**Lab 06**

105060016 謝承儒

1. **Lab06-1 : 12hr/24hr的電子時鐘**
2. **Design Specification :**
3. 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

1. 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

1. 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

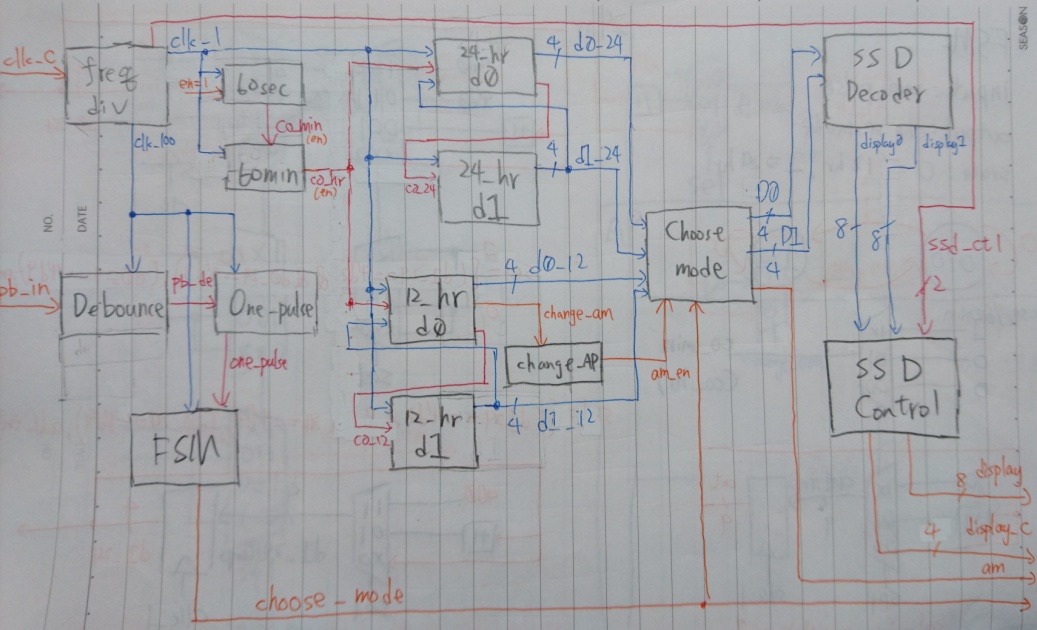


圖1 Lab06-1的區塊圖

1. **Design Implementation :**
2. Logic diagram :

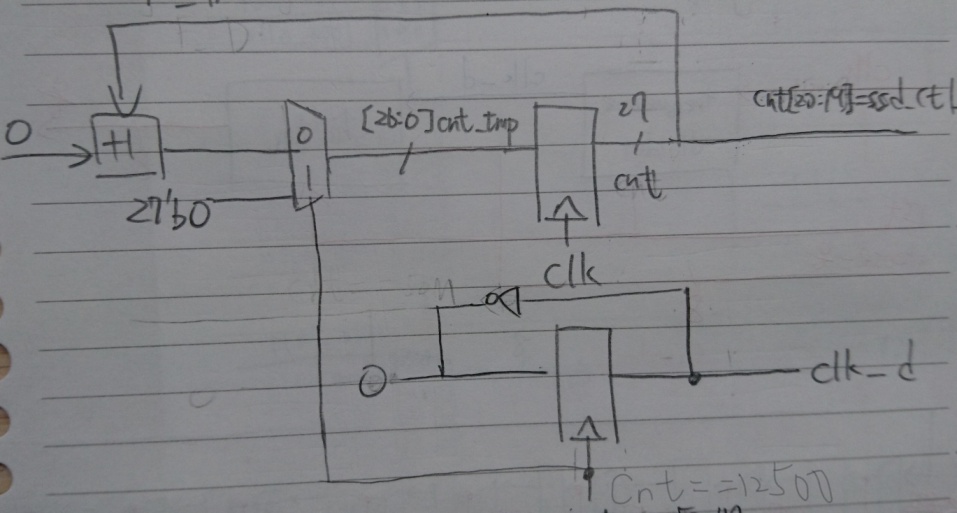


圖2 Lab06-1的除頻器(100MHz->4000Hz)

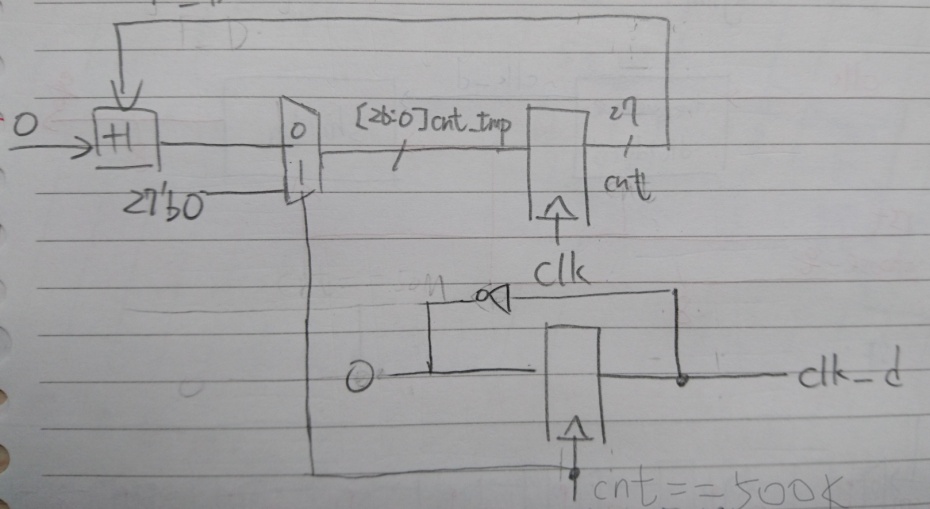


圖3 Lab06-1的除頻器(100MHz->100Hz)

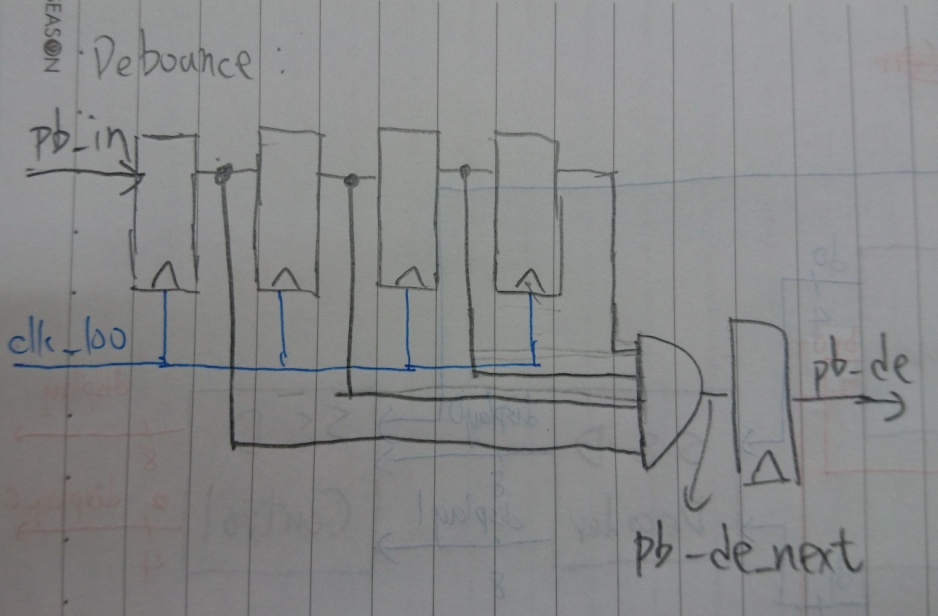


圖4 Lab06-1的Debounce

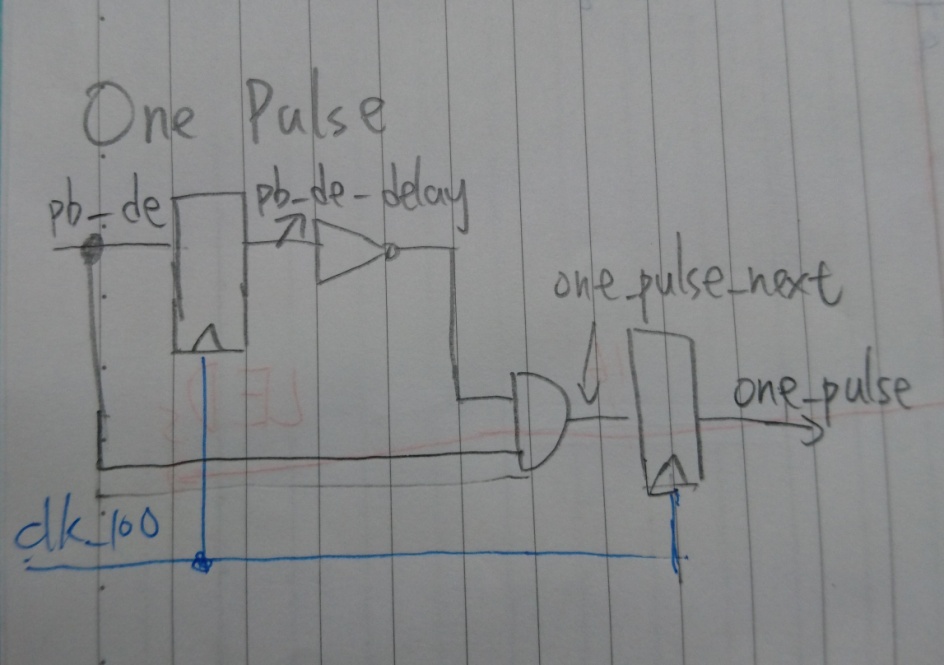


圖5 Lab06-1的One pulse

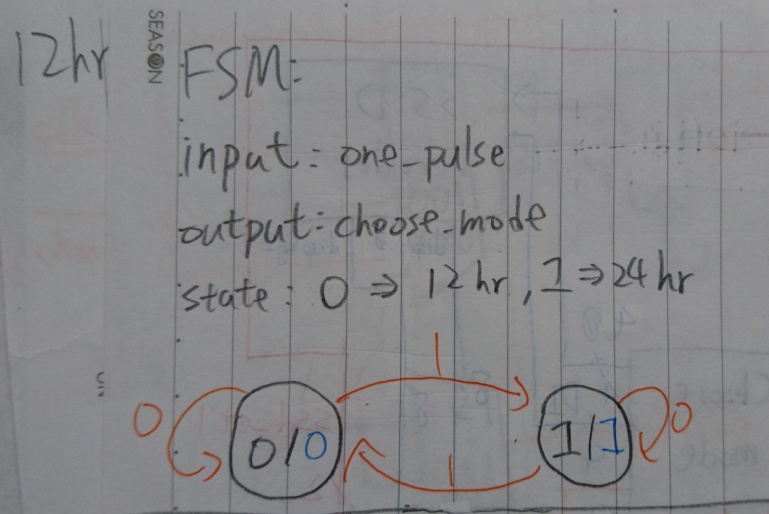


圖6 Lab06-1的FSM

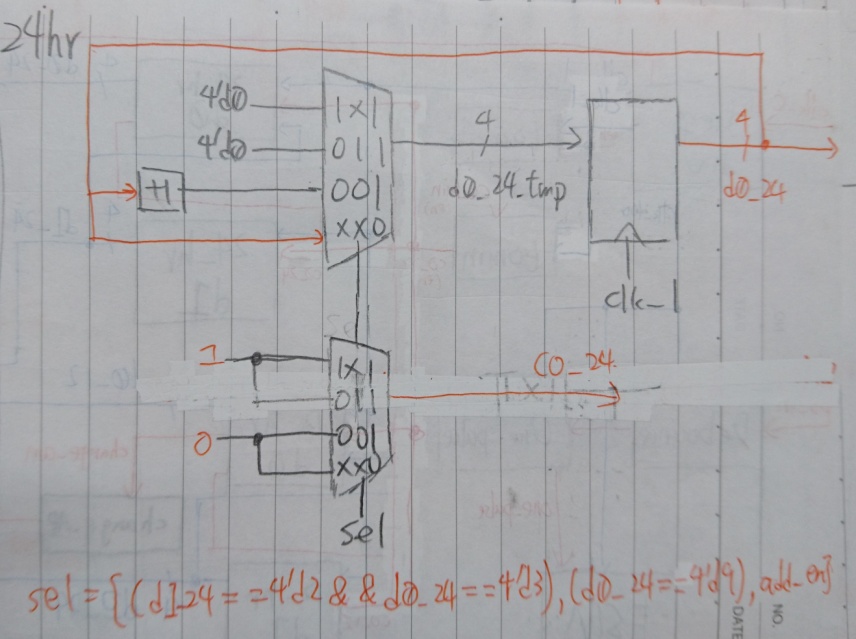


圖7 Lab06-1的24\_hr\_d0 up counter(個位數)

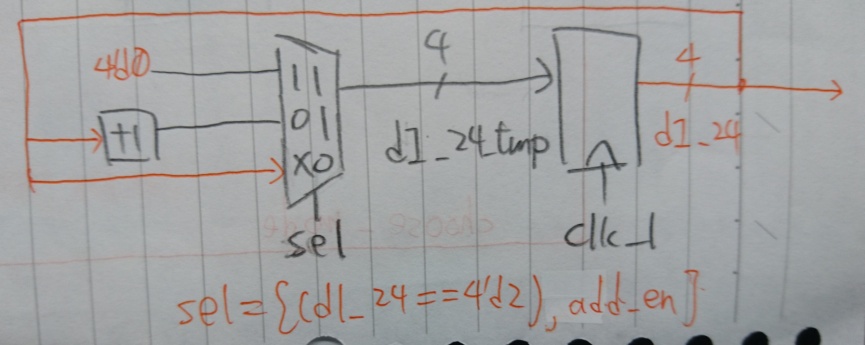


圖8 Lab06-1的24\_hr\_d1 up counter(十位數)

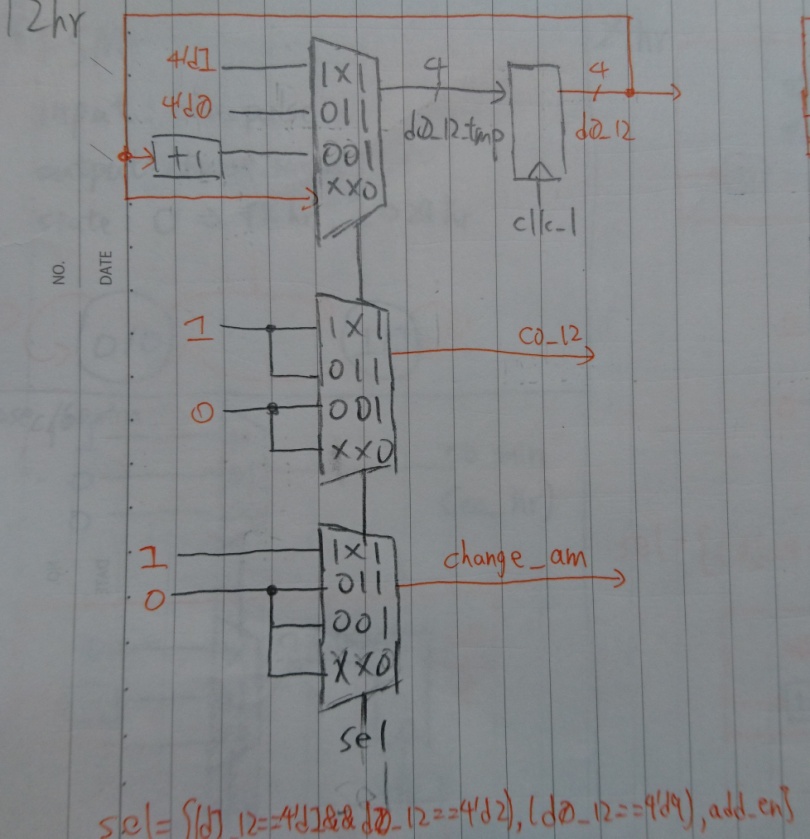


圖9 Lab06-1的12\_hr\_d0 up counter(個位數)

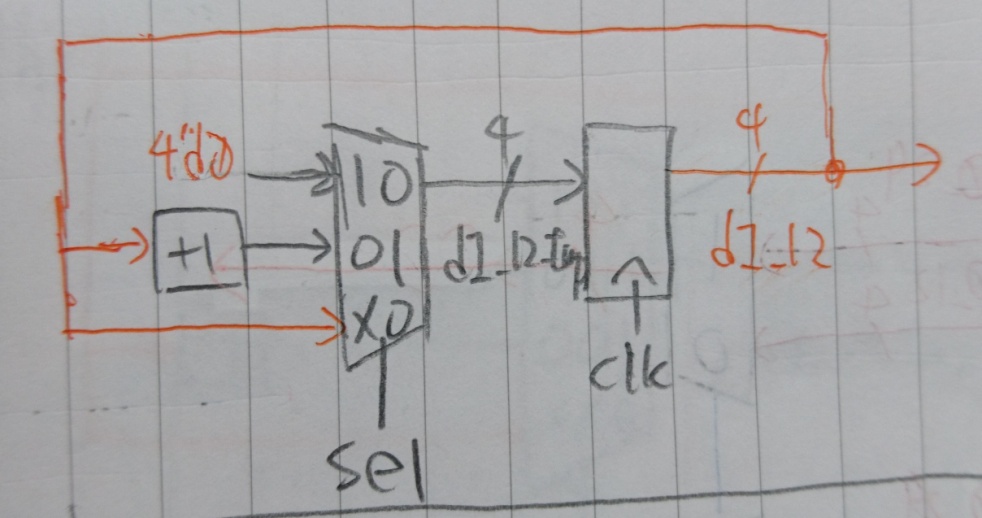


圖10 Lab06-1的12\_hr\_d1 up counter(十位數)

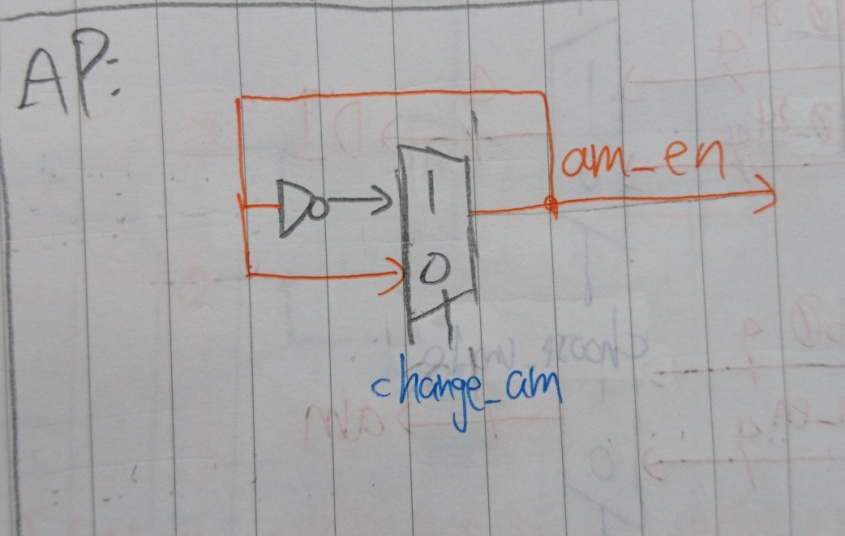


圖11 Lab06-1的change AP(AM/PM)

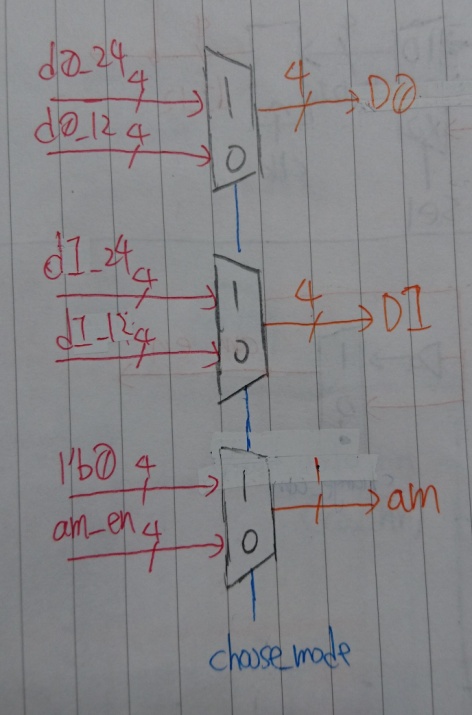


圖12 Lab06-1的choose mode

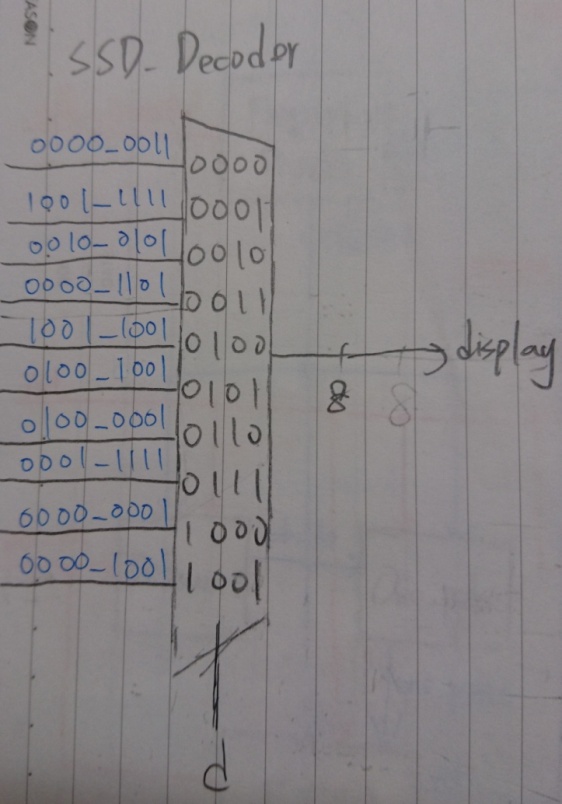


圖13 Lab06-1的SSD Decoder

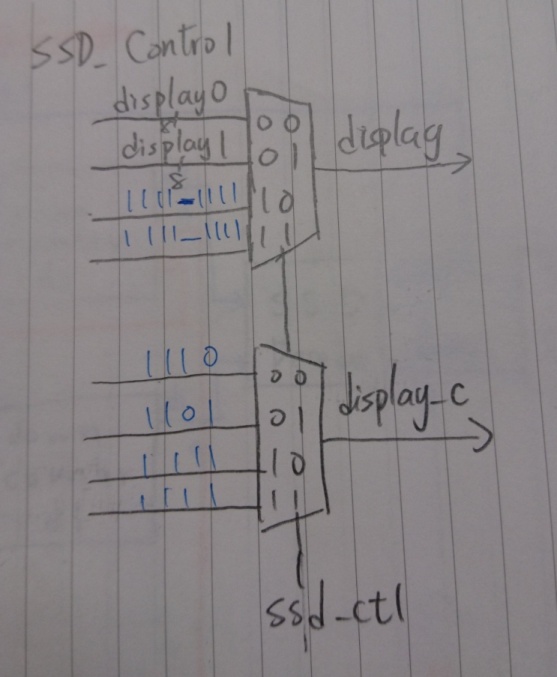


圖14 Lab06-1的SSD Control

1. Pin assignment :
2. Input :
3. clk\_c = W5
4. rst = T17
5. pb\_in = W19
6. Output :
7. display[0] = V7
8. display[1] = U7
9. display[2] = V5
10. display[3] = U5
11. display[4] = V8
12. display[5] = U8
13. display[6] = W6
14. display[7] = W7
15. display\_c[0] = U2
16. display\_c[1] = U4
17. display\_c[2] = V4
18. display\_c[3] = W4
19. am = U16
20. choose\_mode = L1
21. **Discussion :**
22. 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。

1. **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]。

1. **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。

1. **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。

1. **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。

1. **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。

1. **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。

1. 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。

1. **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。

1. **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。

1. **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)。

1. **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。

1. **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。

1. 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。

1. **Lab06-2 : 電子日曆**
   1. **Design Specification :**
      1. 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。

* + 1. 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。

* + 1. 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。

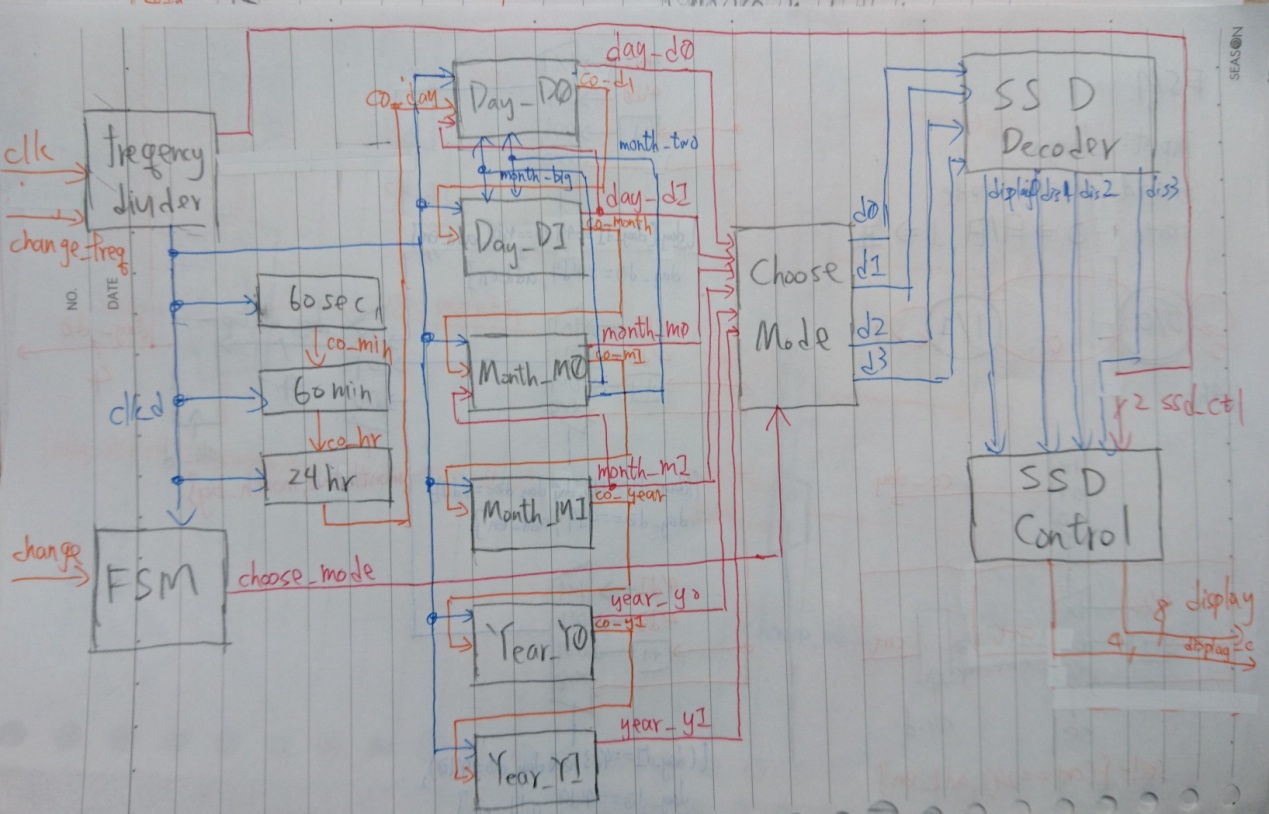


圖15 Lab06-2的區塊圖

* 1. **Design Implementation :**
     1. Logic diagram :

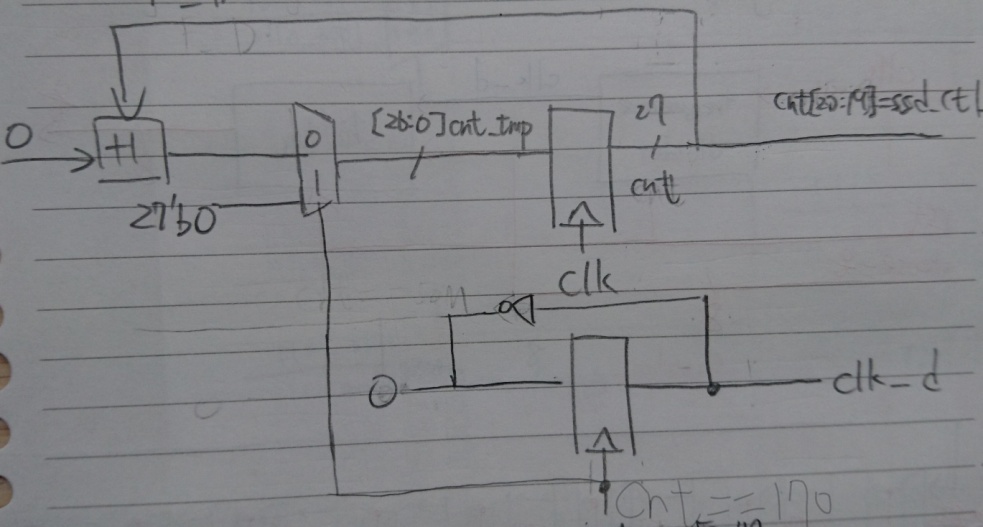


圖16 Lab06-2的除頻器(100MHz->300kHz)

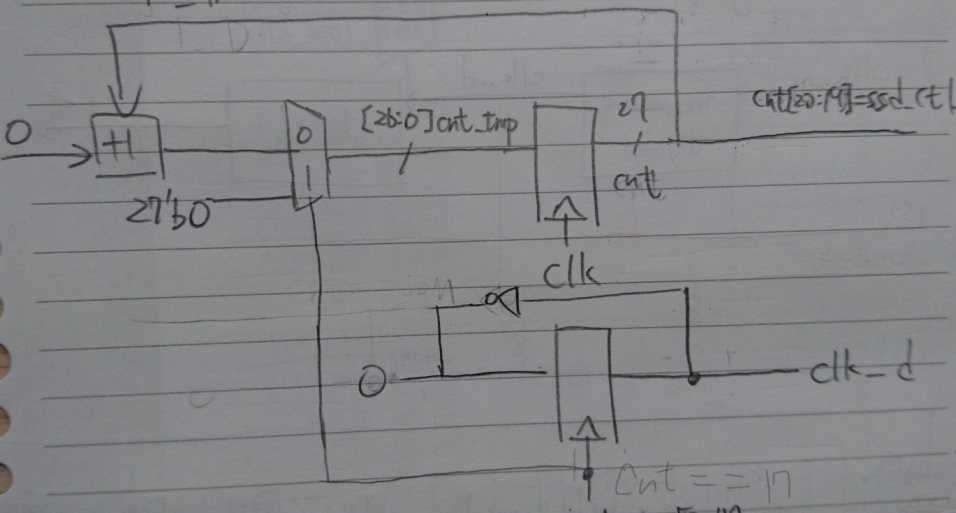


圖17 Lab06-2的除頻器(100MHz->3MHz)

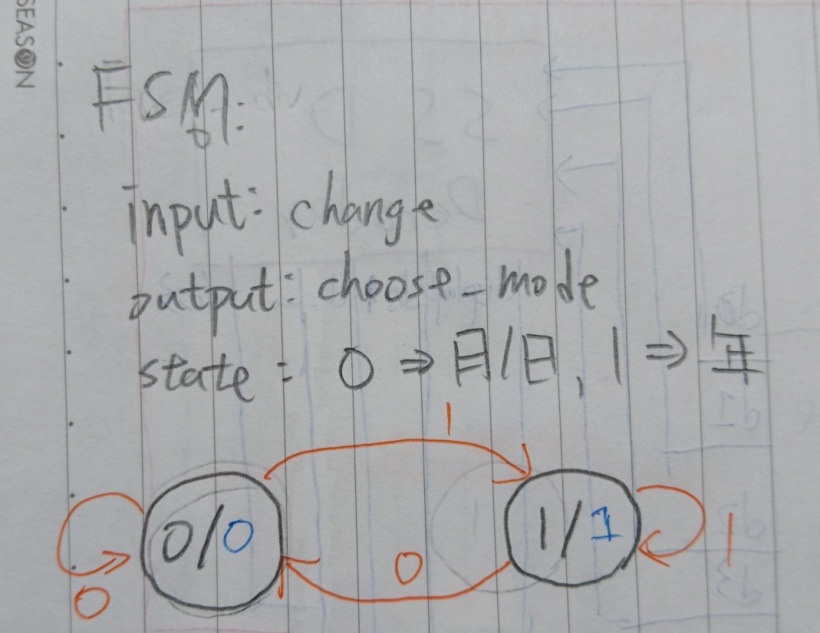


圖18 Lab06-2的FSM

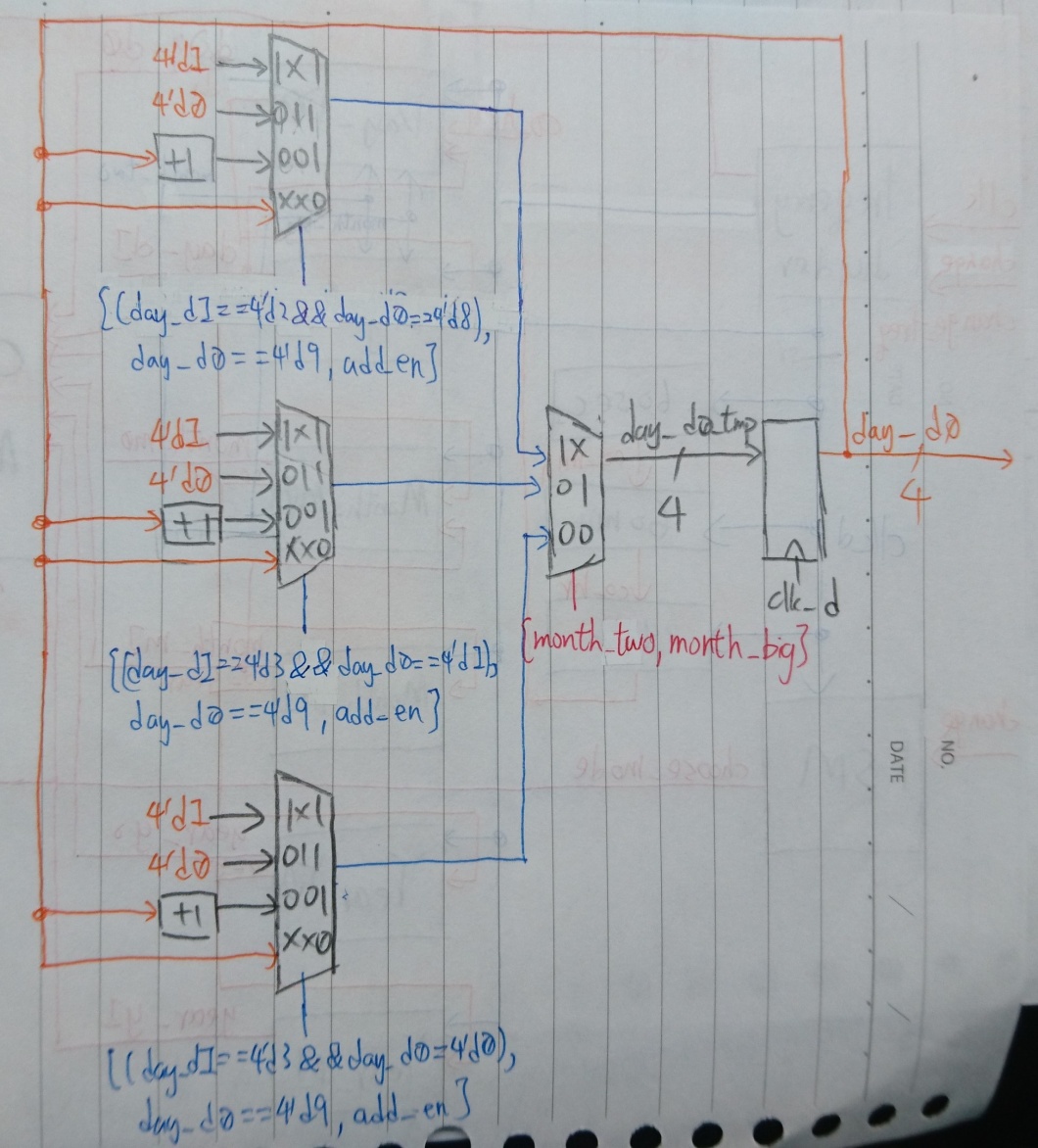


圖19 Lab06-2的day\_d0 up counter(個位數)

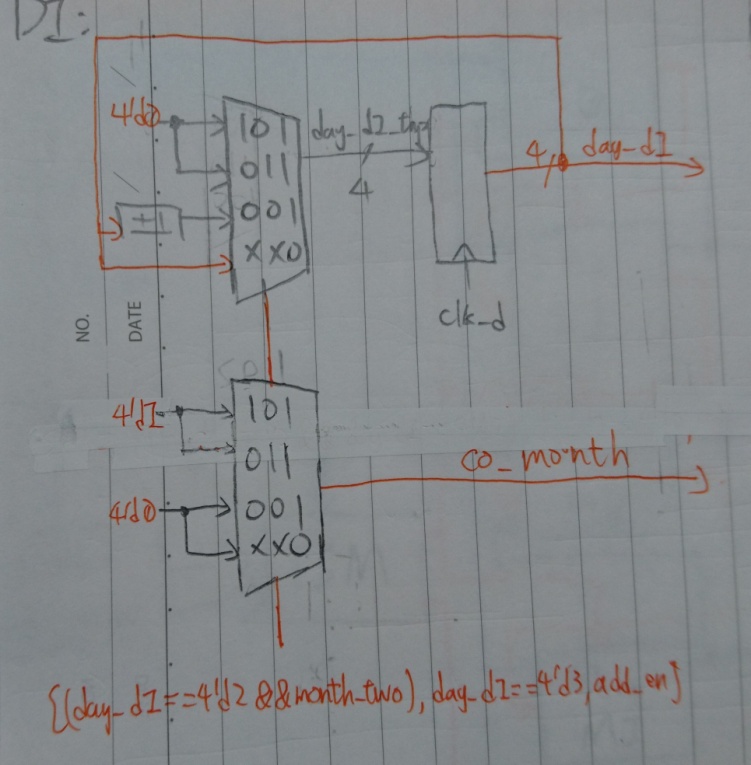


圖20 Lab06-2的day\_d1 up counter (十位數)

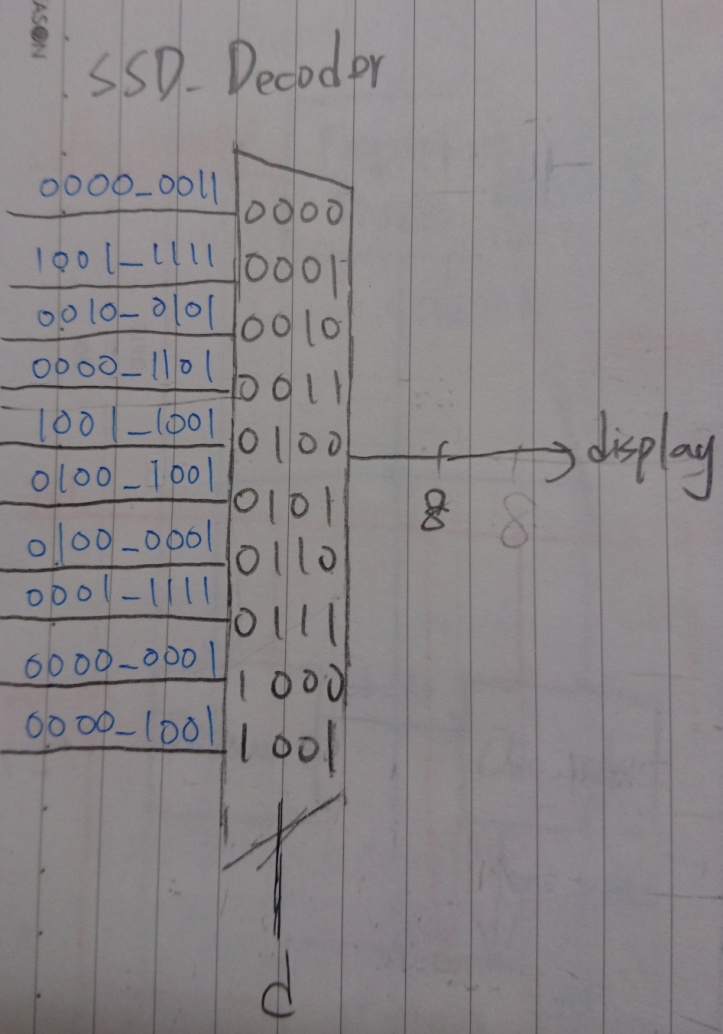


圖21 Lab06-2的SSD Decoder

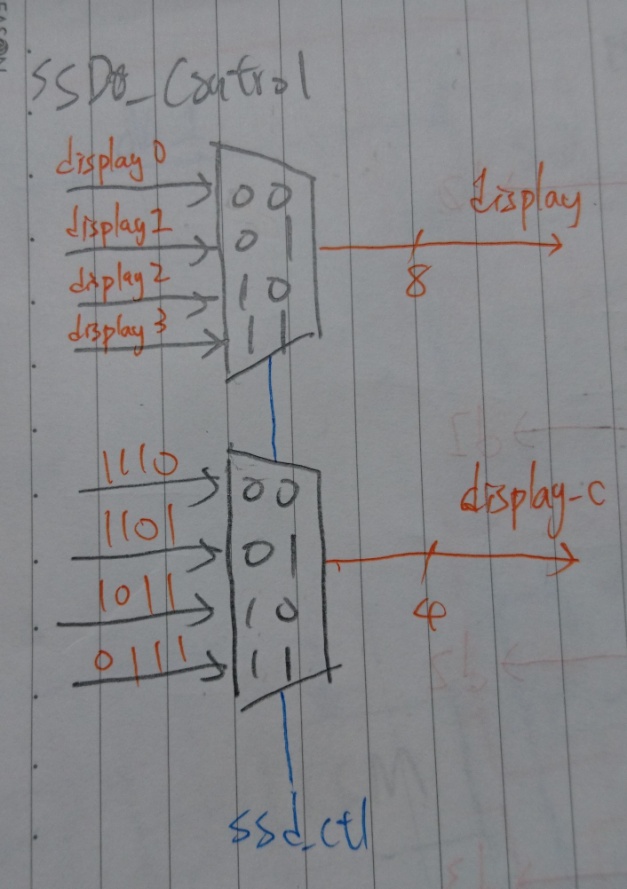


圖22 Lab06-2的SSD Control

* + 1. Pin assignment :
       - 1. Input :

clk : W5

rst : T17

change\_freq : U1

change : R2

* + - * 1. Output :

display[0] = V7

display[1] = U7

display[2] = V5

display[3] = U5

display[4] = V8

display[5] = U8

display[6] = W6

display[7] = W7

display\_c[0] = U2

display\_c[1] = U4

display\_c[2] = V4

display\_c[3] = W4

1. **Discussion :**
   * 1. 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。

* + - * 1. 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。

* + - * 1. 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。

* + - * 1. 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。

* + 1. 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。

1. **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。

1. **Reference :**
   1. 老師給的實驗講義

讓我知道大概的架構是時麼。