

邏輯設計實驗 Lab6 結報

105060012 張育菘

1 . Finish the time display function supporting 24-hour (00-23). 1.1 Support two modes: AM/PM and 24-hour.

hr_ten	hr_one	hr_one	min_one
R2,T1,U1,W2	R3,T2,T3,V2	W13,W14,V15,W15	W17,W16,V16,V17

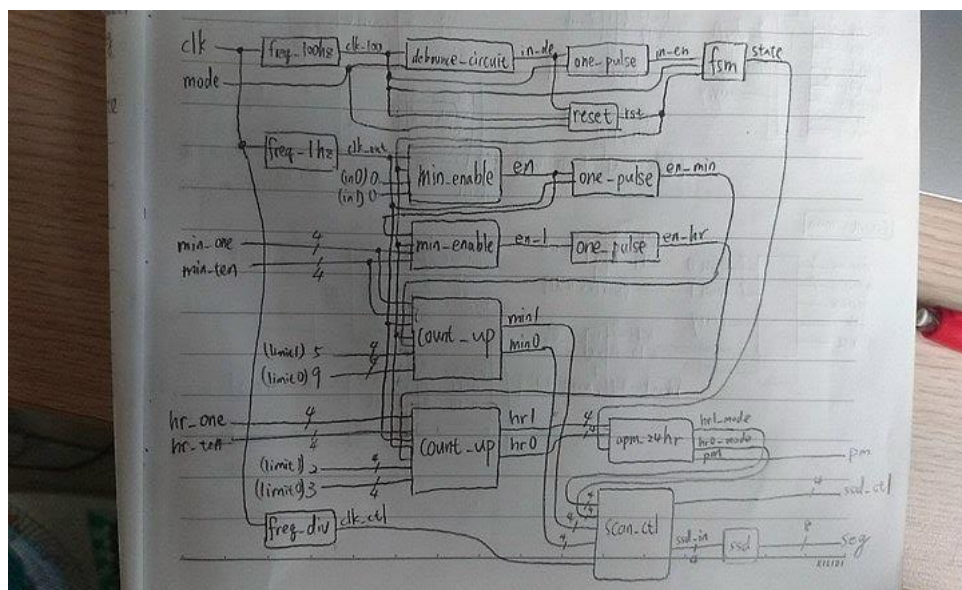
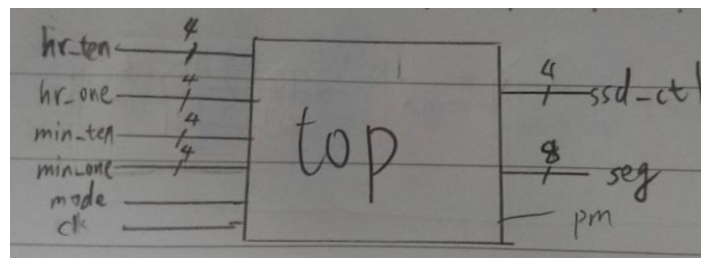
clk	mode	pm
W5	T17	U16

Design Specification

Input : clk,mode,hr_ten[3:0], hr_one [3:0], hr_one [3:0], min_one [3:0];

Output : ssd_ctl[3:0],seg[7:0],pm;

block diagram :



Design Implementation

Logic function :

1. top :

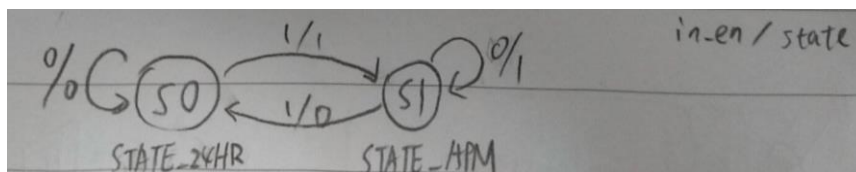
此 top module，專門拿來呼叫其他小 module 的。

2. fsm :

此 module 做為 finite state machine，以接出 state 至 apm_24hr 決定顯示 24hr 抑或是 A/PM 的關鍵。

其 input 為經過 one_pulse 處理過的訊號 in_en，因此當按下 mode 按鈕時，只會顯示一下的 A/PM 形式(1hz)。

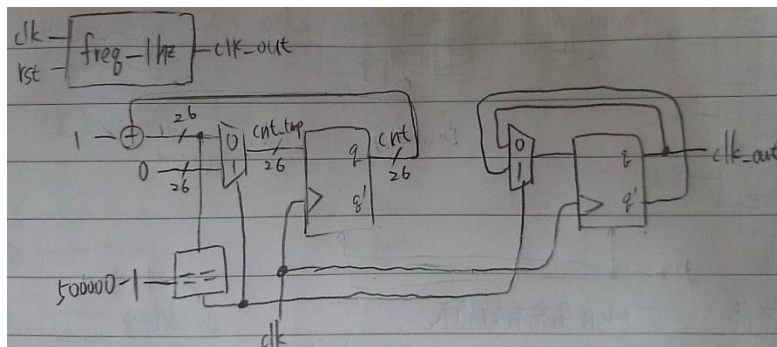
這裡的 clk 我是接 100hz，因為 clk 在 0→1 時讀 one_pulse(其 clk 為 100hz)處理過的值，因此 top 的 clk 不能太快。



3. freq_1hz :

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

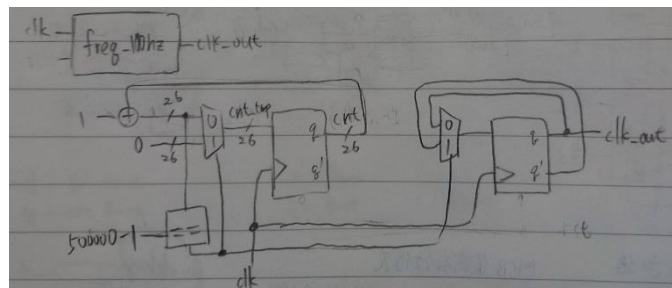
- 這裡不能接 top 的 input rst，因為那裡的 rst 是希望 count_up 重新開始數。當 rst=0 時，freq_1hz 輸出的 clk 持續為 0→freq_1hz 不做事。



4. freq_100hz :

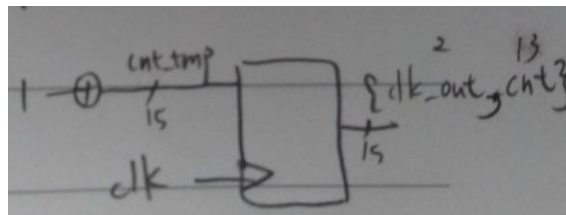
此為除頻的 module，輸出 100hz 的 clk_out 當作 debounce_circuit、one_pulse 以及 fsm 的 clk。

- 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。

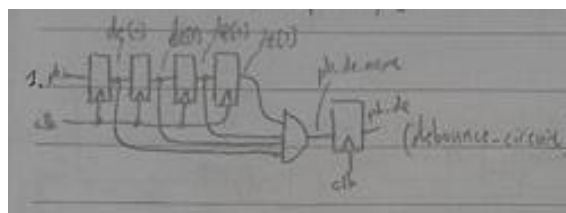


6. debounce_circuit :

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。

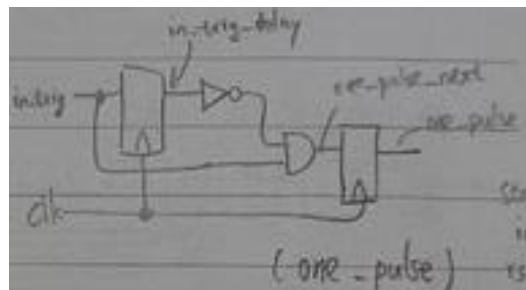


7. one_pulse :

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_up :

這是同時數兩個變數，個位數 cnt_one 跟十位數 cnt_ten 進行往下數。當 cnt_one=9 時把 cnt_ten 的數值加一以及 cnt_one 的數值變成 0；當 cnt_one 跟 cnt_ten 分別為 limi0 & limit1(上界)時，讓兩個數變成 0。而初始化(rst=0)

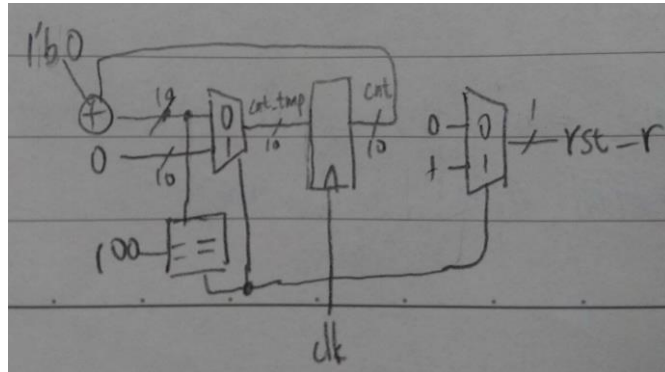
就是把數設為 $\text{cnt_ten}=\text{in1} \ \& \ \text{cnt_one}=\text{in0}$ 。

這裡還接一個 en ，作為是否能往下數的判斷值，若 $\text{en}=1$ ，表示可以往上數；若 $\text{en}=0$ ，則不能繼續往上數。

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。



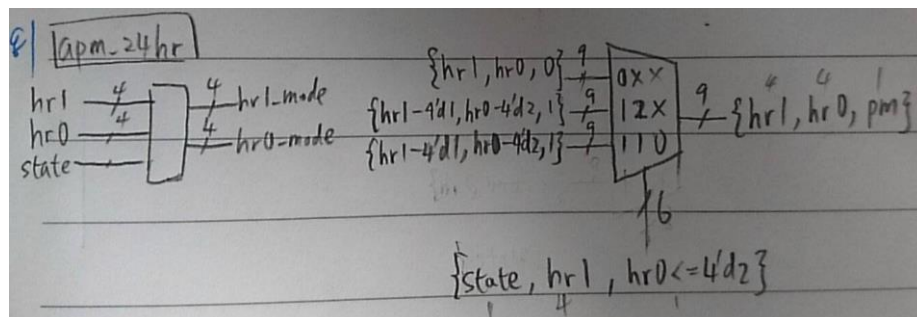
10. apm_24hr :

此 module 是決定要顯示 24hr 還是 A/PM。(STATE_24HR : 0, STATE_APM : 1)

作法就是判斷 $\text{state}, \text{hr1}$ 的值 $\&(\text{hr0} \leq 4'd2)$ 此判斷式的對錯：

當 $(\text{hr1} == 4'd2) \&\&(\text{state} == 1)$ 或是 $(\text{hr1} == 4'd1) \&\&(\sim(\text{hr0} \leq 4'd2)) \&\&(\text{state} == 1)$

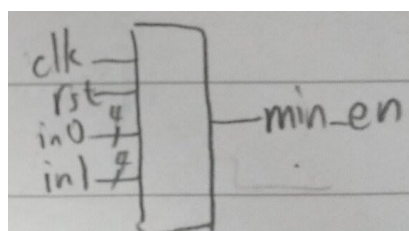
時，要從 24HR 的形式轉換成 A/PM 的形式，且顯示 pm 的燈要亮。



11. min_enable : (從 0 數至 60 的 counter)

此 module 是用來決定處理小時的 counter 和處理分鐘的 counter 能否開始加法。

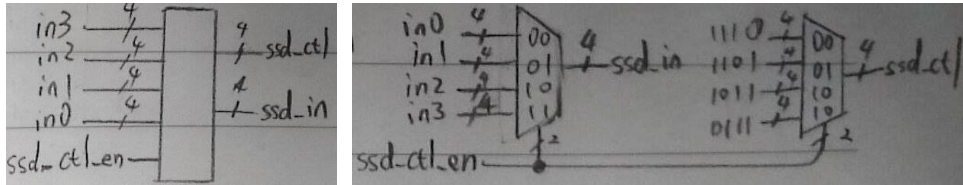
- 其產生的 min_en 接出去後還要再接一個 one_pulse 處理，因為其產生的 min_en 訊號週期為為 60 倍的 clk ，因此要接一個 one_pulse 處理成週期為 clk 的訊號。



12. 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。



13. 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

1. 圖一：當長按 V17，表示 rst=1→數字顯示給定的初始值 23:59。
2. 圖二：當按下 V17，表示轉換 mode→從顯示 24HR 變成顯示 A/PM；且因為 23:59 變成 11:59 時表示此狀態為 PM，因此 U16 亮。
3. 圖三：23:59 經過一分鐘後會回至 00:00。

圖一



圖二



圖三



Discussion

1. 我在這設計的 `hr_ten[3:0]`, `hr_one [3:0]`, `hr_one [3:0]`, `min_one [3:0]`是用來作為初始值的設定，這樣的好處就是我有辦法決定我要從何處開始，也方便用來 debug。
2. `count_up` 判斷 `rst` 要在 `rst` 從 `0→1` 時判斷，因為 `rst` 是一個按鈕，按下的過程是 `0→1`，且要在 `rst==1` 時做出重置，這是跟以往實驗較不同之處。
3. `min_enable` 其產生的 `min_en` 接出去後還要再接一個 `one_pulse` 處理，因為其產生的 `min_en` 訊號週期為為 60 倍的 `clk`，因此要接一個 `one_pulse` 處理成週期為 `clk` 的訊號。

2. For the date functions in clock (no leap year), we have the following functions:
o Day (Jan/March/May/July/Aug/Oct/Dec: 1-31, Feb: 28, Apr/June/Sept/Nov: 30),
o Month (1-12), o Year (00-99).

Implement the following functions:

2.1 Month-Day function display in the 4 14-segment displays. 2.2 Combine the Year and 1.1 to finish a Year-Month-Day timer, and use one DIP switch to select the display of Year (2 Seven-Segment Displays, SSDs) or Month-Day (4 SSDs).)

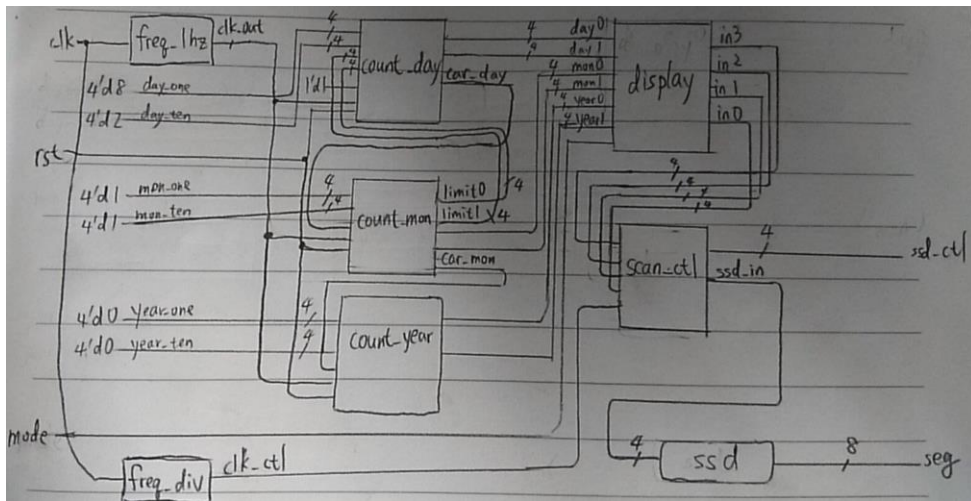
clk	rst	mode
W5	V17	R2

Design Specification

Input : `clk,rst,mode`;

Output : `ssd_ctl[3:0],seg[7:0],state`;

block diagram :



Design Implementation

Logic function :

1. top :

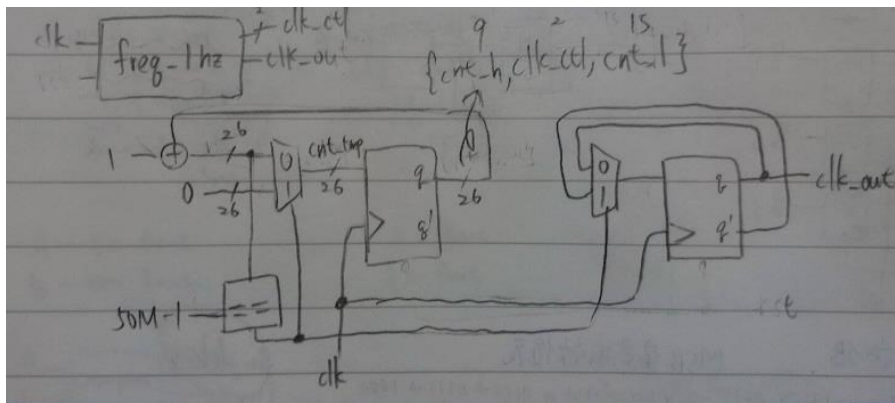
此 top module，專門拿來呼叫其他小 module 的。

並給定初始值，年/月/日分別為 00/11/28。

2. freq_1hz :

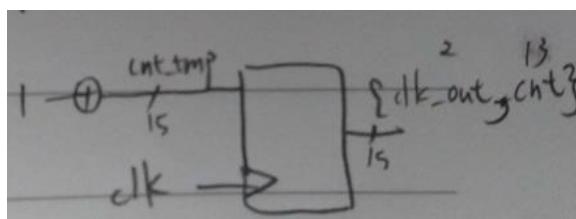
此為除頻的 module，分別輸出 1hz 的 clk_out 控制 count_down 以及控制 scan_ctl 的 clk_ctl。

- 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。



3. freq div :

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



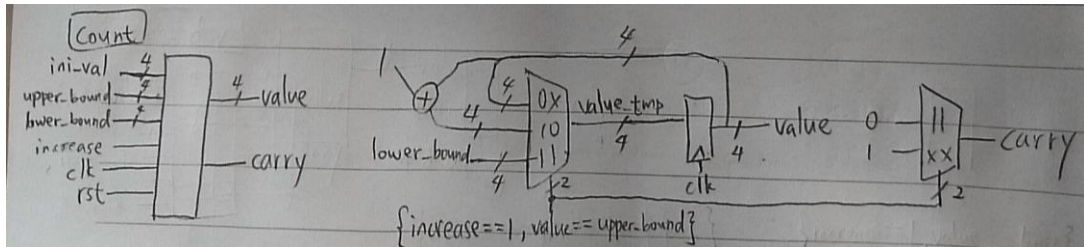
4. count :

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 表示有數近來，因此可以繼續進行加法。



5. count_day :

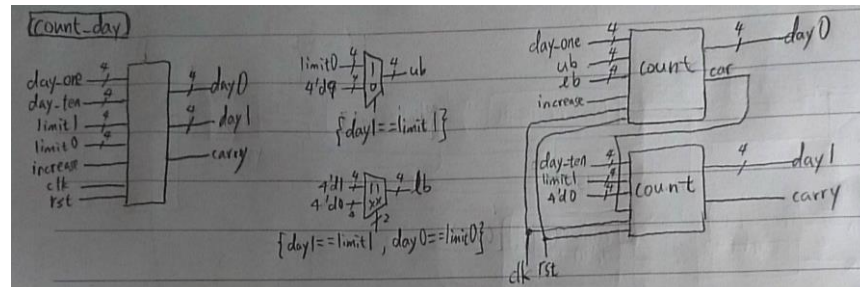
此 module 是用來做日期的顯示。

由於日期是一兩個數，因此內部接兩個 count，分別做個位數及時位數的加法；

而個位數的上界(ub)下界(lb)分別是用(day1==limit1)以及((day1==limit1) && (day0==limit0))來作判別給值。

此接出去的 carry 會作為 count_mon 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。



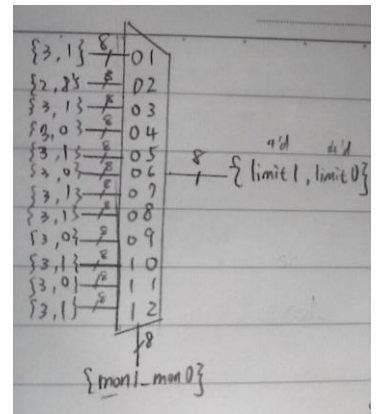
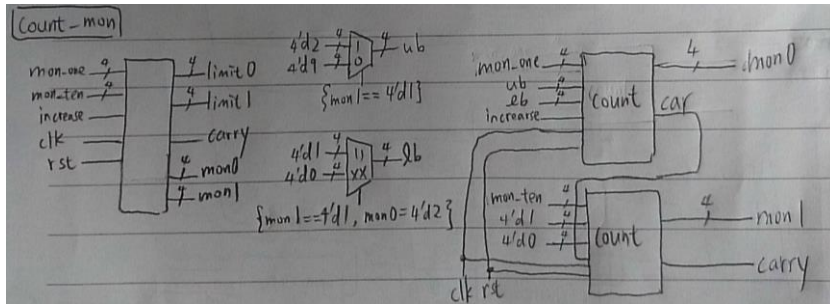
6. count_mon :

此 module 是用來做月份的顯示，還有給 count_day 上界的 limit1(十位數的)和 limit0(個位數的)。

而個位數的上界(ub)下界(lb)分別是用(mon1==4'd1)以及((mon1==4'd1) && (mon0==4'd2))來作判別給值。

此接出去的 carry 會作為 count_year 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。

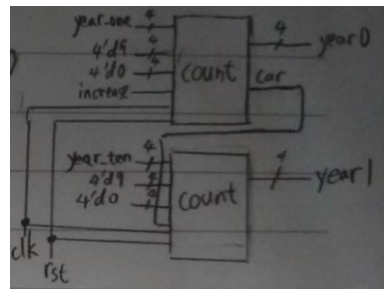
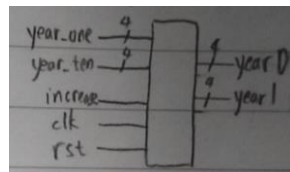


7. count_year :

此 module 是用來做年份的顯示(00~99)。

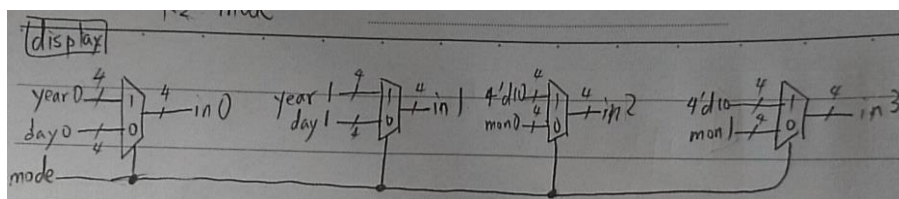
由於年份的十位數進位在 top 內沒有意義，因此就不把處理十位數的 carry 接出來。

- 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。



8. display :

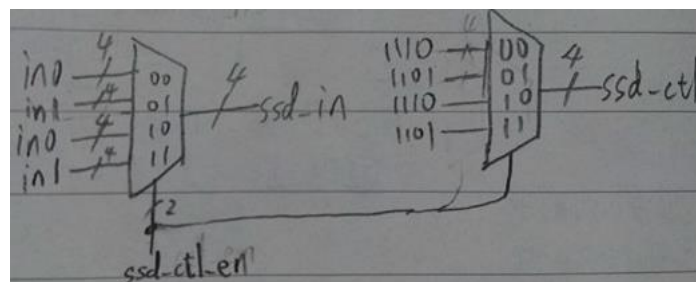
此 module 是用來決定 scan_ctl 的 input :in3, in2, in1, in0。



9. 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。



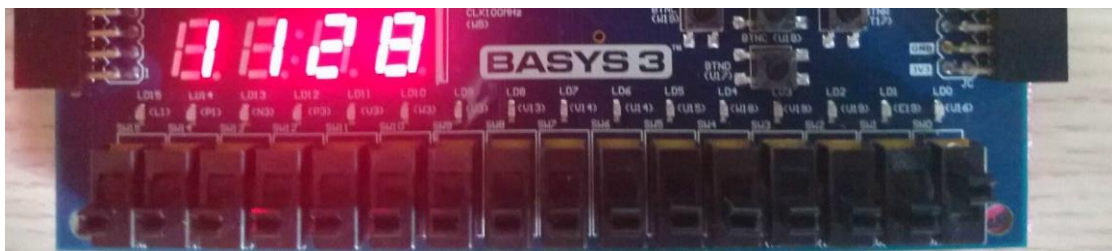
10. ssd :

```
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

1. 圖一、二：當 V17=1，表示 rst=1 → 顯示初始值。
圖一：mode=0 → 顯示 mon/day 的初始值 1128
圖二：mode=1 → 顯示 year 的初始值 00
2. 圖三、四：當 V17=0，表示 rst=0。
圖三目前數到的日期為 12/31，下一刻會變成 01/01，而年份也會加一年(圖四)
3. 圖五、六：當 V17=0，表示 rst=0。
圖五目前數到的日期為 02/28，下一刻會變成 03/01(圖六)

圖一



圖二



圖三



圖四



圖五



圖六



Discussion

1. 在做 count 時特地接 upper_bound 和 lower_bound，這樣子就比較容易 count 有辦法做像 00→12 的循環加法。
2. 在做 count_day, count_mon 和 count_year 時，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。
3. display 在 in3 以及 in2 上有用到 4'd10 是因為我在 ssd 內設 4'd10:segs=8'b11111111，因此當要顯示的數是 10，即其效果就像在七段顯示器上不會顯示任何圖案一樣。

Conclusion :

這次的 lab 讓我們做出一個時鐘，真的是超級猛的，而且也更新了我的 count 變成更有邏輯性的樣子，真的學到很多。