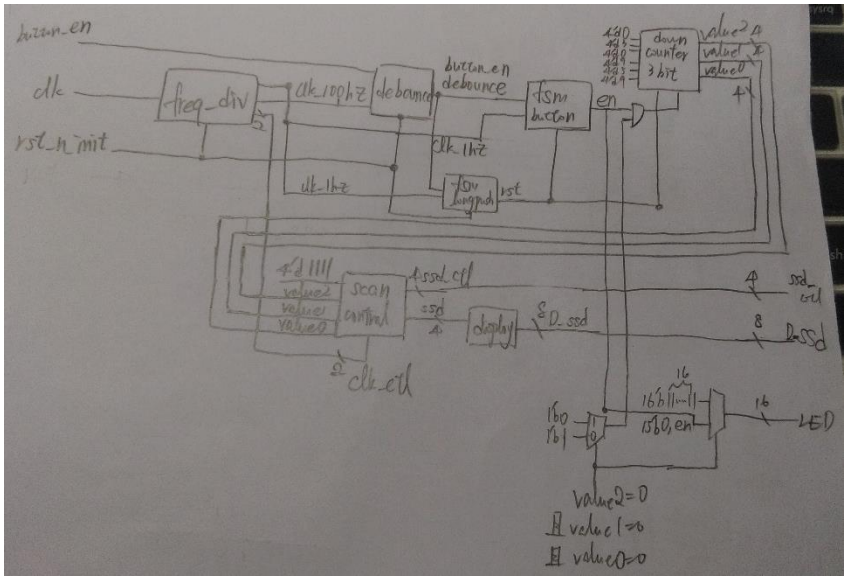
   Output : [3:0] ssd_ctl, [7:0] D_ssd, [16:0] LED



2. **Design implementation**

   使用 lab5-2 所寫的各 module 來 implement， 只是更多 module

   以下為各 module 間 wire 的連結，在 top module 我有用兩個 MUX 來判斷 LED 顯示和是否已經數到零。我還有寫 rst_n_init 來用 DIP reset 那些一開始就要 reset 的 module



I/O	D_ssd[7]	D_ssd[6]	D_ssd[5]	D_ssd[4]	D_ssd[3]	D_ssd[2]	D_ssd[1]	D_ssd[0]
Pin	W7	W6	U8	V8	U5	V5	U7	V7

I/O	Ssd_ctl[3]	Ssd_ctl[2]	Ssd_ctl[1]	Ssd_ctl[0]
Pin	W4	V4	U4	U2

I/O	LED[15]	LED[14]	LED[13]	LED[12]	LED[11]	LED[10]	LED[9]	LED[8]
Pin	L1	P1	N3	P3	U3	W3	V3	V13

I/O	D_ssd[7]	D_ssd[6]	D_ssd[5]	D_ssd[4]	D_ssd[3]	D_ssd[2]	D_ssd[1]	D_ssd[0]
Pin	V14	U14	U15	W18	V19	U19	E19	U16

I/O	button_en	Button_mode	clk	Rst_n_init
Pin	U18	U17	W5	V16

3. Discussion

Lab5-1, 5-2 寫出來之後，lab5-3 就沒什麼問題，只是這次要連接 9 個 module，隨著實驗也做到 lab5，內容也變得越來越複雜。

我發現處理 mode 的 debounce 和 fsm 不能和 en 一起 reset，否則 mode 會被 reset 成 0，這樣永遠都會是倒數 30 秒，因此 mode 的 debounce 和 fsm 要和 frequncy divider, fsm longpush, en 的 debounce 一起在一開始就要 reset

除此之外，我還多設一顆 LED 來顯示 mode，這樣比較好操作，不然 reset 的時候不知道是 mode 幾

4. Conclusion

做完這次實驗，讓我學會如何用長按和兩顆按鈕分別表示 reset, mode, start, pause

5. Reference

因為 lab5-3 和 lab5-1 很像，所以同 lab5-1