106061151 劉安得

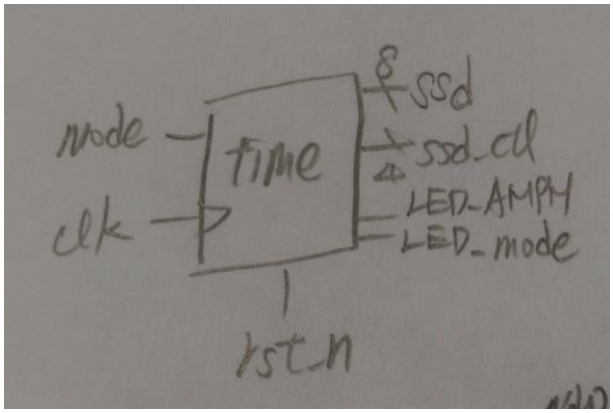
Lab6-1

1. **Specification**

   24hour time display with AM/PM

   Input : mode, clk ,rst_n

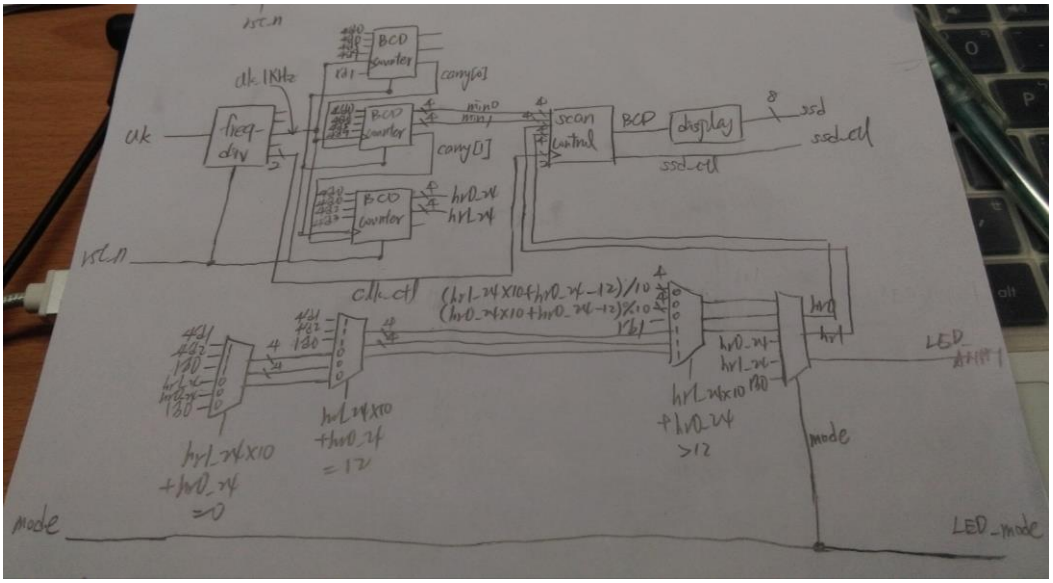
   Output : [7:0] ssd, [3:0] ssd_ctl, LED_AMPM, LED_mode



2. **Implementation**

   因為 lab6 需要加速 clock 這樣比較好驗證，所以 frequency divider 需要新增更快的頻率。而我也修改了之前的 BCD counter 讓它有更多功能，並再寫 2bit BCD counter。至於 display 則是直接使用之前寫過的

   以下為各 module 間 wire 的連結，還有多設兩顆 LED 分別表示是否為 AM/PM 制，以及現在為 AM 還是 PM。在 top module 我還有用 if-else 來換算 AMPM



I/O	Ssd[7]	Ssd[6]	Ssd[5]	Ssd[4]	Ssd[3]	Ssd[2]	Ssd[1]	Ssd[0]
Pin	W7	W6	U8	V8	U5	V5	U7	V7

I/O	Ssd_stl[3]	Ssd_stl[3]	Ssd_stl[3]	Ssd_stl[3]
Pin	W4	V4	U4	U2

I/O	LED_AMPM	LED_mode	Mode	Clk	Rst_n
Pin	U19	E19	V16	W5	V17

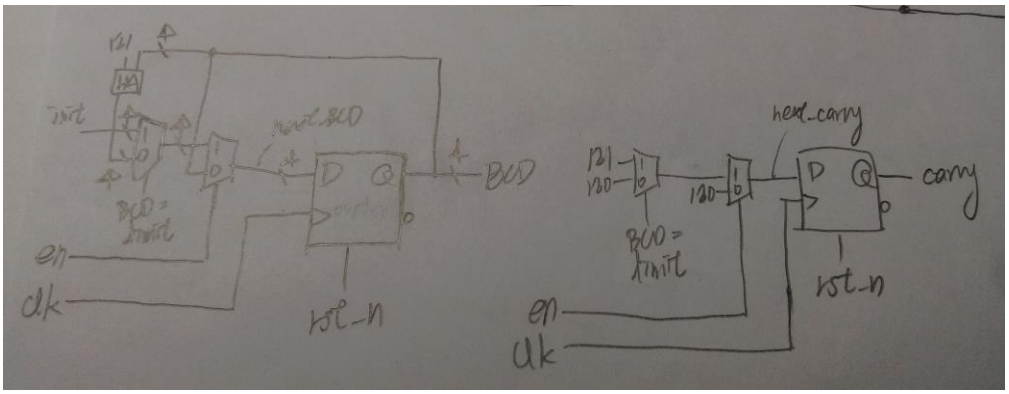
Frequency divider

新增 1K and 100K frequency，因為和之前的寫法很像，所以在此不詳細描述

BCD counter

新增 carry, initial, limit, enable，limit 為 counter 的上限，數到上限後，歸零為 initial，並且 carry 為 1。

If(en = 0),
  Next_BCD = BCD
  Next_carry = 0
Else if(BCD = limit),
  Next_BCD = init
  Next_carry = 1
Else,
  Next_BCD = BCD +1
  Next_carry = 0



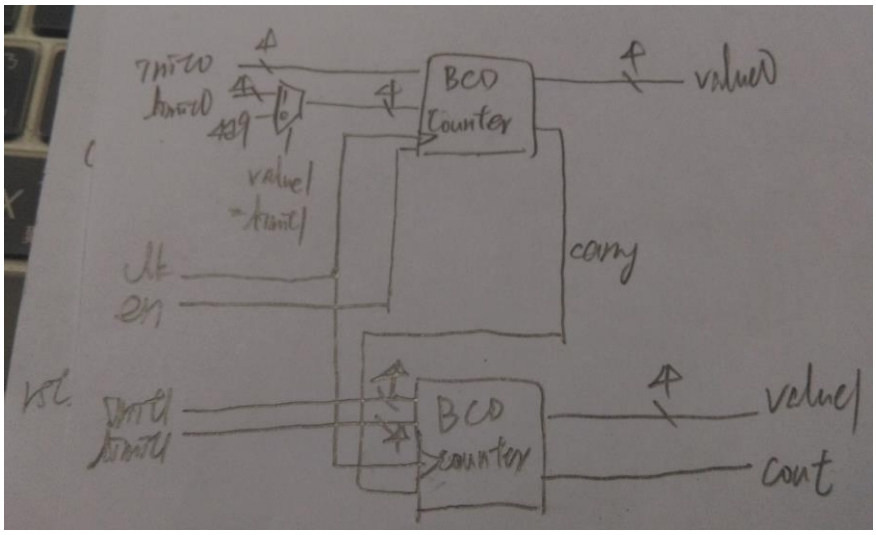

```
always @*
begin
    if(en == 1'b0)
    begin
```

```
            next_BCD = BCD;
            next_carry = 1'b0;
        end
        else if(BCD == limit)
        begin
            next_BCD = init;
            next_carry = 1'b1;
        end
        else
        begin
            next_BCD = BCD + 1'b1;
            next_carry = 1'b0;
        end
    end
```

2bit BCD counter

將兩個 BCD counter 合併在一起，方便計算秒分鐘小時這些兩位數的單位值得注意的是個位數的 limit 平時為 9 否則個位數會數到一半就進為，當 value1=limit1，十位數到達上限時，再將個位數的 limit 設為 limit1

以下為兩個 BCD counter 間 wire 的連結



3. Simulation

Clk	BCD[3:0]	carry
0	X	0
1	0(0000)	0
0	0(0000)	0
1	1(0001)	0
0	1(0001)	0
1	2(0010)	0
0	2(0010)	0
1	3(0011)	
0	3(0011)	0
1	4(0100)	0
0	4(0100)	0
1	5(0101)	0
0	5(0101)	0
1	6(0110)	0
0	6(0110)	0
1	7(0111)	0
0	7(0111)	0
1	8(1000)	0
0	8(1000)	0
1	9(1001)	0
0	9(1001)	0
1	0(0000)	1
0	0(0000)	1

4. Discussion

雖然題目只要小時，但我連分鐘秒都一起寫，然後我是 clock 是從秒開始數，慢慢進位。

這次 lab 我在處理 BCD 數據上遇到很多問題，因為我是寫 BCD counter，它可以設計 initial and limit value，在 implement 有上限的數字的時候會比較好用。我不是用 binary counter，然後再寫一個 BCD converter。

我在 top Module 使用不少 if-else 來轉換 AMPM，我是判斷現在小時數到多少，來決定轉成 AMPM 為多少。值得一提的是，24 小時的 0 點，為 AM12 點，所以不能直接全部-12。

一開始我沒有改 frequency divider，想說盡量不要變動，所以它只有 1hz, 100hz，但後來實在覺得太慢，就寫了比較快的頻率

然後我本來是寫 6bit BCD counter，小時分鐘秒寫在一起，但後來發現 limit 會出問題，舉小時為例，limit 為 24，那當你數到 04 的時候，下一個應該是 05，但結果會是 10。我最後是改寫成 3 個 2bit BCD counter，然後個位數 limit 在十位數到達 limit 前設為 9

5. Conclusion

做完這次 lab，讓我學會如何 implement 時鐘，如何使用 BCD counter，如何使用 case 以及如何處理 BCD 數據

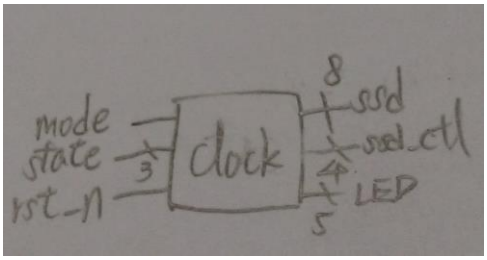
Lab6-2

1. **Specification**

   Calendar(year, month, day) + 24hour time display with AM/PM

   Input : mode, [2:0] state, clk ,rst_n

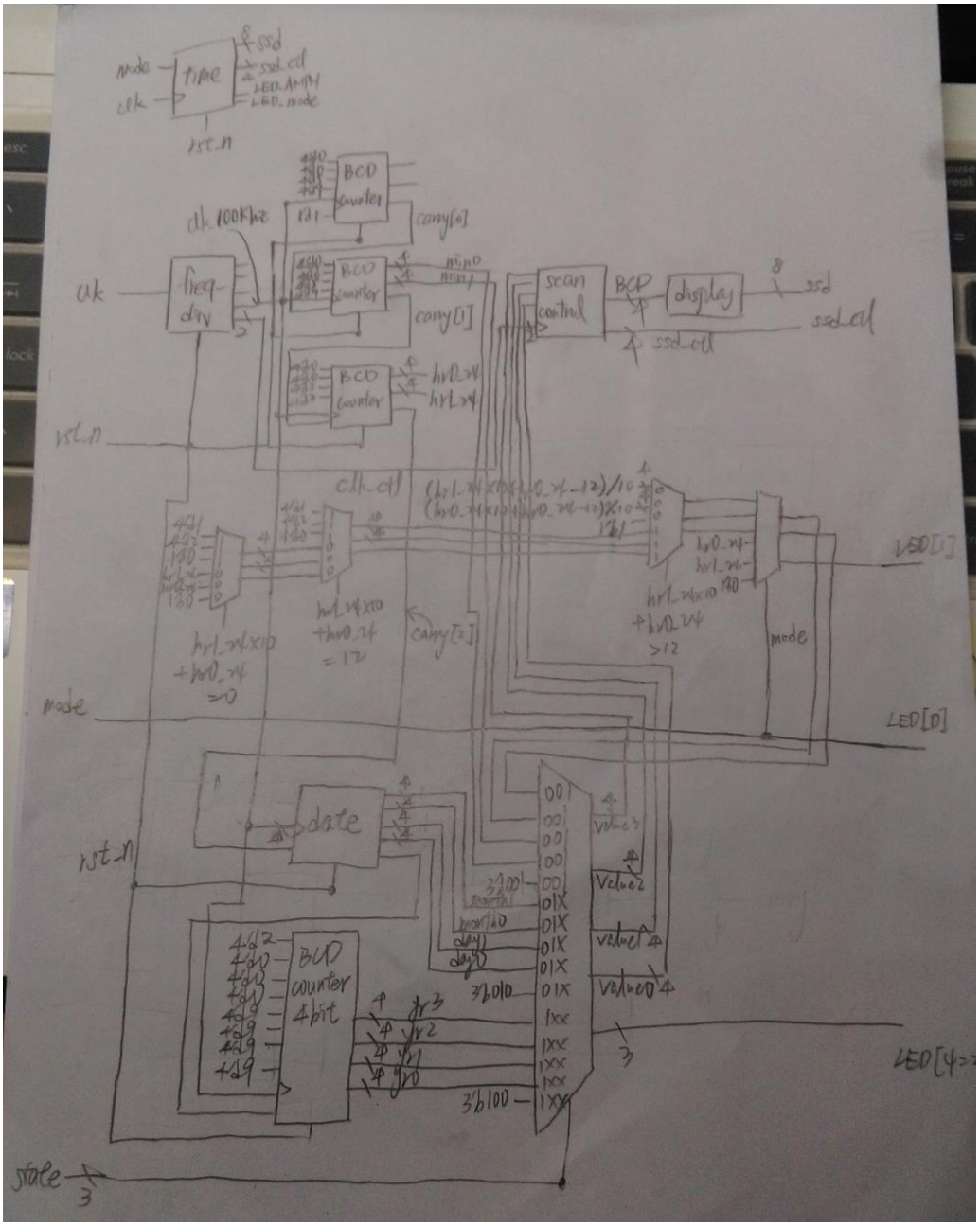
   Output : [7:0] ssd, [3:0] ssd_ctl, [4:0]LED



2. **Implementation**

   比 lab6-1 多了年，月份跟日期，因為月份跟日期有獨特的進位方式，所以我特別寫個 time module，年則是用一般的 4bit BCD counter

   以下為各 module 間 wire 的連結，還有多設兩顆 LED 分別表示是否為 AM/PM 制，以及現在為 AM 還是 PM，跟三顆 LED 分別表示現在為年/月/日。在 top module 我還有用 if-else 來換算 AMPM，並用 state 決定七段顯示器要顯示年/月/日



I/O	Ssd[7]	Ssd[6]	Ssd[5]	Ssd[4]	Ssd[3]	Ssd[2]	Ssd[1]	Ssd[0]
Pin	W7	W6	U8	V8	U5	V5	U7	V7

I/O	Ssd_stl[3]	Ssd_stl[3]	Ssd_stl[3]	Ssd_stl[3]
Pin	W4	V4	U4	U2

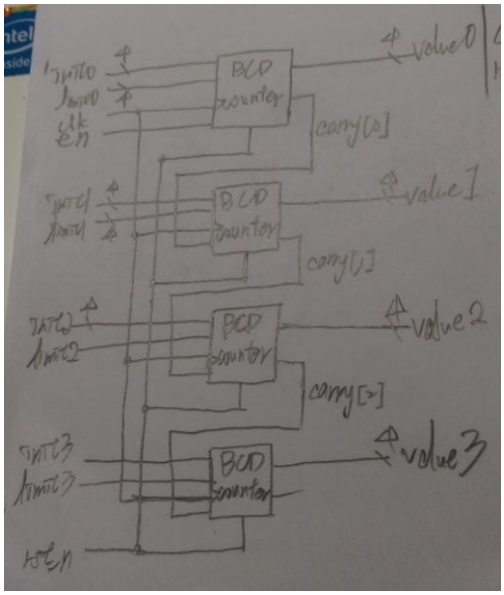
I/O	LED[4]	LED[3]	LED[2]	LED[1]	LED[0]
Pin	U15	W18	V19	U19	E19

I/O	State[2]	State[1]	State[0]	Mode	Clk	Rst_n
Pin	V15	W15	W17	V16	W5	V17

4bit BCD counter

將四個 BCD counter 合併在一起，來計算年

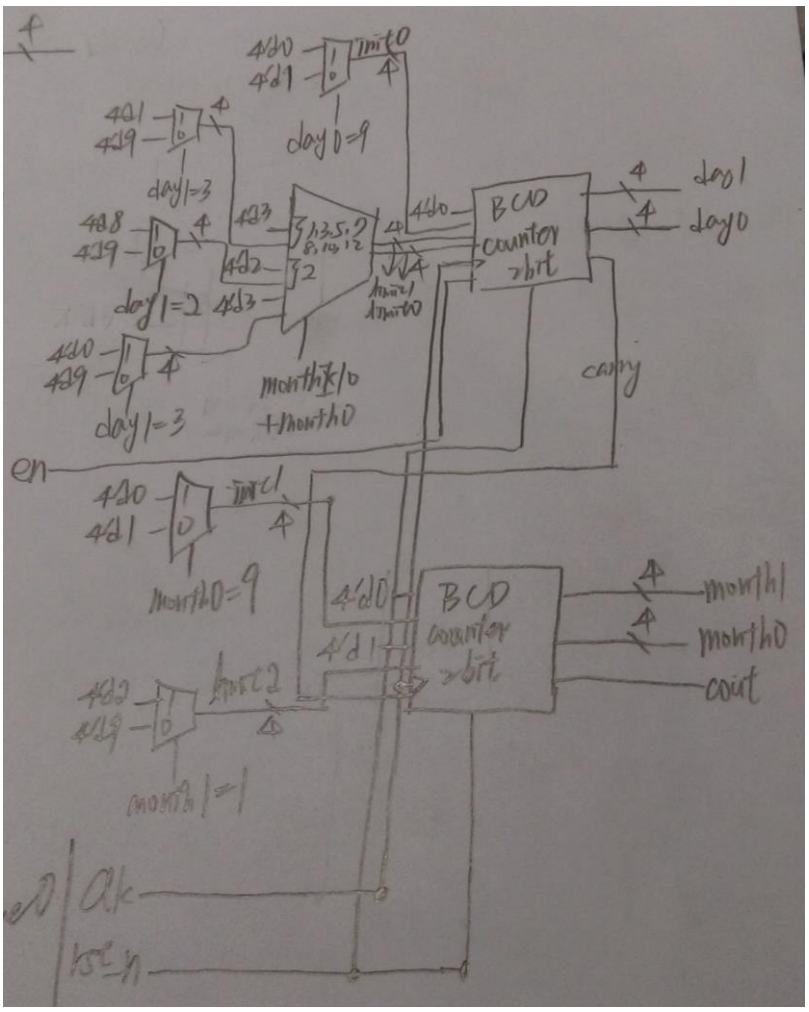
以下為四個 BCD counter 間 wire 的連結



date

由兩個 2bit BCD counter 組成，並用 case 判斷不同月份下日期的 limit

除此之外因為每個月的第一天都是 1 號，而非 0 號，然後當個位數進位時，必須歸零。因此不能直接設 initial value 為 1，這樣個位數進位會有 bug；但也不能設 0，這樣每個月會從 0 號開始數。我後來想到的辦法是，如果個位數等於 9 時，initial value 為 0，這樣代表平時的進位；如果不等於 9，initial value 為 1，這樣代表算到一半因為達到該月日期上限，只好進位，剛好平年一個月只會有 28, 30, 31 天，沒有 9 結尾的，因此不會有 bug。至於閏年怎麼辦會在 lab6-3 解釋。




```
always @*
begin
    if(month0 == 4'd9)
        init1 = 4'd0;
    else
        init1 = 4'd1;

    if(day0 == 4'd9)
        init0 = 4'd0;
    else
        init0 = 4'd1;

    if(month1 == 4'd1)
    begin
        limit2 = 4'd2;
    end
```

```
        else
        begin
            limit2 = 4'd9;
        end

        case(month1*4'd10+month0)
        4'd1, 4'd3, 4'd5, 4'd7, 4'd8, 4'd10, 4'd12:begin
            limit1 = 4'd3;
            if(day1 == 4'd3)
                limit0 = 4'd1;
            else
                limit0 = 4'd9;
        end
        4'd2:begin
            limit1 = 4'd2;
            if(day1 == 4'd2)
                limit0 = 4'd8;
            else
                limit0 = 4'd9;
        end
        default:begin
            limit1 = 4'd3;
            if(day1 == 4'd3)
                limit0 = 4'd0;
            else
                limit0 = 4'd9;
        end
        endcase
    end
```

3. Discussion

我是用三個 DIP 來控制，我認為這樣比較直觀，而非因為有三個 state 可以用 2bit 表示而設兩個

在寫 date 的時候出很多問題，我一開始是把 initial value 設為 01，但我發現這樣 09 數完會直接跳 11，因此我後來才用上面所提的方法避開這問題。

4. Conclusion

做完這次 lab，讓我學會如何 implement 有年，月，日，時，分的時鐘，以

及讓不同月有不同的天數。感覺現在做的實驗越來越生活實用了，不過也隨之越來越複雜

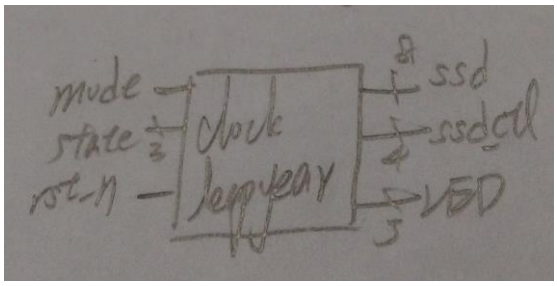
Lab6-3

1. Specification

Calendar(year, month, day) with leap year + 24hour time display with AM/PM

Input : mode, [2:0] state, clk ,rst_n

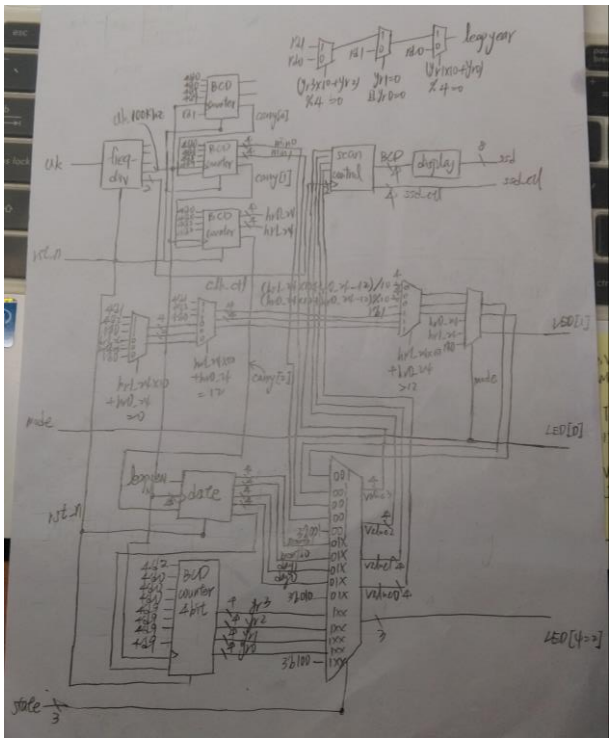
Output : [7:0] ssd, [3:0] ssd_ctl, [4:0]LED



2. Implementation

簡單來說，比 lab6-2 多了閏年。因此要修改 date module，要多 input leapyear

以下為各 module 間 wire 的連結，還有多設兩顆 LED 分別表示是否為 AM/PM 制，以及現在為 AM 還是 PM，跟三顆 LED 分別表示現在為年/月/日。在 top module 我還有用 if-else 來換算 AMPM，並用 state 決定七段顯示器要顯示年/月/日。最後使用 if-else 判斷是否為閏年。



I/O	Ssd[7]	Ssd[6]	Ssd[5]	Ssd[4]	Ssd[3]	Ssd[2]	Ssd[1]	Ssd[0]

Pin	W7	W6	U8	V8	U5	V5	U7	V7

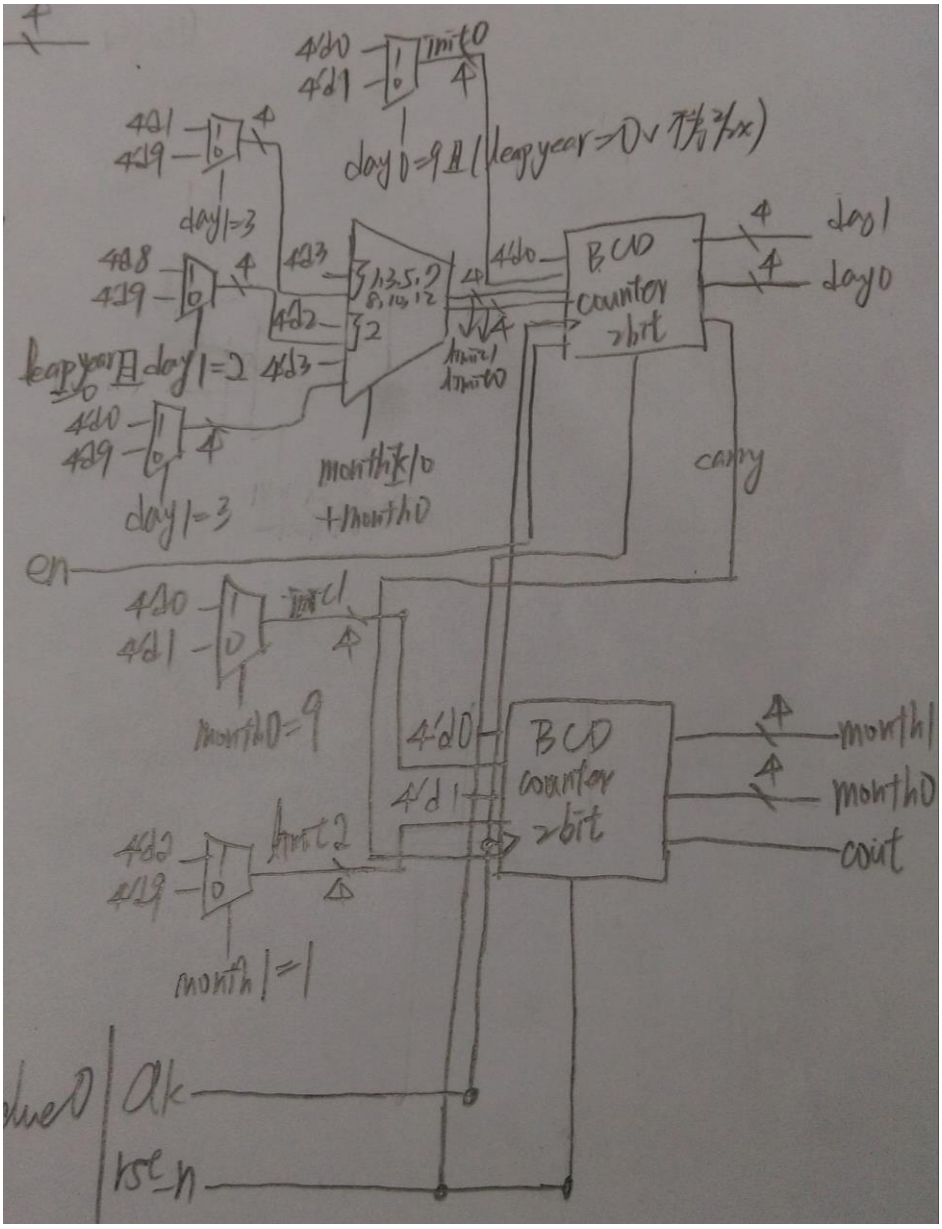
I/O	Ssd_stl[3]	Ssd_stl[3]	Ssd_stl[3]	Ssd_stl[3]
Pin	W4	V4	U4	U2

I/O	LED[4]	LED[3]	LED[2]	LED[1]	LED[0]
Pin	U15	W18	V19	U19	E19

I/O	State[2]	State[1]	State[0]	Mode	Clk	Rst_n
Pin	V15	W15	W17	V16	W5	V17

date

如果為閏年的話，二月份有 29 天。但我之前是用個位數來判斷個位數進位為 0 或 1。2/29 下一天應該是 3/01，但用之前的條件跑會是 3/00，因此要修改條件。




```
if(day0 == 4'd9 && (leapyear != 4'd1 || month1 != 4'd0 || month0 != 4'd2 ||
day1 != 4'd2))
                init0 = 4'd0;
        else
            init0 = 4'd1;
```

3. Discussion

因為和 lab6-2 很像，再加上前車之鑑，所以 lab6-3 打得比較快。

閏年除了被 4 整除外，還有被 100, 400 整除的規則，但由於必須邊注意有沒有 2/29 號，但又要注意年分，所以也不能調太快，因此這部分應該是無法檢查。

4. Conclusion

做完這整個 lab，讓我學到如何寫出時鐘和處理 BCD 數據，乍看之下這次 lab 跟之前很像，做了才發現有很多小細節需要注意。