**(1) Implement a stopwatch function (00:00-59:59) with the FPGA board.**

**1.1 Use the four (Seven-Segment Displays, SSDs) as the display. The left two digits represent the minute and the right two digits represent the second.**

**1.2 Use two push buttons to control the function. Use one button to control start/stop and the other to control the lap and reset. When the stopwatch counts, press the 'lap' button will freeze the SSDs but the stopwatch continues counting, and when press the 'lap' button again, the SSDs will start to show current time.**

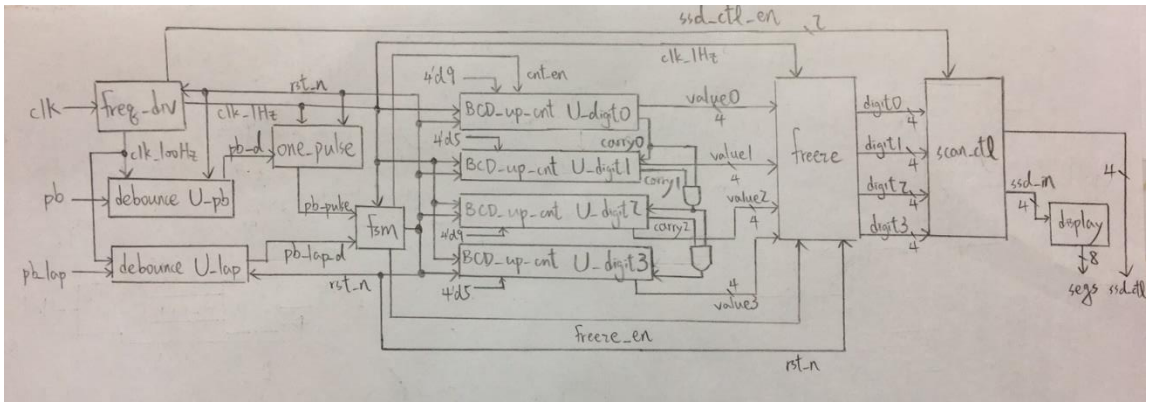
## IO:

Input: clk, pb, pb_lap

Output: [7:0]segs, [3:0]ssd_ctl

## Block diagram:

本次實驗需要使用以下 module 功能，分別為除頻器(freq_div)、debounce 處理*2(debounce U_pb、U_lap)、one pulse 處理(one pulse)、FSM(fsm)、十進位上數器*4(BCD_up_cnt U_digit0～3)、freeze 控制(freeze)、scan control(scan_ctl)以及七段顯示解碼器(display)。




**1. 除頻器:** 將 100MHz 的 clk 輸入通過除頻器除頻成 1 Hz、100Hz 的輸出頻率，也輸出 2bit 的 ssd control enable 作為 scan control 的控制項

**2. debounce U_pb、one pulse:** 將控制 start/stop 的按鈕訊號(pb)經過 debounce 處理再經過 one pulse 處理，連接到 FSM 當作輸入

**3. debounce U_lap:** 將控制 lap/reset 的按鈕訊號(pb_lap)經過 debounce 處理，連接到 FSM 當作輸入

**4. FSM:** 透過按鈕的輸入值選擇 FSM 的 state 和所對應到的輸出(count enable、rst_n、freeze enable)

**5. 十進位上數器:** 將 FSM 的輸出 count enable 連接到十進位上數器秒數個位數的輸入來控制是否要進行上數

**6. freeze:** 將各級上數器(digit0～3)的輸出連接到 freeze 裡當作輸入，也連接 FSM 的 freeze enable 輸出訊號當作輸入，來控制目前七段顯示器是否要跟著上

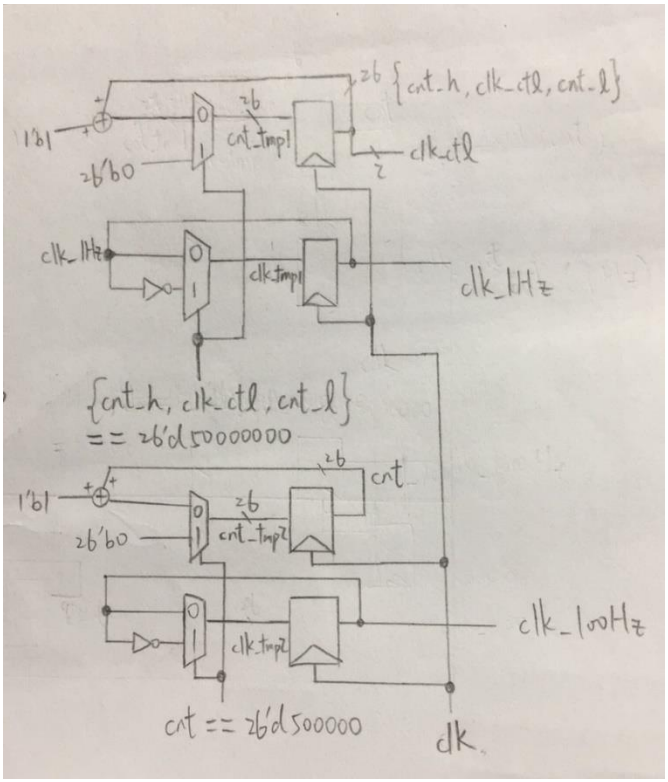
數器的輸出顯示，或是要顯示 lap 狀態下提供紀錄的數字

**7. scan control:** 將 freeze 的輸出 digit3～0 以及除頻器的輸出 ssd control enable 作為輸入，透過快速的輪流顯示，可以讓 4 個七段顯示器各自顯示自己要顯示的數字

**8.七段顯示解碼器:** 將 scan control 給的輸入值(ssd_in)透過解碼之後顯示在七段顯示器上

## Logic diagram:

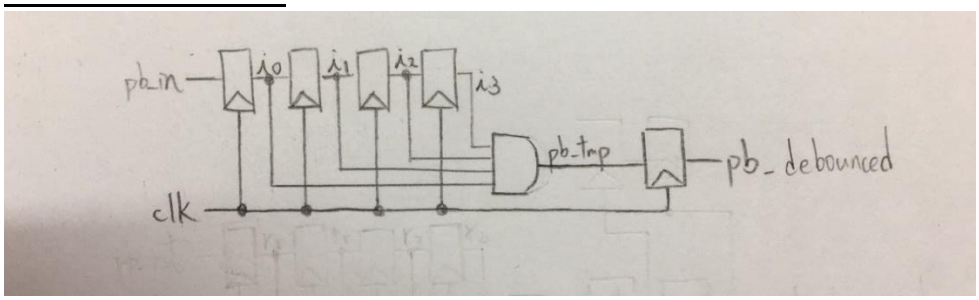
除頻器: 輸出 1Hz、100Hz 以及 2bit 的 clk_ctl 控制




本次除頻器需要製造兩個不同頻率的 clock，分別為 1Hz 和 100Hz，由於 FPGA 板上的頻率為 100MHz，建構理念是分別設計一個上限為 50M 和 500K 的 binary up counter，前者在達到 50M 之前，每個 clk 都讓 counter 加上 1，而當 counter 數到 50M 時，便將輸出值(clk_1Hz) toggle 一次，也將 counter 歸零，後者則是在達到 500K 之前，每個 clk 都讓 counter 加上 1，而當 counter 數到 500K 時，便將輸出值(clk_100Hz) toggle 一次，也將 counter 歸零，如此下來，輸出值(clk_1Hz)在等於 0 和等於 1 的時間各自都是 50M 次的 clk，也就是說，clk_1Hz 的值 0.5 秒會 toggle 一次，因此輸出值的頻率將會等於 100MHz/(50M*2) = 1Hz；輸出值 clk_100Hz 在等於 0 和等於 1 的時間各自都是 500K 次的 clk，也就是說，clk_100Hz 的值 0.005 秒會 toggle 一次，因此輸出值的頻率將會等於

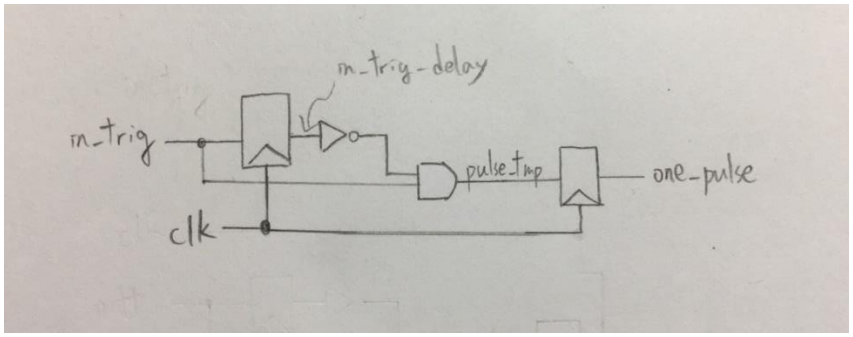
100MHz/(500K*2) = 100Hz。
另外，將上限為 50M 的 counter 中的第 16、17bit 抓出來當作給 scan control 的控制項。

## Debounce circuits:




由於 push bottom 會產生 bouncing 的現象，因此將 push bottom 的輸入通過 4 個 flip flop 後，再將每個 flip flop 的輸出(i0～i3)通過 and gate，可以得到穩定的訊號值(pb_debounced)。

## One pulse:




由於每次按下 push bottom 的時間都不一樣，要確保只在一個 clock 裡有輸入訊號，必須要經過上圖的邏輯來產生。將輸入(in_trig)延遲一個 clock，得到 in_trig_delay，再將 in_trig_delay 與接下來的輸入(in_trig)做 and，可以得到只有在一個 clock 裡有 positive signal 的輸出(one_pulse)。

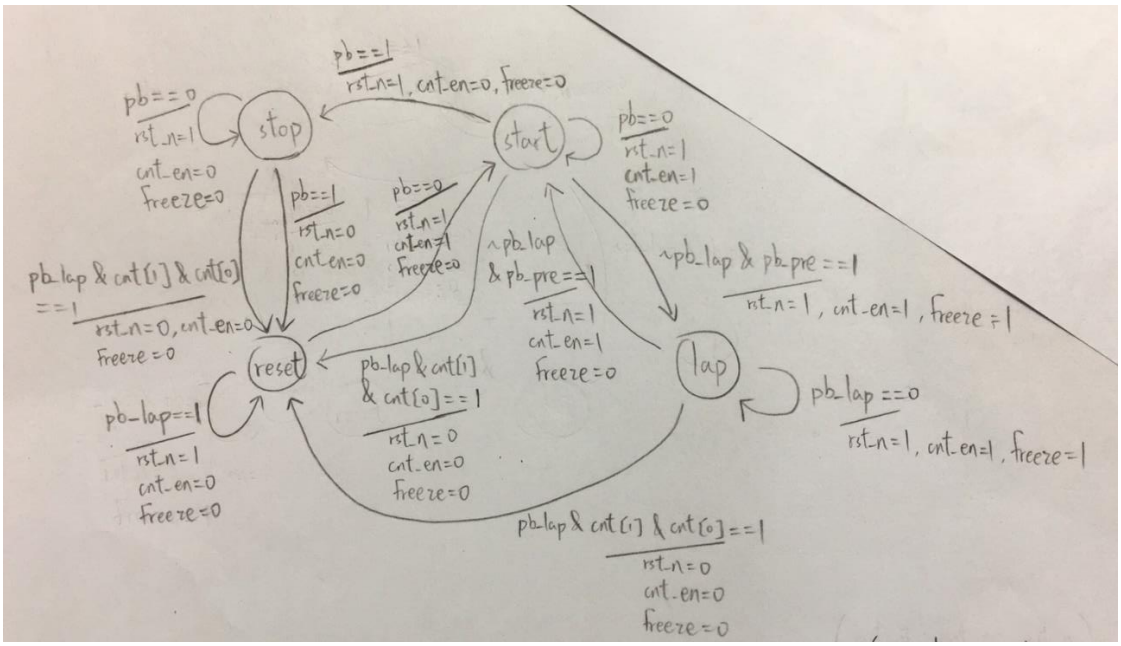
FSM:

圖中的含意為: input/output

(pb 為控制 start/stop 的按鈕輸入)

(pb_lap 為控制 lap/reset 的按鈕輸入)

(rst_n 為 reset、cnt_en 為 count enable、freeze 為 freeze enable)




重置狀態為 START。

**按下一次 pb:**

(1)START→STOP，暫停(count enable=0)，下次從頭開始

(2) STOP→RESET→START，先重置(rst_n=0)，之後從頭開始數

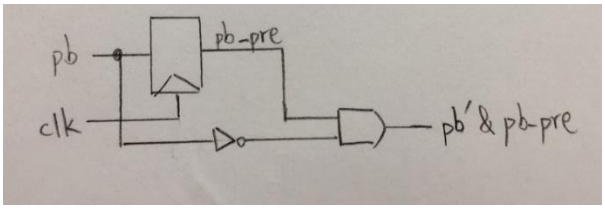
**短時間按下 pb_lap:**

(1)START→LAP，碼錶繼續數(count enable=1)，但顯示器停留在按下瞬間的時間(freeze enable=1)

(2)LAP→START，顯示器回到目前當下的時間

**長時間按下 pb_lap:** 不管當下為哪個狀態都回到 RESET 並重置(rst_n=0)，之後重頭開始數

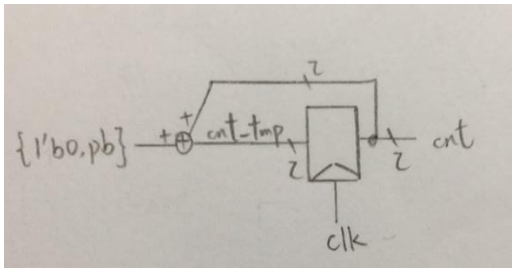
短按控制: (pb_lap 按鈕)




利用一個 flip flop，將上一個 clock 的 push bottom 訊號暫存下來(pb_pre)，再將 pb_pre 與當下的 push bottom 訊號(pb)的 invert 通過 and gate，就能夠判斷輸入

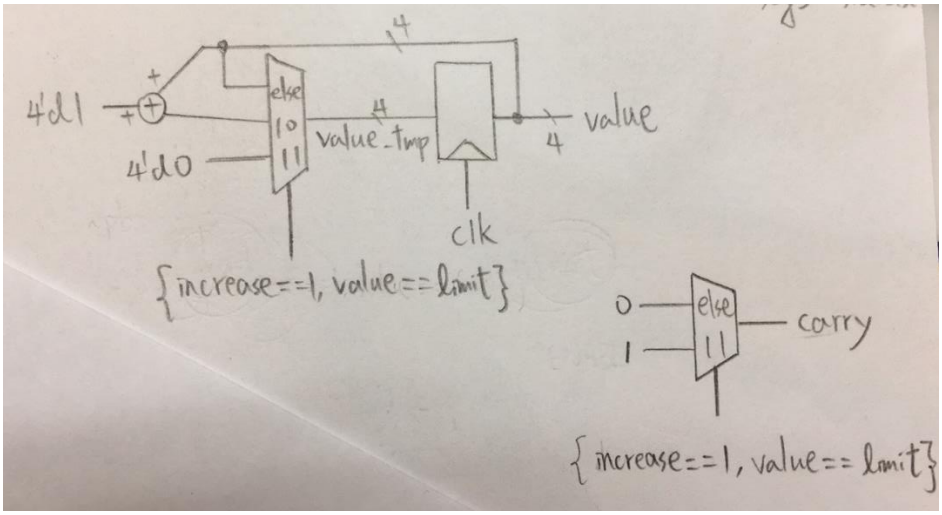
訊號是否在上一個 clock 為 1，而現在是 0，也就是在放開 push bottom 的瞬間 pb' & pb_pre 會等於 1，以達到短按控制的效果。

長按控制: (pb_lap 按鈕)



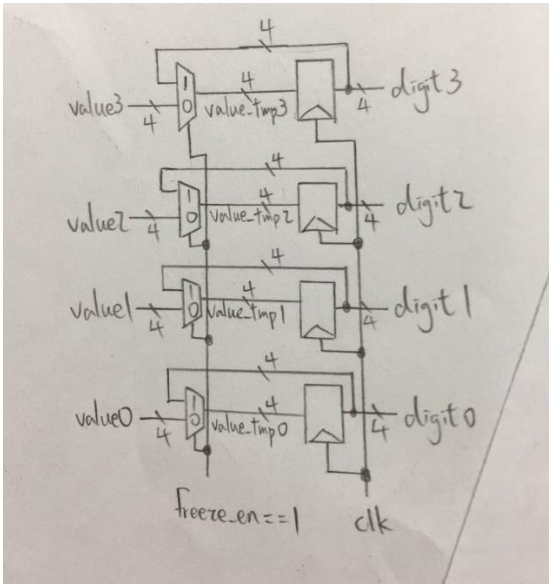

每次 clock 會讓 cnt 加上 push bottom 的輸入值(pb)，並在輸入值(pb)的 falling edge 將 cnt 歸零，因此若是長按 push bottom，使得 cnt 數到 2'b11，則 pb = cnt[1] = cnt[0] = 1，FSM 在下一個 clock 後將會轉換成 reset 的狀態，以達到長按控制的效果。

十進位上數器*4:




**digit0:** (increase 為 fsm 的輸出 count enable，limit=9)
**digit1:** (increase 為 digit0 的 carry，limit=5)
**digit2:** (increase 為 digit0 和 digit1 的 carry(皆為 1 時)，limit=9)
**digit3:** (increase 為 digit0、digit1 和 digit2 的 carry(皆為 1 時)，limit=5)
當 increase 等於 1 且 value 等於 limit 時，代表下次需要進位，carry 輸出 1 告知下一級，value_tmp = 0；當 increase 等於 1 但 value 不等於 limit 時，代表下次要加 1 但不用進位，carry = 0，value_tmp = value + 1；其他狀況代表不需要加 1 也不需要進位，carry = 0，value 維持現狀，value_tmp = value，當每次 clk 的 positive edge 來時，將 value 輸入 value_tmp 的值。

## freeze 控制:




當 freeze enable = 1 時，各個 value_tmp 等於各個 digit 的值，將輸出值凍結起來；當 freeze enable = 0 時，各個 value_tmp 等於各個輸入(value0～3)的值，跟著上數器給的值輸出。每次 clk 的 positive edge 來時，將各個 digit 輸入各個 value_tmp 的值。

## Scan control:

ssd_ctl_en = 00 → ssd_ctl = 0111
ssd_ctl_en = 01 → ssd_ctl = 1011
ssd_ctl_en = 10 → ssd_ctl = 1101
ssd_ctl_en = 11 → ssd_ctl = 1110

四個七段顯示器輪流顯示，ssd_ctl_en = 00 時→顯示第一個七段顯示器(分鐘十位數)；ssd_ctl_en = 01 時→顯示第二個七段顯示器(分鐘個位數)；ssd_ctl_en = 10 時→顯示第三個七段顯示器(秒數十位數)；ssd_ctl_en = 11 時→顯示第四個七段顯示器(秒數個位數)，以快速輪流顯示的方式達成視覺暫留的效果。

## 7-segment display decoder:

in = 0 → segs = 8'b0000_0011
in = 1 → segs = 8'b1001_1111
in = 2 → segs = 8'b0010_0101
in = 3 → segs = 8'b0000_1101
in = 4 → segs = 8'b1001_1001
in = 5 → segs = 8'b0100_1001

in = 6 → segs = 8'b0100_0001
in = 7 → segs = 8'b0001_1111
in = 8 → segs = 8'b0000_0001
in = 9 → segs = 8'b0000_1001
default 為 segs = 8’b1111_1111(全暗)

結合以上功能，可以得到一個具有 start/stop、lap/reset 功能，且可以從 00:00 上數到 59:59 的碼表。

## I/O pin assignment:

| segs[7] | segs[6] | segs[5] | segs[4] | segs[3] | segs[2] | segs[1] | segs[0] |
|---|---|---|---|---|---|---|---|
| W7 | W6 | U8 | V8 | U5 | V5 | U7 | V7 |

| ssd_ctl[3] | ssd_ctl[2] | ssd_ctl[1] | ssd_ctl[0] | clk | pb | pb_lap |
|---|---|---|---|---|---|---|
| W4 | V4 | U4 | U2 | W5 | W19 | U18 |

**(2)Implement a timer (can support as long as 23:59) with the following functions.**

**2.1 Use one DIP switch as the 'setting' control. When the 'setting' is ON, you can use two buttons to set the minute and second.**

**2.2 Use other two buttons to control the timer operation. One button for start/stop and the other button for pause/resume.**

**2.3 When the time goes to 0, light up all the LEDs.**

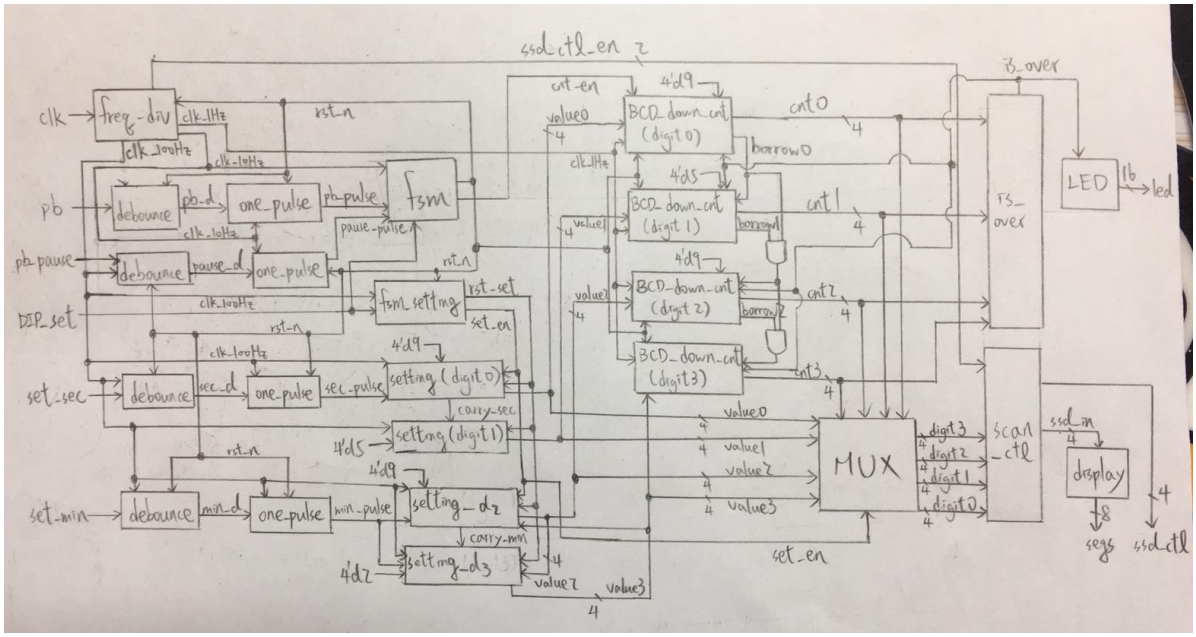
## IO:

Input: clk, pb, pb_pause, DIP_set, set_min, set_sec

Output: [3:0]ssd_ctl, [7:0]segs, [15:0]led

## Block diagram:

本次實驗需要使用以下 module 功能，分別為除頻器(freq_div)、debounce 處理*4(debounce)、one pulse*4 處理(one pulse)、FSM*2(fsm、fsm_setting)、十進位上數器*4(setting)、十進位下數器*4(BCD_down_cnt)、多工器(MUX)、is_over 控制(is_over)、scan control(scan_ctl)以及七段顯示解碼器(display)、led 顯示(LED)。




**1. 除頻器:** 將 100MHz 的 clk 輸入通過除頻器除頻成 1 Hz、10Hz、100Hz 的輸出頻率，也輸出 2bit 的 ssd control enable 作為 scan control 的控制項

**2. debounce:** 將 4 個按鈕的訊號都通過 debounce 處理之後得到穩定的訊號

**3. one pulse:** 將 4 個 debounce 過後的訊號都再經過 one pulse 處理，得到只會在一個 clock 內為 1 的訊號，並連接到 FSM 當作輸入

**4. FSM(fsm):** 透過 2 個按鈕(pb、pb_pause)和 1 個 DIP_switch 的輸入值(DIP_set)選擇此 FSM 的 state 和所對應到的輸出(count enable、rst_n)

**5. FSM(fsm_setting):** 透過 1 個 DIP_switch 的輸入值(DIP_set)選擇此 FSM 的 state 和所對應到的輸出(set enable、rst_set)

**6.十進位上數器:** 將 fsm_setting 的輸出 set enable 連接到十進位上數器秒和分的個位數來控制是否准許進行上數，而觸發分和秒上數器要加 1 的分別為兩個按鈕(set_sec、set_min)的輸入值，set_min 每按一次分會加 1，set_sec 每按一次秒會加 1；另外，fsm_setting 的另一個輸出 rst_set 則是專屬給這四個設定時間的十進位上數器的，目的在於確保其他功能在重置時，之前設定好要倒數的分鐘和秒數不會被重置掉。

**7.十進位下數器:** 將 fsm 的輸出(count enable) 連接到秒數個位數下數器來控制是否要進行下數

**8. MUX**: 透過 fsm_setting 的輸出 set enable 來判斷現在要顯示的是下數的畫面還是設定時間的畫面，並輸出給 scan control 當作輸入

**9. is_over**: 由於要在倒數至 00:00 時停止，將下數器的四位數輸出值都接近來當作輸入，若四者都為 0，則輸出一個訊號為 1 的輸出，告知下數器停止並通知 led 燈全亮

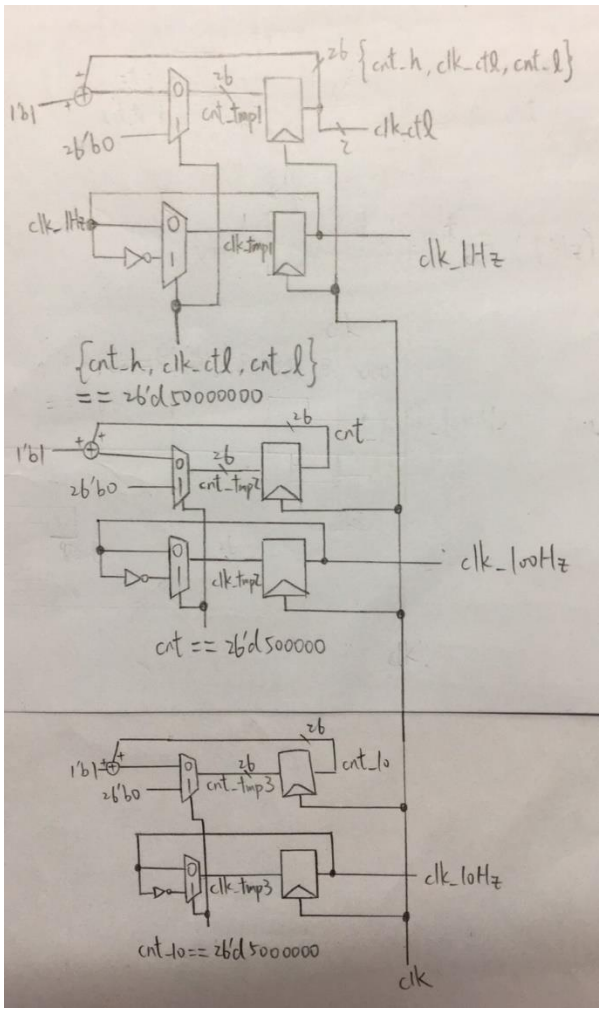
**10. LED**: 當接收到 is_over 輸出為 1 的訊號時，讓所有 led 燈亮起

**11. scan control:** 將 MUX 的輸出 digit3～0 以及除頻器的輸出 ssd control enable 作為輸入，透過快速的輪流顯示，可以讓 4 個七段顯示器各自顯示自己要顯示的數字

**12.七段顯示解碼器:** 將 scan control 給的輸入值(ssd_in)透過解碼之後顯示在七段顯示器上

**Logic diagram:**

<u>除頻器:</u> 輸出 1Hz、10Hz、100Hz 以及 2bit 的 clk_ctl 控制




這個除頻器需要製造三個不同頻率的 clock，分別為 1Hz、10Hz 和 100Hz，由於 FPGA 板上的頻率為 100MHz，建構理念是分別設計三個上限為 50M、5M 和 500K 的 binary up counter:

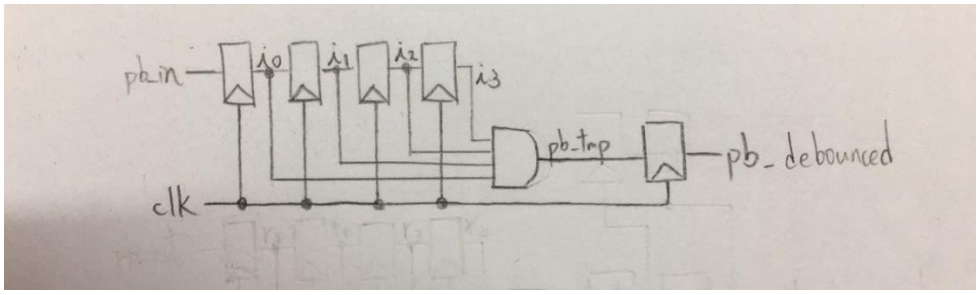
(1)**1Hz**: 在達到 50M 之前，每個 clk 都讓 counter 加上 1，而當 counter 數到 50M 時，便將輸出值(clk_1Hz) toggle 一次，也將 counter 歸零，如此一來，輸出值(clk_1Hz)在等於 0 和等於 1 的時間各自都是 50M 次的 clk，也就是說，clk_1Hz 的值 0.5 秒會 toggle 一次，因此輸出值的頻率將會等於 100MHz/(50M*2) = 1Hz

(2)**10Hz**: 在達到 5M 之前，每個 clk 都讓 counter 加上 1，而當 counter 數到 5M 時，便將輸出值(clk_10Hz) toggle 一次，也將 counter 歸零，如此一來，輸出值 clk_10Hz 在等於 0 和等於 1 的時間各自都是 5M 次的 clk，也就是說，clk_10Hz 的值 0.05 秒會 toggle 一次，因此輸出值的頻率將會等於 100MHz/(5M*2) = 10Hz。

(3)**100Hz**: 在達到 500K 之前，每個 clk 都讓 counter 加上 1，而當 counter 數到 500K 時，便將輸出值(clk_100Hz) toggle 一次，也將 counter 歸零，如此一來，

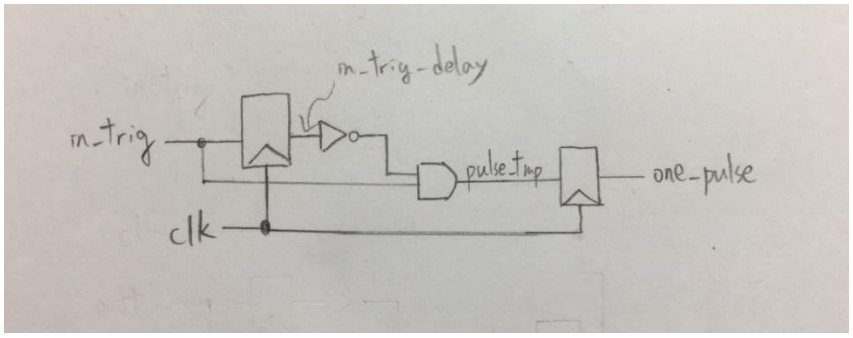
輸出值 clk_100Hz 在等於 0 和等於 1 的時間各自都是 500K 次的 clk，也就是說，clk_100Hz 的值 0.005 秒會 toggle 一次，因此輸出值的頻率將會等於 100MHz/(500K*2) = 100Hz。
另外，再將上限為 50M 的 counter 中的第 16、17bit 抓出來當作給 scan control 的控制項。

## Debounce circuits:




由於 push bottom 會產生 bouncing 的現象，因此將 push bottom 的輸入通過 4 個 flip flop 後，再將每個 flip flop 的輸出(i0～i3)通過 and gate，可以得到穩定的訊號值(pb_debounced)。

## One pulse:




由於每次按下 push bottom 的時間都不一樣，要確保只在一個 clock 裡有輸入訊號，必須要經過上圖的邏輯來產生。將輸入(in_trig)延遲一個 clock，得到 in_trig_delay，再將 in_trig_delay 與接下來的輸入(in_trig)做 and，可以得到只有在一個 clock 裡有 positive signal 的輸出(one_pulse)。

FSM(fsm):

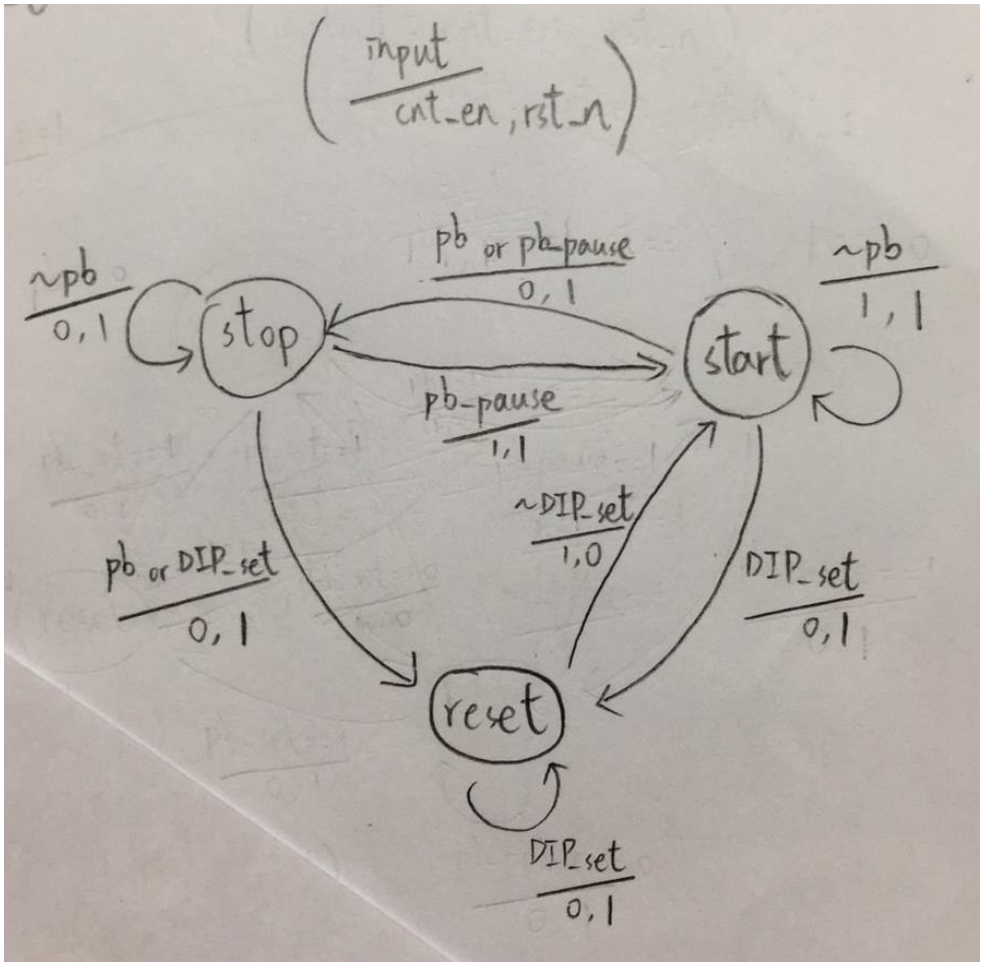
圖中的含意為: input/output

(pb 為控制 start/stop 的按鈕輸入)

(pb_pause 為控制 pause/resume 的按鈕輸入)

(DIP_set 為控制現在是否要設定時間的 DIP_switch 開關訊號)

(rst_n 為 reset、cnt_en 為 count enable)




重置狀態為 START。

**按下一次 pb:**

(1)START→STOP，暫停倒數 (count enable=0)

(2)STOP→RESET→START，先重置(rst_n=0)，再從先前設定好的時間從頭開始倒數

**按下一次 pb_pause:**

(1)START→STOP，暫停倒數 (count enable=0)

(2)STOP→START，繼續倒數

**開啟 DIP_set:** 不管當下為哪個狀態都回到 RESET，並維持在 RESET

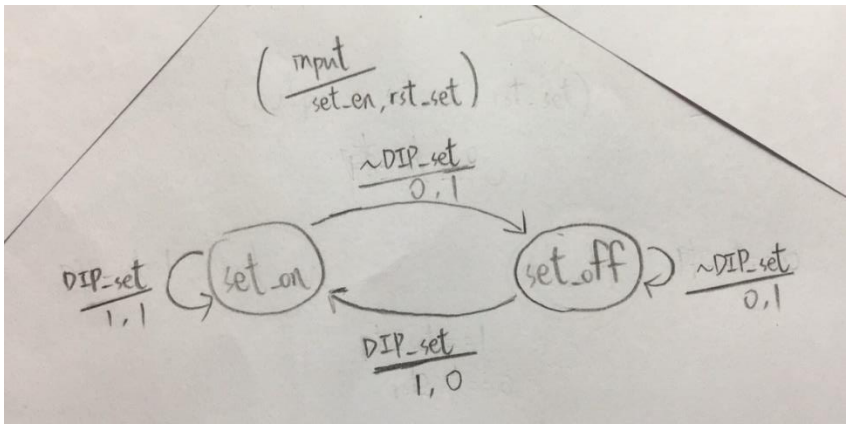
**關閉 DIP_set:** RESET→START，先重置後再回到 START，由於有先重置，因此可以將設定好的值輸入進下數器，並從新設定好的分鐘和秒數開始倒數

## FSM(fsm_setting):

圖中的含意為: input/output

(DIP_set 為控制現在是否要設定時間的 DIP_switch 開關訊號)

(set_en 為 set enable、rst_set 為用來設定時間的上數器的重置訊號)




重置狀態為 SET_OFF。

**打開 DIP_set:**

(1)SET_OFF→SET_ON，set enable = 1，可以設定時間；rst_set = 0，重置用來設定時間的上數器至 00:00

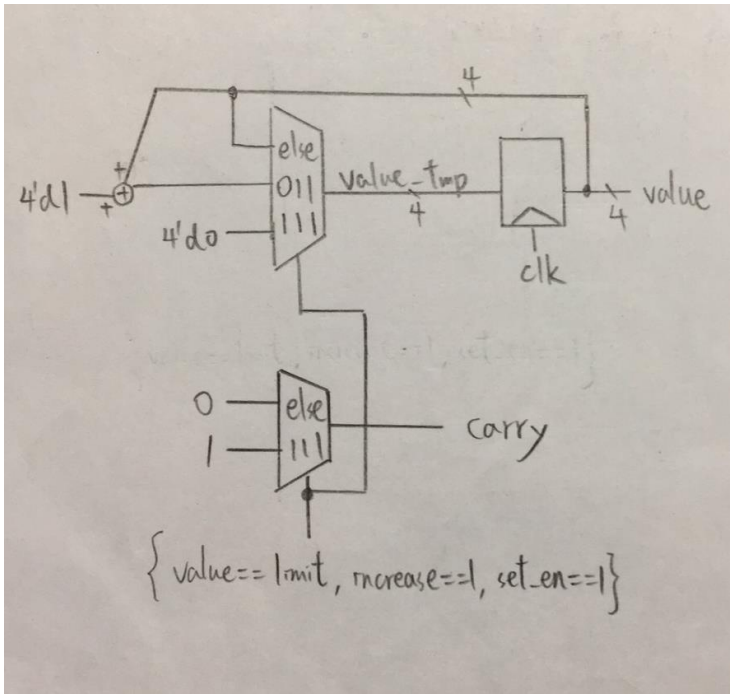
(2)SET_ON→SET_ON，維持在 SET_ON 狀態

**關閉 DIP_set:**

(1)SET_ON→SET_OFF，set enable = 0，不能設定時間

(2)SET_OFF→SET_OFF，維持在 SET_OFF 狀態

## 十進位上數器*4:



依照秒鐘和分鐘分成兩組:

**(1)秒鐘設定:**

**digit0:** (increase 為按鈕 set_sec 的輸入訊號，limit=9)

**digit1:** (increase 為 digit0 的 carry，limit=5)

(set_en 為 fsm_setting 的輸出值，代表的是現在是否可以進行設定時間的操作)

當 increase 等於 1 且 value 等於 limit 且 set_en 等於 1 時，代表下次需要進位，value_tmp = 0；當 increase 等於 1 但 value 不等於 limit 而 set_en 等於 1 時，代表下次要加 1 但不用進位，value_tmp = value + 1；其他狀況代表不需要加 1，value 維持現狀，value_tmp = value，當每次 clk 的 positive edge 來時，將 value 輸入 value_tmp 的值。

當 increase = 1 且 value = limit 且 set_en 等於 1 時，代表需要進位，carry 輸出 1 告知下一級；其餘狀況不需要進位，carry 輸出 0。

**(2)分鐘設定:**

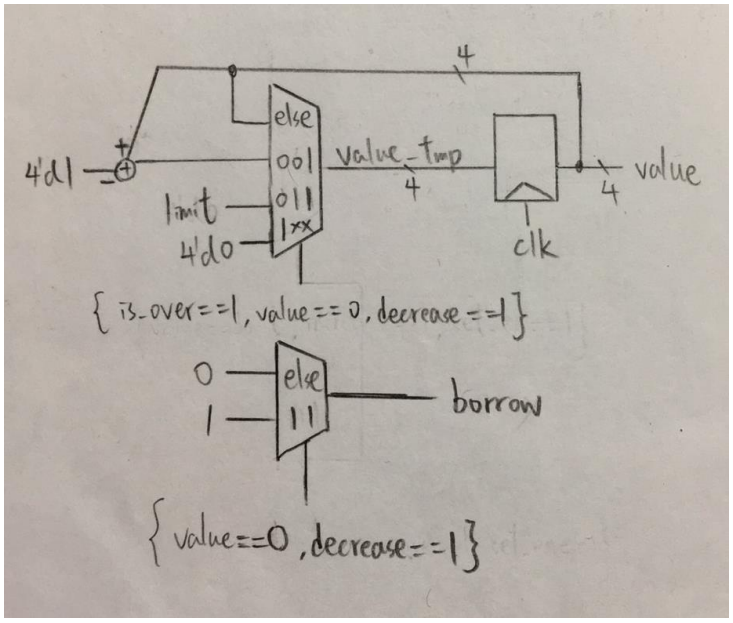
**digit2:** (increase 為按鈕 set_min 的輸入訊號，limit=9)

**digit3:** (increase 為 digit2 的 carry，limit=2)

分鐘設定的邏輯基本上與上述完全相同，而由於分鐘的上限為 23 分鐘，因此另外再加上一個判斷為當{digit2,digit3} = 23 時，digit2 和 digit3 的 value_tmp = 0，因此下一個 clk 時 digit2 和 digit3 的 value 會輸入 0，達到歸零的效果。

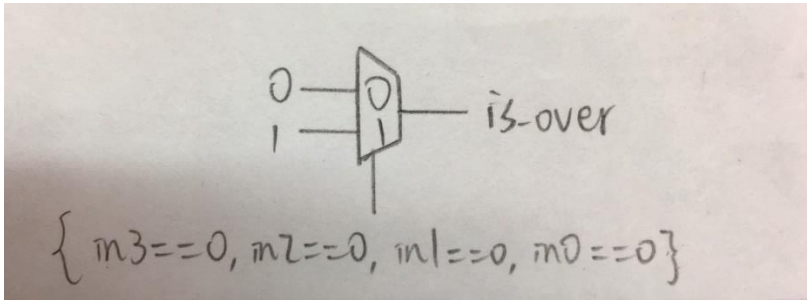
此外，由於這四個上數器的重置訊號是 fsm_setting 另外給的，因此當別的功能像是在操作 start/stop 時，設定好的時間不會被重置，而是在打開 DIP_set 的開關時才會進行重置。

<u>十進位下數器*4:</u>



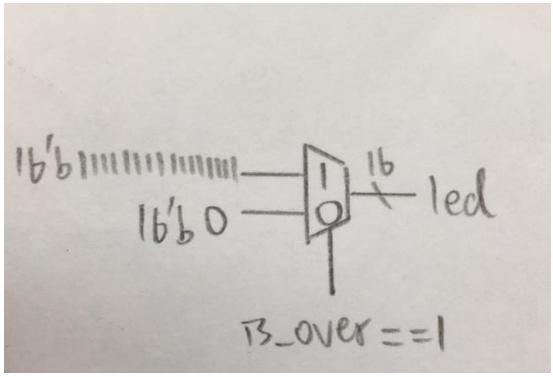
**digit0:** (decrease 為 fsm 的輸出 count enable，limit=9)
**digit1:** (decrease 為 digit0 的 borrow，limit=5)
**digit2:** (decrease 為 digit0 和 digit1 的 borrow (皆為 1 時)，limit=9)
**digit3:** (decrease 為 digit0、digit1 和 digit2 的 borrow (皆為 1 時))
(is_over 為下數器數到 00:00 的通知信號)
當 is_over 等於 1 時，代表已經倒數完畢，各級下數器維持 0 的輸出，因此 value_tmp = 0；當 decrease 等於 1 且 value 等於 0 時，代表需要借位，borrow = 1 告知下一級，value_tmp = limit；當 decrease 等於 1 但 value 不等於 0 時，代表下次要減 1 但不用借位，borrow = 0，value_tmp = value - 1；其他狀況代表不需要減 1 也不需要借位，borrow = 0，value 維持現狀，value_tmp = value，當每次 clk 的 positive edge 來時，將 value 輸入 value_tmp 的值。
此外，這四個下數器重置時會輸入設定好的時間，在上面的 fsm 中有提到，因為關掉設定時間的開關後會先重置後再回到 START，由於有先重置，因此可以將設定好的值輸入進下數器，並從新設定好的分鐘和秒數開始倒數

## is_over 控制:



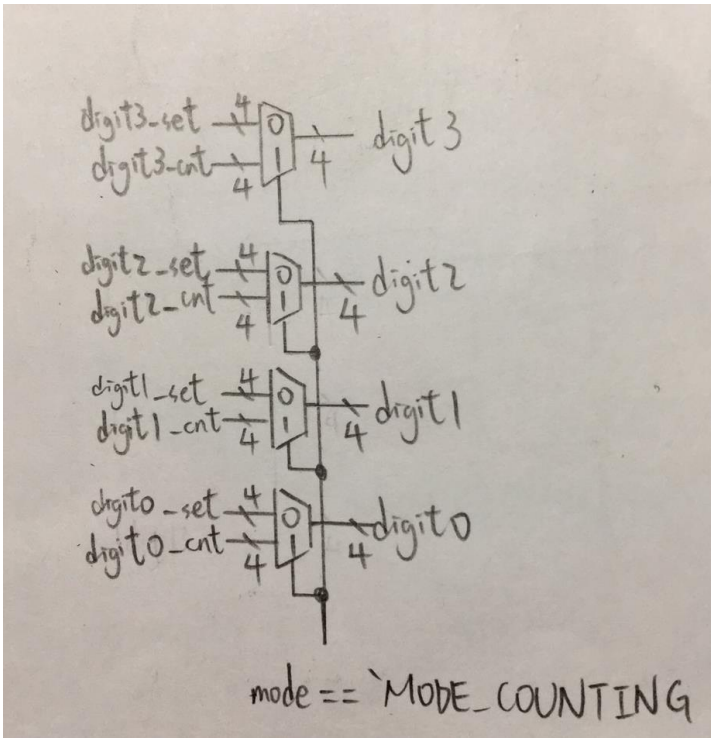

In3～in0 為下數器的 4 個位數的輸出，判斷當 4 位數輸出都為 0 時，輸出 is_over 為 1 告知下數器要停止下數，避免出現一直循環倒數的狀況。

## LED:




由上述 is_over 訊號來判斷是否已經倒數至 0，若是的話，16bit 的輸出訊號全部等於 1，LED 燈全部亮起；若不是倒數至 0，則輸出訊號全部等於 0，LED 燈全暗。

MUX:




以 mode 連接 fsm_setting 的輸出 set enable，digit3～0_cnt 連接下數器的輸出值，以及 digit3～0_set 連接設定時間的上數器的輸出值，藉由 mode 的訊號來選擇現在要顯示的是下數的畫面(mode = `MODE_COUNTING)還是設定時間的畫面(mode = `MODE_SETTING)，並輸出(digit3～0)給 scan control 當作輸入。

Scan control:

ssd_ctl_en = 00 → ssd_ctl = 0111
ssd_ctl_en = 01 → ssd_ctl = 1011
ssd_ctl_en = 10 → ssd_ctl = 1101
ssd_ctl_en = 11 → ssd_ctl = 1110

四個七段顯示器輪流顯示，ssd_ctl_en = 00 時→顯示第一個七段顯示器(分鐘十位數)；ssd_ctl_en = 01 時→顯示第二個七段顯示器(分鐘個位數)；ssd_ctl_en = 10 時→顯示第三個七段顯示器(秒數十位數)；ssd_ctl_en = 11 時→顯示第四個七段顯示器(秒數個位數)，以快速輪流顯示的方式達成視覺暫留的效果。

7-segment display decoder:

in = 0 → segs = 8'b0000_0011
in = 1 → segs = 8'b1001_1111
in = 2 → segs = 8'b0010_0101
in = 3 → segs = 8'b0000_1101

in = 4 → segs = 8'b1001_1001
in = 5 → segs = 8'b0100_1001
in = 6 → segs = 8'b0100_0001
in = 7 → segs = 8'b0001_1111
in = 8 → segs = 8'b0000_0001
in = 9 → segs = 8'b0000_1001
default 為 segs = 8’b1111_1111(全暗)

結合以上功能，可以得到一個可以設定時間，且具有 start/stop、pause/resume 功能的倒數計時器。

## I/O pin assignment:

| led[15] | led[14] | led[13] | led[12] | led[11] | led[10] | led[9] | led[8] |
|---|---|---|---|---|---|---|---|
| L1 | P1 | N3 | P3 | U3 | W3 | V3 | V13 |

| led[7] | led[6] | led[5] | led[4] | led[3] | led[2] | led[1] | led[0] |
|---|---|---|---|---|---|---|---|
| V14 | U14 | U15 | W18 | V19 | U19 | E19 | U16 |

| segs[7] | segs[6] | segs[5] | segs[4] | segs[3] | segs[2] | segs[1] | segs[0] |
|---|---|---|---|---|---|---|---|
| W7 | W6 | U8 | V8 | U5 | V5 | U7 | V7 |

| ssd_ctl[3] | ssd_ctl[2] | ssd_ctl[1] | ssd_ctl[0] | clk |
|---|---|---|---|---|
| W4 | V4 | U4 | U2 | W5 |

| pb | pb_pause | DIP_set | set_min | set_sec |
|---|---|---|---|---|
| W19 | U17 | R2 | U18 | T17 |

## Conclusion:

這次實驗當中我花了最久時間的就是畫出 block diagram、debug 以及做這份報告，由於實驗越趨複雜，block diagram 要愈連愈多， module 也愈來愈多個，各個 module 之間的線路與關係也愈來愈雜亂，而我雖然有先把每個小的功能都一個一個先確認過沒問題之後再連接訊號，但有時還是會出現 bug，像是 FSM

中我有單獨拉出來測試 count enable 和 reset 的輸出是沒問題的，但是接上下數器時卻不能正常的 pause/resume 或是 stop/start，我花了很多時間思考這個問題，最後我換一個邏輯一樣但是寫法不同的方式試試看，功能就變正常了，因此 debug 的時間遠遠比預想的還要花得更久，甚至是比設計過程還要久的時間，再來則是由於功能越來越多，報告中要討論的東西也越來越多，因此寫報告的時間也跟著拉長了，之後的實驗只會愈來愈複雜，還是希望我可以盡快熟悉 verilog 的格式等問題，才不會讓像是上述 FSM 中的 bug 花費掉我太多的時間，因為只要一個小地方出錯，可能整個大的功能都會出問題，因此還是老話一句，希望我盡量不要在非必要的地方出錯，並且在以後的實驗也能夠更加的順利。