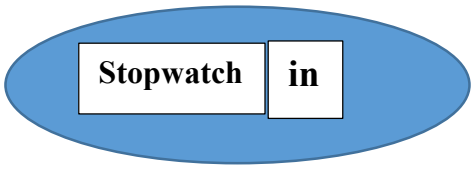
1.
Design Specification

✓ **Input / Output**

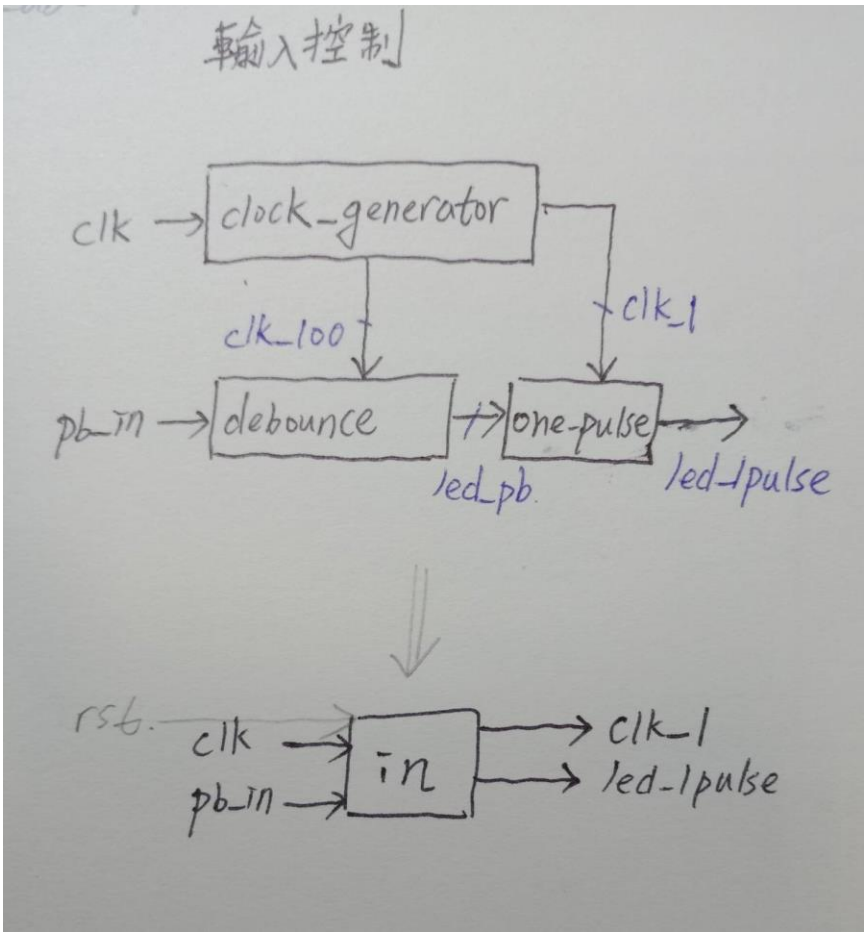
For a 30-second down counter with pause function:
Input: clk, rst, pb_in.
Output: D_ssd[7:0], ssd_ctl[3:0], led[15:0].

✓ **Block Diagram**

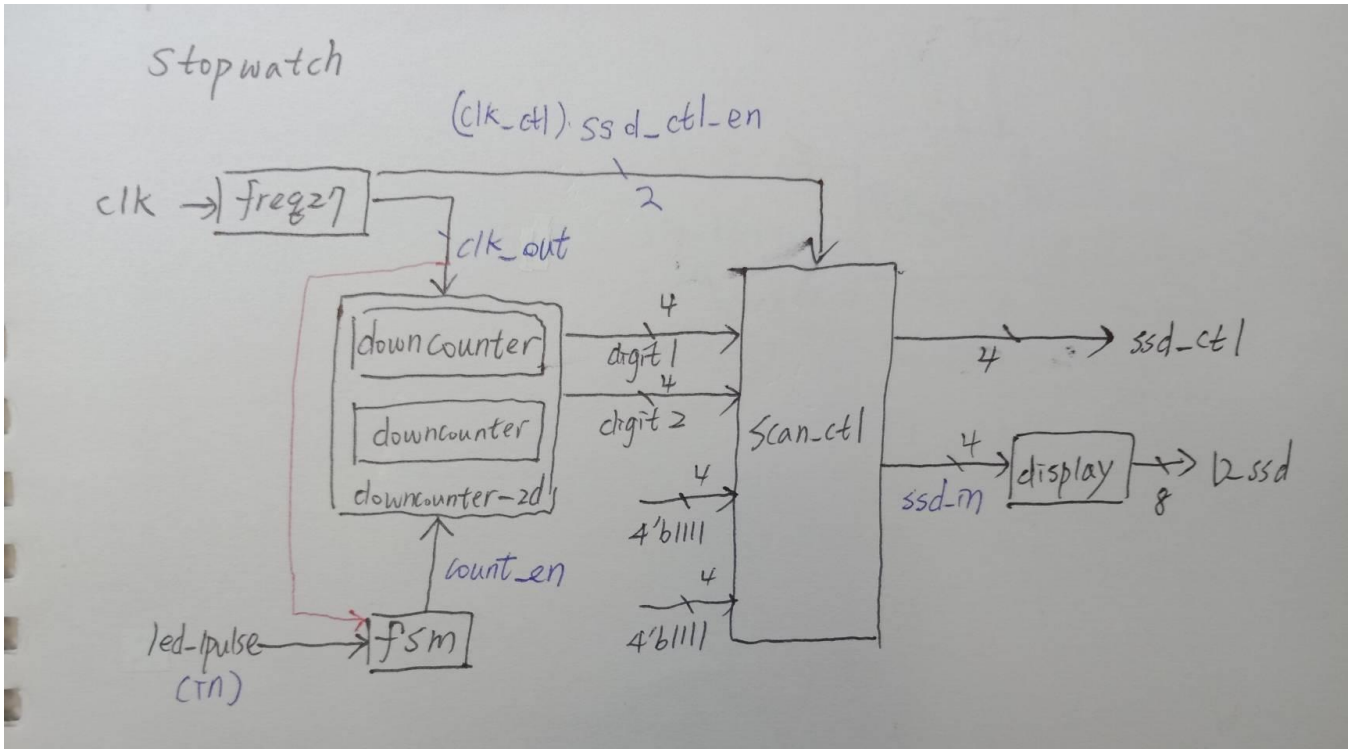
- **top**



- **In**



- **stopwatch**



Design Implementation

✓ **Function Descriptions**

此功能為從 30 數到 0 的倒數計時器，並且用兩個 push button 作為 reset 與 pause/start 功能。我把整個系統分為輸入控制與核心功能(stopwatch)來做。輸入控制的部分，有除頻器、debounce 與 one pulse。除頻器輸出 100hz 與 1 hz 分別輸入 debounce 與 one pulse.

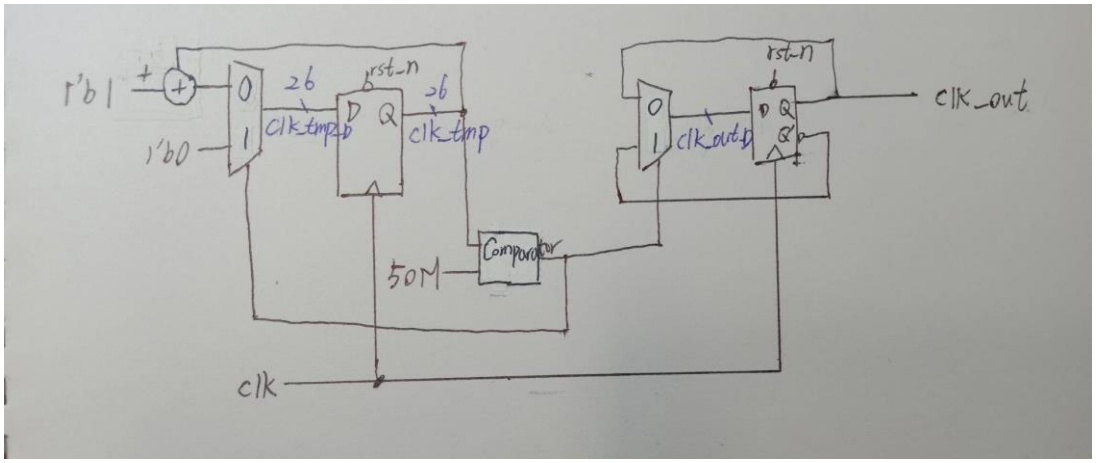
Debounce 的做法為 在震盪的波用 register 拉出四個值把他 AND 在一起，把震盪的那邊全部變成 0.one pulse 的做法為 將原本的波延遲一個 clk 再做 inverse ,接著將他跟原本的波型做 AND 就會出現一個 clk 的 one pulse.

Stopwatch module 包含了除頻器，downcounter, fsm, scan_ctl, display。除了 fsm，其餘功能皆在之前的實驗做過，因此這裡解釋一下 fsm 怎麼做。除頻器輸入 1hz 的頻率到 fsm，這裡很重要的是輸入的頻率需與輸入到 one pulse module 的頻率相同。接著在 fsm 裡，有兩個 state，在 pause state 裡若 in 為 1，則進入到 start state, 且輸出 count_en = 1，若 in=0 則維持現狀，若是在 start state 裡 in 為 1 則進入到 pause state, count_en = 0 若 in = 0 時則維持在原狀態。用 mux 來決定輸出為甚麼。Fsm 輸出的 count_en 會輸進 downcounter 當作要不要數的條件。

Led 亮燈的控制也用 mux。若 digit0 digit1 同時為零，則燈全亮，且停止倒數。

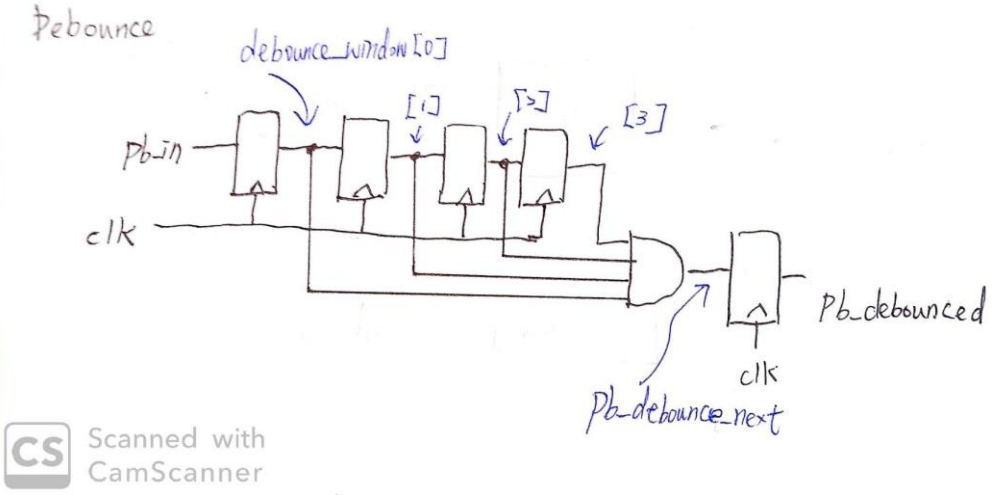
✓ **Logic Diagram**

- **clk_genertor**

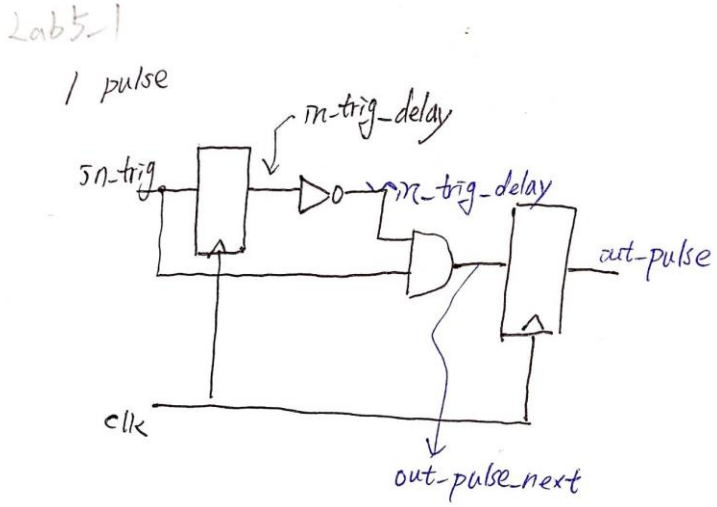


PS.把圖中 comparator 的輸入 50M 改成 5000K 就可以變成輸出為 100HZ

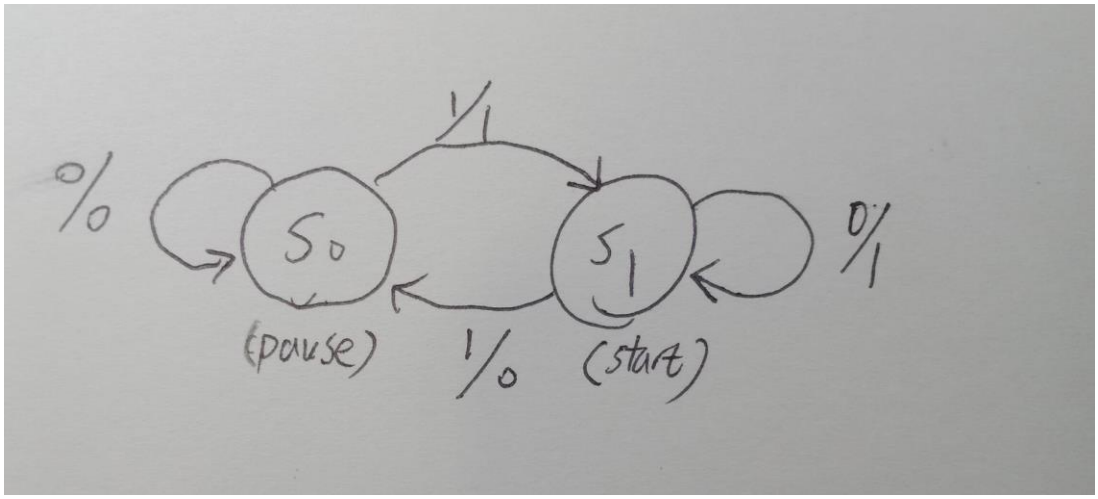
- **debounce**



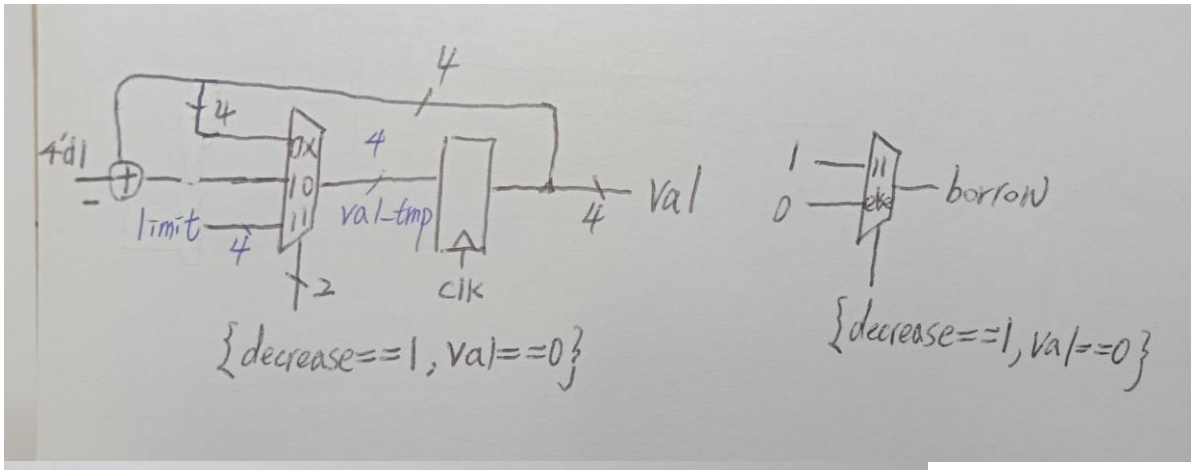
- **one pulse**

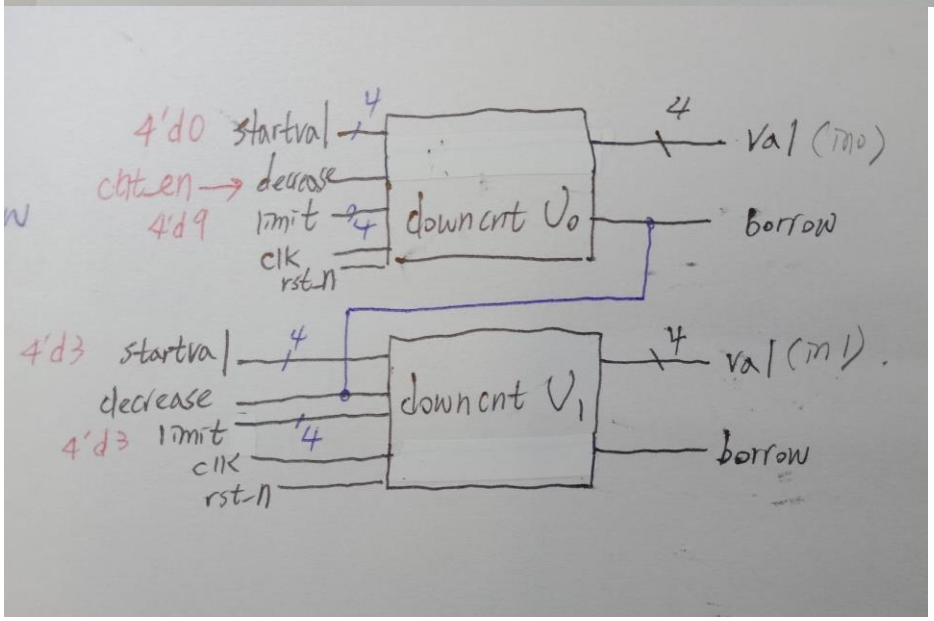


- **fsm**



- **downcounter**





✓ **Function Table**

- **fsm**

Present state	**Input**	**Next state**	**Output**
	in		**count_en**
0	**0**	**0**	**0**
0	**1**	**1**	**1**
1	**0**	**1**	**1**
1	**1**	**0**	**0**

✓ **I/O pin**

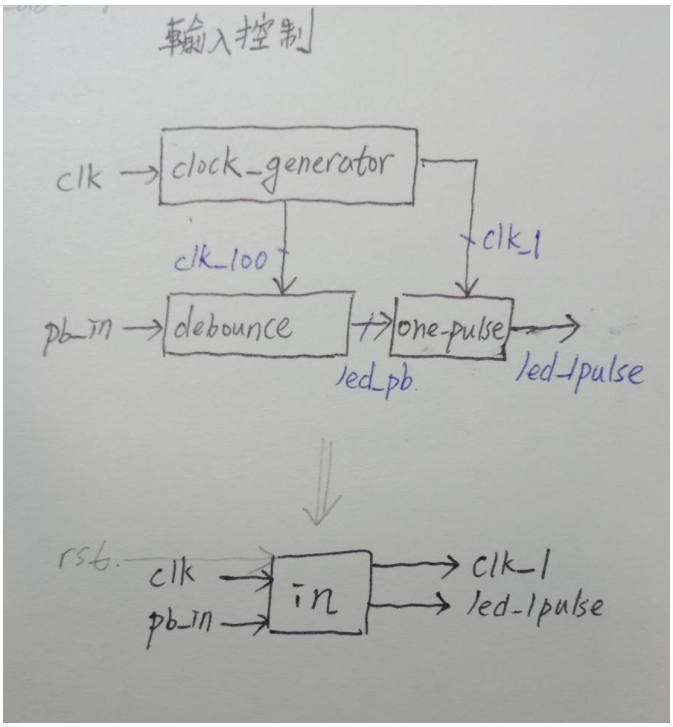
I/O	clk	rst	pb_in	ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]	
LOC	W5	U18	T17	U2	U4	V4	W4	
I/O	D_ssd[7]	D_ssd[6]	D_ssd[5]	D_ssd[4]	D_ssd[3]	D_ssd[2]	D_ssd[1]	D_ssd[0]
LOC	W7	W6	U8	V8	U5	V5	U7	V7
I/O	led[0]	led[1]	led[2]	led[3]	led[4]	led[5]	led[6]	led[7]
LOC	U16	E19	U19	V19	W18	U15	U14	V14
I/O	led[8]	led[9]	led[10]	led[11]	led[12]	led[13]	led[14]	led[15]
LOC	V13	V3	W3	U3	P3	N3	P1	L1

2.
Design Specification

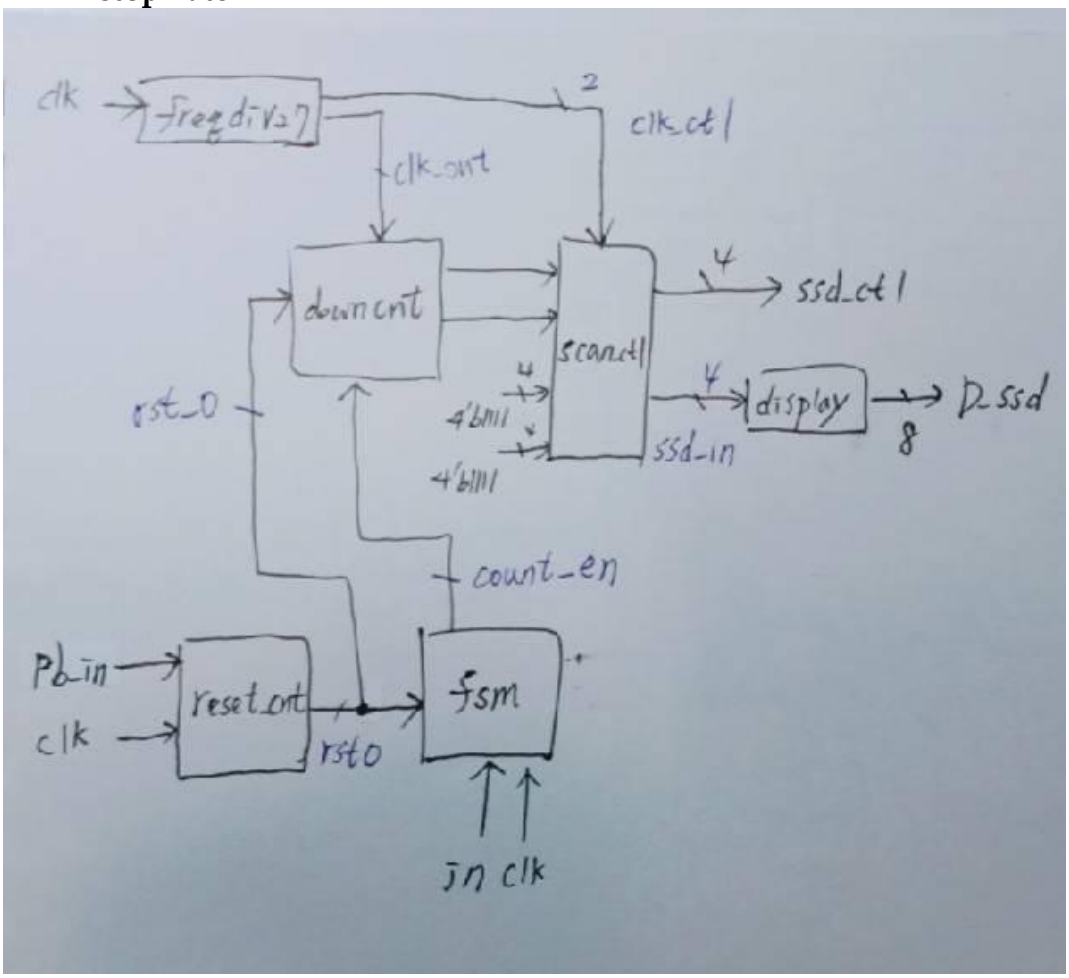
- ✓ **Input / Output**

  For a 30-second down counter with pause function:
  Input: clk, pb_in.
  Output: D_ssd[7:0], ssd_ctl[3:0], led[15:0].

- ✓ **Block Diagram**
  - **in**



- **stopwatch**



Design Implementation

✓ **Function Descriptions**

此題與上題唯一的不同在於只用一個 push button 來控制 reset, pause , start.因此我在這題多加了一個 reset_cnt 來處理 push button 的 input 要為 reset 還是 pause/start。做法為，用一個 counter，當(push button input) in=1 時開始數，數到 150M 時，rst_en = 1 啟動 reset。若沒有數到 150M 則 rst_en = 0，沒有 reset, 則 in 就是 pause/start 的功能。其他的功能都與上題相同。

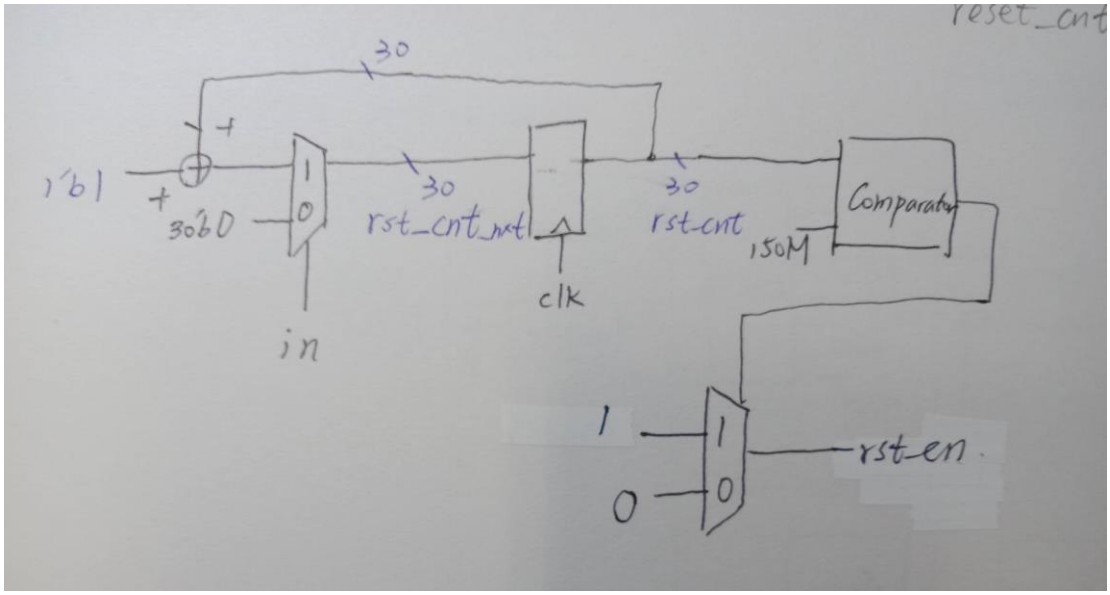
✓ **Logic Functions**

- **reset_cnt**

```
if (in)        rst_cnt_nxt = rst_cnt + 1;
else         rst_cnt_nxt = 30'b0;
if (rst_cnt == 30'd150000000)
      begin
         rst_en <= ~rst_en;
         rst_cnt<=30'd0;
      end
```

else
rst_cnt <= rst_cnt_nxt;

- ✓ **Logic Diagram**
  - **reset_cnt**



- ✓ **I/O pin**

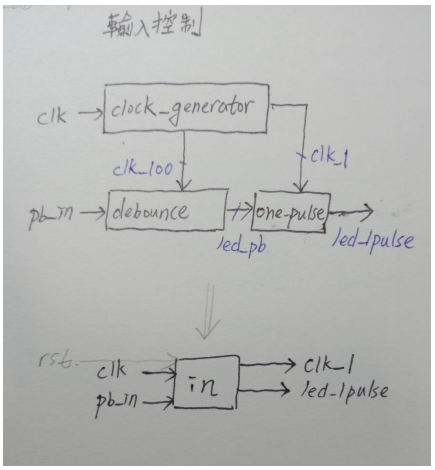
I/O	clk	pb_in	ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]		
LOC	W5	T17	U2	U4	V4	W4		
I/O	D_ssd[7]	D_ssd[6]	D_ssd[5]	D_ssd[4]	D_ssd[3]	D_ssd[2]	D_ssd[1]	D_ssd[0]
LOC	W7	W6	U8	V8	U5	V5	U7	V7
I/O	led[0]	led[1]	led[2]	led[3]	led[4]	led[5]	led[6]	led[7]
LOC	U16	E19	U19	V19	W18	U15	U14	V14
I/O	led[8]	led[9]	led[10]	led[11]	led[12]	led[13]	led[14]	led[15]
LOC	V13	V3	W3	U3	P3	N3	P1	L1

3.
Design Specification

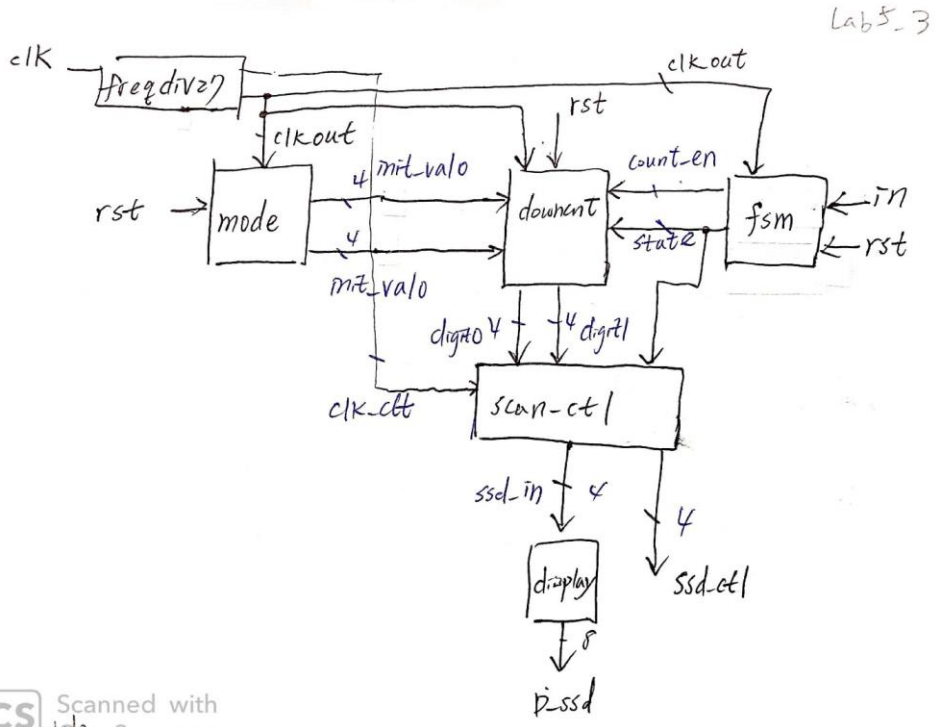
- ✓ **Input / Output**

  For a multi-function timer which has two modes: 30-second/1-minute countdown:
  Input: clk, rst, pb_in.
  Output: D_ssd[7:0], ssd_ctl[3:0].

- ✓ **Block Diagram**
  - **in**



  - **stopwatch**



Design Implementation

✓ **Function Descriptions**

此題為一個可以數 1 分鐘倒計時或者 30 秒倒計時的多功能倒數計時器。中間倒計時的方法都與第一題相同，不同在於一開始要從哪倒數。因此我做了一個 mode 的 module，為一個 finite state machine，用來選擇輸出到 downcounter 的初始值應該為多少。若現在的 state 在 30s，若 rst=1 則 state 轉換到 1min, 初始值為 0 0，rst=0 狀態不變。若現在在 1min 的 state, rst=1 則 state 轉換到 30s, 初始值為 3 0，rst=0 狀態不變。接著初始值輸入到 downcounter 中，在之前的題中若初始值為 00 不會倒數，但現在 count enable 會多做一個判斷，若 state 在剛開始的 pause 狀態則可以從 00->59 開始倒數。

在 scan_ctl 也有針對 1:00 的 1 甚麼時候要顯示做判斷，判斷為若兩個 digit0, digit1 皆為 0 且狀態在 pause(剛開始時)要顯示 1,其餘皆不顯示。

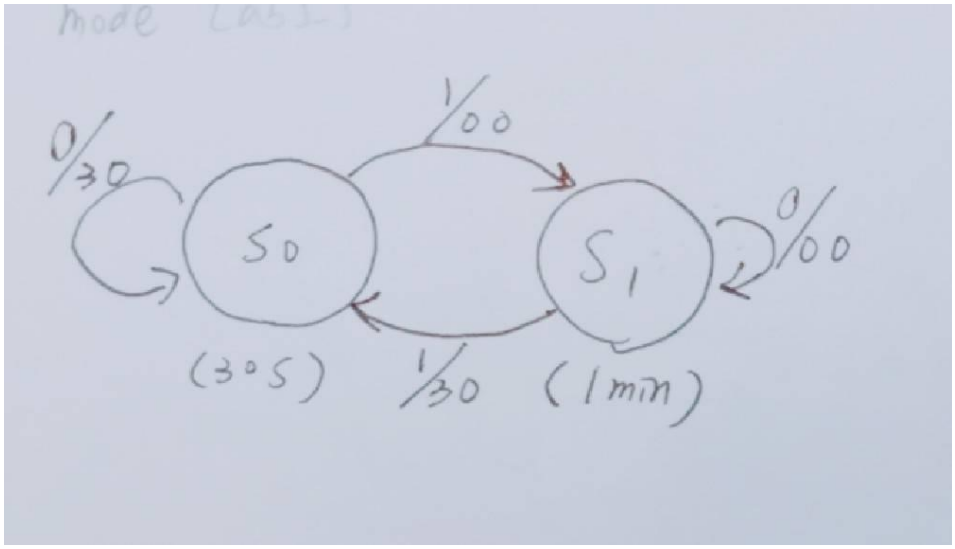
✓ **Function Table**

- **mode**

Present state	**Input**	**Next state**	**Output**	
	rst		**init_val0[3:0]**	**init_val1[3:0]**
0	**0**	**0**	**3**	**0**
0	**1**	**1**	**0**	**0**
1	**0**	**1**	**0**	**0**
1	**1**	**0**	**3**	**0**

✓ **Logic Diagram**

- **mode**



- ✓ **I/O pin**

I/O	clk	rst	pb_in	ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]	
LOC	W5	U18	T17	U2	U4	V4	W4	
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]
LOC	W7	W6	U8	V8	U5	V5	U7	V7
I/O	led[0]	led[1]	led[2]	led[3]	led[4]	led[5]	led[6]	led[7]
LOC	U16	E19	U19	V19	W18	U15	U14	V14
I/O	led[8]	led[9]	led[10]	led[11]	led[12]	led[13]	led[14]	led[15]
LOC	V13	V3	W3	U3	P3	N3	P1	L1

Discussion

這次的實驗，過程中實在是辛酸滿滿。感覺都煩惱到白頭髮長出來了。
從 lab5_1 就開始受挫，寫完是一回事，debug 又是一回事。我的問題是按下 push button 可是他完全不理我。我一開始先檢查 stopwatch，檢查方式是把 input 從 button 改為 DIP switch，發現他可以正常運作，這時我就想說應該是 one pulse 或 debounce 的問題，可是檢查了好久都沒問題，也試過不要把 push button 經過處理，甚至這樣都可以正常運作。最後的答案再過了一個禮拜的課，教授說 one pulse 跟 fsm 要接一樣的 clk! 我從來都沒有考慮過 clk 的問題，果真改了就可以運作了!
接下來的實驗我都有思考 clk 的問題，果真有幫助到很多，這次的實驗花了我許多心血也打擊了我很多，但反過來想這就是在訓練我們受挫卻能再站起來的能力。況且以後只會越來越難，要把心變成鋼鐵之心!

Conclusion

這次實驗最主要學到如何處理 push button 的問題，而經過這次比較複雜的問題也更清楚 module 間的連結與合作。要注意 clk 的問題，fsm 跟 one pulse 的 clk 必須要一樣，不然 fsm 會抓不到訊號。
長按短按可以用一個 clk 去計算做區別功能。Mode 的轉換可以用 finite state machine 的方法來做。