邏輯設計實驗 Lab7 結報

105060012 張育菘

1 Implement a stopwatch function 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.

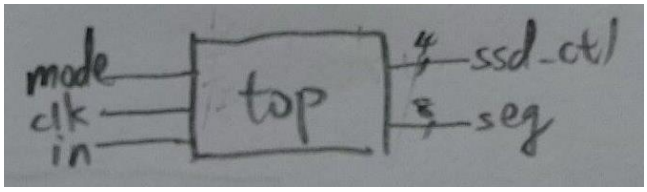
clk	mode	in
W5	T17	U16

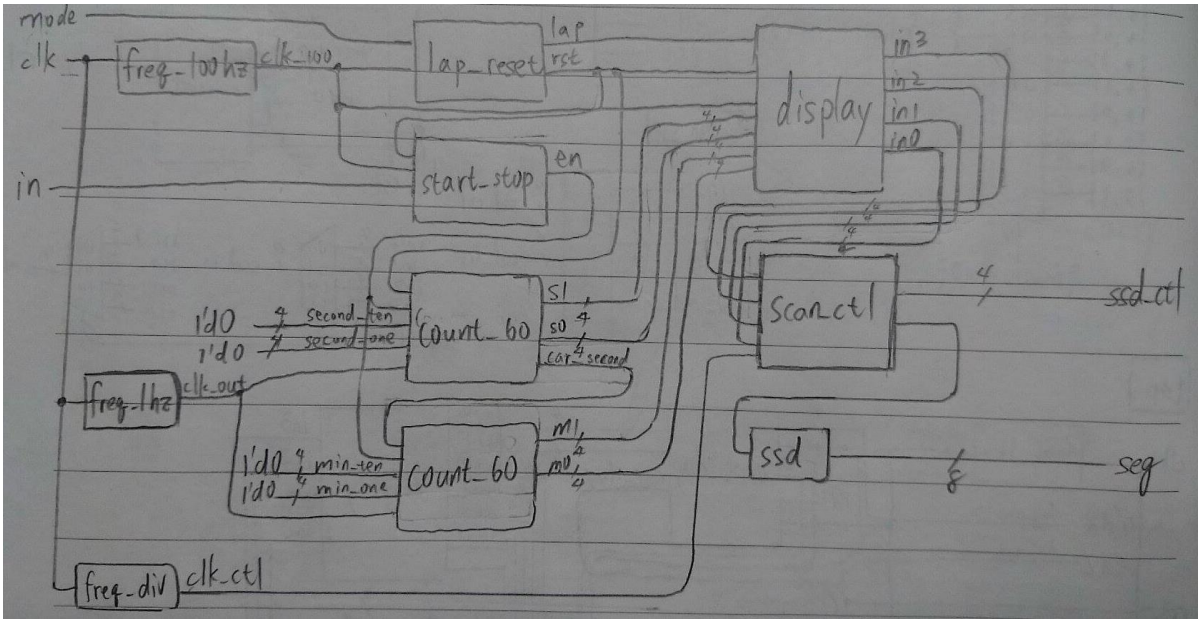
Design Specification

Input：clk,mode,in;

Output：ssd_ctl[3:0],seg[7:0],pm;

block diagram :





Design Implementation

Logic function :

1. top：

   此 top module，專門拿來呼叫其他小 moudule 的。

2. fsm：

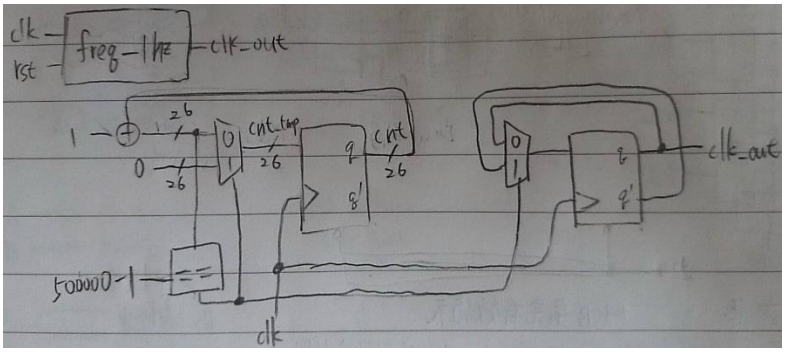
   此 module 做為 finite state machine，以接出 state(在 display 內)決定要顯示 lap 抑或是顯示正在數的狀態。

   這裡的 clk 我是接 100hz，因為 clk 在 0→1 時讀 one_pulse(其 clk 為 100hz)處理過的值，因此 top 的 clk 不能太快。

3. freq_1hz：

   此為除頻的 module，輸出 1hz 的 clk_out 控制 count_down。

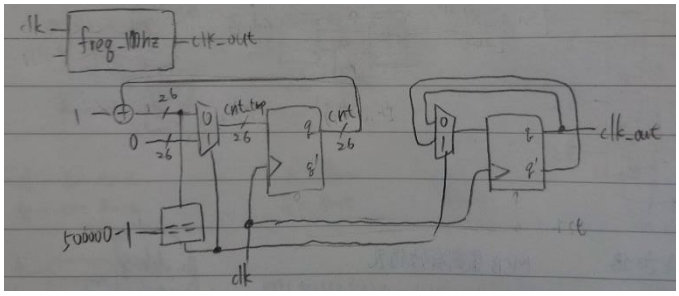
   - 這裡不能接 top 的 input rst，因為那裡的 rst 是希望 count_up 重新開始數。當 rst=0 時，freq_1hz 輸出的 clk 持續為 0→freq_1hz 不做事。



4. freq_100hz：

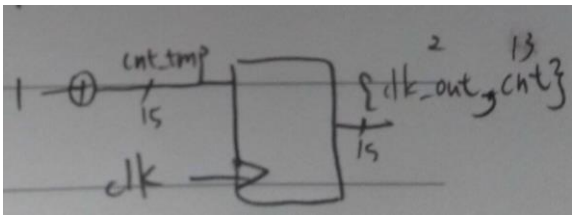
   此為除頻的 module，輸出 100hz 的 clk_out 當作 debounce_circuit、one_pulse 以及 fsm 的 clk。

   - 這裡不能接 top 的 input rst，因為那裡的 rst 是希望 count_up 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk 持續為 0→freq_100hz 不做事。



5. freq_div：

   此為除頻的 module，輸出的 clk_ctl 當作 scan_ctl 的 clk，其頻率極高。

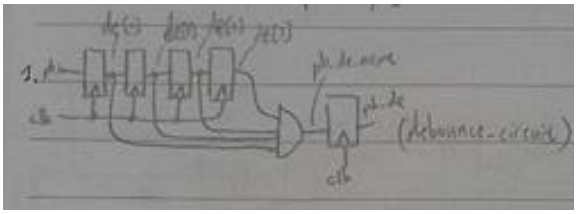


6. debounce_circuit：

   此為**除按鈕雜訊**的 module，原理是，當 debounce_circuit 收到連續 4 個 1 時才產生一個 0→1 的訊號，這樣可以避免收到前段起伏不定的雜訊，達到除雜訊的效果。

   且要接 100hz 的 clk，因為若接此的 clk 頻率太快就無法達到除雜訊的效果，太慢可能會超過按鈕維持穩定值的時間，因此我選擇 100hz 作為 debounce_circuit 的 clk。

   - 這裡不能接 top 的 input rst，因為那裡的 rst 是希望 count_up 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk_100 持續為 0→ debounce_circuit 不做事。

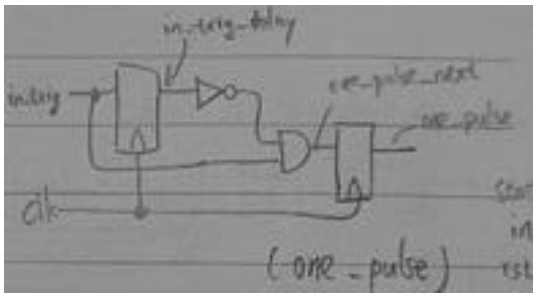


7. one_pulse：

   輸入訊號經過 debounce_circuit 處理後，其週期可能會超過 1 個 clk 的時間，因此需要此 module 使訊號的週期為 1 個 clk 的時間。

   這裡的 clk 我也是接與 debounce_circuit 同的 100hz。

   - 這裡不能接 top 的 input rst，因為那裡的 rst 是希望 count_up 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk_100 持續為 0→ one_pulse 不做事。

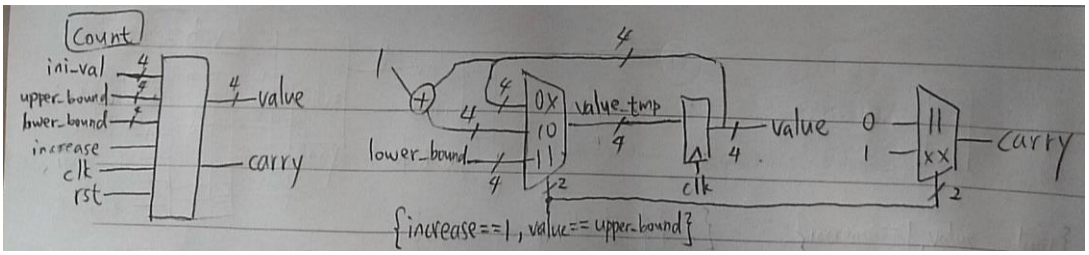


8. count：

   此 module 是用來做一個數的加法。

   carry 表示進位；increase 表示低一為數的進位；upper_bound 表示上界；lower_bound 表示下界；ini_val 表示初始值。

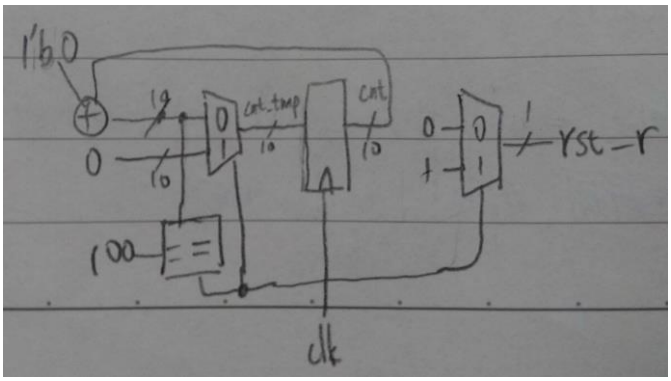
   當 increase=1 表示有數進來，因此可以繼續進行加法。



9. reset：

   此 module 功能是為產生 rst 訊號。

   而此 input 訊號 rst 來自只經過 debounce_circuit 處理過的輸入訊號 in_de，因為此週期可能為多個週期的 clk_100，因此可以拿來作為判斷長按的依據。



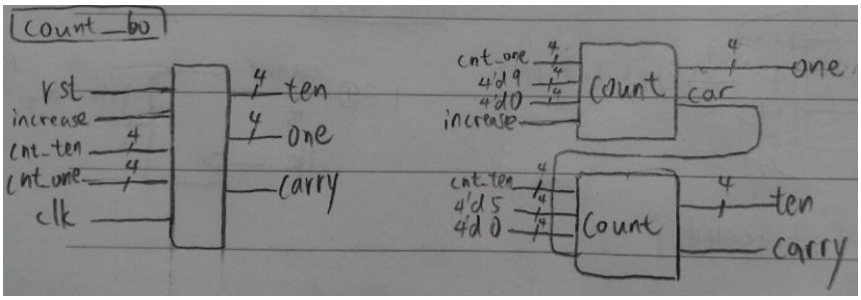
10. count_60：

    此 module 是用來做 0~60 的加法。

    由於 60 是一兩個數，因此內部接兩個 count，分別做個位數及十位數的加。

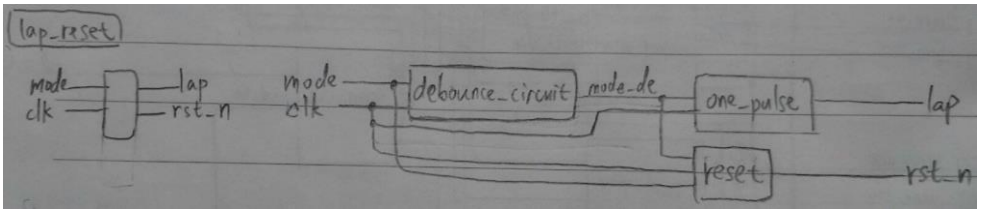
    此接出去的 carry 會作為另一個 count_60 的 increase。

    - 個位數的 carry 即為十位數的 increase，由於 carry 是在作暫存值時就給值了，因此當個位數要進位時，個位數以及十為數會同時變化。



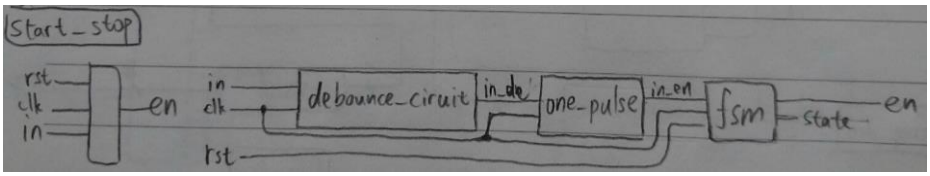
11. lap_reset：

    此 module 是用來產生 lap(短按)以及 reset(長按)的訊號。



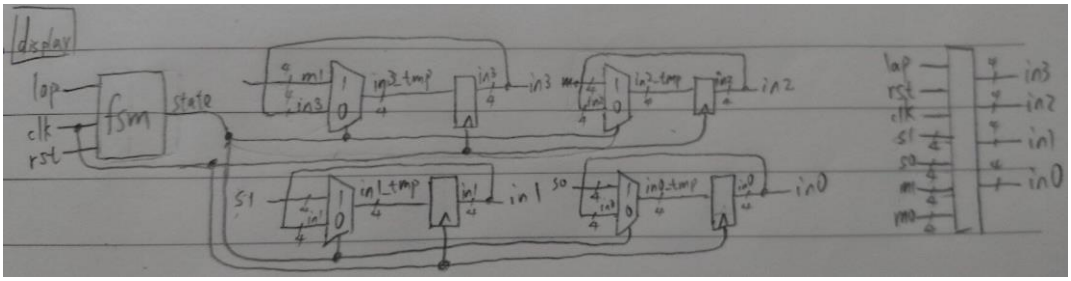
12. start_stop：

此 module 是用來產生 count_60 能否開始數的 en 訊號，因此內部有接一個 fsm。



13. display：

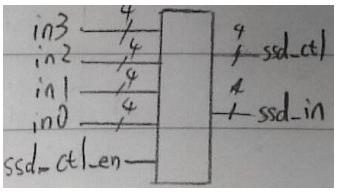
此 module 是用來決定七段顯示器要顯示何種狀態，因此內部亦有接一個 fsm。當 state==1 表示要顯示還在繼續數的狀態；當 state==1 時表示要顯示當下數到的時間，並停在此時，但是因為這只是在 display 動的手腳，因此背後的 counter 仍然持續在往上數。

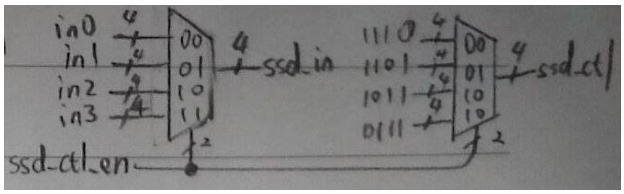


14. scan_ctl：

此 module 是讓七段顯示器同時顯示不同位數的 module。

ssd_ctl_en(from freq_div 的 clk_ctl[2:0])的用意為顯示兩個不同的數，因為七段顯示器一次只能顯示一種數次，因此需要一個頻率介於內建時間到 1hz 的 ssd_ctl_en 產生同時顯示兩個數的錯覺。





15. ssd：

當 bcd 為 4'd0: segs = 8'b00000011; //七段顯示器顯示”0”

當 bcd 為 4'd1: segs = 8'b10011111; //七段顯示器顯示”1”

當 bcd 為 4'd2: segs = 8'b00100101; //七段顯示器顯示”2”

當 bcd 為 4'd3: segs = 8'b00001101; //七段顯示器顯示”3”

當 bcd 為 4'd4: segs = 8'b10011001; //七段顯示器顯示”4”

當 bcd 為 4'd5: segs = 8'b01001001; //七段顯示器顯示”5”

當 bcd 為 4'd6: segs = 8'b01000001; //七段顯示器顯示”6”

當 bcd 為 4'd7: segs = 8'b00011111; //七段顯示器顯示”7”

當 bcd 為 4'd8: segs = 8'b00000001; //七段顯示器顯示”8”

當 bcd 為 4'd9: segs = 8'b00001001; //七段顯示器顯示”9”

當 bcd 為 4'd10:segs=8'b11111111;　　//七段顯示器顯示” “
default: segs = 8'b00000000;　　//七段顯示器顯示”8.”

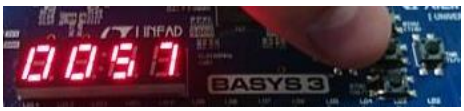
Result (T17 控制 lap(短按)和 rst(長按)；W19 控制 start_stop)

1. 在數到 30 時按下 T17(圖一)，表示表示要顯示當下數到的時間，並停在此時，但背後的 count_60 仍未停止繼續數。→過一段時間後再次按下 T17(圖二)，顯示現在真正數到的時間 52。
2. 在數到 57 時按下 W19(圖三)→暫停在當下→再次按下 W19(圖四)，再次開始數。
3. 在數到 1 分 11 秒時常按 T17(圖五) →產生 rst 訊號→回到 00:00(圖六)。

圖一

圖二

圖三



圖四

圖五

圖六

Discussion

1. 我在這設計 start_stop 的功能是使顯示暫停/開始的訊號
2. 在處理 lap 訊號時我是在 display 內處理的，這樣就不會影響到 count_60 的運作，只會影響到七段顯示器顯示的數字而已。

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 hour and minute.

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.

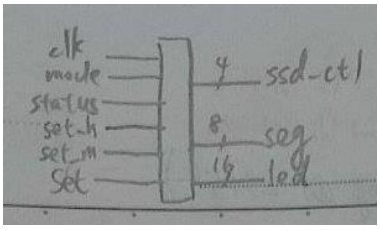
status	mode	set	set_h	set_m	clk
W19	T17	V17	T18	U17	W5

Design Specification

Input：status,mode,set,set_h,set_m,;

Output：ssd_ctl[3:0],seg[7:0],led[15:0];

block diagram :



Design Implementation

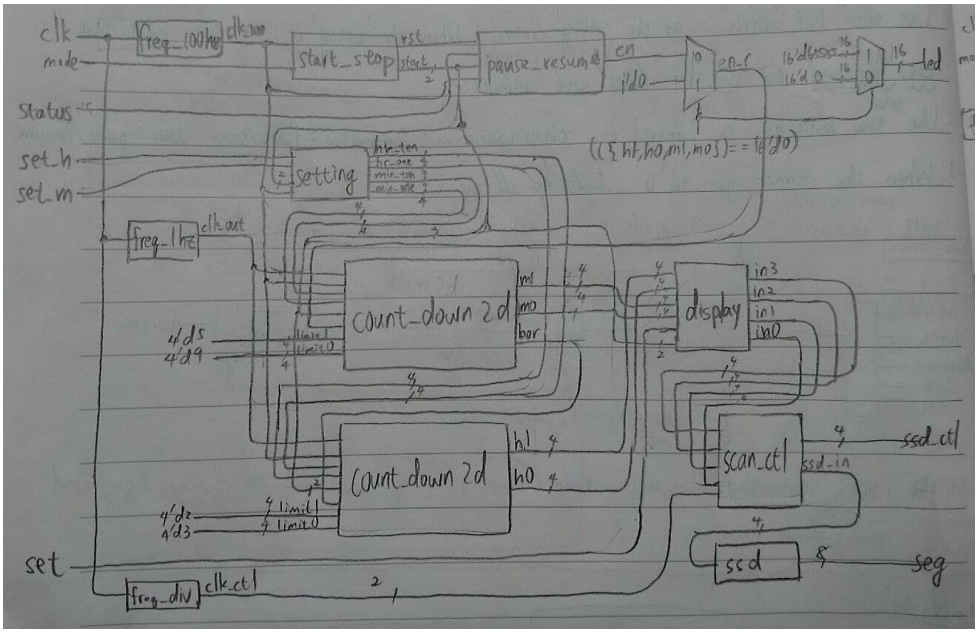
Logic function :

1. top：

此 top module，專門拿來呼叫其他小 moudule 的。

並判斷 led 何時要亮。(在({h1,h0,m1,m0})==16'd0 時亮(顯示到 00:00 時))

還有決定接近去 count_down2d 的 en_r 的值，因為當({h1,h0,m1,m0})==16'd0 時，表示倒數完了，要停在 00:00。



2. fsm：

   此 module 做為 finite state machine，以接出 en 作為判斷 count_down2d 能否運作的關鍵。

   - 這裡的 clk 我是接 100hz，因為 clk 在 0→1 時讀 one_pulse(其 clk 為 100hz)處理過的值，因此 top 的 clk 不能太快。

3. fsm_start：

   此 module 做為 finite state machine，以接出 start[1:0]作為判斷”電子鐘”現在的狀態。

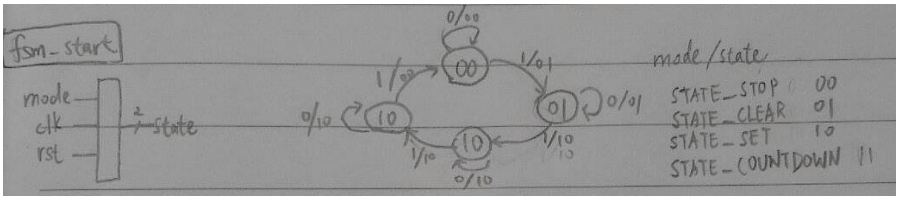
   00 表示 STATE_STOP，顯示最終時間，並無法利用”開始暫停”按鍵改變。

   01 表示 STATE_CLEAR，會將七段顯示器及 counter 內的數歸零。

   10 表示 STATE_SET，此時可以利用 set_h(小時)及 set_m(分鐘)設定時間。

   11 表示 STATE_COUNTDOWN，可以開始進行倒數。

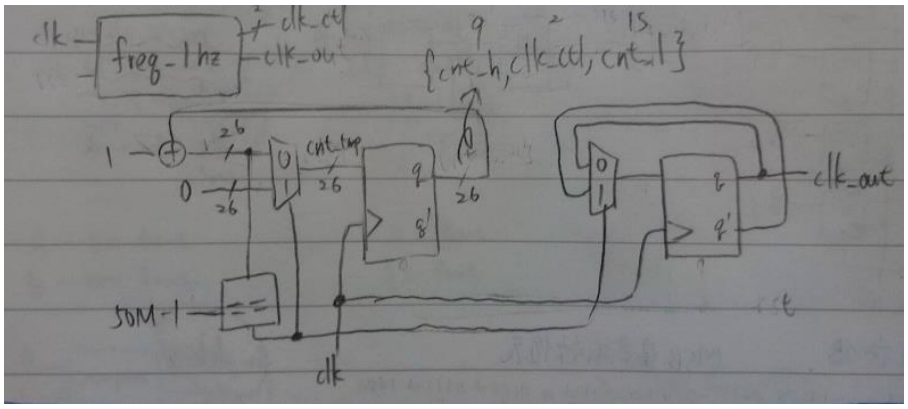
   - 這裡的 clk 我是接 100hz，因為 clk 在 0→1 時讀 one_pulse(其 clk 為 100hz)處理過的值，因此 top 的 clk 不能太快。



4. freq_1hz：

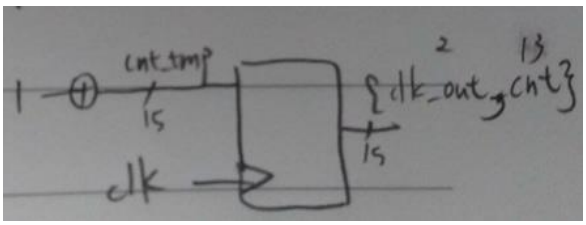
   此為除頻的 module，分別輸出 1hz 的 clk_out 控制 count_down 以及控制 scan_ctl 的 clk_ctl。

   - 這裡不能接 display 的 input rst，因為那裡的 rst 是希望 count_down 重新開始數。當 rst=0 時，freq_1hz 輸出的 clk 持續為 0→freq_1hz 不做事。



5. freq_div：

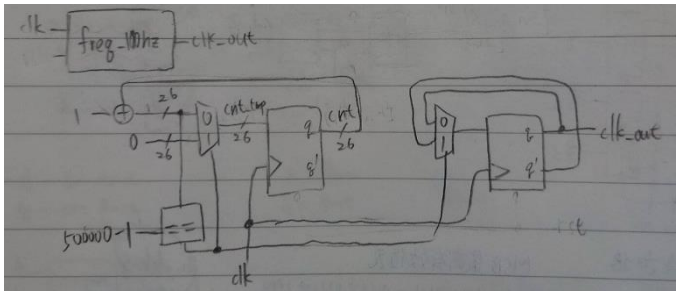
   此為除頻的 module，輸出的 clk_ctl 當作 scan_ctl 的 clk，其頻率極高。



6. freq_100hz：

   此為除頻的 module，輸出 100hz 的 clk_out 當作 debounce_circuit、one_pulse 以及 fsm 的 clk。

   - 這裡不能接 top 的 input rst，因為那裡的 rst 是希望 count_up 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk 持續為 0→freq_100hz 不做事。

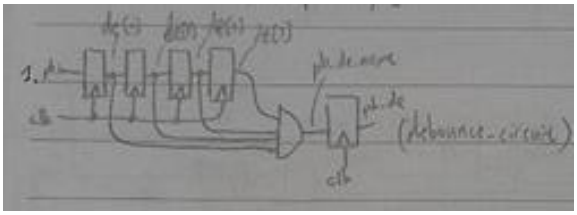


7. debounce_circuit：

   此為**除按鈕雜訊**的 module，原理是，當 debounce_circuit 收到連續 4 個 1 時才產生一個 0→1 的訊號，這樣可以避免收到前段起伏不定的雜訊，達到除雜訊的效果。

   且要接 100hz 的 clk，因為若接此的 clk 頻率太快就無法達到除雜訊的效果，太慢可能會超過按鈕維持穩定值的時間，因此我選擇 100hz 作為 debounce_circuit 的 clk。

   - 這裡不能接 top 的 input rst，因為那裡的 rst 是希望 count_up 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk_100 持續為 0→ debounce_circuit 不做事。

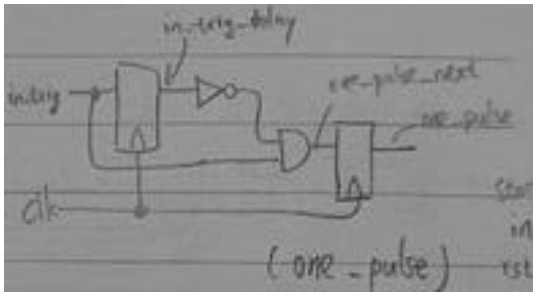


8. one_pulse：

   輸入訊號經過 debounce_circuit 處理後，其週期可能會超過 1 個 clk 的時間，因此需要此 module 使訊號的週期為 1 個 clk 的時間。

   這裡的 clk 我也是接與 debounce_circuit 同的 100hz。

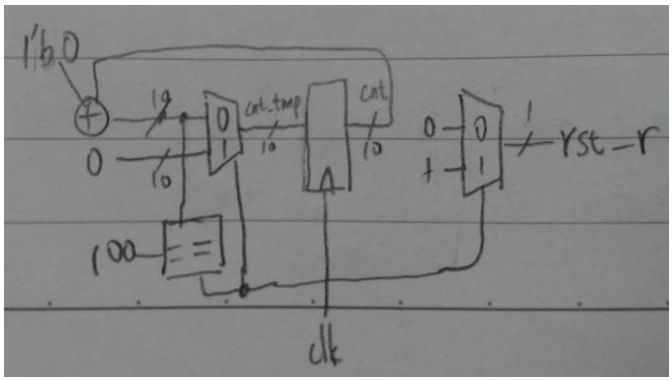
   - 這裡不能接 top 的 input rst，因為那裡的 rst 是希望 count_up 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk_100 持續為 0→ one_pulse 不做事。



9. reset：

   此 module 功能是為產生 rst 訊號。

   而此 input 訊號 rst 來自只經過 debounce_circuit 處理過的輸入訊號 in_de，因為此週期可能為多個週期的 clk_100，因此可以拿來作為判斷長按的依據。

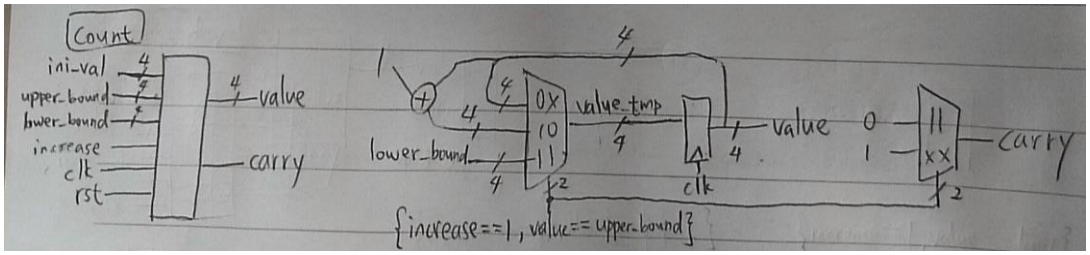


10. count：

    此 module 是用來做一個數的加法。

    carry 表示進位；increase 表示低一位數的的進位；upper_bound 表示上界；lower_bound 表示下界；ini_val 表示初始值。

    當 increase=1 表示有數進來，因此可以繼續進行加法。

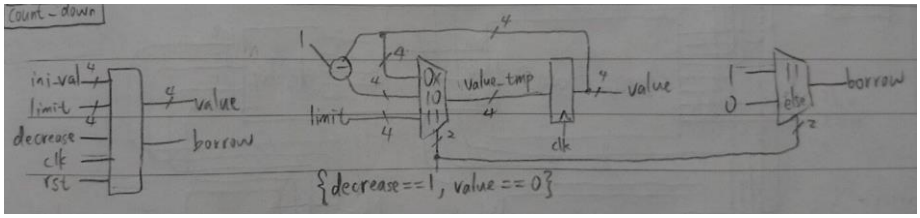


11. count_down：

    此 module 是用來做一個數的減法。

    borrow 表示借位；decrease 表示低一位數的借位；limit 表示此 digit 得 ceiling；ini_val 表示初始值。

    當 decrease=1 表示有數要借位，因此可以繼續進行減法。

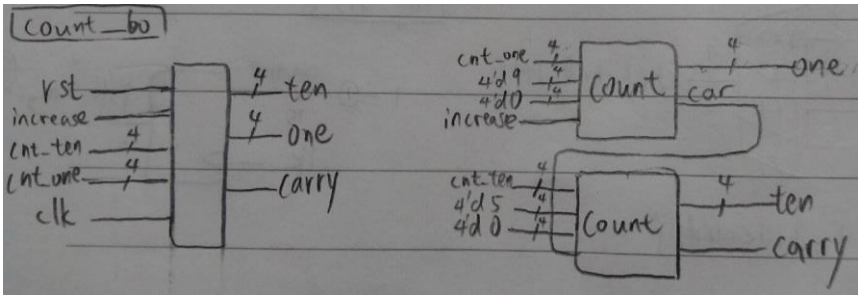


12. count_60：

此 module 是用來做 0~60 的加法。

由於 60 是一兩個數，因此內部接兩個 count，分別做個位數及十位數的加。

- 個位數的 carry 即為十位數的 increase，由於 carry 是在作暫存值時就給值了，因此當個位數要進位時，個位數以及十為數會同時變化。
- 其 increase 來自 set_m 經 one_pluse 處理過，再經 mux 選擇過後的 min_en

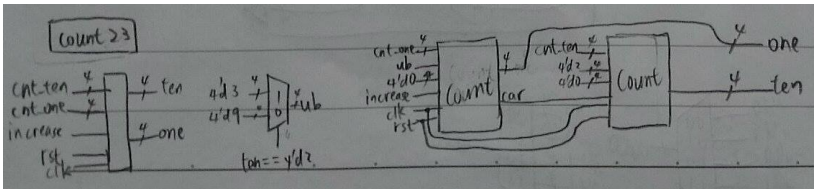


13. count_23：

此 module 是用來做 0~23 的加法。

由於 23 是一兩個數，因此內部接兩個 count，分別做個位數及十位數的加。

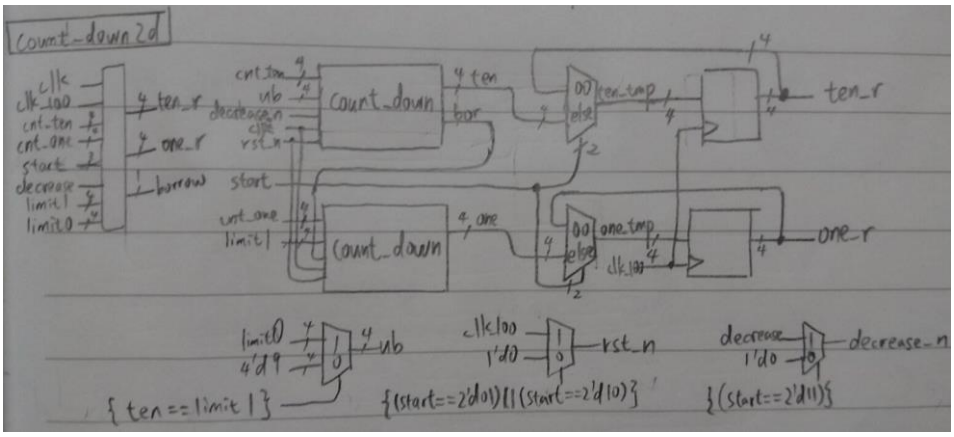
- 數個位數的 count 的上界 ub 待 ten 是否為 4’d2 決定。
- 個位數的 carry 即為十位數的 increase，由於 carry 是在作暫存值時就給值了，因此當個位數要進位時，個位數以及十為數會同時變化。
- 其 increase 來自 set_h 經 one_pluse 處理過，再經 mux 選擇過後的 hr_en



14. count_down2d：

此 module 是用來做 2 位數的減法。

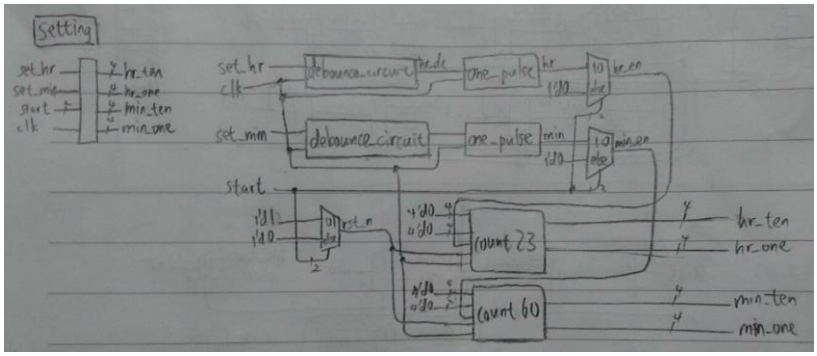
- 由於當 start==2’d0(STATE_STOP)，要一直顯示在當下的時間，因此有接 DFF 去做處理。
- 數個位數的 count_down 的上界 limit 待 ten 是否為 limit1 決定。
  其 rst_n 訊號由((start==2'b01)||(start==2'b10))決定，當 start 的狀態為 STATE_CLEAR 或 STATE_SET 時，rst_n==clk_100，表示隨時讀取經 count23 及 count60 處理過的 hr_ten,hr_one,min_ten,min_one 值，這樣的好處就是，七段顯示器上顯示的時間只要從這裡讀去就好，不用再接 count 去選擇讀值。
- decrease_n 訊號由(start==2'b11)(STATE_COUNTDOWN)決定，當 start 在 STATE_COUNTDOWN 時，表示可以進行倒數，因此此時 decrease_n 就為 decrease。



15. setting：

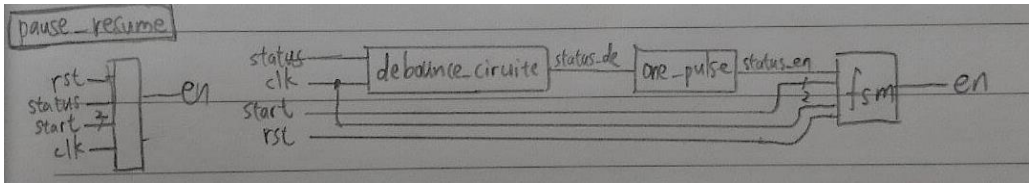
此 module 的用意在當作 start==2'b10 即在(STATE_SET)狀態時最主要使用到的部分，用這個 module 作為設定時間的功能。

- 這邊有用個小巧思就是 counter 的 increase 來自設定時間的按鍵訊號經 one_pluse 處理過，再經 mux 選擇過後的產生的 enable，用意在於當按下一次按鍵就能使要調的部分(小時/分鐘)時間+1。
- 當 start==2'b01 即在(STATE_CLEAR)狀態，使 rst_n 為 1’d1，使其作 reset。



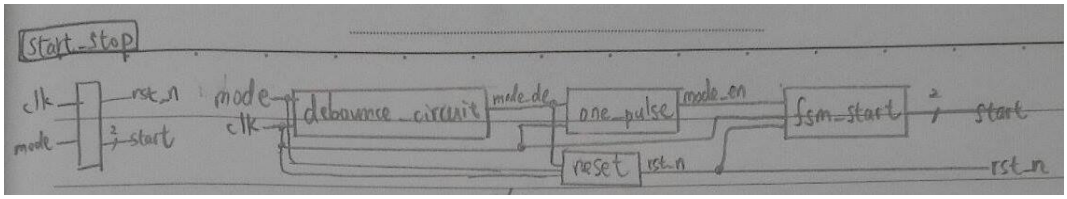
16. pause_resume：

此 module 是用來產生 en 訊號作為 count_down2d 能否開始倒數的關鍵，因此內部有接一個 fsm。



17. start_stop：

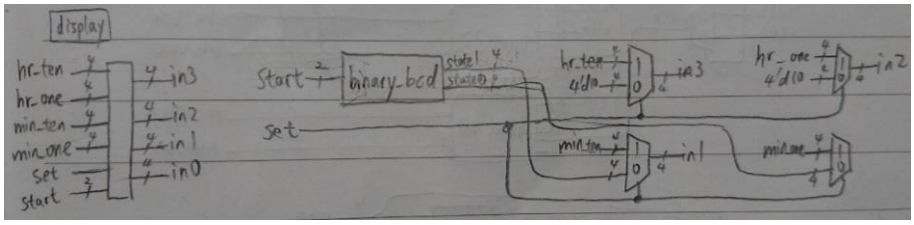
此 module 是用來產生 start(短按)作為”電子鐘”的狀態以及 reset(長按)的訊號。



18. display：

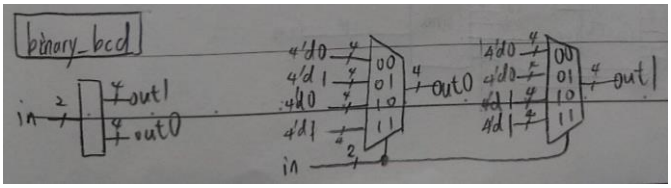
此 module 是用來決定 scan_ctl 的 input :in3, in2,in1,in0。

- 而此接 set(switch)訊號進來的原因是，我想要顯示”start”現在的狀態，這樣也比較好藉由開關判斷現在的狀態。
- 由於 start 訊號為一 2 bits 的數，因此有接一個 binary 轉 bcd 的 module。



19. binary_bcd：

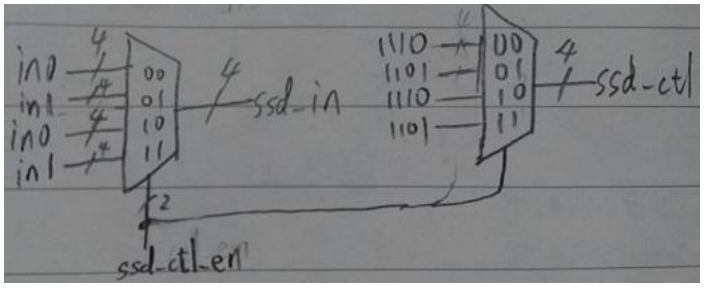
此是用來做 2 bits 的 binary 轉 bcd 的 module。



20. scan_ctl：

此 module 是讓七段顯示器同時顯示不同位數的 module。

ssd_ctl_en(from freq_1hz 的 clk_ctl[2:0])的用意為顯示兩個不同的數，因為七段顯示器一次只能顯示一種數次，因此需要一個頻率介於內建時間到 1hz 的 ssd_ctl_en 產生同時顯示兩個數的錯覺。



21. ssd：

當 bcd 為 4'd0: segs = 8'b00000011; //七段顯示器顯示”0”
當 bcd 為 4'd1: segs = 8'b10011111; //七段顯示器顯示”1”
當 bcd 為 4'd2: segs = 8'b00100101; //七段顯示器顯示”2”
當 bcd 為 4'd3: segs = 8'b00001101; //七段顯示器顯示”3”
當 bcd 為 4'd4: segs = 8'b10011001; //七段顯示器顯示”4”
當 bcd 為 4'd5: segs = 8'b01001001; //七段顯示器顯示”5”
當 bcd 為 4'd6: segs = 8'b01000001; //七段顯示器顯示”6”
當 bcd 為 4'd7: segs = 8'b00011111; //七段顯示器顯示”7”
當 bcd 為 4'd8: segs = 8'b00000001; //七段顯示器顯示”8”
當 bcd 為 4'd9: segs = 8'b00001001; //七段顯示器顯示”9”

當 bcd 為 4'd10:segs=8'b11111111;　　//七段顯示器顯示” “
default: segs = 8'b00000000;　　//七段顯示器顯示”8.”

Result

(T17:mode(長按為 rst;短按為切換 start 的狀態)、W19:status(控制 pause/resume)、T18:set_h、U17:set_m、V17:set(set==1，顯示 start 的狀態;set==0，顯示當下的時間))

1. 當 start 的狀態為 10(STATE_SET)(圖一)→表示可以設定時間(set_h 設定小時;set_m 設定分鐘)(圖二)。
2. 當 start 的狀態為 11(STATE_COUNTDOWN)(圖三)→表示可以開始倒數，因為此時 pause/resume 停留在 pause 的狀態，因此需按一下 status 始知開始倒數(圖四)
3. 當 start 的狀態為 00(STATE_STOP)(圖五)→表示會停留在當下的時間，此時不管有沒有按下 status 鍵，都不會改變現在的狀態(圖六)
4. 當 start 的狀態為 01(STATE_CLEAR)(圖七)→會將時間歸零(圖八)
5. 當數到 00:00 時(圖九)，下面那排的 LED 燈都會亮，並停在 00:00。

圖一

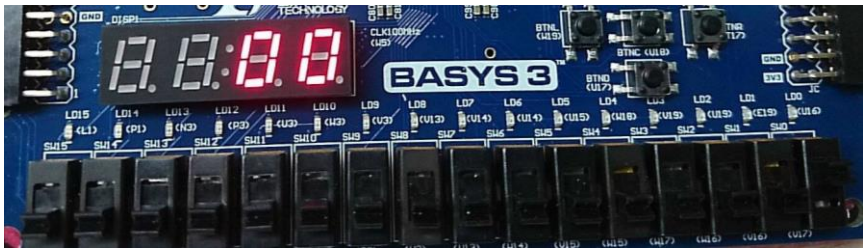
圖二

圖三

圖四


BASYS 3
UNIVERSITY PROGRAM

圖五


BASYS 3

圖六


BASYS 3

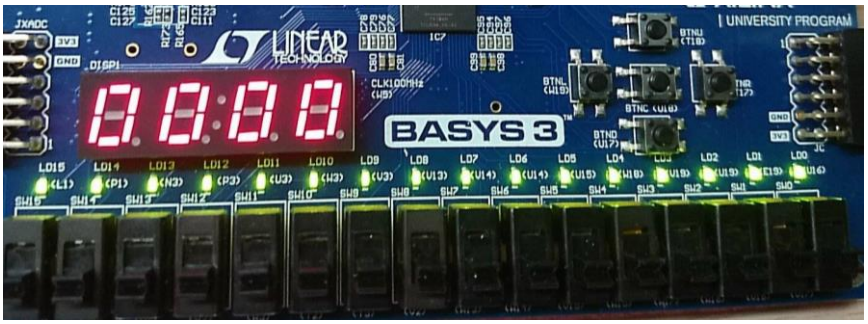
圖七


BASYS 3

圖八


BASYS 3
UNIVERSITY PROGRAM

圖九



Discussion

1. start
   00 表示 STATE_STOP，顯示最終時間，並無法利用”開始暫停”按鍵改變。
   01 表示 STATE_CLEAR，會將七段顯示器及 counter 內的數歸零。
   10 表示 STATE_SET，此時可以利用 set_h(小時)及 set_m(分鐘)設定時間。
   11 表示 STATE_COUNTDOWN，可以開始進行倒數。
2. 在做 count_down2d 時，數個位數的 count_down 的上界 limit 待 ten 是否為 limit1 決定。
   其 rst_n 訊號由((start==2'b01)||(start==2'b10))決定，當 start 的狀態為 STATE_CLEAR 或 STATE_SET 時，rst_n==clk_100，表示隨時讀取經 count23 及 count60 處理過的 hr_ten,hr_one,min_ten,min_one 值，這樣的好處就是，七段顯示器上顯示的時間只要從這裡讀去就好，不用再接 count 去選擇讀值。
3. 在做 setting 時，counter 的 increase 來自設定時間的按鍵訊號經 one_pluse 處理過，再經 mux 選擇過後的產生的 enable，用意在於當按下一次按鍵就能使要調的部分(小時/分鐘)時間+1。
4. display 接 set(switch)訊號進來的原因是，我想要顯示”start”現在的狀態，這樣也比較好藉由開關判斷現在的狀態。

Conclusion：

這次的 lab 做出來真的很有成就感，雖然過程中挫折不斷，也通了宵，但是也學到了一些東西，像是對 fsm 的功用更加了解之外，也懂得如何產生類似像碼錶的功能，真的很棒呢。