Prelab5 Report

5-1. 30_Second Down Counter with Pause Function

●Design Specification

Input:

clk,(接上板子的原本的震盪頻率 W5)

rst_n, (控制整個功能的開關)

in,(讓倒數器暫停倒數，或是重新開始倒數)

pb_rst(讓倒數器可以臨時回歸 30 秒重新倒數)

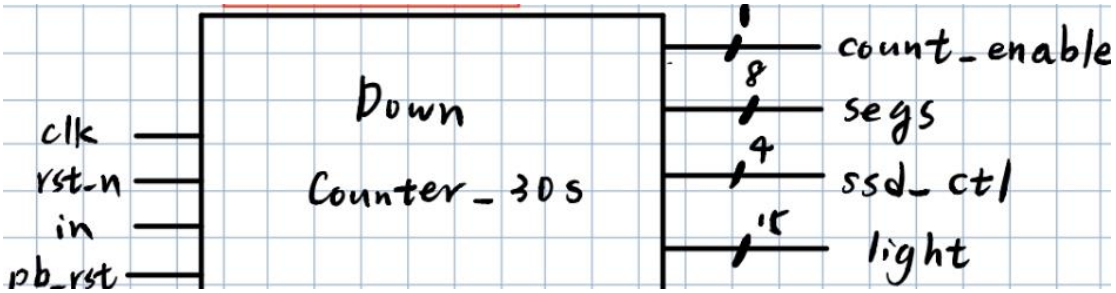
Output:

[7:0]segs,(傳給七段顯示器的 output)

[3:0] ssd_ctl,(控制每一個七段顯示器的亮與暗)

[14:0]light(讓倒數器歸零的時候會讓所有的)

Count_enable(設定一個 led 燈表示當時狀態)



Function Table:

Input				Output			
clk	rst_n	in	pb_rst	cnt_en	segs	ssd_ctl	light
x	↓	x	x	reset 30, cnt_en = 0			
↑	x	x	H	reset 30, cnt_en = 0			
↑	x	H	x	start ← → pause, cnt_en = ~cnt_en			

↑: posedge trigger

↓: negedge trigger

H=1’b1

cnt_en 是控制 downcounter 的倒數與否

segs 是控制七段顯示器 8 段的顯示

ssd_ctl:是控制分別四段七段顯示器的亮與暗

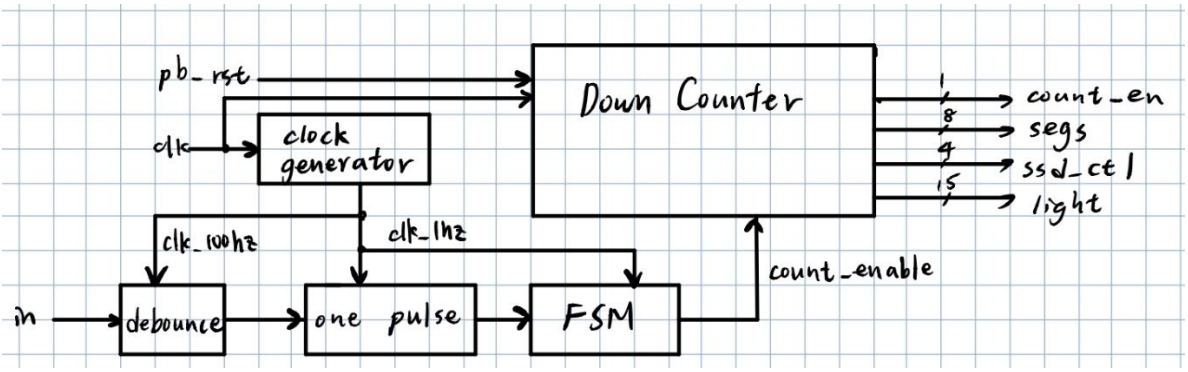
Light 是當倒數器歸零的時候 15 個燈會亮

●Design Implementation(1.1)

1.Outline

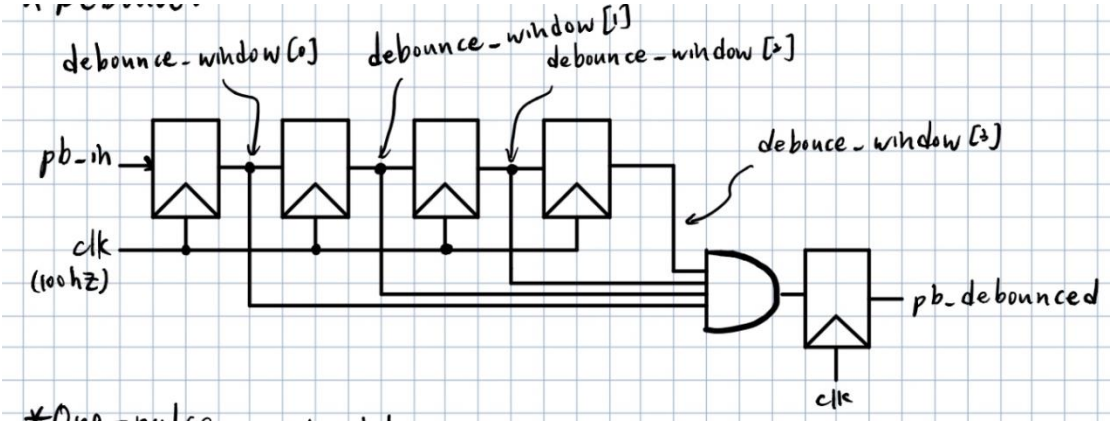
這個實驗主要是要利用按鈕來操作暫停與啟動還有歸零的功能，因為按鈕不像是開關，可以直接向上或是向下扳來提供穩定 0 或 1 的 input；按鈕是操作者按下去一下就要對整個系統產生效果，但是每個人的習慣按的時間不一樣，更何況按鈕還有彈簧的特性，在使用者按下去的時候並不會出現震盪，這會讓系統不穩定，所以需要用到 debounce 來提供一個穩定輸出為 one_pulse 的訊號來解決這個問題。另外特別的是，這次利用到 fsm 來控制 down counter 的 countable 與否，因為持續倒數與暫停倒數是兩種狀態，不同的 input 而改變當時的狀態，不會有其他的狀態產生，所以非常適合使用 fsm。其餘部分都與上次 Lab4.5 的題目一樣。

2.Logic Diagram



↑ 上圖是整個系統的邏輯圖，每個小功能會在下面解釋(Down Counter、clock generator 已經在之前的 Lab 解釋過，所以只放 block diagram 和標示 input、output)

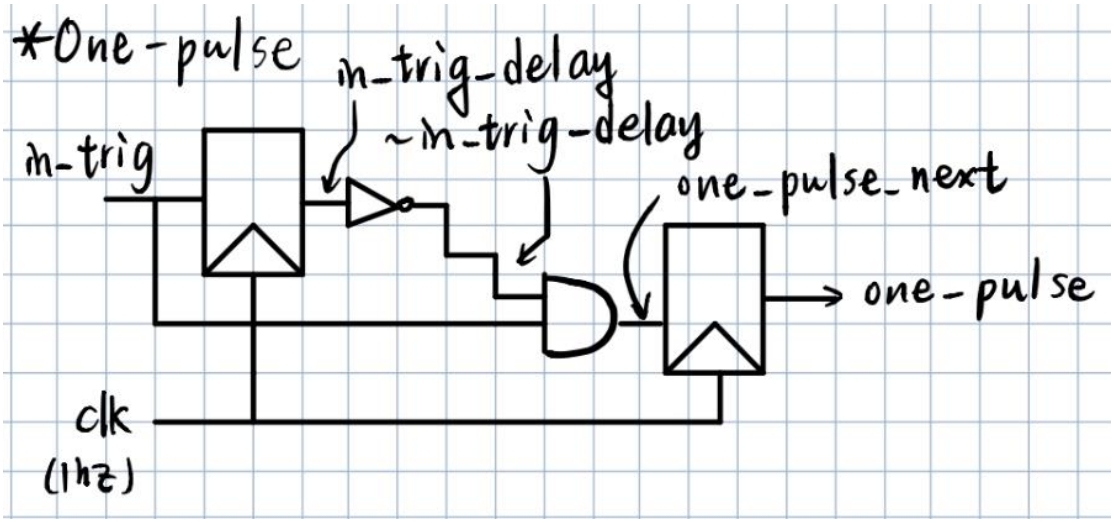
▲Debounce



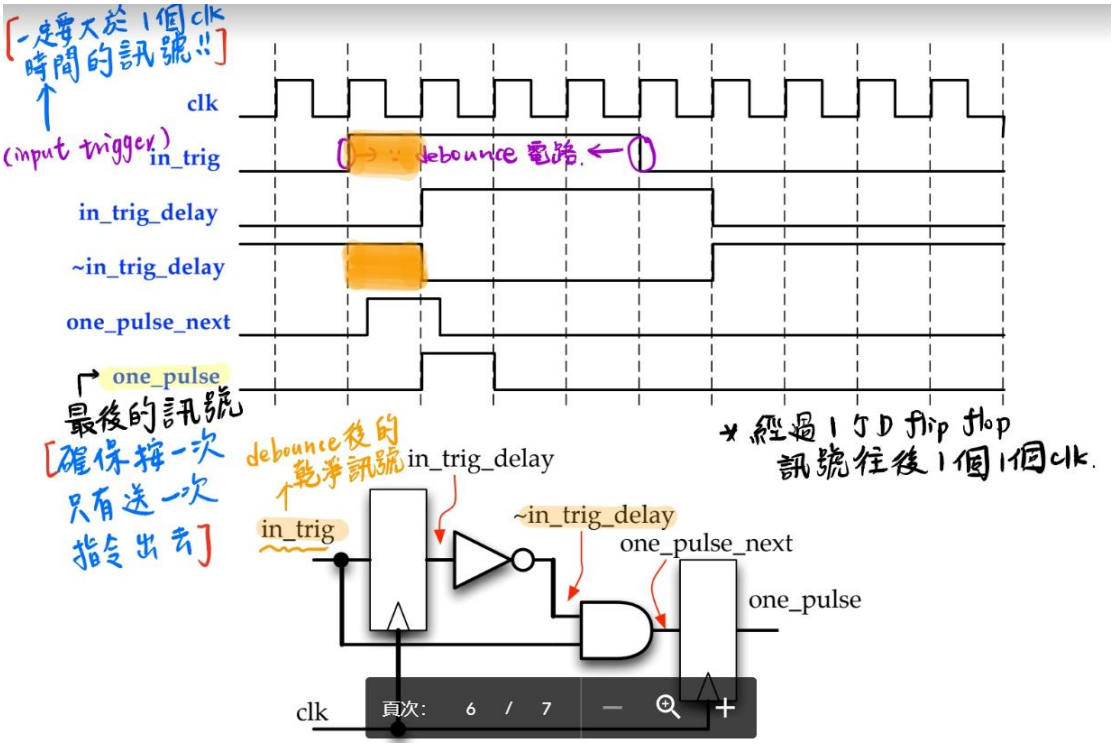
↑ 上圖是利用四個 D-flip flop 來改善按下按鈕會震盪的問題，如果把經過四個 D-flip flop 的結果全部 and 起來，就要當所有值(debounce_window[3:0])全部都等於 1 才會有一個 step 出來，後來再接上一個 D-flip flop 是為了要讓

訊號更穩定，因為可以讓 pb_debounced 的結果與 clk 排隊。另外值得一提的是這些 D-flip flop 接的 clk 是經過 clock generator 出來的 100hz，因為與按鈕震盪的頻率比較接近，可以比較準確的判斷按鈕的 trigger 與否。

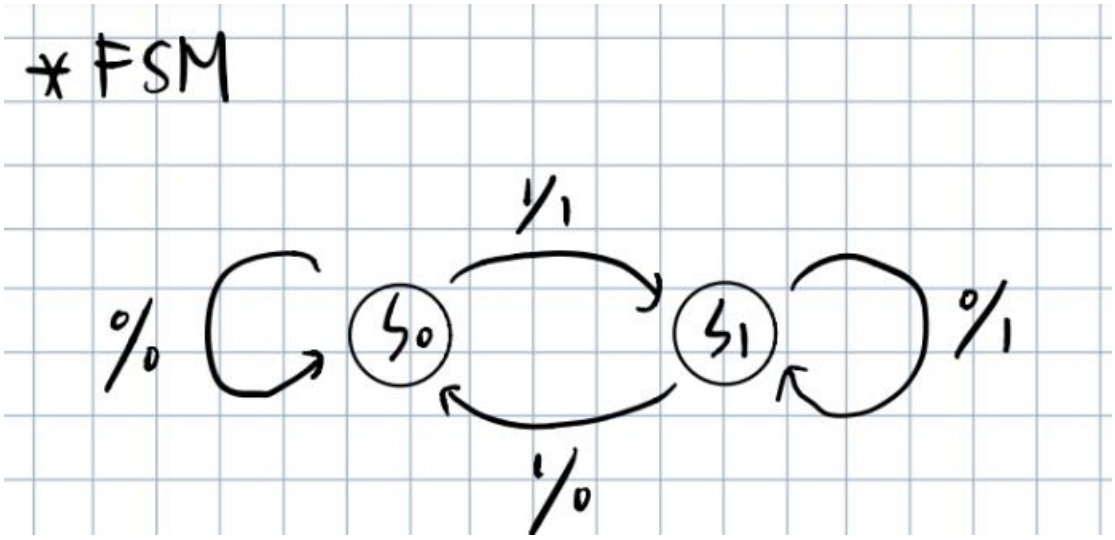
▲One-pulse



↑ 上圖是利用本身的訊號再經過 clk 經過一個 1hz 的 delay 後，利用 invert 來反轉信號，再將兩個訊號 and 起來，最後再利用一個 D flip flop 來把訊號調整到 clk posedge trigger 的位置上，即可以成為控制 down-counter 的 en，也就是 count-enable。(如下圖所示)

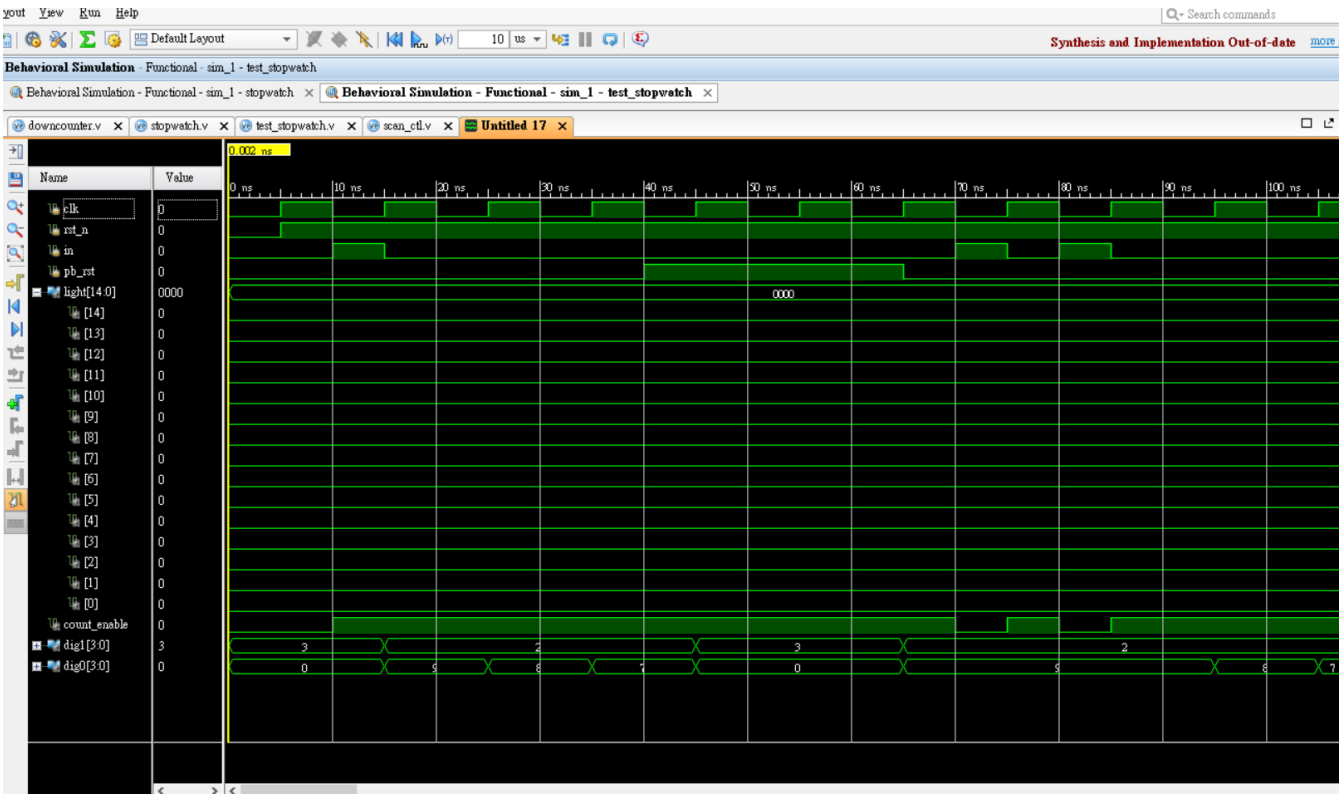


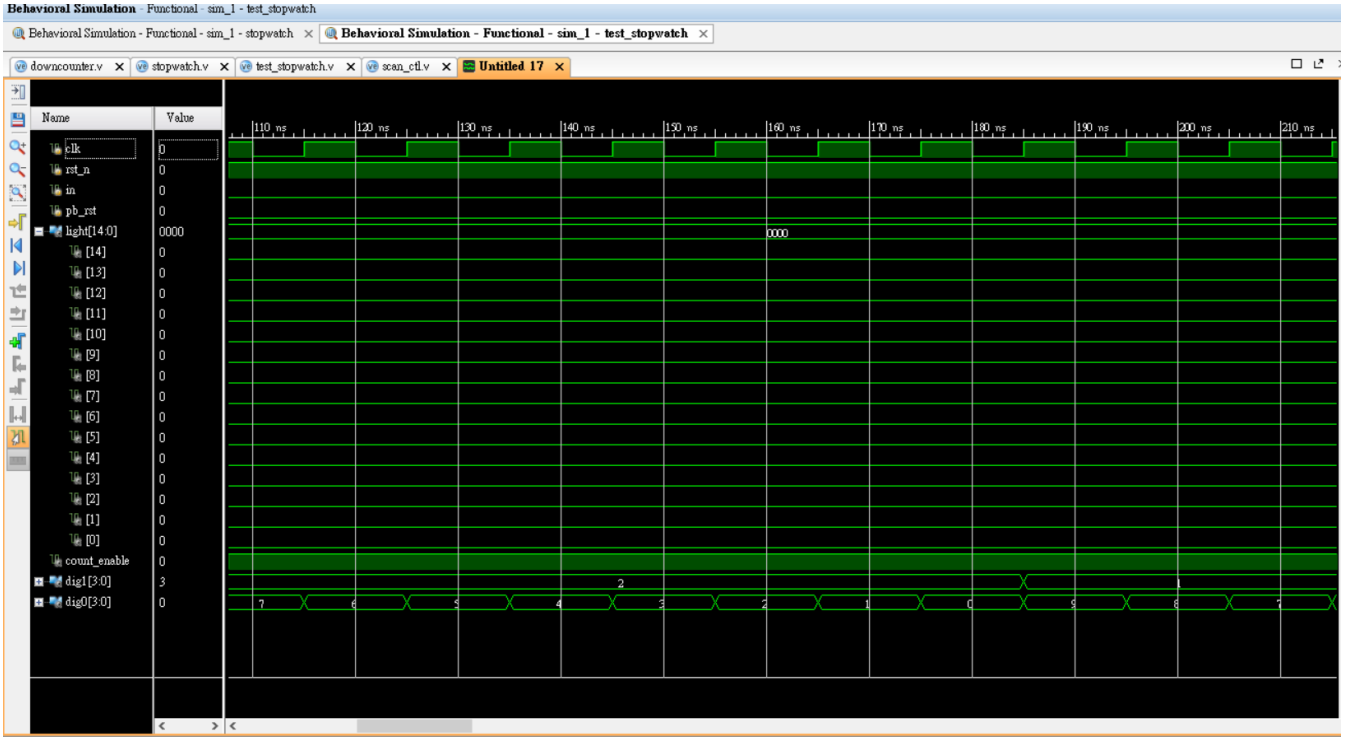
▲FSM

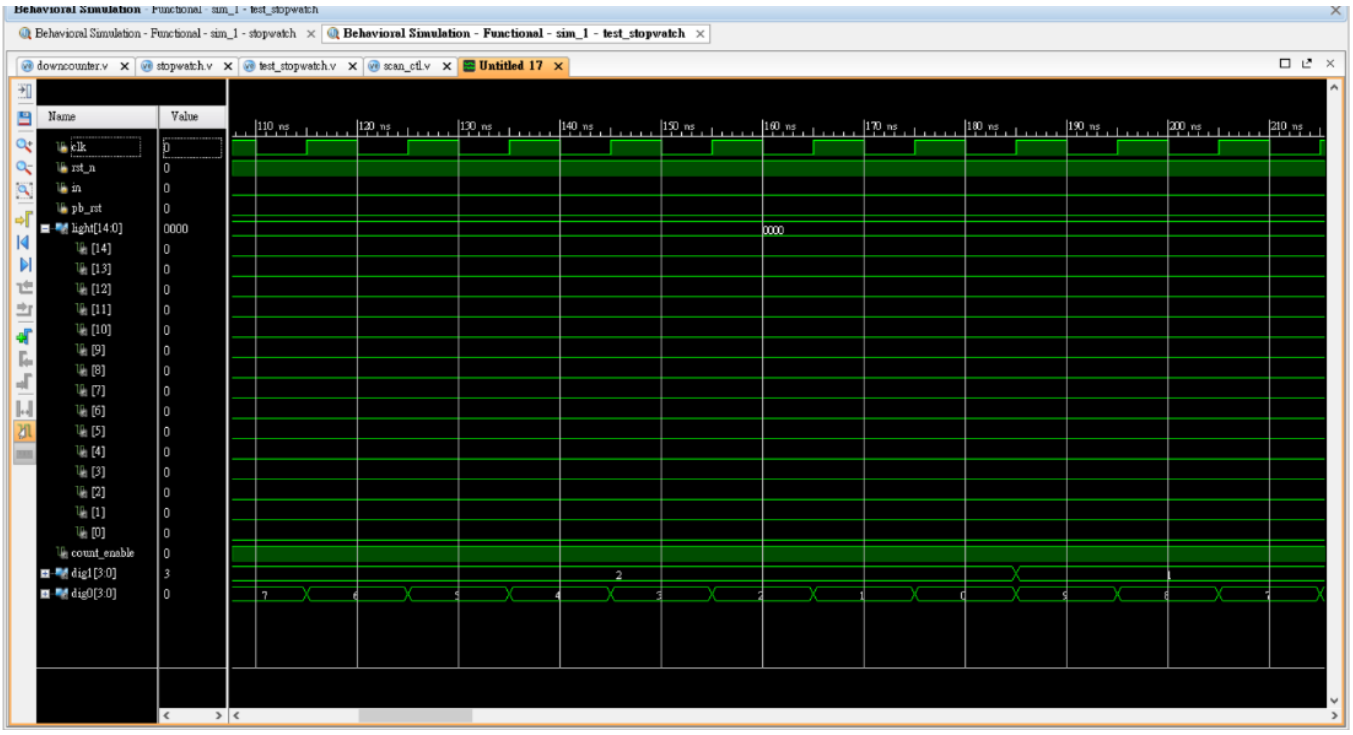


↑利用 FSM 來控制 downcounter 的狀態，S0 是暫停、S1 則是開始倒數，箭頭上的分子代表的是 input、而分母則代表的是 output，利用這個圖表及可以用 case 的語法來寫 verilog 了。(用 case 的語法，state 為括號裏面的變數，利用 input 來變換 state)

●Stimulation(1.4)







因為要利用 testbench 來檢測結果，所以我把原本寫在板子上的版本稍微改了一下，把 output 改接成 digit1、digit0 這樣才可以看到實際的數字結果。還有 clk 改成接到 fsm 的 1hz 的 clk，不然要讓 testbench 震盪原本的 100MHz 頻率有點不太實際。可以看到結果與原本預期的一樣，當 pb_rst trigger 的時候就重新從 30 開始數；到最後 00 的時候，所有的 light 都亮了。

●Discussion

我這次的作業是把老師 ilms 上面給的程式碼合起來，再加上適當的修改，我覺得值得一提的是原本的 stopwatch 並沒有接上 one pulse 的功能，所以當你按下 stop/start 的按鈕的時候，過一秒會暫停，持續按著，再經過一秒還會重新再繼續動，也就是說他持續在 State S0 和 State S1 左右變動，因為 input 一直是 1，所以我在這個作業的 in 就加上 one pulse，所以不管按多久都會是只算是一個 pulse。另外我在 pb_rst 沒有加上 debounce 還有 one pulse 的功能原因是並不需要，因為 pb_rst 並不像是 stop/start 的功能一樣是兩種 state 要利用 fsm 來描述，而是一種按了就回歸原本狀態的功能，所以不需要上述功能。

●Conclusion

這次的 prelab 也是讓我吃足苦頭，原本想利用之前自己打的程式碼來修改，但是用了三個晚上都弄不出來，一直是卡在 30，變成按一下倒數成 29 的，再按一下變成 28 的效果，但是一直 debug 不出來，所以最後改成用老師的程式碼來改，結果卻成功了，讓我很納悶……會在實驗課的時候詢問助教的。