**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:

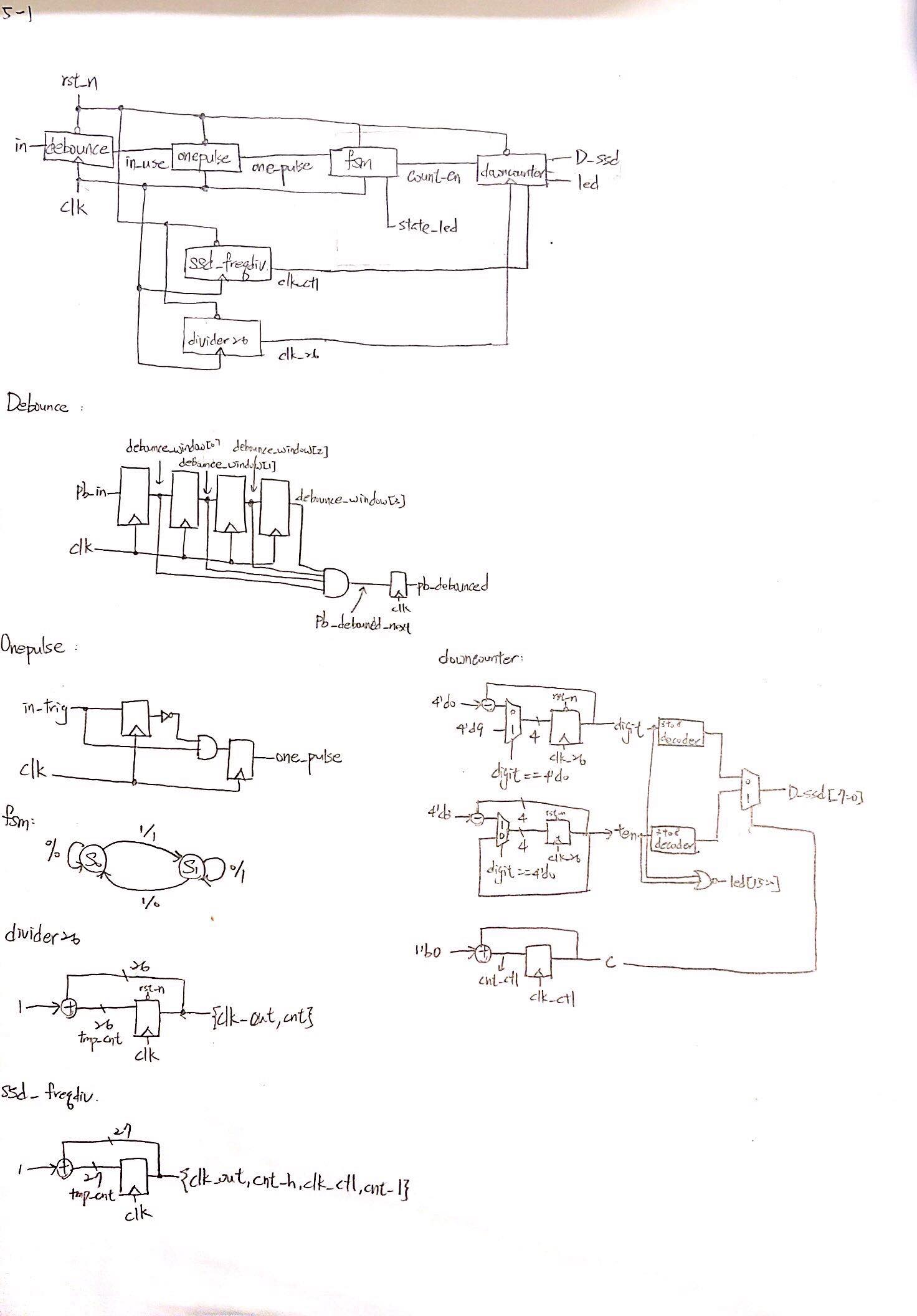
D\_ssd[7:0] // 7-segment display

d[3:0]

led[14:0] // light up when count to 0

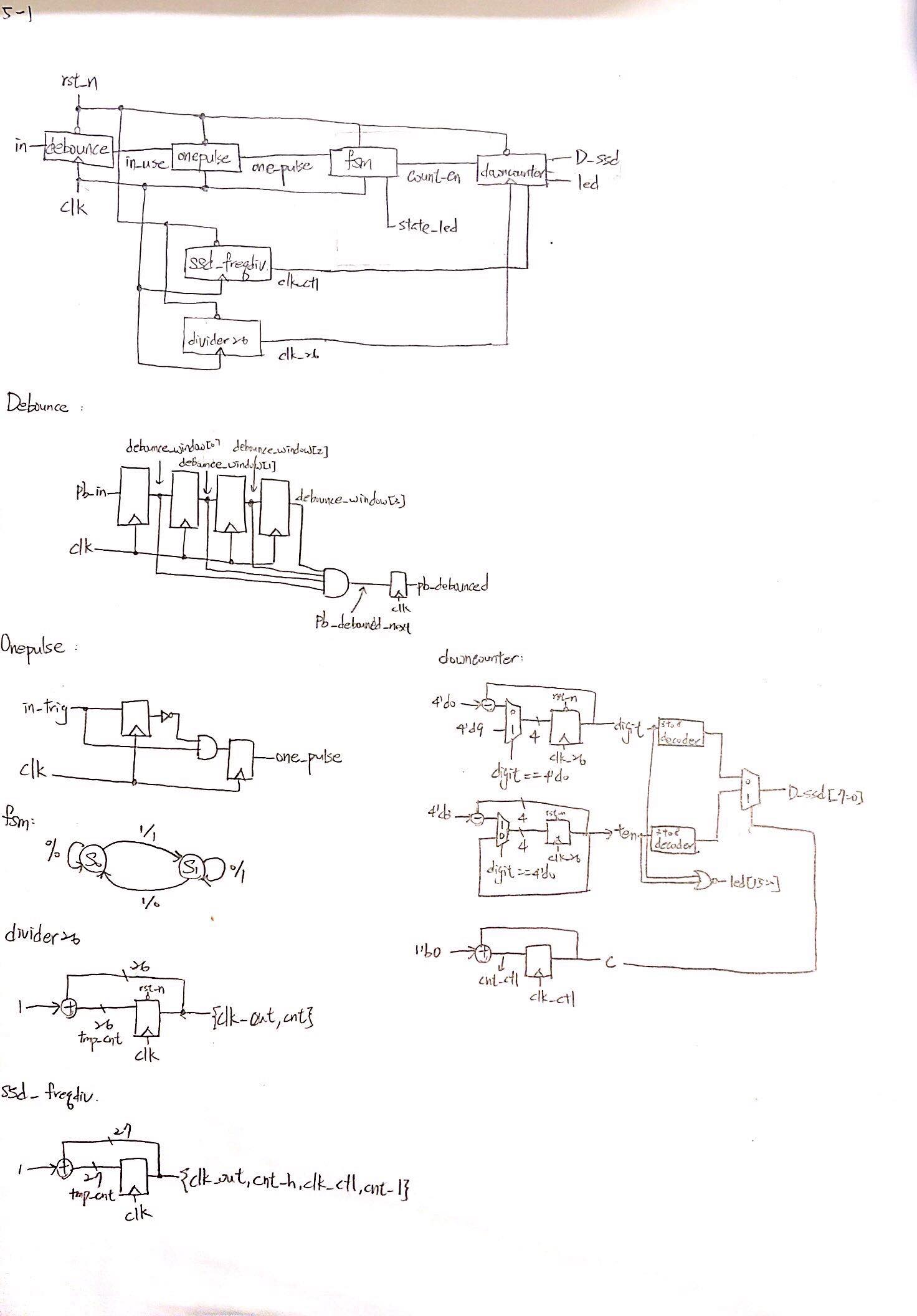
state\_led // show the state

* Draw the block diagram of the design.

****

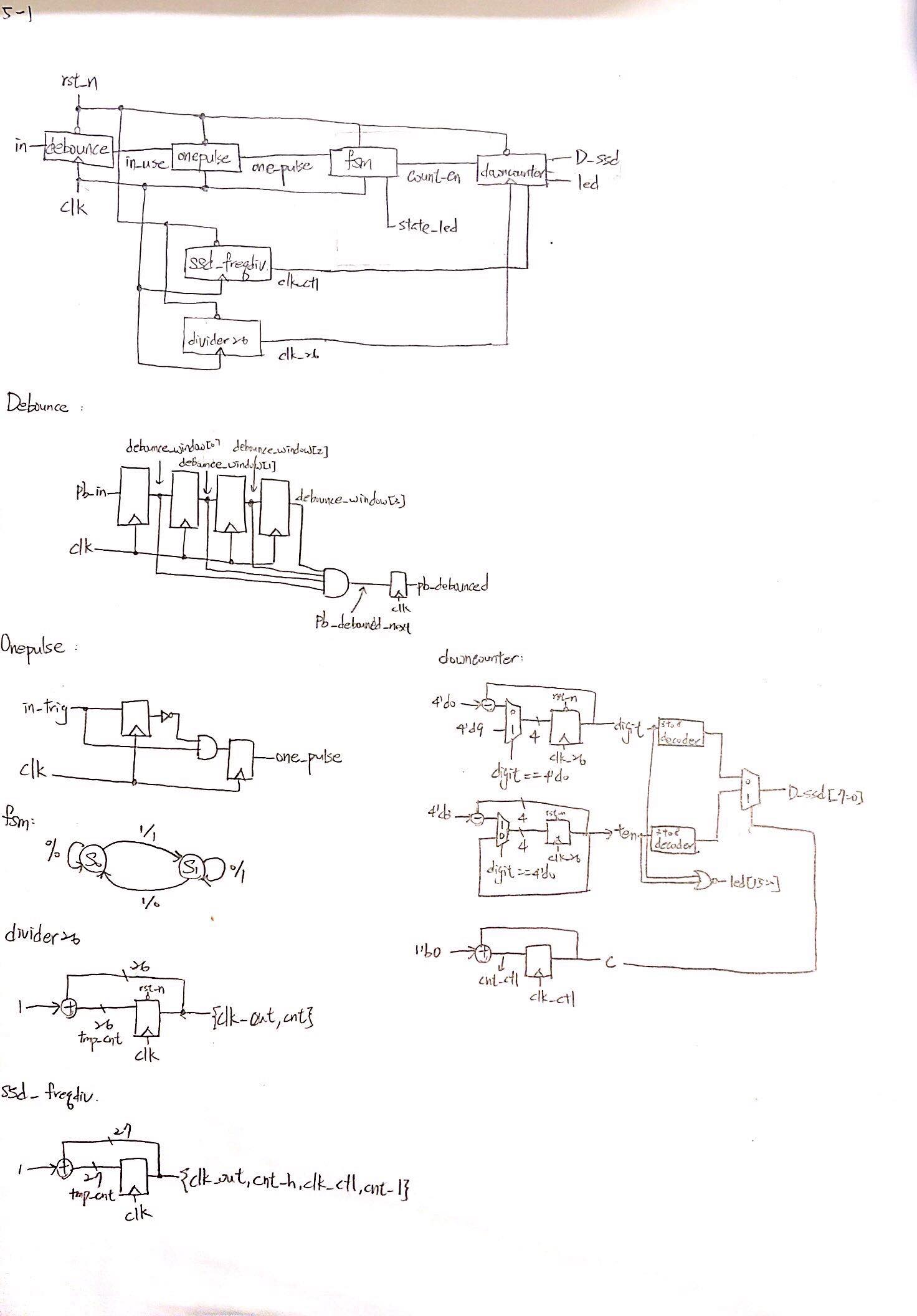
**Design Implementation**

* 本題由debounce、onepulse、fsm、ssd\_freqdiv、divider26、downcounter六個module組成
* Debounce

****

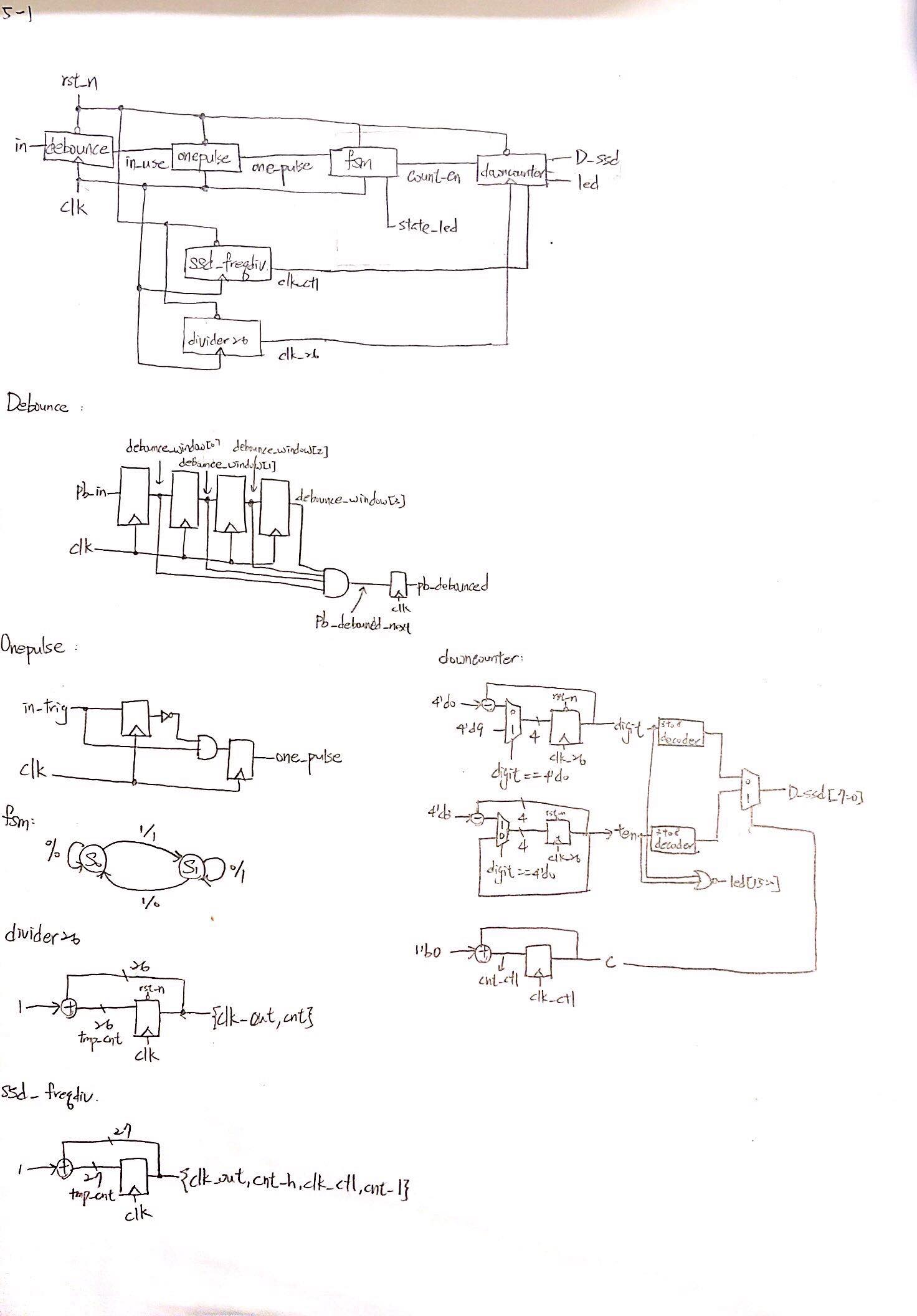
Debounce是為了避免按按鍵時所產生的浮動訊號而設計，藉此得到穩定的訊號。每經過一個clk，下一個input就會進來。當debounce\_window皆為一時，下個output值為1，藉此達到穩定的波形。

* 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

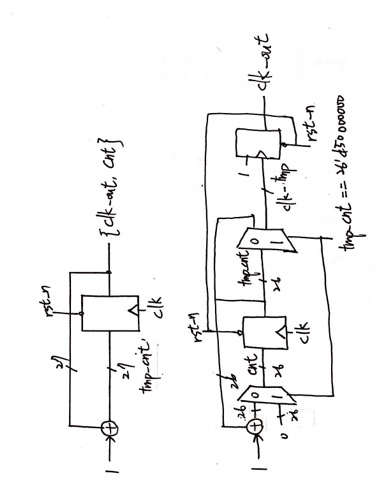
****

Input：rst\_n, clk, one\_pulse

Output：state\_led, count\_en

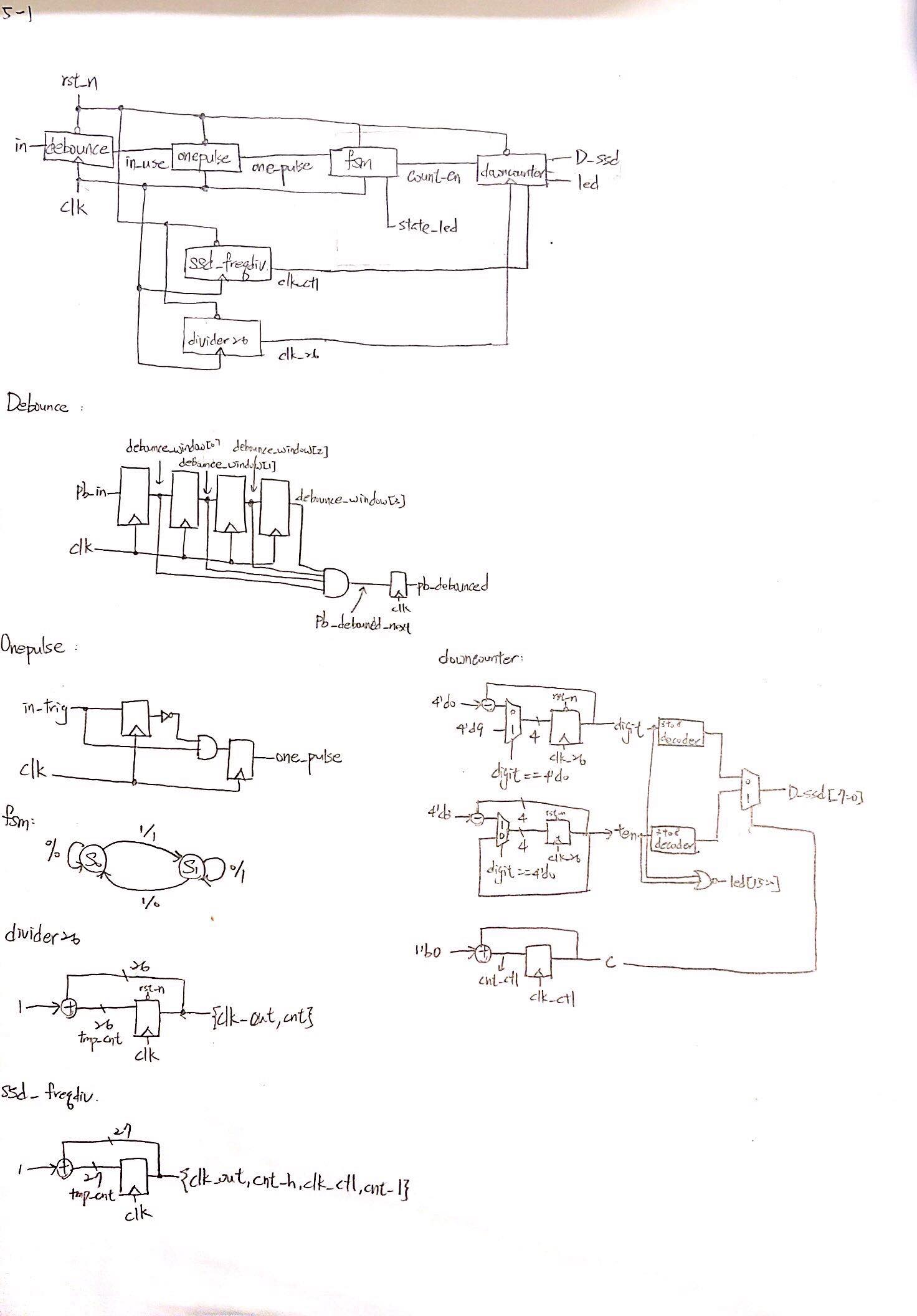
此為控制倒數計時器開始與停止的module，在我的設計中，當one\_pulse為1時，state會做切換，S0為停止，S1為開始。而當狀態為S1時，state\_led為1。

* Divider26



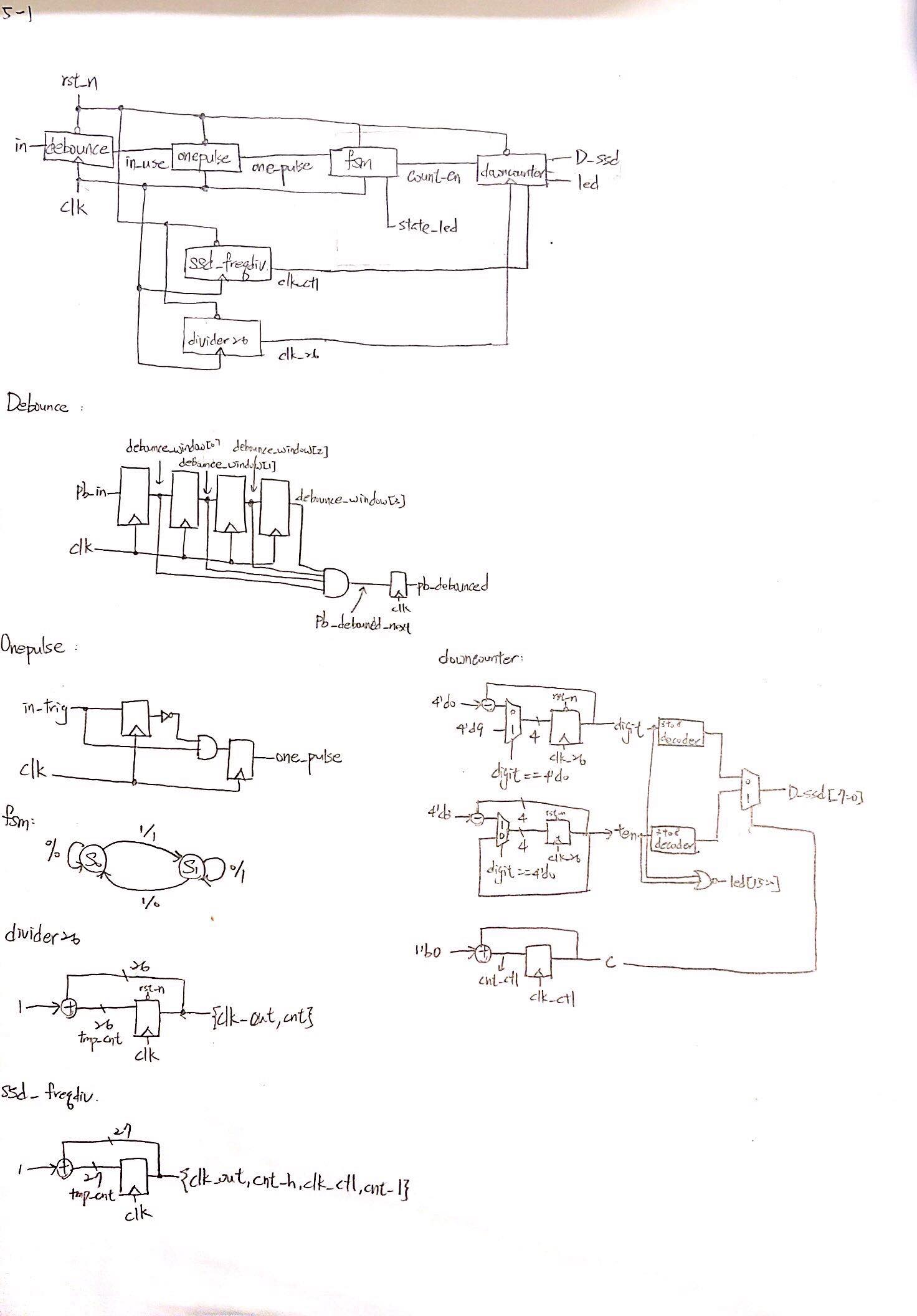
將原本clk的頻率除頻為1Hz

* 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**

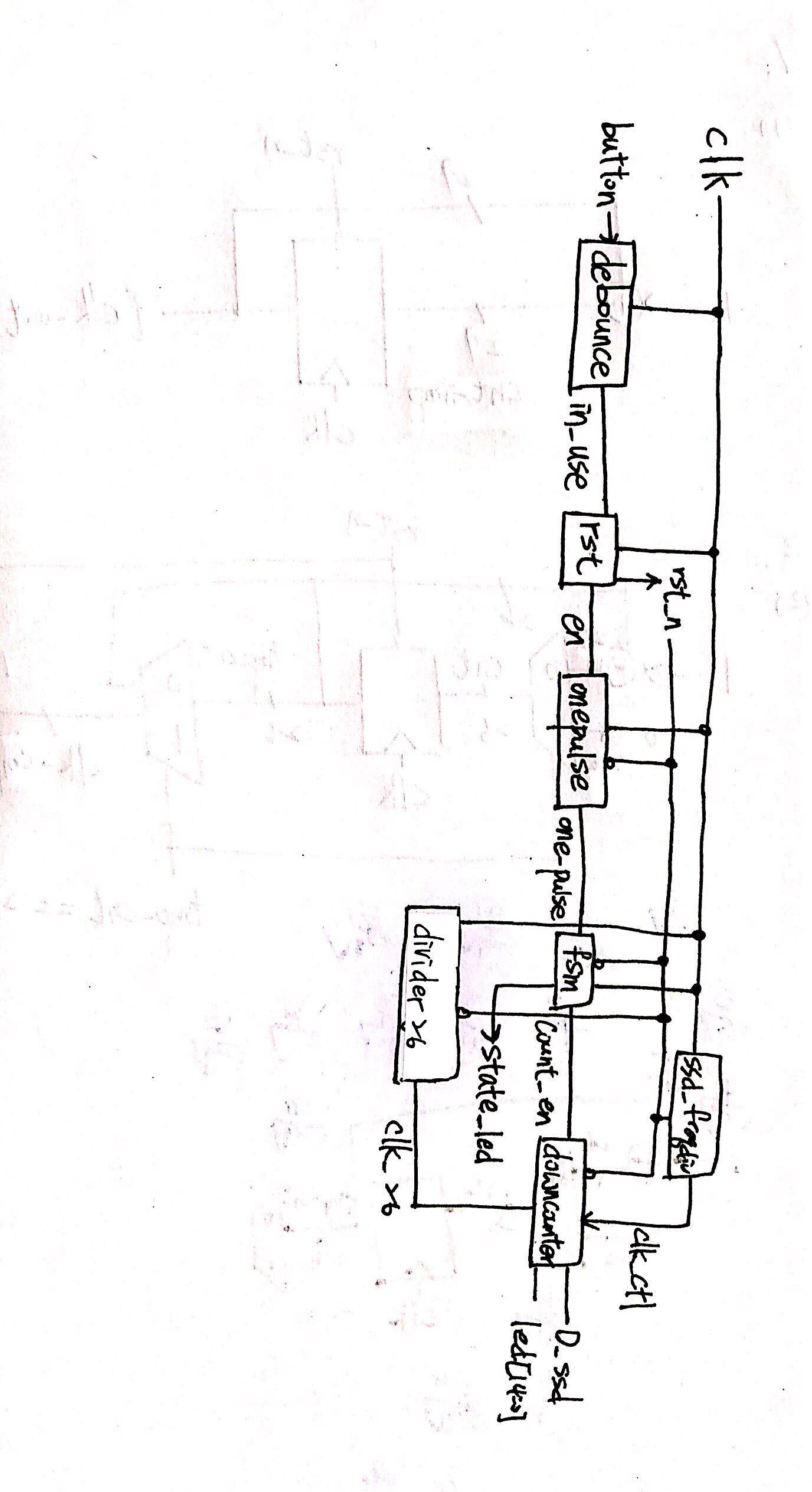
**Design Specification**

* 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。