105061254 林士平 邏輯設計實驗報告 Lab7

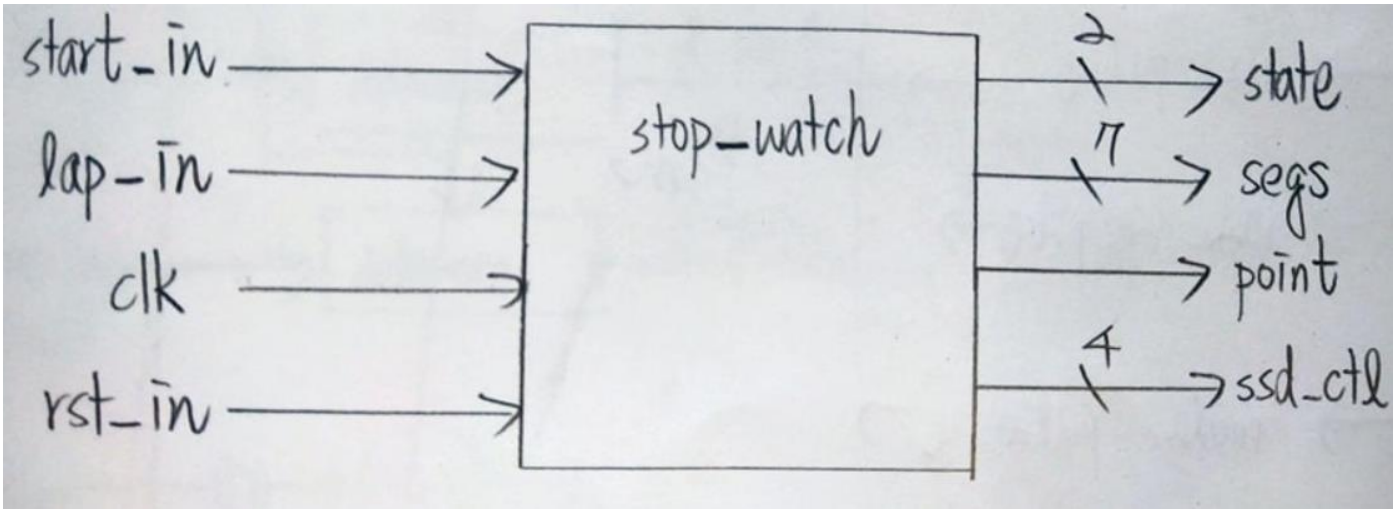
1.

(1) Design specification :

A. Inputs and outputs(表一)：

Inputs	clk, rst_in, start_in, lap_in
Outputs	ssd_ctl[3:0], segs[6:0], point, state[1:0]
↑表一：Inputs and outputs of 1	

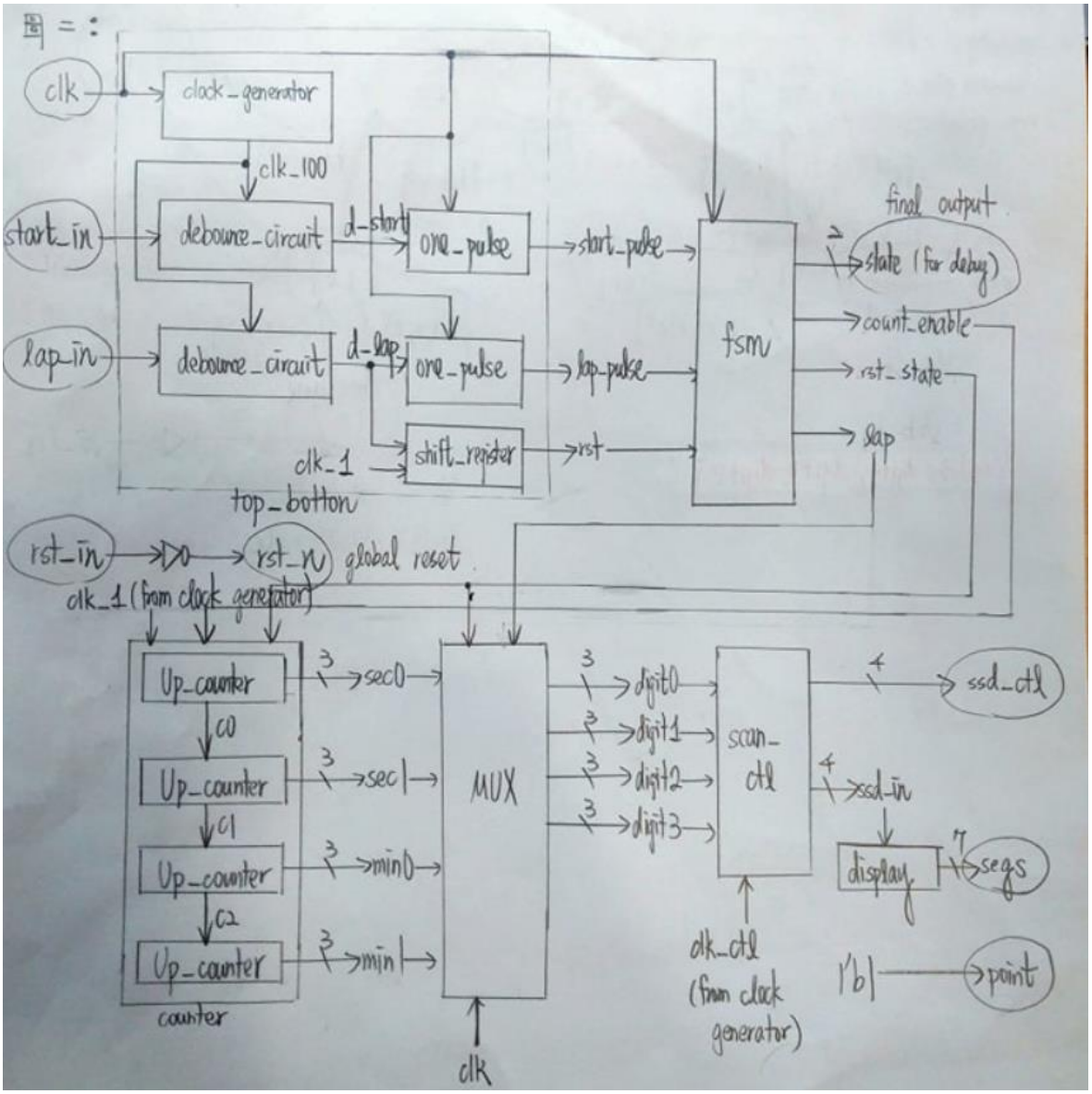
B. Block diagram(function table)(圖一)：



↑圖一：The block diagram of 1

(2) Design implementation :

A. Logic diagram(function table)(圖二)：



↑圖二：logic diagram of 1

B. I/O pin assignment(表二)：

I/O	ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]	point	segs[6]	segs[5]
LOC	W4	V4	U4	U2	V7	W7	W6
I/O	segs[4]	segs[3]	segs[2]	segs[1]	segs[0]	state[1]	state[0]
LOC	U8	V8	U5	V5	U7	E19	U16
I/O	rst_in	clk	start_in	lap_in			
LOC	W19	W5	T17	U17			
↑表二：I/O pin assignment of 1							

C.功能與做法說明：

本題為製作一個顯示秒和分的碼表，並且要有 start/stop、lap 和 rst 的功能。

首先是 top_botton 模組，由兩個按鈕輸入 start_in、lap_in 訊號，並經過 debounce 和 one_pulse 處理，得到 finite state machine 的控制訊號 start_pulse 和 lap_pulse，除了這兩個控制訊號之外還有 rst。rst 是由 lap_in 經過 debounce 之後再經過 shift_register 判斷而來，當長按 lap_in 按鍵超過 4 秒時，rst 便會=1，其他情況下 rst = 0，即利用 shift_register 來判定是否有長按，原理如下：因為 shift_register 一秒會 shift 一個 bit，所以設一個四個 bit 的 shift_register，input 為 d_lap，當四個 bit 均變成 1 時，表示長按至少 4 秒，在此狀況下便輸出 rst = 1。

接著是 fsm，fsm 總共有四個 state：pause、start、lap 和 reset。在 pause state 碼表會暫停、在 start state 碼表會開始運作、在 lap state 碼表會運作但七段顯示器不會改變數值，最後是 reset state，也就是將碼表歸零後會進入的 state，跳出這個 state 的方法就是按一下 start_in 按鈕，如此便會進入 start state，碼表便會開始運作。

再來是 counter，由 4 個 Up_counter 組成，分別輸出秒的個位和十位，以及分的個位和十位，比較特別的地方是 counter 的輸入有來自 fsm 的 state output：count_enable 和 rst_state，count_enable 決定 counter 是否要數、rst_state 決定 counter 是否要歸零。

然後是 MUX，用來處理最後七段顯示器要顯示哪四個數值：如果 rst_state == 1’b1，則顯示 0000；如果 rst_state == 1’b0 && lap == 1’b1 則讓數字不要改變；其他狀況正常顯示 counter 數出來的數字。比較重要的地方是在 rst_state == 1’b1 顯示 0000，這行看似有點多餘，因為 counter 在 rst_state == 1’b1 便會把數值歸零，如此一來顯示 counter 數出來的數字不就好了？但是如果不加這行的話，七段顯示器不會在 reset 完的當下立即顯示 0000，而是要等一秒後，因為 counter 的數值一秒才會更新一次。

最後便把 MUX 選出來的結果放入顯示模組(scan_ctl 和 display)來使七段顯示器顯示。

2.

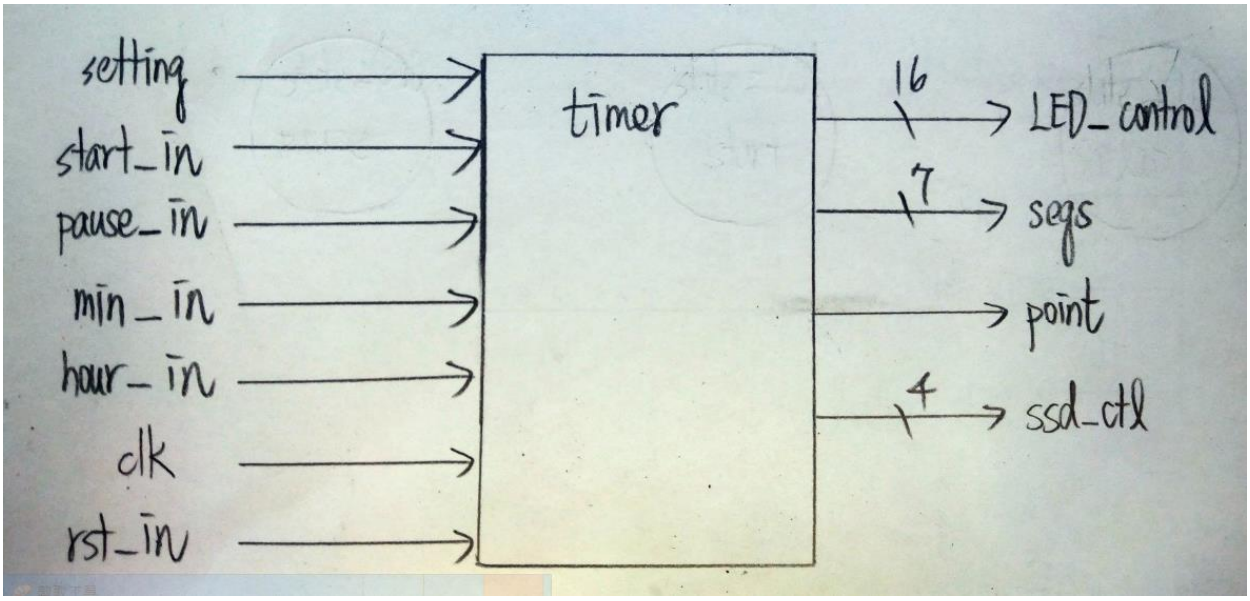
(1) Design specification :

A. Inputs and outputs(表三)：

Inputs	clk, rst_in, start_in, pause_in, min_in, hour_in, setting
Outputs	ssd_ctl[3:0], segs[6:0], point, LED_control[15:0]

↑表三：Inputs and outputs of 2

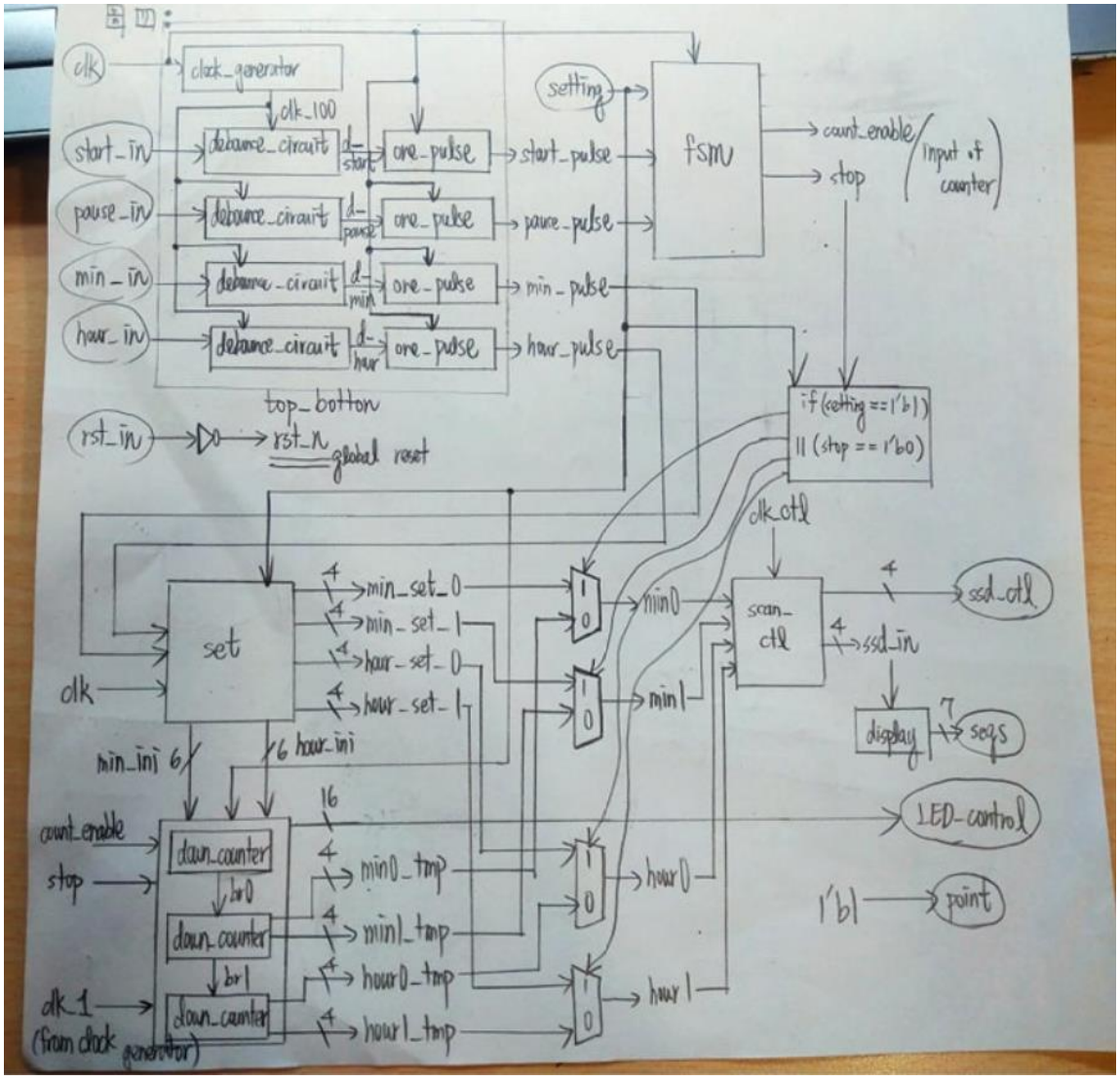
B. Block diagram(function table)(圖三)：



↑圖三：The block diagram of 2

(2) Design implementation :

A. Logic diagram(function table)(圖四)：



↑圖四：logic diagram of 2

B. I/O pin assignment(表四)：

I/O	ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]	point	segs[6]	segs[5]
LOC	W4	V4	U4	U2	V7	W7	W6
I/O	segs[4]	segs[3]	segs[2]	segs[1]	segs[0]	LED_ control[15]	LED_ control[14]
LOC	U8	V8	U5	V5	U7	L1	P1
I/O	LED_ control[13]	LED_ control[12]	LED_ control[11]	LED_ control[10]	LED_ control[9]	LED_ control[8]	LED_ control[7]
LOC	N3	P3	U3	W3	V3	V13	V14
I/O	LED_ control[6]	LED_ control[5]	LED_ control[4]	LED_ control[3]	LED_ control[2]	LED_ control[1]	LED_ control[0]
LOC	U14	U15	W18	V19	U19	E19	U16
I/O	start_in	pause_in	min_in	hour_in	setting	clk	rst_in
LOC	T17	U17	T18	W19	V17	W5	U18

↑表四：I/O pin assignment of 2

C.功能與做法說明：

本題為製作一個倒數計時器，當搬開 setting switch 時，可以設定要從幾時幾分開始倒數，當扳回 setting switch 後，一個按鈕負責開始/回到初始值、一個按鈕負責暫停/繼續下數。

首先是 top_botton 模組，將輸入的按鈕訊號(start_in, pause_in, min_in, hour_in)做 debounce 和 one_pulse，最後得到 start_pulse、pause_pulse、min_pulse 和 hour_pulse。

再來是 fsm 模組，總共有三個 state：stop(setting) state 把倒數計時器歸回初始值、start state 開始下數、pause state 暫停倒數，fsm 的 state transition 由三個控制訊號控制：start_pulse、pause_pulse 和 setting。比較重要的地方是，不管在哪個 state 只要 setting = 1’b1 就會跳到 state0(stop state)，所以 stop state 其實也可稱為 setting state。

接著是 set 模組，也是本題最重要的地方，負責設定 min 和 hour 的初值，也就是設定要從幾時幾分開始向下倒數。其實 set 模組就是兩個 counter，當 setting 開關 = 1’b1 時，每按一次 min_in 按鈕，min_ini(數分鐘的 down_counter 的初始值)就會加 1；每按一次 hour_in，hour_ini(數小時的 down_counter 的初始值)就會加 1，min_ini 和 hour_ini 會連入 counter 模組決定 counter 的初始值。當 setting = 1’b0 以及 fsm 在 start state 時，counter 就會開始運作，由 initial value 開始往下數。

然後是 counter 模組，裡面總共有三個 down_counter，分別下數秒、分、小時，秒的 down_counter 只是用來進位用，所以結果不用 output 出來。分和小時的 down_counter 分別數出 hour0_tmp、hour1_tmp、min0_tmp、min1_tmp，來表示時分的個位和十位。

接著是本題也非常重要的地方，如果直接將 hour0_tmp、hour1_tmp、min0_tmp、min1_tmp 接到顯示模組，那調整時間的過程七段顯示器要過一秒才會換一個數值，但是我們按按鈕的頻率一定大於 1，也就是說七段顯示器顯示的數值會不連續(比如我一秒按三下，那七段顯示器只會顯示 0000 和 0003，0001 和 0002 不會顯示出來)，這樣顯然不盡理想，所以我的解決方法如下：當 setting == 1’b1 時，也就是說當在設定時間的時候，顯示的數字由 set 模組決定(min_set_0、min_set_1、hour_set_0、hour_set_1)，而不是由 counter 模組決定(hour0_tmp、hour1_tmp、min0_tmp、min1_tmp)，因此我加了四個多功器來選擇要把何者丟入顯示模組顯示。

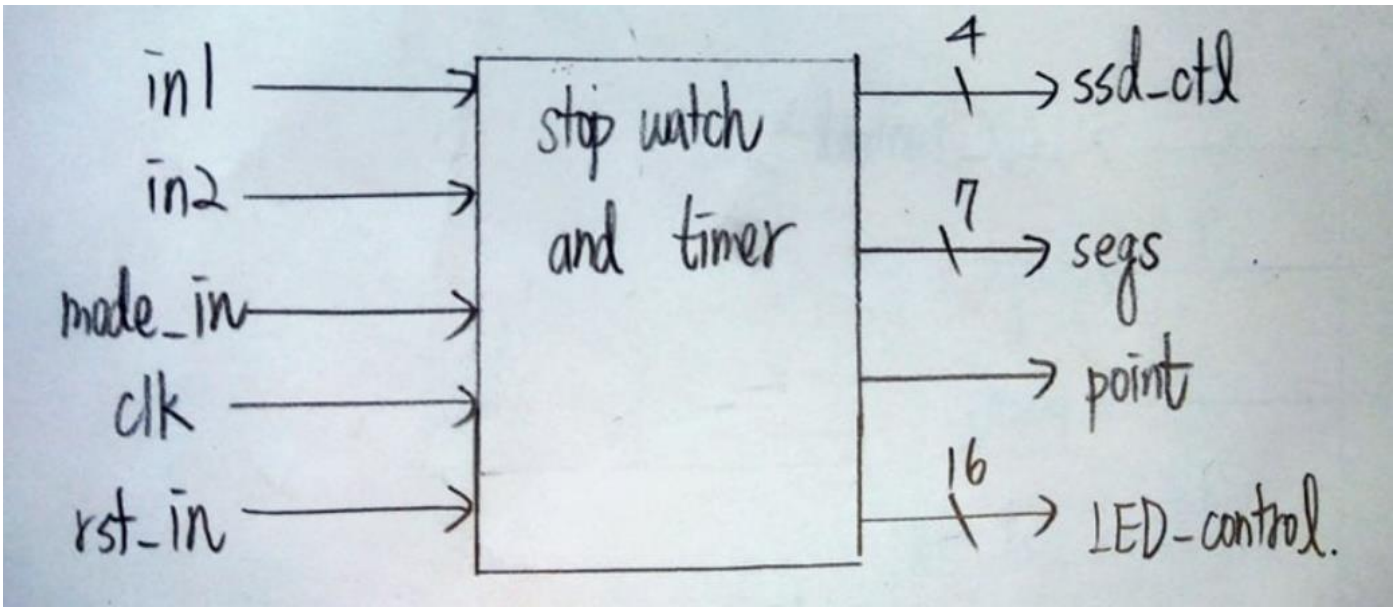
3.

(1) Design specification :

A. Inputs and outputs(表五)：

Inputs	clk, rst_in, in1, in2, mode_in
Outputs	ssd_ctl[3:0], segs[6:0], point, LED_control[15:0]
↑表五：Inputs and outputs of 3	

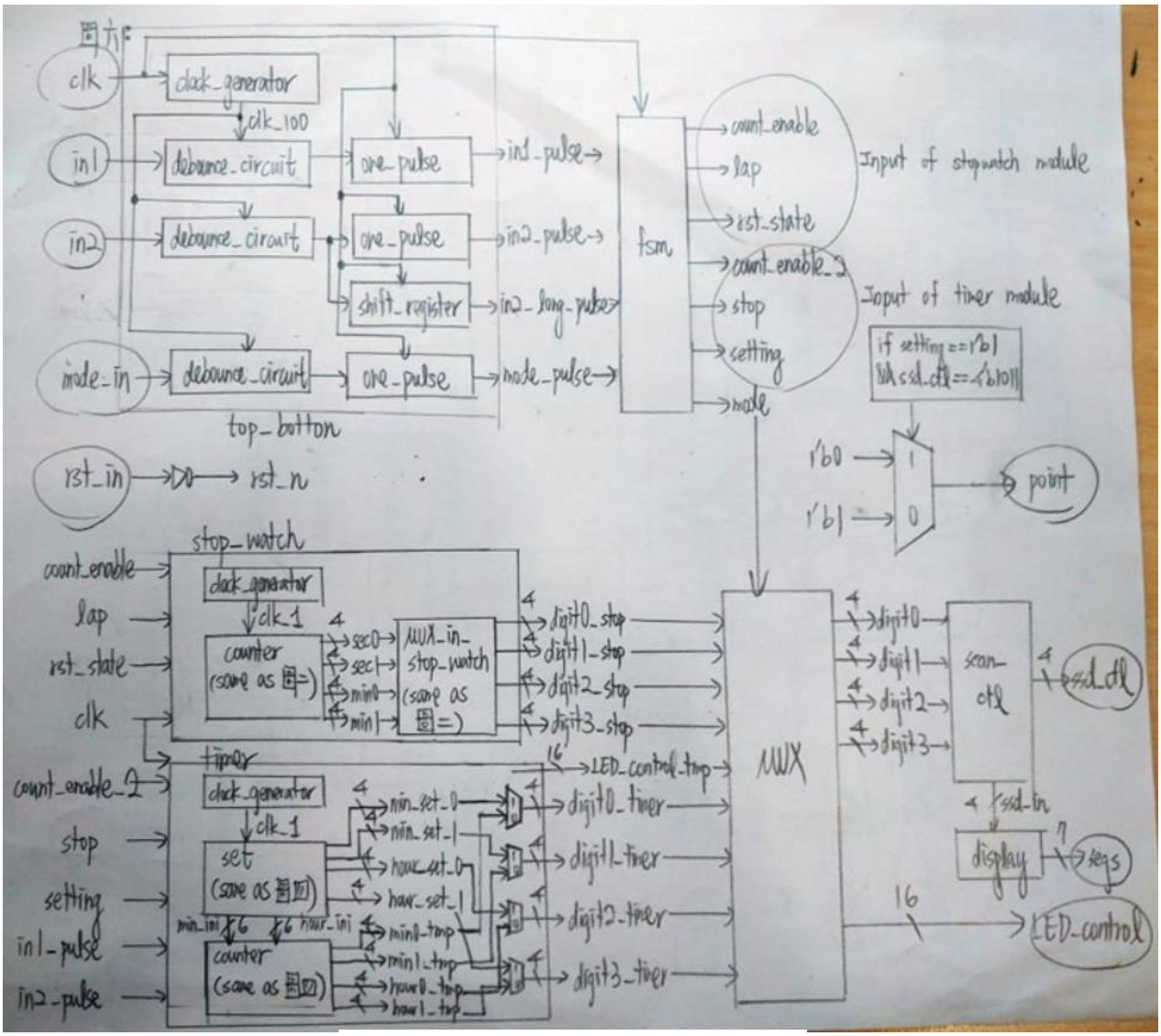
B. Block diagram(function table)(圖五)：



↑圖五：The block diagram of 3

(2) Design implementation :

A. Logic diagram(function table)(圖六)：



↑圖六：logic diagram of 3

I/O	ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]	point	segs[6]	segs[5]
LOC	W4	V4	U4	U2	V7	W7	W6
I/O	segs[4]	segs[3]	segs[2]	segs[1]	segs[0]	LED_ control[15]	LED_ control[14]
LOC	U8	V8	U5	V5	U7	L1	P1
I/O	LED_ control[13]	LED_ control[12]	LED_ control[11]	LED_ control[10]	LED_ control[9]	LED_ control[8]	LED_ control[7]
LOC	N3	P3	U3	W3	V3	V13	V14
I/O	LED_ control[6]	LED_ control[5]	LED_ control[4]	LED_ control[3]	LED_ control[2]	LED_ control[1]	LED_ control[0]
LOC	U14	U15	W18	V19	U19	E19	U16
I/O	in1	in2	mode_in	clk	rst_in		
LOC	T17	U17	U18	W5	W19		

↑表六：I/O pin assignment of 3

C.功能說明：

本題為利用三個按鍵整合前面兩題的功能(碼表和倒數計時器)。

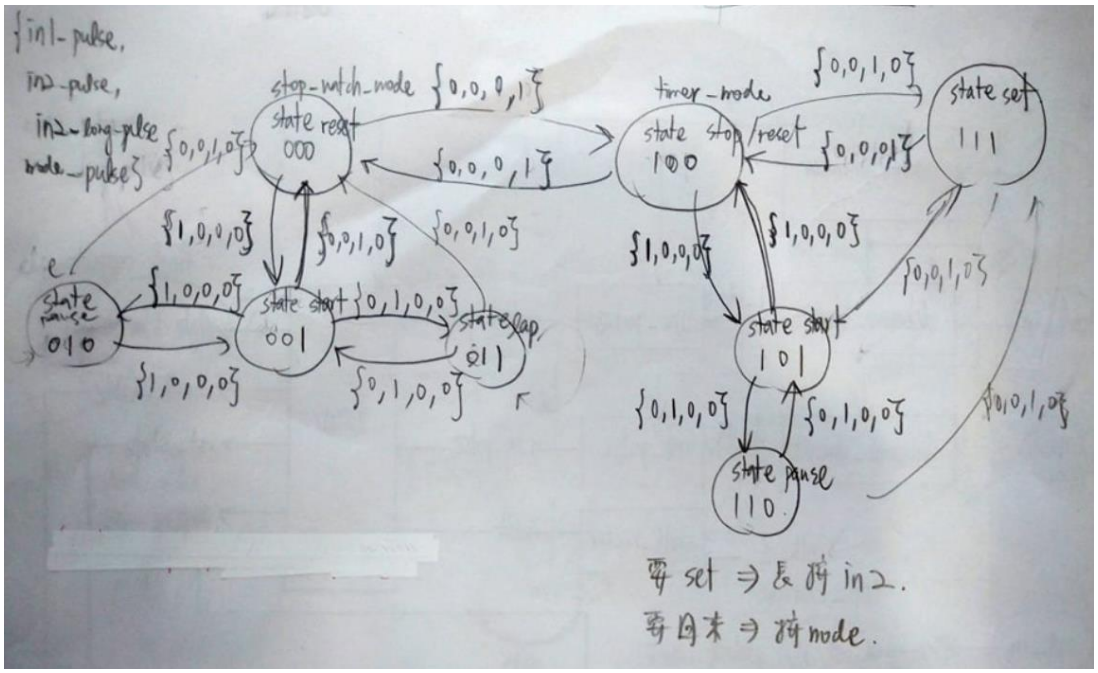
U18 按鍵固定作為碼表和倒數計時器兩個 mode 的轉換鍵，一開始 mode 在碼表，按一下即可到倒數計時器模式，再按一下便又回到碼表模式。當碼表在運作時，按一下 mode 亦可到倒數計時器模式，反之亦然，需注意的是，當一開始到達某個模式，會回到該模式的初始狀態(碼表模式即 0000、倒數計時器模式為使用者之前設置的初始時間)。

當在碼表模式時，T17 按鍵為 start/stop 鍵，按一下可以啟動碼表，再按一下可以讓碼表暫停；U17 按鍵為 lap/reset 鍵，當碼表為 start 狀態時按下 lap 可以讓七段顯示器上的數字停止改變，再按一下可以顯示現在碼表數到的時間，並繼續往上數。長按 U17 按鍵可以讓碼表歸零。

當在倒數計時器模式時，首先要先設定初始時間，長按 U17 按鍵可以到達設定模式，當到達設定模式時，七段顯示器最中間的點會亮起來，此時 T17 按鍵可以設定分、U17 按鍵可以設定時，當設定完時間後，按一下 U18 按鍵即可跳出設定模式，此時七段顯示器最中間的點會熄滅。在此時 T17 按鍵為 start/stop 鍵，按一下即可開始下數，再按一下會回到初始時間；U17 為 pause/resume，按一下可以暫停下數，再按一下可以繼續下數。

D.作法說明：

本題最重要的是 finite state machine 的設計，藉由 finite state machine 來控制 stop_watch 和 timer，並決定最後七段顯示器要顯示哪四個數字，下圖為本題的 state transition diagram：



↑圖七：The state transition diagram of 3

簡單來說：控制訊號為 in1_pulse、in2_pulse、in2_long_pulse 和 mode_pulse，總共有八個 state，stop_watch 模式和 timer 模式各四個，負責兩種模式的各種功能。每個 state 都會有分別對應的 state output，所以為 Moore machine。

5. Discussion

本次的 Lab 模組相當多，考驗到我的整理能力，尤其是 Lab7_3，必須把 Lab7_1 和 Lab7_2 的功能整合在一起，而且 Lab7_1 和 Lab7_2 模組就已經不少，另外需要注意 finite state machine 的設計，它是讓多個功能合而為一的關鍵。

本次 Lab 卡最久的地方除了 Lab7_3 的 finite state machine 設計，還有設置倒數計時器初始值這個功能，為了要讓使用者在設定的過程可以隨時看到目前已經設置到的時間，我設計當在設定模式時，七段顯示器顯示的是 set 模組的 output 而不是 counter 的(詳細見 7_2 題的功能與做法說明)。

6. Conclusion

這次的 Lab 是有史以來最複雜的一個 Lab，寫完之後很有成就感。讓我更了解 finite state machine 的使用與 counter 的應用。寫完之後覺得自己進步不少。