Lab 06

105060016 謝承儒

一.Lab06-1 : 12hr/24hr 的電子時鐘

1. Design Specification :

A. Input : clk_c //輸入的頻率(100MHz)
rst //當=1 時，使數字回到 30 並暫停
pb_in //切換顯示模式(24hr/12hr)的按鈕

B. Output : [7:0] display //七段顯示器的顯示碼
[3:0] display_c //控制 4 個顯示器中哪個會改變
am //1 代表為早上，0 代表為晚上
choose_mode //1 代表為 24 小時制，0 代表為 12 小時制

C. Wire : clk_1 //除頻後的頻率(4000Hz)，近似每秒跑 1hr。
[1:0] ssd_ctl //從除頻器輸出，解碼後為 display_c
clk_100 //除頻後頻率(100Hz)，驅動 FSM 相關功能
pb_de //經過 Debounce 後的 pb_in
one_pulse
//經過 One_pulse 處理後，產生為期 1 週期的
co_min / co_hr / co_24 / co_12
//代表進位，分別是分、小時、24hr 的十位數、12hr 的十位數
[3:0] d0_24、d0_12、d1_24、d1_12
//24hr/12hr 的個位數和十位數
change_am //是否改變早晚，若為 1 就改變 am_en
am_en //代表早晚，輸入到 am
[3:0] D0 //個位數的值
[3:0] D1 //十位數的值
[7:0] display0 //個位數的顯示碼
[7:0] display1 //十位數的顯示碼

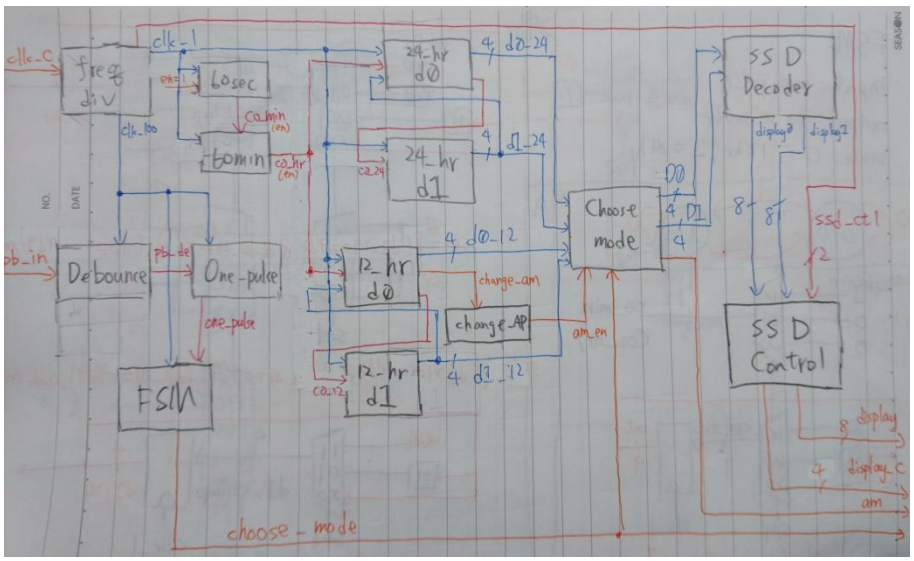

圖 1 Lab06-1 的區塊圖

2. Design Implementation :

A. Logic diagram :

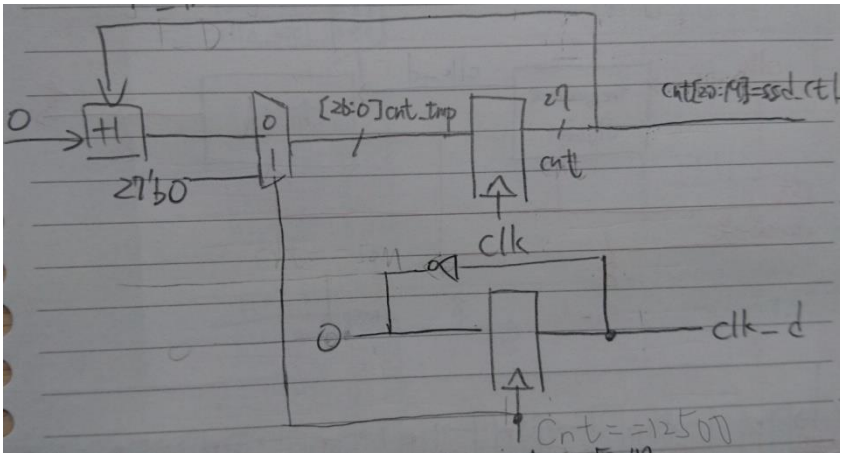

圖 2 Lab06-1 的除頻器(100MHz->4000Hz)

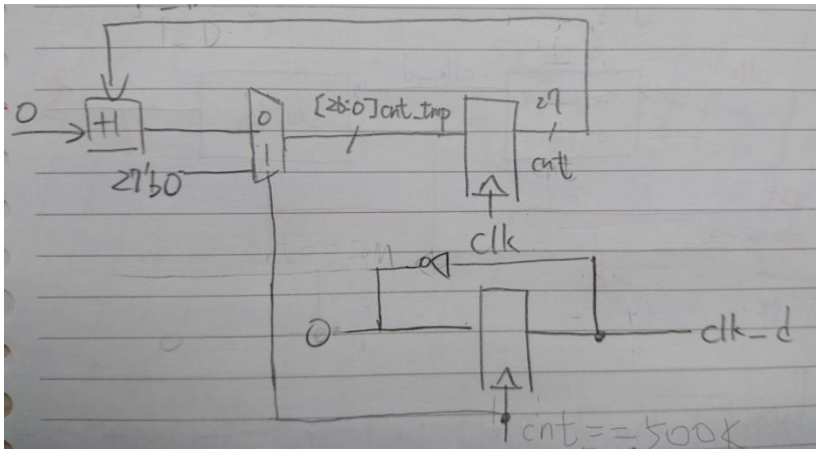

圖 3 Lab06-1 的除頻器(100MHz->100Hz)

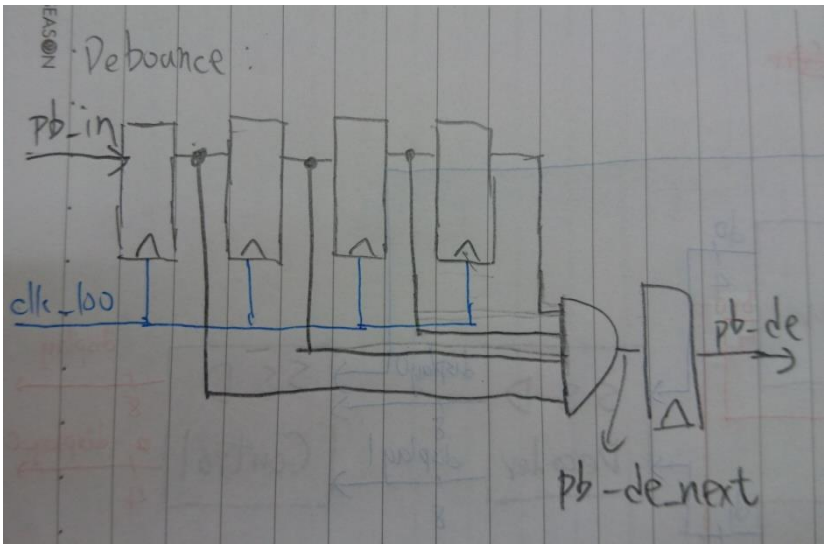

圖 4 Lab06-1 的 Debounce

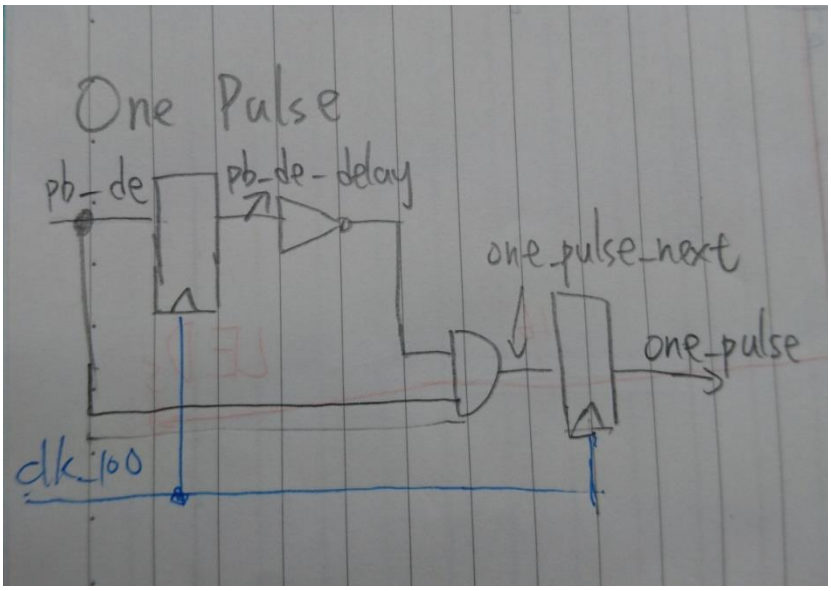

圖 5 Lab06-1 的 One pulse

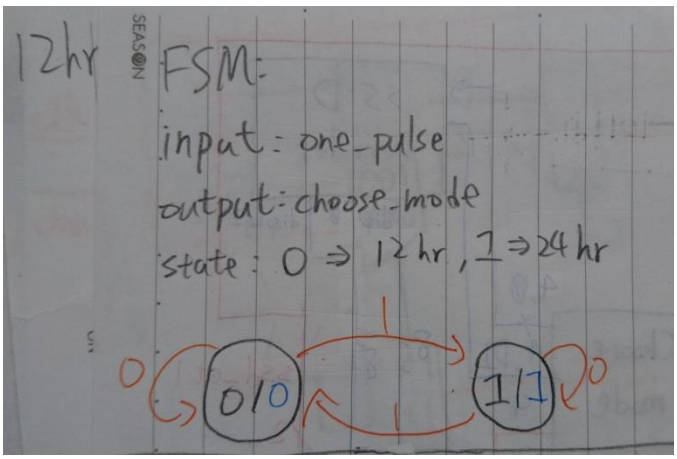

圖 6 Lab06-1 的 FSM

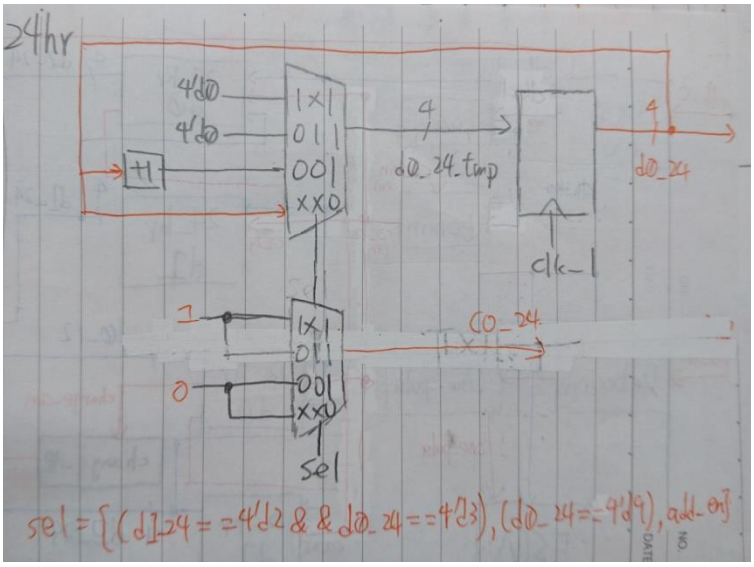

圖 7 Lab06-1 的 24_hr_d0 up counter(個位數)

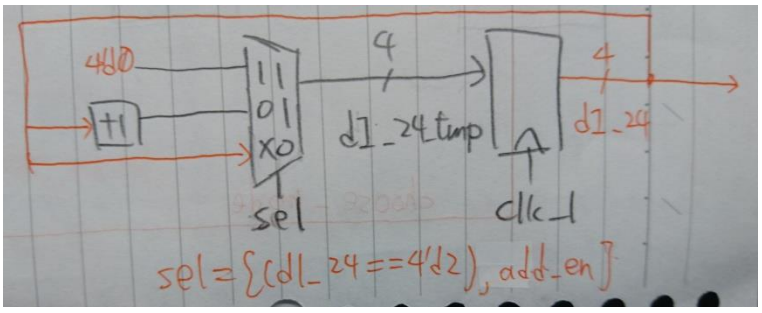

圖 8 Lab06-1 的 24_hr_d1 up counter(十位數)

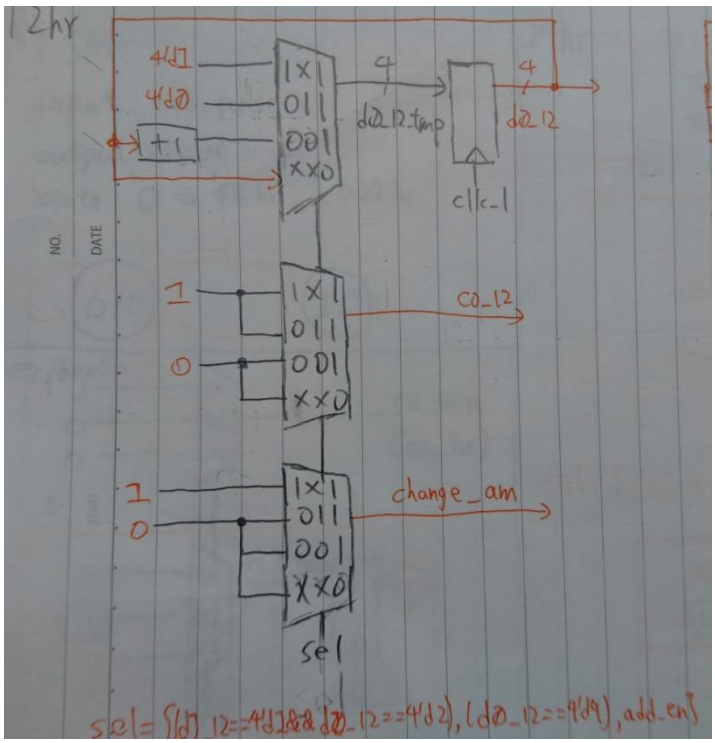

圖 9 Lab06-1 的 12_hr_d0 up counter(個位數)

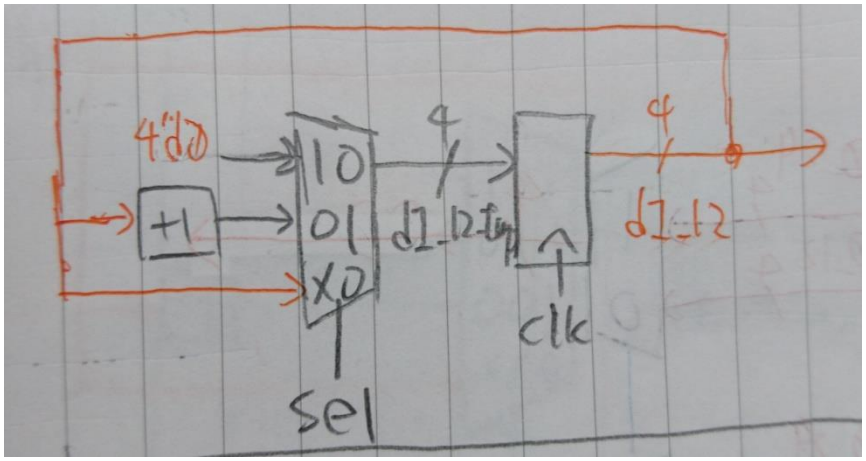

圖 10 Lab06-1 的 12_hr_d1 up counter(十位數)

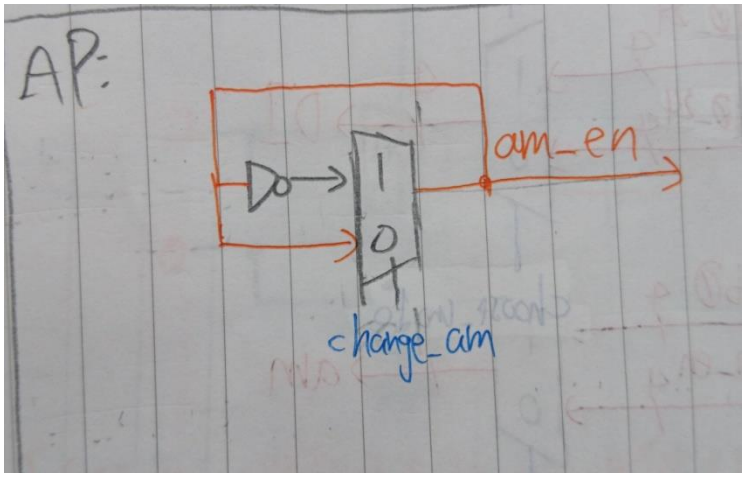

圖 11 Lab06-1 的 change AP(AM/PM)

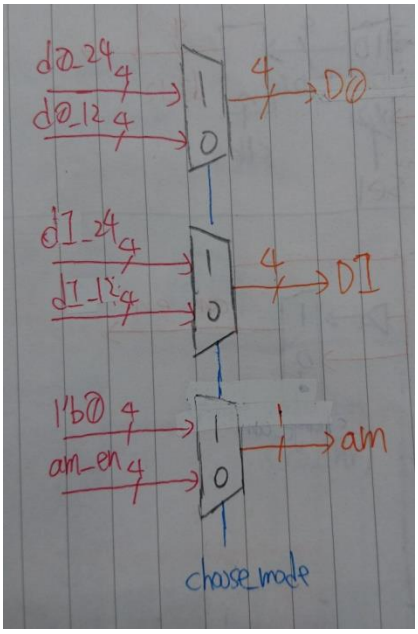

圖 12 Lab06-1 的 choose mode

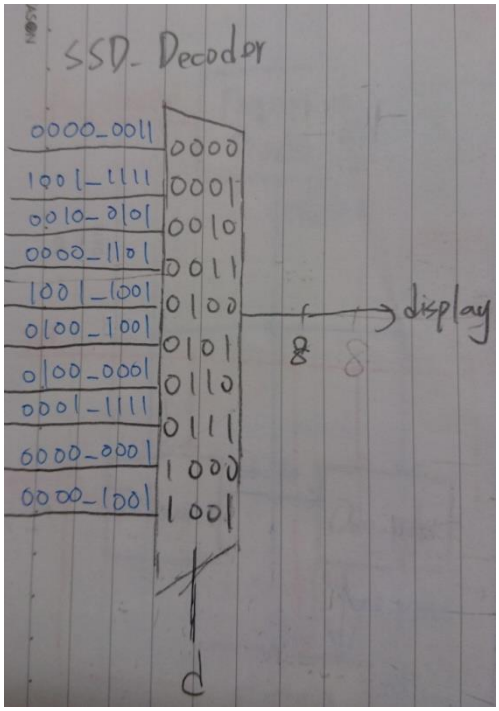

圖 13 Lab06-1 的 SSD Decoder

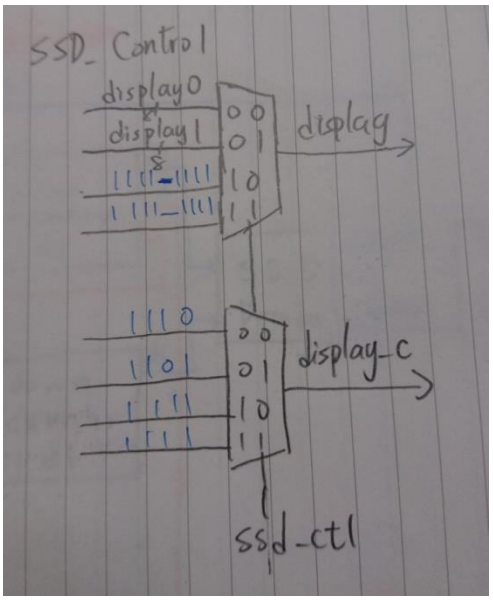

圖 14 Lab06-1 的 SSD Control

B. Pin assignment :
   a. Input :
      1) clk_c = W5
      2) rst = T17
      3) pb_in = W19

b. Output :
   1) display[0] = V7
   2) display[1] = U7
   3) display[2] = V5
   4) display[3] = U5
   5) display[4] = V8
   6) display[5] = U8
   7) display[6] = W6
   8) display[7] = W7
   9) display_c[0] = U2
   10) display_c[1] = U4
   11) display_c[2] = V4
   12) display_c[3] = W4
   13) am = U16
   14) choose_mode = L1

3. Discussion :

A. 思考過程 :

這次要做的是電子時鐘，有 24hr/12hr 兩種模式。按照圖 1 的區塊圖來製作。

a. 除頻器:用來製造 4000Hz、100Hz

跟之前的 Lab 相同，用輸入的 clk(clk_c)來驅動 27-bit 的 Counter(其值為[26:0]cnt)。在 cnt=12500 時，讓 clk_d = ~clk_d 且 cnt 歸零，達到 4000Hz 的效果，如此便有接近每秒跑 1hr 的效果。

而將條件改成 cnt=500K，便可以製造出 100Hz。

除此之外，將其中的兩位輸出(ssd_ctl)，作為待會 SSD_Control 的其中一個 Input，此次選擇 cnt[7:6]。

b. Debounce : 消除不穩定的震盪

當我們按下按鈕時，其內部的彈簧會發生震盪，有時表面上只按 1 次，實際上內部發生的震盪導致按了 3 次。

利用一個 4-bit 的 Shfiter，每次 clk 都將現在按鈕的狀態輸入(pb_in)，若 4 個都為 1，則輸出 pb_de=1。因為是由電腦模擬出來的，不會有不穩定的震盪。

c. **One Pulse**： 製造只會為期 1 週期的 1=>短按

將剛得到的 pb_de 輸入進 Flip Flop(FF)，只有當 clk posedge 時，FF 才會打開讓值通過，因此 FF 後的值(pb_de_delay)會和進來前的(pb_de)有 1clk 的誤差。

將~(pb_de_delay)和 pb_de 做 and，便可以製造出只會為期 1 週期的 1，這可以來當作短按的效果。

d. **FSM**:用來決定是 24hr/12hr 狀態

state 有 0 和 1 兩種狀態。

當 state 為 0 時，就代表為 12hr 狀態且輸出 choose_mode=0；而當 state 為 1 時，就代表為 24hr 狀態且輸出 choose_mode=1。

若按下按鈕後(pb_in=1)，就會換到另一個狀態(0->1 或 1->0)；反之，若沒按下(pb_in =0)，就甚麼事都沒發生(0->0 或 1->1)。

若 rst 的話，state 便回到 0。

e. **60sec/60min up counter**

只需要設計一個 Counter，當它內部的值(cnt)數到 60 就回到 0，和輸出 co_min/co_hr=1 來讓下一個時間進位。

f. **24hr up counter**

分為個位數和十位數兩個 Down Counter。

1) 個位數(d0_24)

將十位數的值輸入進來(d1_24)。

若現在已經數到 23(d1_24=2 且 d0_24=3)，下一個就會是 01。

因此，d0_24_tmp=1、co_24=1。

若只是數到 9(ex:19->20)，則 d0_24_tmp=0、co_24=1。

一般加上去的情況(ex:18->19)，就只需要 d0_24_tmp=d0_24+1 就可以且 co_24=0(不進位)。

前面沒有進位上來的話(co_hr=0)，就維持原樣。

2) 十位數(d1_24)

若現在已經數到 23，下一個就會是 01。因此，d1_24_tmp=0。

一般加上去的情況(ex:18->19)，只需要 d1_24_tmp=d1_24+1。

前面沒有進位上來的話(co_24=0)，就維持原樣。

g. **12hr up counter**

基本上和 24hr 的機制一樣，只需要注意在早晚互換時(11->12)，得額外輸出 change_am=1 來讓後面的 change AP 改變早晚。

h. **change AP :** 改變早晚

當 change_am=1 時，便把 am_en = ~ am_en，也就早晚互換。

若 rst=1，便把 am_en =1，回到早上。

i. **Choose_mode :** 選擇為何種模式
把剛剛在 FSM 得到的 choose_mode 作為選擇依據。
當 choose_mode=0 時，就代表為 12hr，D0=d0_12、D1=d1_12、am=am_en。
當 choose_mode=1 時，就代表為 24hr，D0=d0_24、D1=d1_24、am=0(沒有早晚的差別)。

j. **SSD Decoder :** 將數字轉換成顯示碼
將 Choose_mode 得到的值(D0、D1)輸入，作為多工器的依據(在 code 裡可用 case 得到相同效果)，解碼成 8-bit2 的顯示碼(display0、display1)，將它們輸入到下個 SSD Control。

k. **SSD Control :** 決定哪塊板子的值改變
把在除頻器得到的 2-bit 的 ssd_ctl 輸入，作為選擇 SSD 四個顯示器的依據。
當 ssd_ctl_en=00 時，便將 display0 輸出(display)，並將 display_c 輸出為 1110。如此顯示器上便只有最後一位可以改變，保留其他三個的字母。其他 ssd_ctl 的情況也是一樣，不過因為這是只用到後面 2 位，因此當 ssd_ctl 為 10、11 時，便把 display 和 display_c 所有 bit 都設成 1，讓它們不會亮，避免 bug 出現。

B. 過程中的 Bug :
本來沒有 change AP 這個 block，但在 rst 時，應該回到早上(am=1)，卻因為 am 本來在的 always 無法放入 rst，只好把跟 am 有關的獨立出來成為 1 個 block。

二.Lab06-2：電子日曆

1. Design Specification :

A. Input : clk_c　　//輸入的頻率(100MHz) 。
　　rst　　//當=1 時，回到 30 並暫停。
　　change_freq　　//改變時鐘頻率。
　　change　　//改變顯示模式，此開關為 switch。

B. Output : [7:0] display　　//七段顯示器的顯示碼。
　　[3:0] display_c　　//決定哪個顯示器改變。

C. Wire : clk_d　　//除頻後頻率(300kHz)，近似每秒跑 3 天。
　　clk_month　　//除頻後頻率(3MHz)，近似每秒跑 1 月。
　　[1:0] ssd_ctl　　//從除頻器輸出，解碼後為 display_c。
　　choose_mode　　//決定顯示模式，0 為月/日，1 為年。
　　co_min、co_hr、co_day、co_d1、co_month、co_m1、co_year、co_y1
　　//代表是否進位到下一個時間單位，或個位數進位到十位數
　　[3:0] day_d0/d1、month_m0/m1、year_y0/y1
　　//各個時間單位的個、十位數值
　　[3:0] d0、d1、d2、d3
　　//各個位數的值
　　[7:0] display0、display1、display1、display1
　　//各個位數的顯示碼，由 d0~d3 解碼出來。

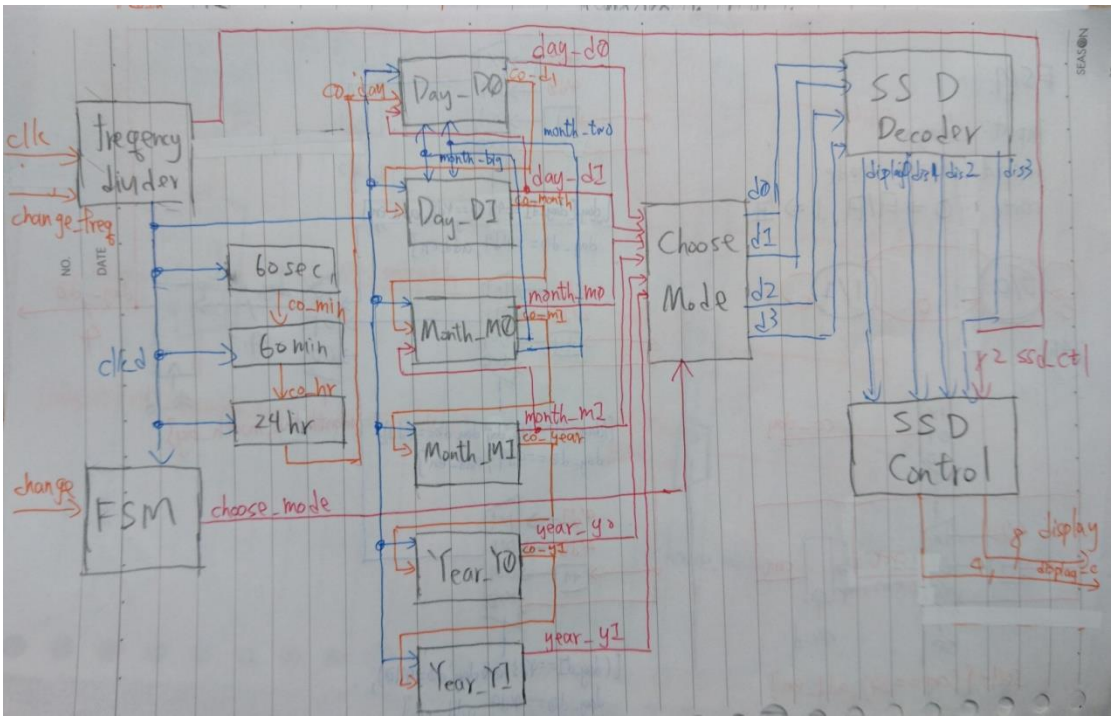

圖 15 Lab06-2 的區塊圖

2. Design Implementation :

A. Logic diagram :

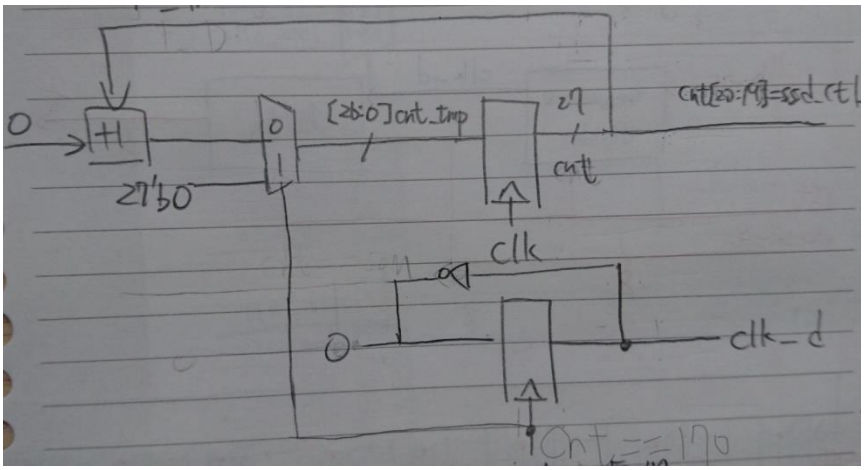

圖 16 Lab06-2 的除頻器(100MHz->300kHz)

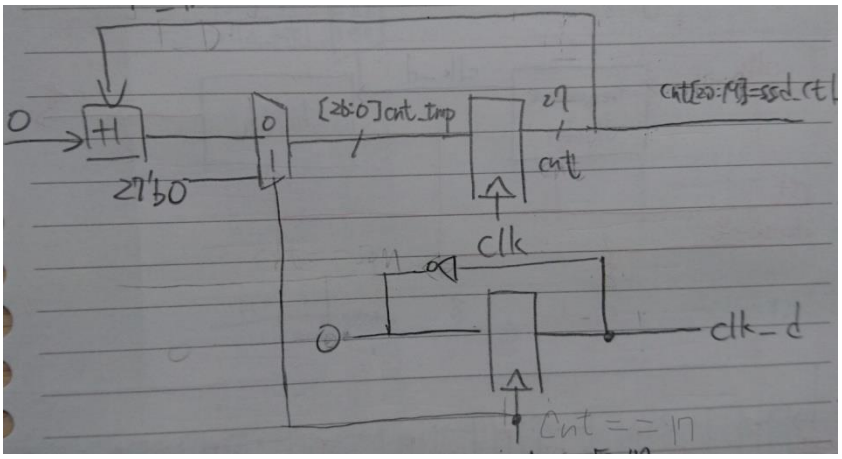

圖 17 Lab06-2 的除頻器(100MHz->3MHz)

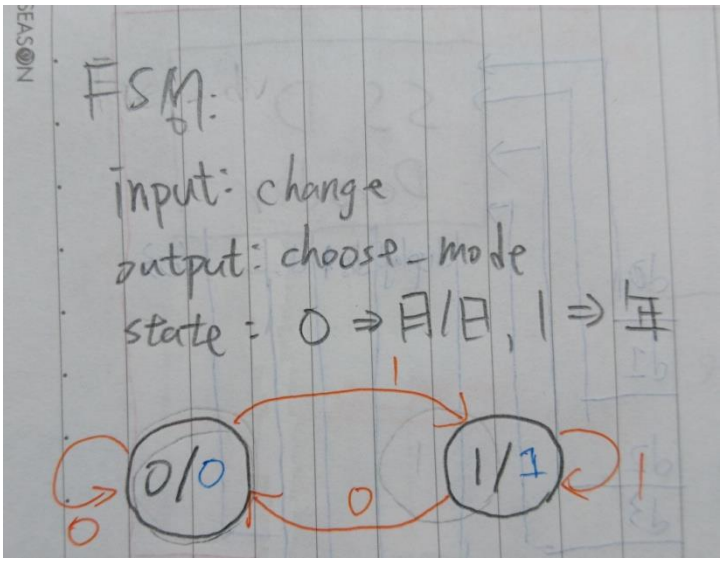

圖 18 Lab06-2 的 FSM

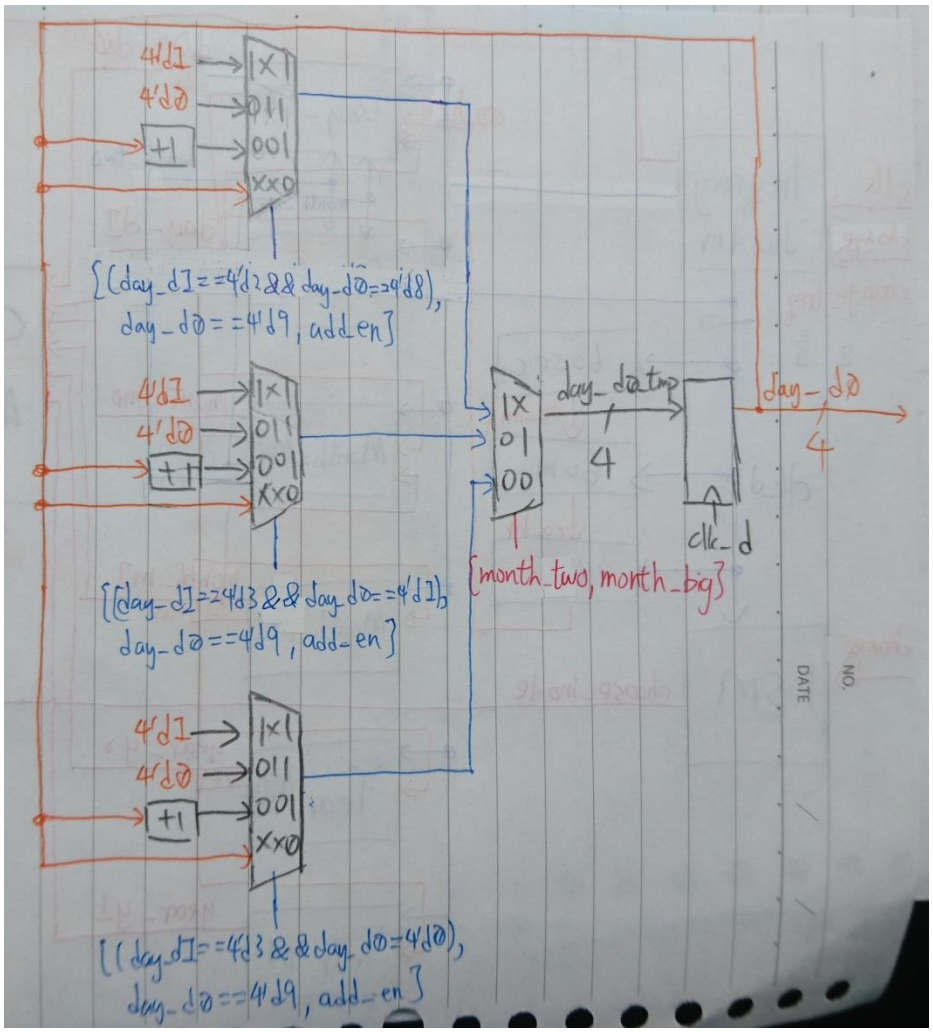

圖 19 Lab06-2 的 day_d0 up counter(個位數)

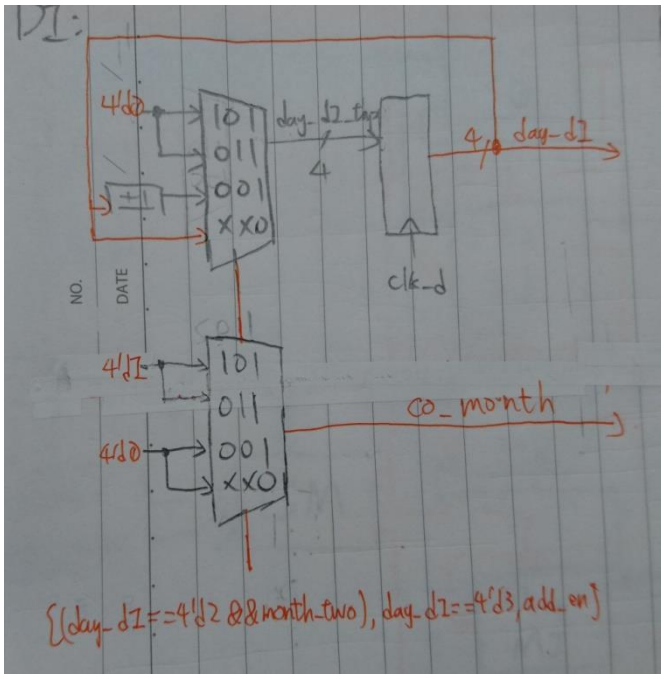

圖 20 Lab06-2 的 day_d1 up counter (十位數)

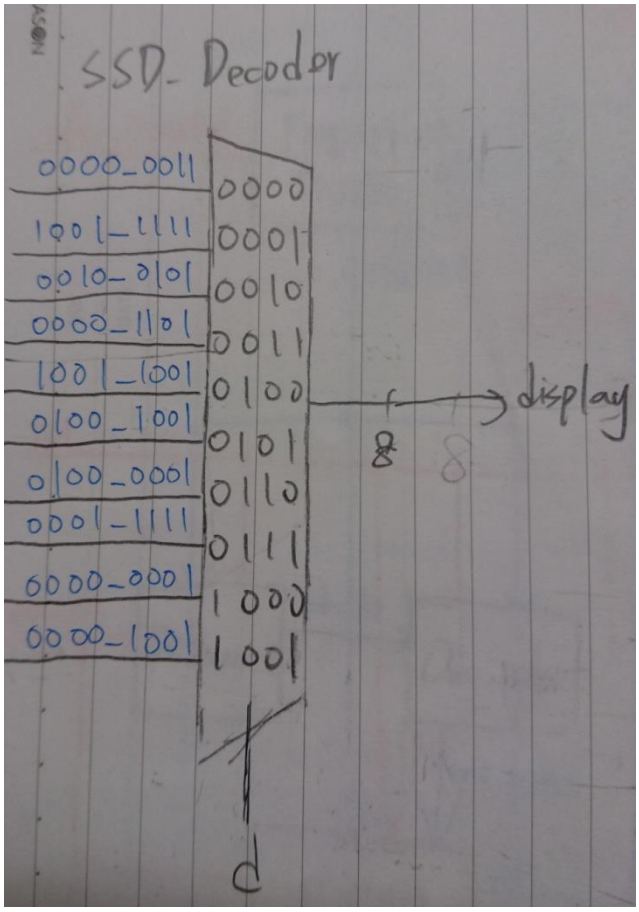

圖 21 Lab06-2 的 SSD Decoder

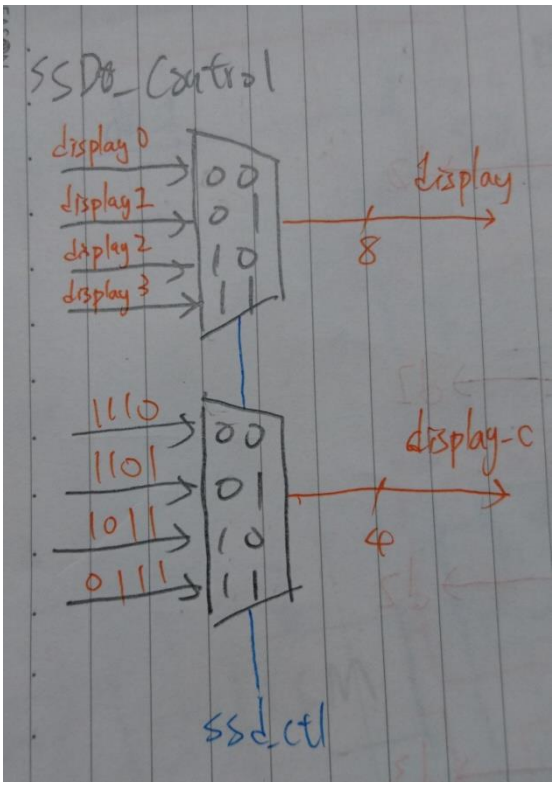

圖 22 Lab06-2 的 SSD Control

B. Pin assignment :

a. Input :

1) clk : W5
2) rst : T17
3) change_freq : U1
4) change : R2

b. Output :

1) display[0] = V7
2) display[1] = U7
3) display[2] = V5
4) display[3] = U5
5) display[4] = V8
6) display[5] = U8
7) display[6] = W6
8) display[7] = W7
9) display_c[0] = U2
10) display_c[1] = U4
11) display_c[2] = V4
12) display_c[3] = W4

三. Discussion :

A. 思考過程 :

這次要做的是電子日曆，基本上想法和 Lab06-1 相同，只需要修改架構即可。

a. 第二個除頻器 : 製造兩種不同頻率

為了能更快看出實驗結果是否正確，我做出 2 種不同的頻率，利用一個 switch(change_freq)來切換。

一個是 300KHz，每秒大約等於 3 天；另一個則是 3MHz，每秒大約等於 1 月。

b. 修改 FSM : 將按鈕(bottom)切換改成開關(switch)切換

因為改成用 switch，就不需要 Debounce 和 One_pulse，不過也需要修改 FSM 裏頭的機制。

state 有 0 和 1 兩種狀態。

當 state 為 0 時，就代表為月/日狀態且輸出 choose_mode=0；而當 state 為 1 時，就代表為年份狀態且輸出 choose_mode=1。

若打開開關後(change=1)，就換到顯示年分 1(0->1 或 1->1)；反之，若沒打開開關(change=1)，就換到顯示月/日(0->0 或 1->0)。

若 rst 的話，state 便回到 0。

c. 時間的 up counter
基本上和 Lab06-1 的方法相同，只是需要判斷這時候是大月(31)還是小月(30)，或者是特例的二月(28)，可以從月份的那部分輸出 month_two、month_big 輸入到日期那邊來幫助判斷。

B. 過程中的 Bug :
起初發現日期不會在月底(28、30、31)時就進位，反而會數到 9 在進位(29、39)。檢查後發現是 day_d1 忘記宣告成 4-bit，當 0011(3)、0010(2)輸入到 day_d1 時只會有 1、0。
如此便不會滿足 day_d1==4'd3、day_d1==4'd2 的條件，才會無法進位。

四.Conclusion :

實驗做到越後面，雖然每次還是做很久，但已經沒有初期那種"不知從何下手"的感覺，看到題目後其實都能有個大概的架構，不會有做白工的感覺。

五.Reference :

1. 老師給的實驗講義
讓我知道大概的架構是時麼。