Lab5_1

Design Specification

✓ For a 30-second down counter with pause function:

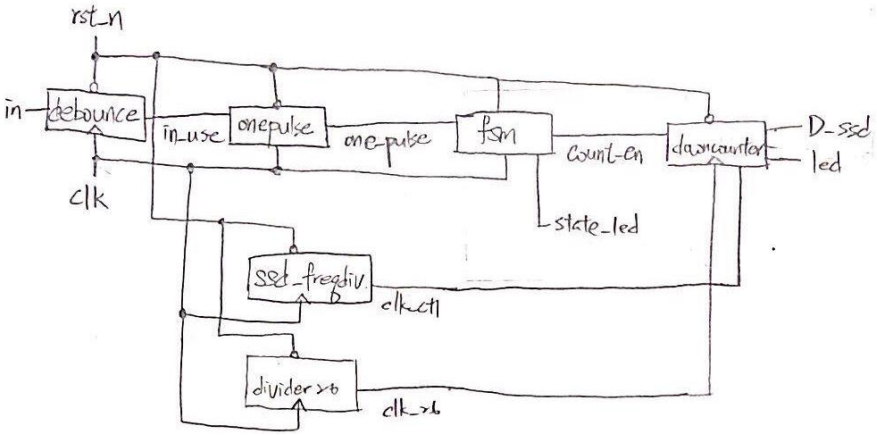
Input:

in // control start/pause button
rst_n // control rst_n button
clk

Output:

D_ssd[7:0] // 7-segment display
d[3:0]
led[14:0] // light up when count to 0
state_led // show the state

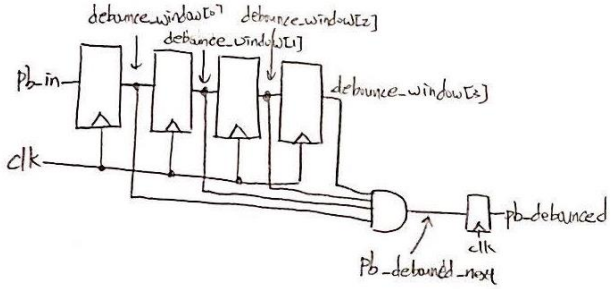
✓ Draw the block diagram of the design.



Design Implementation

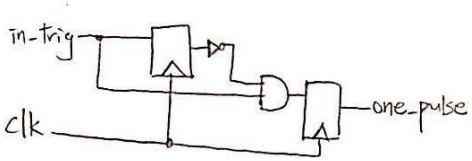
✓ 本題由 debounce、onepulse、fsm、ssd_freqdiv、divider26、downcounter 六個 module 組成

✓ Debounce



Debounce 是為了避免按按鍵時所產生的浮動訊號而設計，藉此得到穩定的訊號。每經過一個 clk，下一個 input 就會進來。當 debounce_window 皆為一時，下個 output 值為 1，藉此達到穩定的波形。

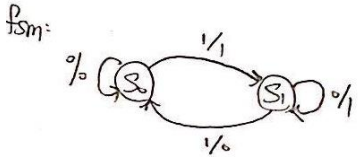
✓ Onepulse



其目的為製造一個 button 的訊號，讓按下 button 後可以一直保持在下個 state，而不用一直按著 button。當 in_trig 進入 Dff 後，在下一個 clk 會得到一個 not 的訊號，與原本的 in_trig

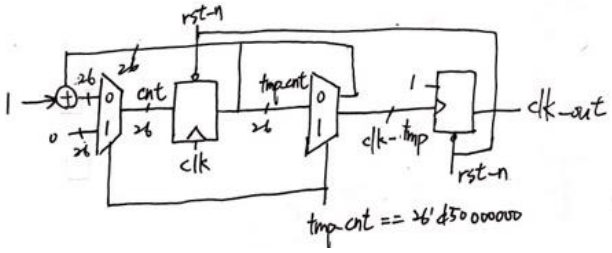
and 後再經過一個 Dff，便會得到 one_pulse 訊號。

✓ Fsm



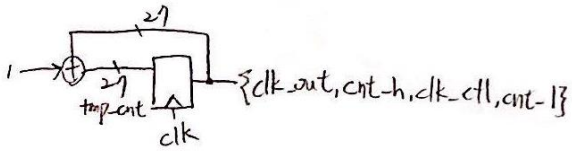
Input：rst_n, clk, one_pulse
Output：state_led, count_en
此為控制倒數計時器開始與停止的 module，在我的設計中，當 one_pulse 為 1 時，state 會做切換，S0 為停止，S1 為開始。而當狀態為 S1 時，state_led 為 1。

✓ Divider26



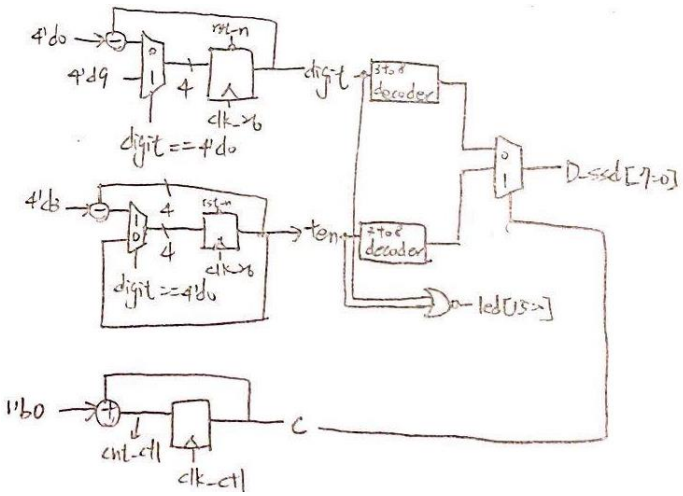
將原本 clk 的頻率除頻為 1Hz

✓ ssd_freqdiv



擷取中間的頻率以達到視覺暫留的效果，讓 7-segment display 看起來同時顯示不同的數字。

✓ downcounter



將十位數與個位數分開做，並利用 digit 的值判斷是否繼續減或輸入下一個值。另外，用 c 控制 7-segment display 輸出十位數或個位數。

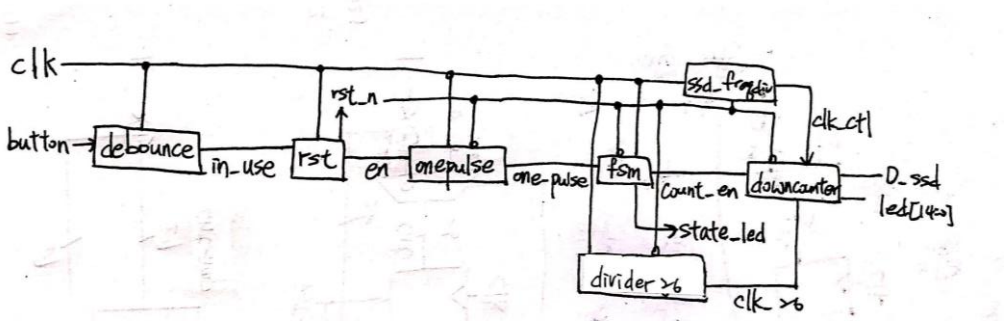
✓ I/O pin

I/O	in	rst_n	clk	d[3]	d[2]	d[1]	d[0]	state_led
VOC	U18	T17	W5	W4	V4	U4	U2	U16
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]
VOC	W7	W6	U8	V8	U5	V5	U7	V7
I/O	led[14]	led[13]	led[12]	led[11]	led[10]	led[9]	led[8]	led[7]
VOC	L1	P1	N3	P3	U3	W3	V3	V13
I/O	led[6]	led[5]	led[4]	led[3]	led[2]	led[1]	led[0]	
VOC	V14	U14	U15	W18	V19	U19	E19	

Lab5_2
Design Specification

✓ For a 30-second down counter with pause function:
Input: button, clk
Output: D_ssd[7:0], d[3:0], led[14:0], state_led

✓ Draw the block diagram of the design.



Design Implementation

✓ 此設計與 lab5-1 不同的是，reset 與 in(開始或停止)是由同一個 button 控制。因此我在此設計中多加了一個 rst 的 module，在其中加入一個 counter，當長按 button 兩秒時，rst_n 等於 1。而 rst_n 與 in 為反向，因此 button = in & (~rst_n)

✓ I/O pin

I/O	button	clk	d[3]	d[2]	d[1]	d[0]	state_led	
VOC	U18	W5	W4	V4	U4	U2	U16	
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]
VOC	W7	W6	U8	V8	U5	V5	U7	V7
I/O	led[14]	led[13]	led[12]	led[11]	led[10]	led[9]	led[8]	led[7]
VOC	L1	P1	N3	P3	U3	W3	V3	V13
I/O	led[6]	led[5]	led[4]	led[3]	led[2]	led[1]	led[0]	
VOC	V14	U14	U15	W18	V19	U19	E19	

Discussion

這次的兩個題目很類似，唯一的差別為如何控制 rst_n 與 in。再跟同學討論後知道其實有很多種做法可以同時用一個 input 控制，例如用 debounce；而我則是選擇比較直觀的利用 counter 控制按幾秒為 reset 的 input。

另外，我發現 debounce 的設計其實滿容易被忽視的。因為一開始設計第一題時是由 prelab 改寫

的，因此沒有加入 debounce 的設計，在測試的時候也沒有感覺到明顯不同。這部分可能需要在請問教授或助教，在哪種情況下 debounce 的有無會導致明顯差別。

Conclusion

這次題目是由很多 module 所組成的設計，讓我了解到之前一次次實驗所累積的成果的重要性。其實題目都是由很簡單的 module 組成，真正實驗的重點在於如何設計與連接各個 module。

