Lab5 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)

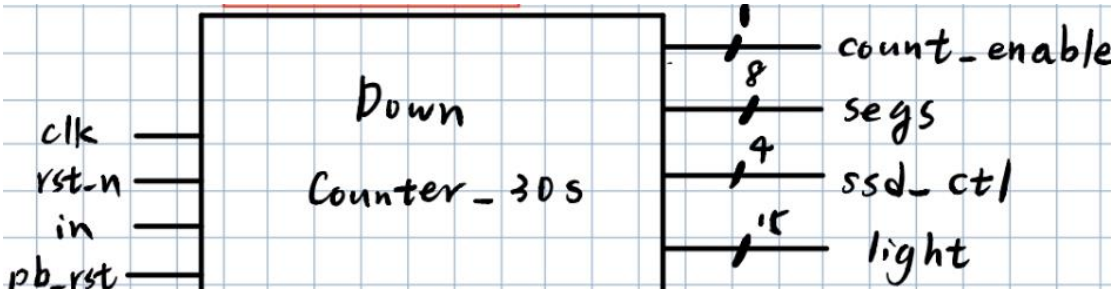
Output:

[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

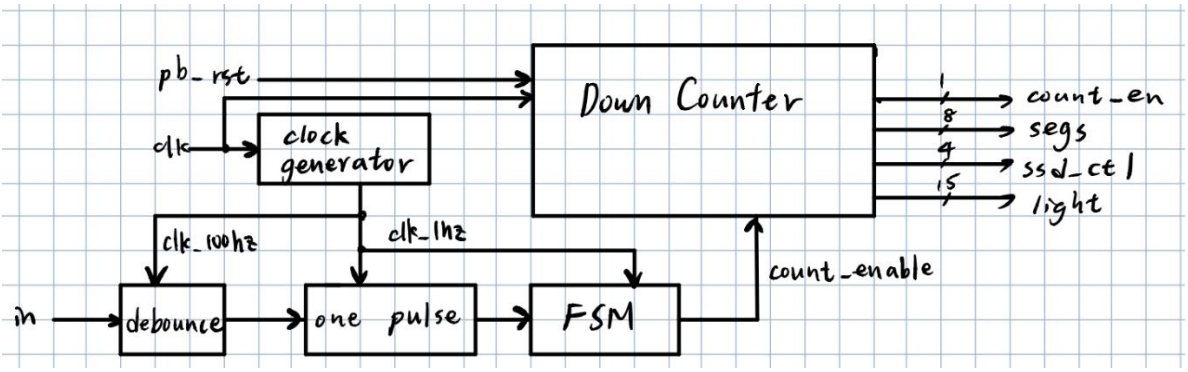
Light 是當倒數器歸零的時候 15 個燈會亮

●Design Implementation(1.1)

1.Outline

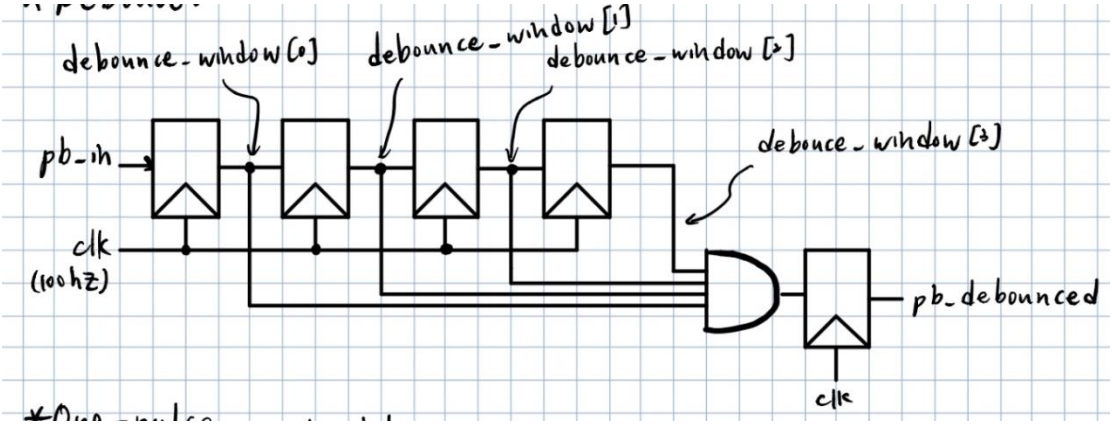
這個實驗主要是要利用按鈕來操作暫停與啟動還有歸零的功能，因為按鈕不像是開關，可以直接向上或是向下扳來提供穩定 0 或 1 的 input；按鈕是操作者按下去一下就要對整個系統產生效果，但是每個人的習慣按的時間不一樣，更何況按鈕還有彈簧的特性，在使用者按下去的時候並不會出現震盪，這會讓系統不穩定，所以需要用到 debounce 來提供一個穩定輸出為 one_pulse 的訊號來解決這個問題。另外特別的是，這次利用到 fsm 來控制 down counter 的 countable 與否，因為持續倒數與暫停倒數是兩種狀態，不同的 input 而改變當時的狀態，不會有其他的狀態產生，所以非常適合使用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

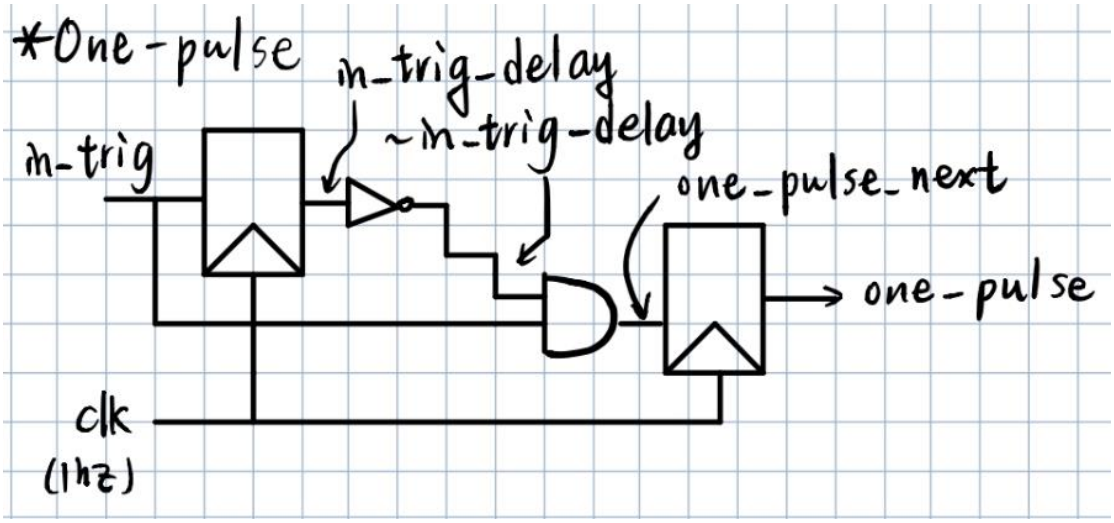
▲Debounce



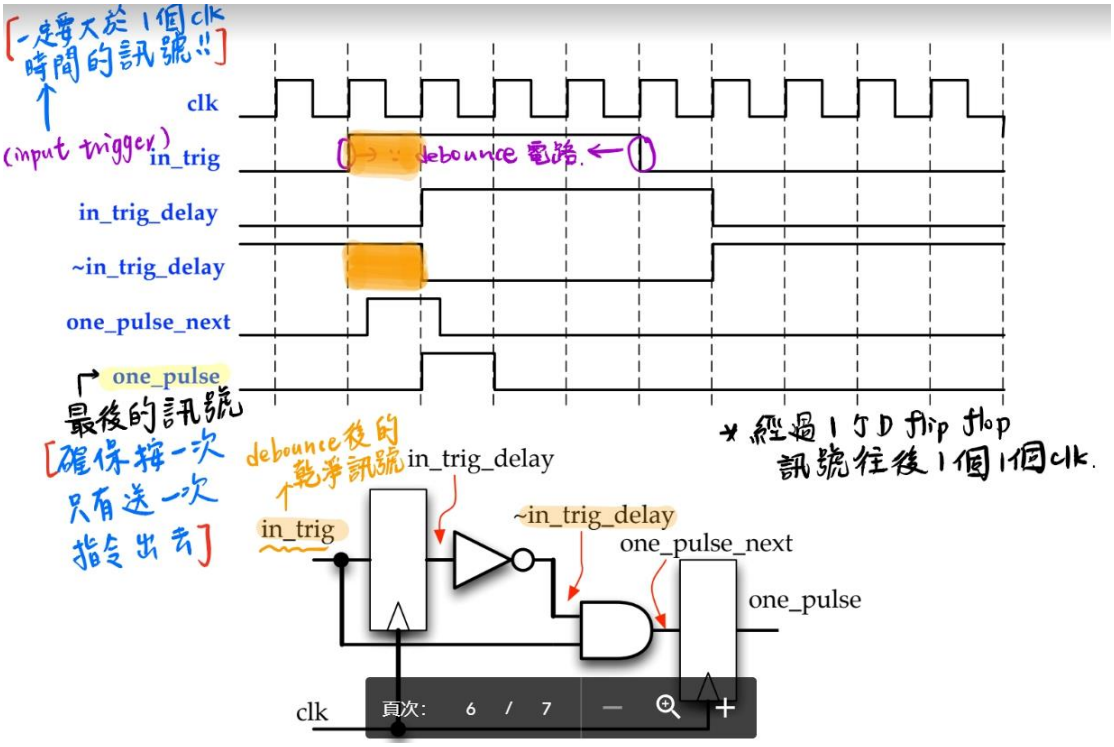
↑ 上圖是利用四個 D-flip flop 來改善按下按鈕會震盪的問題，如果把經過四個 D-flip flop 的結果全部 and 起來，就要當所有值(debounce_window[3:0])全部都等於 1 才會有一個 step 出來，後來再接上一個 D-flip flop 是為了要讓

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。

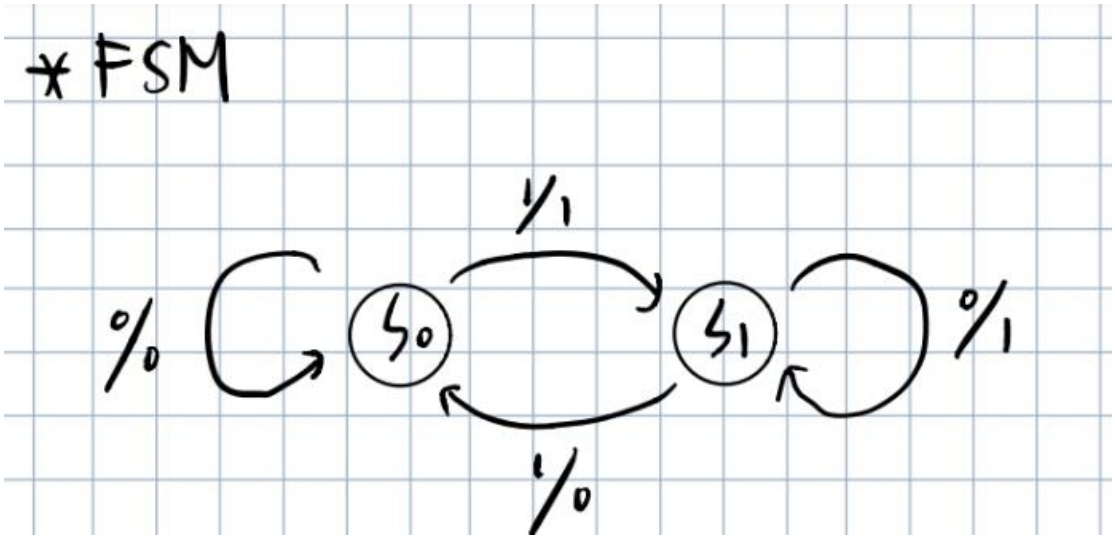
▲One-pulse



↑ 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)

↓以下是 I/O 接腳

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

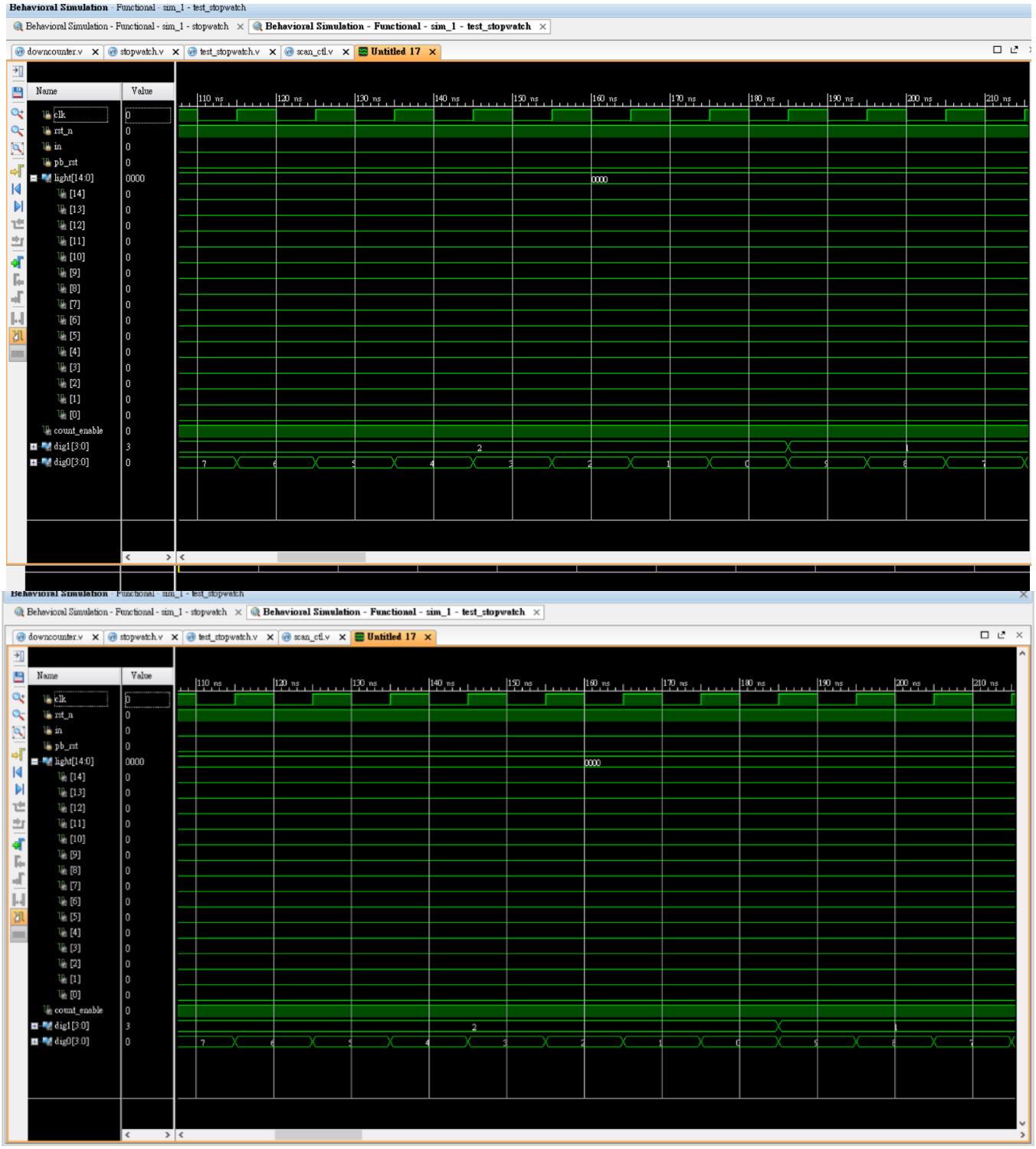
I/O	ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]
Pin	W4	V4	U4	U2

I/O	segs [7]	segs [6]	segs [5]	segs [4]	segs [3]	segs [2]	segs [1]	segs [0]	Clk	Rst_n
Pin	V14	U14	U15	W18	V19	U19	E19	U16	W5	R2

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

I/O	in	pb_rst	Count_enable
Pin	U18	U17	L1

●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。

5-2. 30_Second Down Counter with a multifunctional button

●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(短按讓倒數器暫停、開始；長按讓倒數器 reset)

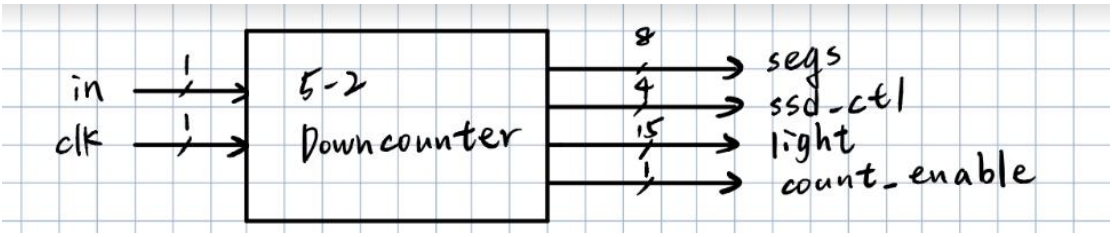
Output:

[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)



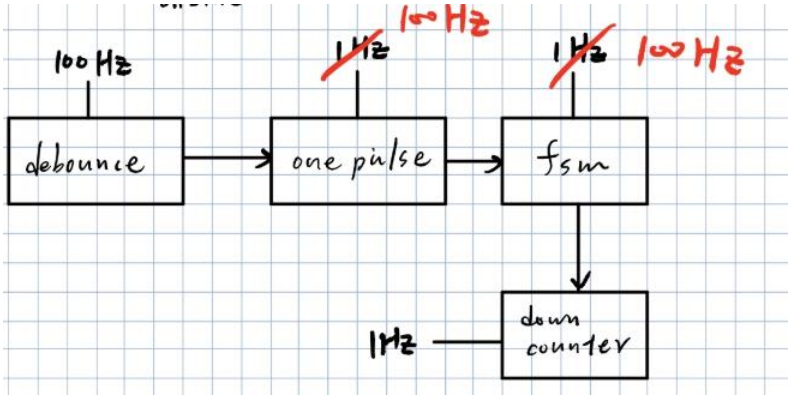
●Design Implementation

1.Outline:

這個實驗與 5-1 很像，只是要把兩個按鈕的功能合再一起，短按是原本暫停/開始的功能；長按則是 reset 的功能。

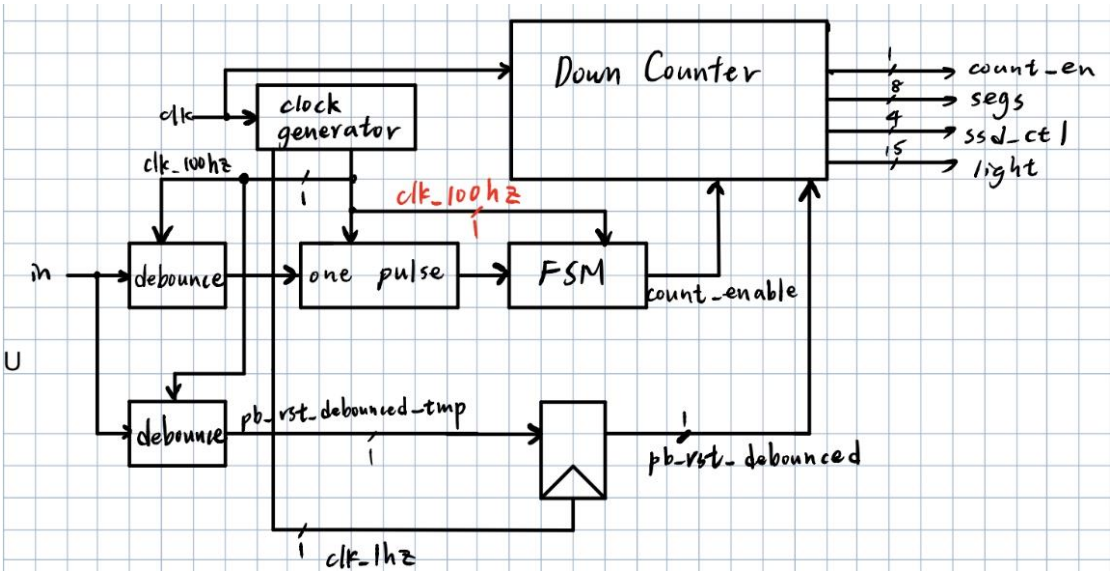
為了達到這個效果，我把原本 one pulse 還有 fsm 的 module 的 input 從原本的 1Hz 改到 100Hz，經過這個步驟，可以讓原本要按一秒才會產生反應的按鈕，變成只要輕輕按一下就會有效果了，這樣可以更容易區分長按及短按的差別。至於在 reset 功能方面，因為 reset 不用接上 fsm 還有 one pulse，為了讓他達到長按才會啟動的特性，我在該 module 後面再接上一個 clk_1hz 的 D

flip flop，利用經過 D flip flop 會延遲一個 clk(也就是 1 秒)的特性來產生長按才會觸發的效果。



←左圖說明了要把原本 5-1 中 onepulse、fsm 的 clk 改成 100hz，會讓需要讓按鈕觸發的時間變短，因為只要 fsm 能接受到 one pulse 的訊號的話，即使變換時脈效果仍然不變。

2.Logic Diagram



因為所有的 block 在前一題或是之前的報告中都有解釋過了，所以這邊只簡單說明不同之處:

因為 reset 沒有 state 的問題，所以不用接上 one pulse、fsm ，只要接上 debounce 即可，為了達到長按的特性，設 reg: pb_rst_debounced_tmp 作為暫時的變數，後面再加上一個 clk 為 1hz 的 D-flip flop，也就是說要按超過 1 秒才會 reset。

↓以下是 I/O 接腳

I/O	segs [7]	segs [6]	segs [5]	segs [4]	segs [3]	segs [2]	segs [1]	segs [0]	Clk	Rst_n
Pin	V14	U14	U15	W18	V19	U19	E19	U16	W5	R2

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

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

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

I/O	in	Count_enable
Pin	U18	L1

●Discussion

一開始完全沒有辦法想像要怎麼用同一個按鈕表現出兩個效果，還有一個問題是如果想要啟動第二個效果，但是不想要第一個的話要怎麼辦? 但是仔細想想，在這題中如果要啟動 reset 的效果的話，即使有產生暫停/開始的效果也無妨，反正最終都會回歸到 30。所以只要 delay 長按的效果即可。(很好奇能不能做出反過來的效果，也就是說把短按做成 reset、長按做成是暫停/啟動的效果，之後問問看教授或助教 XD)

5-3. 30/60_Second Down Counter with a multifunctional button

●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(短按讓倒數器暫停、開始；長按讓倒數器 reset)

pb_mode(可以切換 60s、30s 模式的按鍵)

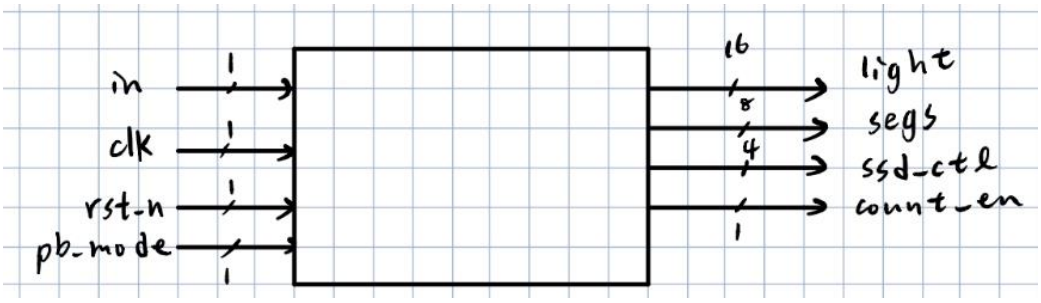
Output:

[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)



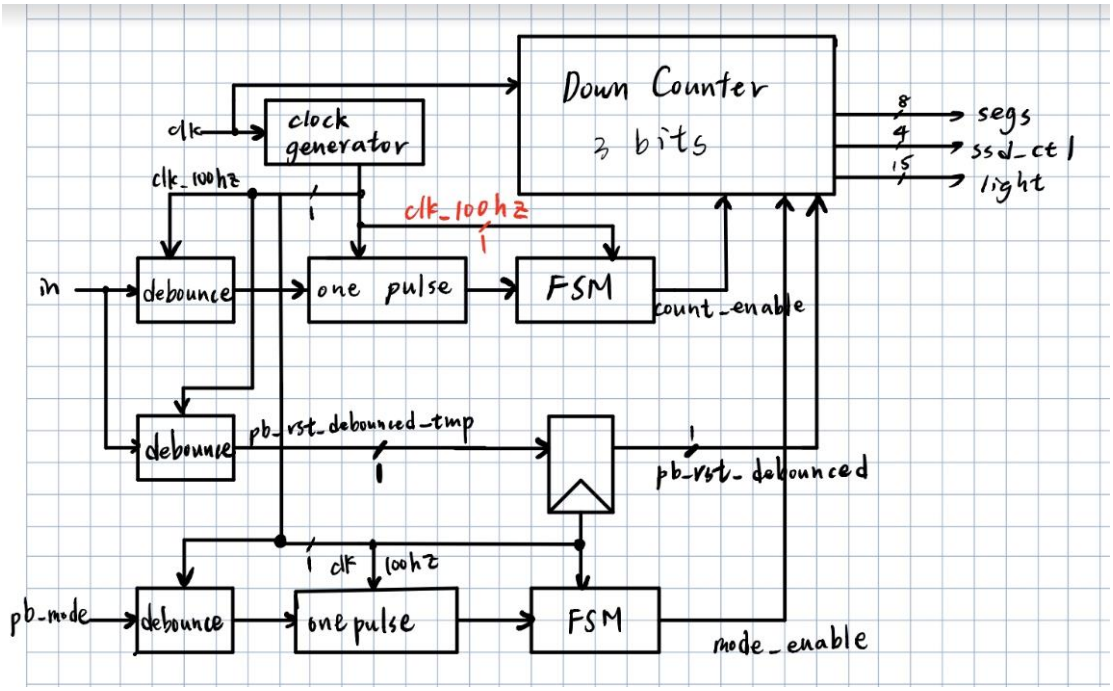
●Design Implementation

1.Outline:

這個實驗是想要另外加上一個 60 秒的模式，利用另外一個額外的按鍵，按下的時候也會順便重新 reset。

主要與 5-2 不同的地方是要另外再加上一個 debounce 還有 one pulse 來給 mode button 使用，出來的訊號再給一個 FSM 處理，分為兩個 state，當 state 是零的時候就代表著 30s 的狀態、而相反的 state 為 1 的時候就代表著是 60s 的狀態。另外，原本只有兩個 bits 的 downcounter 也要改成 3 個 bits 因為 60s 的狀態會需要用到三位數。

2.Logic Diagram



因為所有的 block 在前一題或是之前的報告中都有解釋過了，所以這邊只簡單說明不同之處:

另外加上的 pb_mode 所加上另一串的 block 所接上的所有 clk 都是 100hz 的原因也如之前所說的，只要 onepulse 和 fsm 的 clk 頻率一樣，可以讓 Fsm 接受到 one pulse 的訊號即可，所以利用 100hz 的頻率可以讓按下到啟動的時間變短，也就是輕輕按一下即可產生變換模式的效果。

↓以下是 I/O 接腳

I/O	segs [7]	segs [6]	segs [5]	segs [4]	segs [3]	segs [2]	segs [1]	segs [0]	Clk	Rst_n
Pin	V14	U14	U15	W18	V19	U19	E19	U16	W5	R2

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

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

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

I/O	in	pb_mode	Count_enable
Pin	U18	U17	L1

●Discussion

這一題讓我和我的朋友吃足苦頭 XD，因為我當初自己打的時候就花了非常多的時間，因為這個題目用到非常多的 module，而且我也不太會整理，我都是我之前的作業來傳進來慢慢改得，因此裡面有很多 top module，但是每次我 run 的時候結果都有錯誤，遲遲找不出錯誤，所以只好開大招去找大神幫我 debug，但是結果我被他臭罵了一頓，他說我的程式碼看起來非常的爛，原因是我沒有縮排、也沒有幫每個 module 取讓人可以馬上知道他功能的名稱，甚至裡面有很多功能一樣，但是名子不一樣的 module 名稱，之後我才知道最好的方式是要把每一個功能放在一個資料夾裡面，當要的時候就直接從那邊複製，而不是從之前的作業直接拿出來，因為第一名稱會非常的奇怪、第二是裡面的變數名稱可能與你新作的題目不一樣，讓其他人非常難以理解。

●Conclusion

原本想要第一天下課的時候，就直接把 lab5 給完成，但是花了將近 4 小時卻連第一題的 prelab 都沒有完成，一開始是卡在 reset 的問題，我原本以為是整個系統只有一個 reset 但是這樣要怎麼給 pb_rst 一個 reset 呢……?也是請教其他之前修過的同學才能理解大概要做的方向，下次不能再這樣先傻傻的直接去做，會花非常多冤枉的時間。

另外，與其他同學聊天的時候聊到，他們之前做的 final 成果有猜拳、甚至是俄羅斯方塊，感覺都非常困難……，都需要接上螢幕、鍵盤才可以完成，甚至還有音效，希望我也能成功完成……