106061151 劉安得

Lab7-1.

1. **Specification**

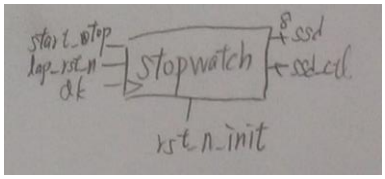
A stopwatch

Input : start_stop, lap_rst_n, rst_n_init, clk

Output : [7:0] ssd, [3:0] ssd_ctl

當按下 start/stop 時，開始計時，再按下 start/stop 時，暫停計時

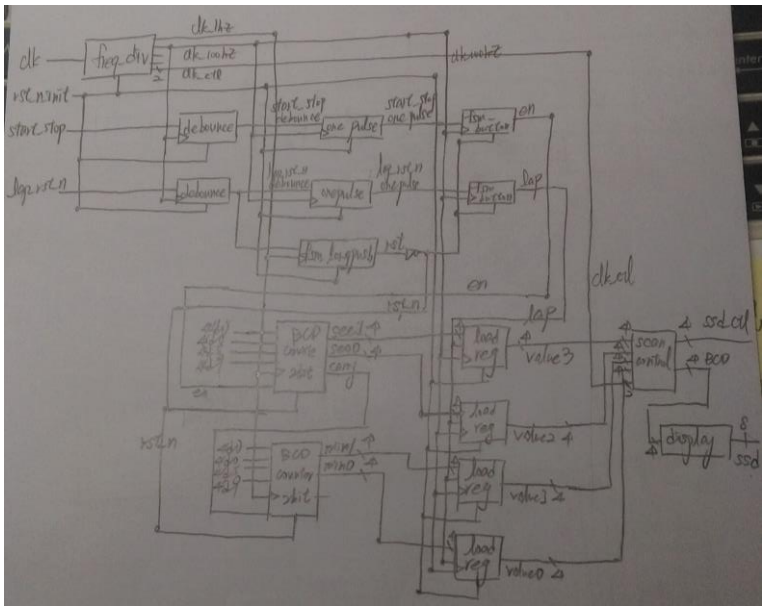
當短按 lap_rst_n 時，代表 lap，SSD 會停止更新數字，但 stopwatch 仍然在計時；再短按一次 lap_rst_n 時，SSD 便會更新現在數到多少；如果長按 lap_rst_n，便代表 reset



2. **Implementation**

大部分 module 都是之前寫的，這次只有多寫了 one pulse 和 load register。之前沒寫 one pulse 的時候，因為我 clock 設 1hz，所以每次都要按一陣子，而且不能多不能少。現在我 clock 設 100hz，而 one pulse 則只會產生短短的一次 100hz 的 cycle，這樣只要按一下就可以了。

以下為各 module 間 wire 的連結



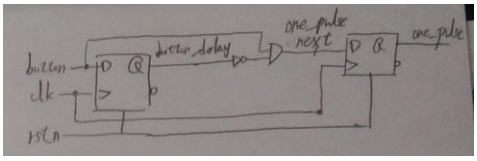
I/O	Ssd[7]	Ssd[6]	Ssd[5]	Ssd[4]	Ssd[3]	Ssd[2]	Ssd[1]	Ssd[0]
Pin	W7	W6	U8	V8	U5	V5	U7	V7

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

I/O	Start_stop	Lap_rst_n	Rst_n_init	Clk
Pin	U18	U17	V17	W5

One-pulse

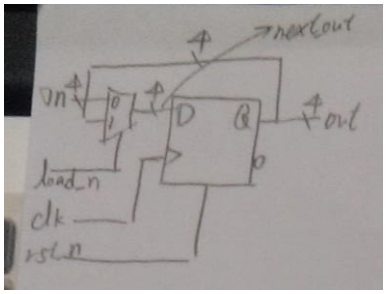
我參考之前講義的打法



assign one_pulse_next = button & ~button_delay;

Load register

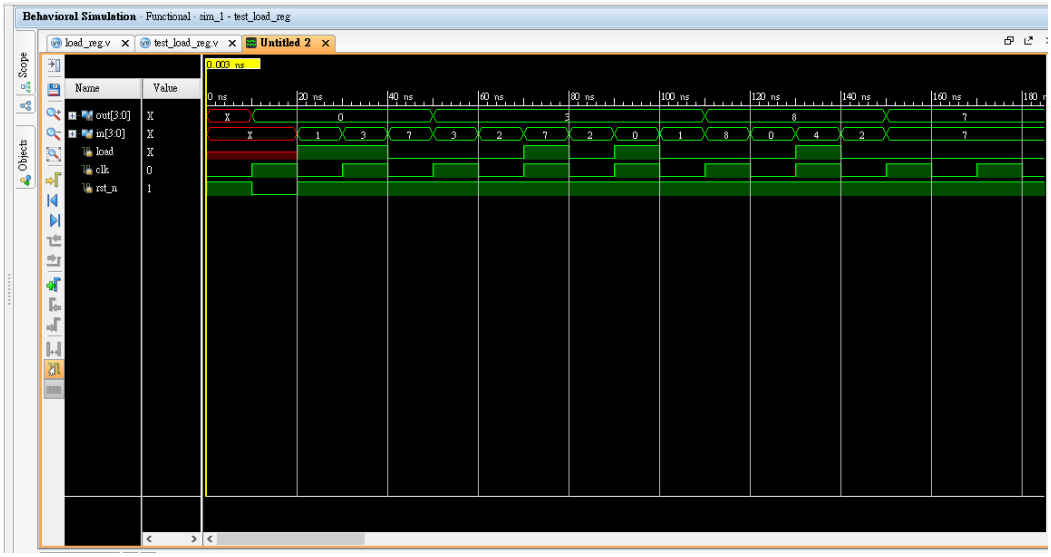
當 load 為 1 時，load input；load 為 0 時，便暫存現在的值




```
always @*
begin
    if(load_n == 0)
        next_out = in;
    else
        next_out = out;
end
```

3. **Simulation**

   Load register



clk	Load_n	In[3:0]	Next_out[3:0]	Out[3:0]
0	1	1		0
1	1(unload)	3	0	0
0	0	7		0
1	0(load)	3	3	3
0	0	2		3
1	1(unload)	7	3	3
0	0	2		3
1	1(unload)	0	3	3
0	0	1		3
1	0(load)	8	8	8
0	0	0		8
1	1(unload)	4	8	8
0	0	2		8
1	0(load)	7	7	7

4. Discussion

我本來打算把 load register 寫進 scan control 裡面，修改裡面的內容讓它可以被暫停，但後來決定還是分開寫。原因是因為首先 scan control 本來是 combinational logic，但加了 load register 的話就要多加 clock 和 reset，不會比分開寫來的方便多少；再來是因為如果分開寫的話，可用性比較廣，說不定未來 lab 還會用到 load register

這次 lab 充分的使用按鈕，把 lap 跟 reset 寫在一起，這樣就可以比較省按鈕，不然每個功能都設一個按鈕，就要設很多顆。說不定之後還能做出按兩下這種功能。

我之前沒有寫 one pulse 這個功能，design button 的時候就變得很麻煩。這次 lab 做出 one pulse 後，只要輕輕按一下即可

5. Conclusion

做完這次 lab，讓我學會如何做 stopwatch，還有如果使用 lap

6. Reference

05_push_buttons

The logic diagram of one pulse

Lab7-2.

1. **Specification**

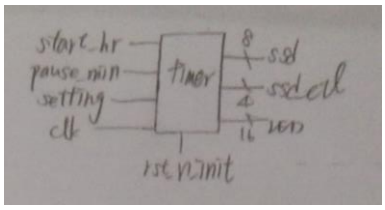
   A timer

   Input : start_hr, pause_min, setting, rst_n_init, clk

   Output : [7:0] ssd, [3:0] ssd_ctl, [15:0] LED

   當 setting 為 0 時，短按 start/hr 時，代表 start，開始倒數，再短按 start/hr 時，代表 stop，歸零；短按 pause/min 時，代表 pause，會暫停倒數，再短按 pause/min 時，代表 resume，會繼續倒數

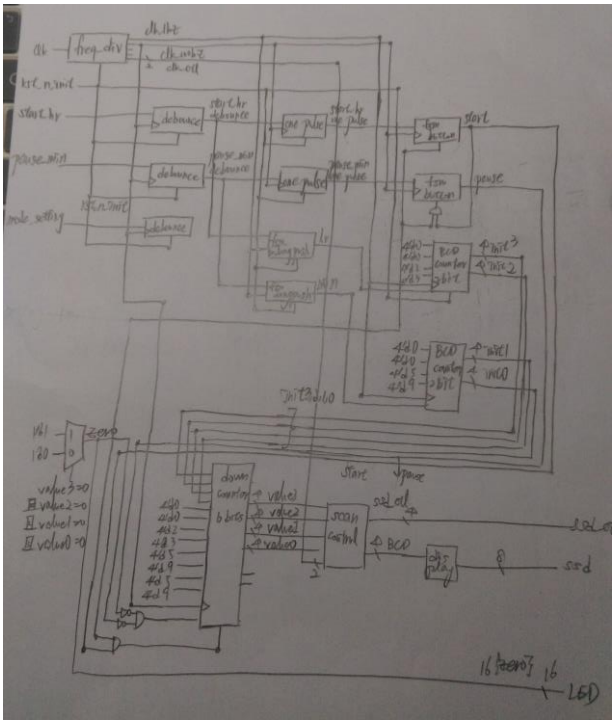
   當 setting 為 1 時，代表 setting mode，此時設定倒數時間；長按 start/hr，代表 hour，每長按一秒 hour 會加一；長按 pause/min，代表 minute，每長按一秒 minute 會加一



2. **Implementation**

   Lab7-1 是往上加，lab7-2 則是往下數。而我大都是用之前寫過的 module，只有使用之前的 down counter 再寫一個 6-bits down counter。

   以下為各 module 間 wire 的連結，我另外呼叫兩個 counter 計算 setting 的值，我還有寫一個 if-else 判斷是否數到 0，數到 0 時，將 down counter disable



I/O	Ssd[7]	Ssd[6]	Ssd[5]	Ssd[4]	Ssd[3]	Ssd[2]	Ssd[1]	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

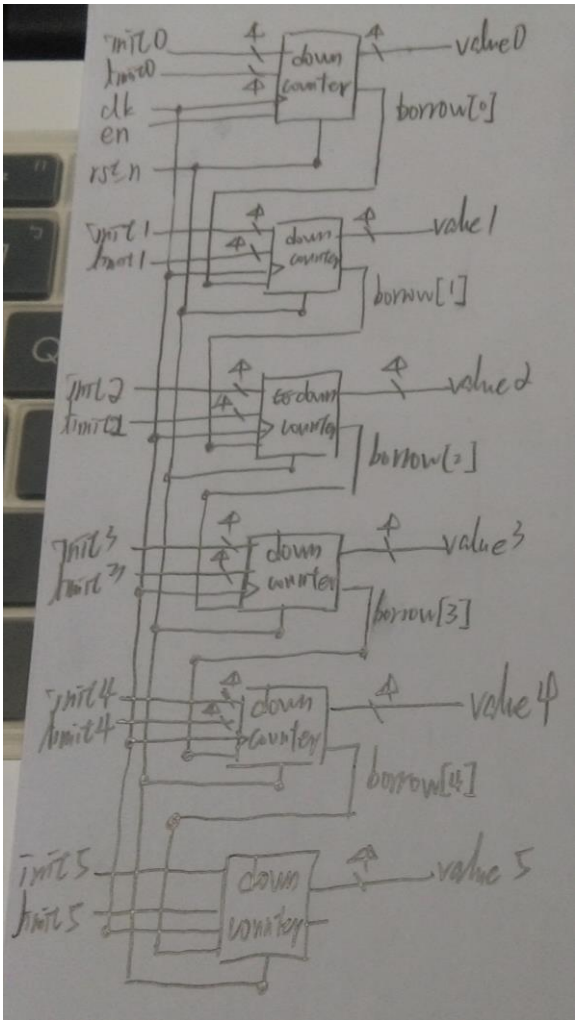
I/O	Start_hr	Pause_min	Setting	Rst_n_init	Clk
Pin	W19	T17	V16	V17	W5

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	LED[7]	LED[6]	LED[5]	LED[4]	LED[3]	LED[2]	LED[1]	LED[0]
Pin	V14	U14	U15	W18	V19	U19	E19	U16

6-bits down counter

6 個 down counter 之前的 wire 連接如下



3. Discussion

這次 lab 我把 start 跟 hour 寫在一起，pause 跟 minute 寫在一起，這樣就可以比較省按鈕。因為當 setting mode 時，start 跟 pause 功能用不到，所以這樣子就可以讓一個按鈕有多個功能。

我使用 100hz 去跑，因為單位是秒，所以七段顯示器每變一次為 0.6 秒

4. Conclusion

做完這次 lab，讓我學會如何做 timer，還有 start/stop，pause/resume

Lab7-3.

1. **Specification**

   A stopwatch/timer

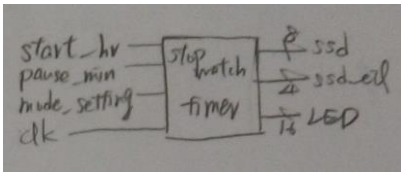
   Input : start_hr, pause_min, mode_setting, rst_n_init, clk

   Output : [7:0] ssd, [3:0] ssd_ctl, [15:0] LED

   當 mode 為 0 時，代表為 stopwatch mode；短按 start/hr，代表 start，會開始計時，再短按 start/hr 時，代表 stop，暫停計時；短按 pause/min 時，代表 pause，SSD 會停止更新數字，但 stopwatch 仍然在計時；再短按 start/min 時，代表 resume，SSD 便會更新現在數到多少

   當 mode 為 1 時，代表為 timer mode；短按 start/hr，代表 start，開始倒數，再短按 start/hr 時，代表 stop，歸零；短按 pause/min 時，代表 pause，會暫停倒數，再短按 pause/min 時，代表 resume，會繼續倒數

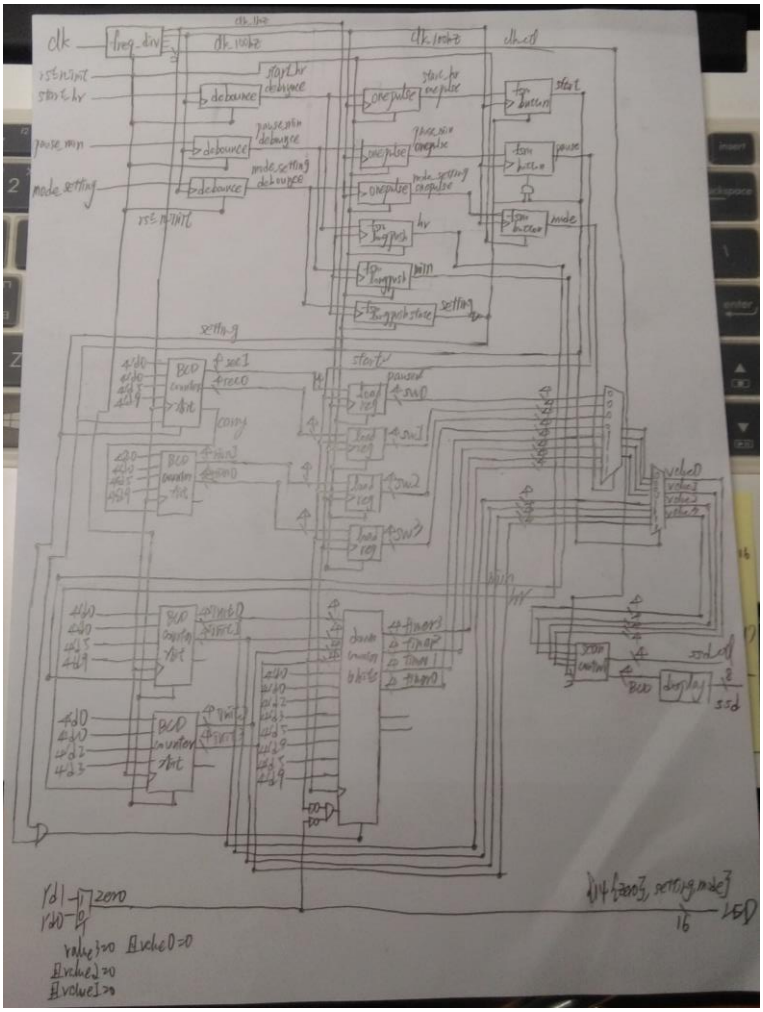
   短按一下 mode/setting，會切換 mode；當長按 mode/setting 時，代表 setting mode，此時設定倒數時間；長按 start/hr，代表 hour，每長按一秒 hour 會加一；長按 pause/min，代表 minute，每長按一秒 minute 會加一；再長按 mode/setting 便會切換回去

2. **Implementation**

   Lab7-3 就是把前兩個 lab 合起來，因為這裡的 setting 有點像是長按和按一下換一個 state 的概念結合，所以我有再另外寫一個 module 來 implement 長按換一個 state。

   以下為各 module 間 wire 的連結，我另外呼叫兩個 counter 計算 setting 的值，我還有寫一個 if-else 判斷是否數到 0，數到 0 時，將 down counter disable。最後我 LED 有設最後兩個分別為 setting 和 mode，方便我知道現在是什麼模式。



I/O	Ssd[7]	Ssd[6]	Ssd[5]	Ssd[4]	Ssd[3]	Ssd[2]	Ssd[1]	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

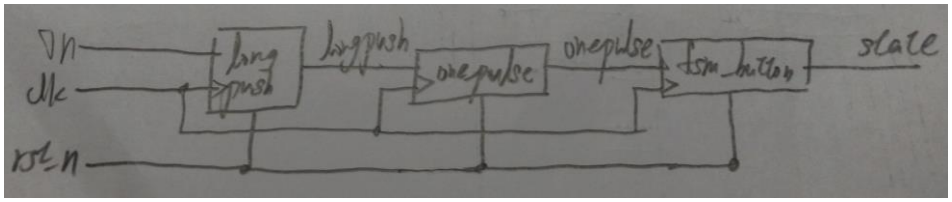
I/O	Start_hr	Pause_min	Mode_Setting	Rst_n_init	Clk
Pin	W19	T17	U18	V17	W5

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	LED[7]	LED[6]	LED[5]	LED[4]	LED[3]	LED[2]	LED[1]	LED[0]
Pin	V14	U14	U15	W18	V19	U19	E19	U16

fsm_longpush_state

我連結了 longpush, one pulse and fsm_state，連結 one pulse 是因為要確保一次長按只會有一個 cycle 進到 fsm_state。



3. Discussion

這次 lab 我用三個按鈕設定了六個功能，每個按鈕分別有短按跟長按兩種模式。這次連結的 module 有多達二十幾個，雖然幾乎不用打任何 module，只要打 top module 就好，但因為有很多 wire 要設，還要確定有打對，有連結好，因此打的時候都要特別注意

一開始我只有長按 setting 會 reset，但我發現這樣子不會一直保持在 setting mode，除非一直按著，所以我才會有想把長按和按一下切換一次結合在一起的想法

4. Conclusion

因為大部分 module 都是用之前寫的，所以現在比較考驗如果使用這些現有工具並連結它們。而且所使用的 module 也從個位數，十幾個變成二十幾個了。而且每次 lab 都是前一次 lab 的功能再延伸，有種前呼後應的感覺，例如這次 lab 就有用到很久以前寫的 display, frequency divider 等等。