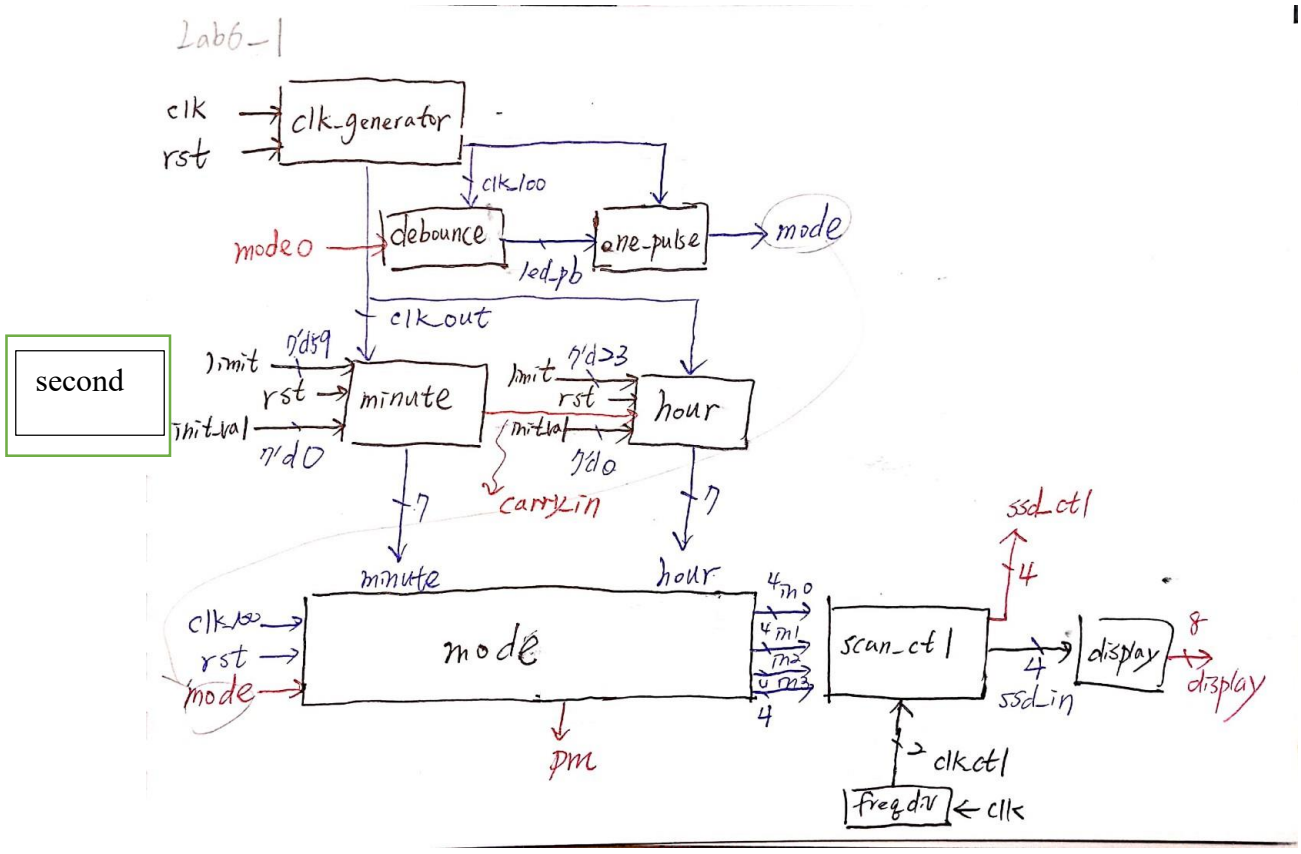
1.
Design Specification

✓ **Input / Output**

For an electric watch with AM/PM and 24-hour:
Input: clk, rst, modeo.
Output: display [7:0], ssd_ctl[3:0],pm.

✓ **Block Diagram**

- **top**



Design Implementation

- ✓ **Function Descriptions**

  此功能為一個可以切換 24/12 小時制，並顯示秒分時的電子錶。

  (i) 計時器的部分分為三種 second , minute, hour　各自輸進 limit , rst, init_val, carry_in(exclude second)來判斷何時 cout＝1,(exclude hour) 與歸到初始值。

  方法為如果現在 minute<limit 與 carry_in = 1 那就選擇加一輸出，如果 minute 等於 limit ,　且 carry_in 等於 1 那就選擇回歸到原始值且輸出 cout = 1，若以上狀況都違反，那就選擇不變的值輸出且 cout = 0.

  接著把 second cout 當作 minute carry_in , 把 minute cout 當作 hour carry_in。這三個 module 各自輸出 second, minute, hour 的值輸入到 mode 裡。

  (ii) Push button 的輸入 in 控制 mode, 先把 in 通過 debounce, onepulse(此作法已在前題講述)，再把他輸進 mode module 裡用以控制 second, minute, hour 誰該輸出到 scan_ctl 最後 display 出來。

  方法為　設定三種 state　(1)24 小時(2)12 小時 與(3)顯示秒, 做法為 finite state machine。

  在 24 小時制的 state 裡　選擇 pm=0 輸出　且輸出不變的 minute 跟 hour 輸出。

  在 12 小時制的 state 裡，做一連串的判斷。

  如果現在 hour>12（ex. 13 (24)= pm1） 則選擇輸出 1 做 pm, 把現在的值減 12 輸出。

  又如果 hour==12 (12(24)＝pm12）則也選擇 1 做 pm 輸出，並且輸出不變的 hour。

  又如果 hour==0（0(24) = am12）則選擇 0 做 pm 輸出, 輸出 12 做 hour。

  若以上條件皆不符合，則 pm = 0, 輸出不變的 hour.

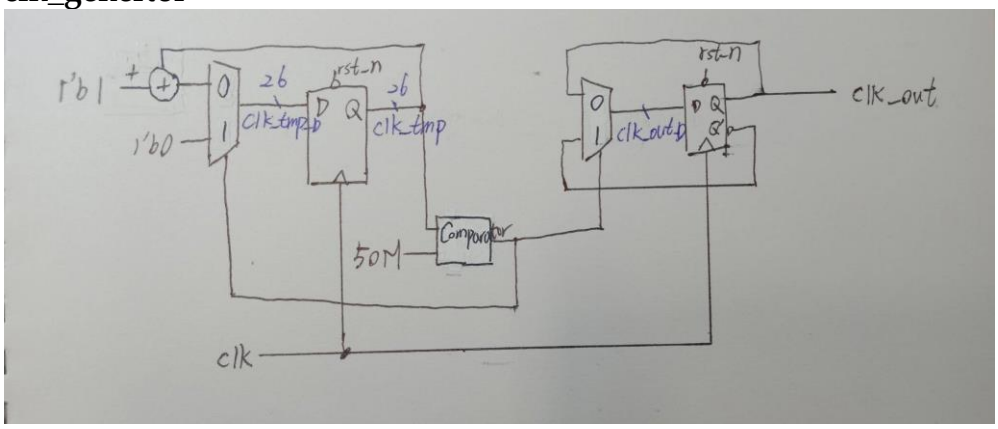
  最後也是選擇 minute 跟 hour 輸出。

  在顯示秒的 state 裡，則選擇 second 輸出。

  三個 state 的轉換是靠判斷 in 若為 1 則到另一個 state。(1)->(2)->(3).

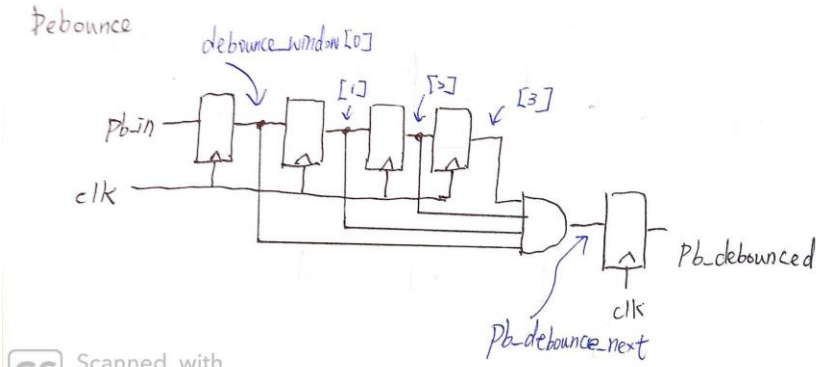
  (iii)scan_ctl 跟 display 都與之前相同作法。

- ✓ **Logic Diagram**
  - **clk_genertor**

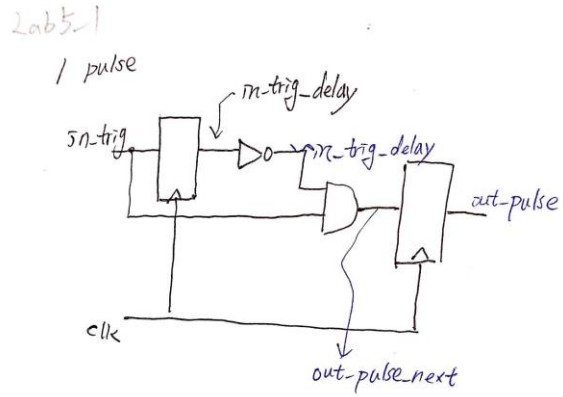


PS.把圖中 comparator 的輸入 50M 改成 5000K 就可以變成輸出為 100HZ

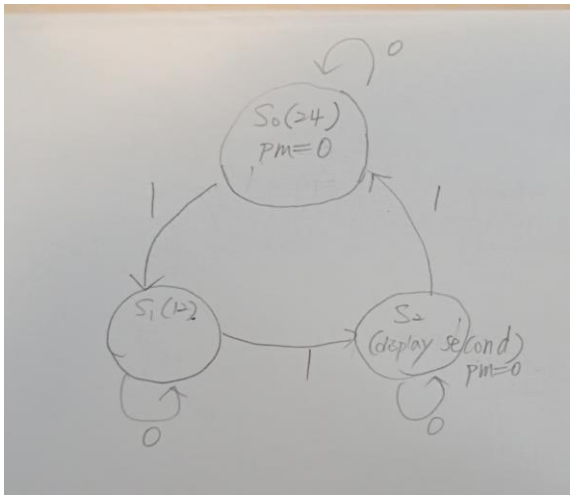
- **debounce**



- **one pulse**



- **mode**



✓ **Logic Functions**

- **minute**

```
if (minute<limit && carry_in) begin
        minute_next = minute + 1;
        carry_out = 0;
        end
    else if (carry_in) begin
        minute_next = init_val;
        carry_out = 1;
        end
    else begin
        minute_next = minute;
        carry_out = 0;
        end
```

- **mode**

state 0(24hour)：

```
pm = 0;
```

state 1(12 hour)：

```
if (hour>12) begin
        pm = 1;
        hour_tmp = hour - 12;
    end
    else if (hour==12) begin
        pm = 1;
        hour_tmp = hour;
    end
    else if (hour==0) begin
        pm= 0;
        hour_tmp = 12;
    end
    else begin
        pm = 0;
        hour_tmp = hour;
    end
  end
```

✓ **I/O pin**

I/O	clk	rst	modeo	pm	ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]
LOC	W5	U18	T17	L1	U2	U4	V4	W4
I/O	display[7]	display[6]	display[5]	display[4]	display[3]	display[2]	display[1]	display[0]
LOC	W7	W6	U8	V8	U5	V5	U7	V7

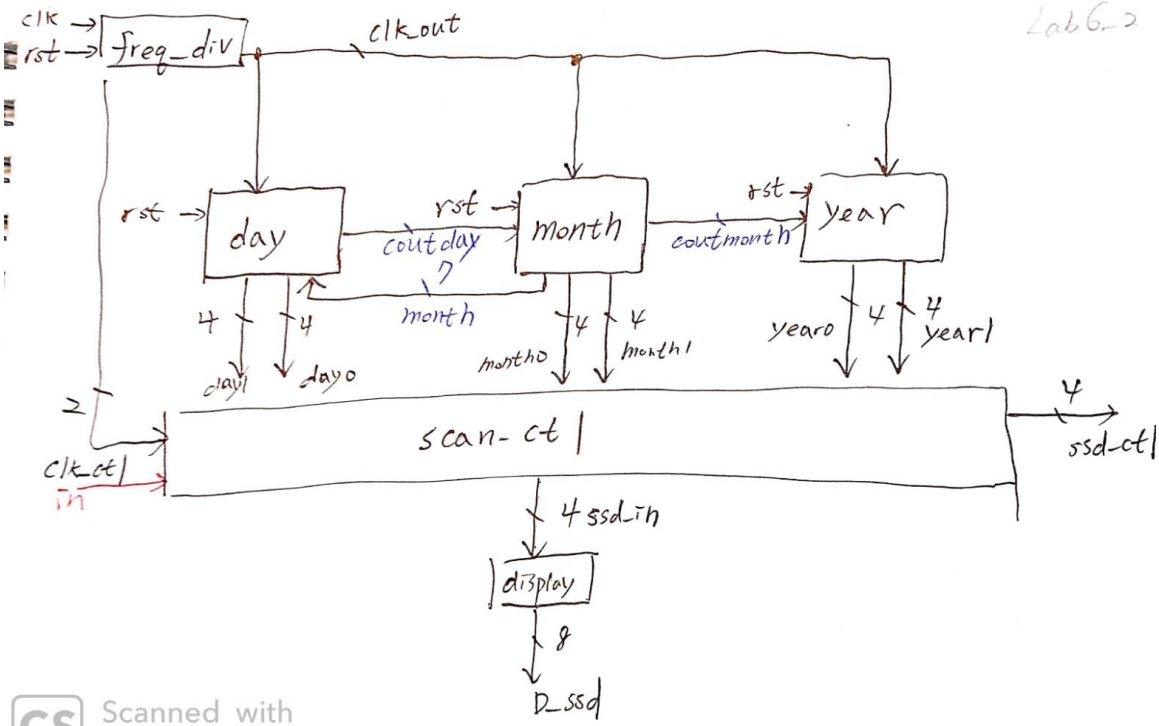
2.
Design Specification

✓ **Input / Output**

For an electric watch with YEAR-MONTH-DAY:
Input: clk, rst, in.
Output: D_ssd [7:0], ssd_ctl[3:0].

✓ **Block Diagram**

- **top**



Design Implementation

✓ **Function Descriptions**

這題的功能為能夠顯示年月日的計時器。
此題與上題切換顯示內容的方式不一樣在於此題控制 mode 的輸入 DIP switch 因此不需要有 debounce onepulse。

(i) 計數器的部分有三種 day month year 做法大致與上題相同，比較特別的是 day 的 limit 需靠 month 來判斷，因此 month 有輸出到 day 中。
此處判斷 day limit 的方法是用查表(會在 logic function 提供)。
把 day cout 當作 month carry_in , 把 month cout 當作 year carry_in。這三個 module 各自輸出 day, month, year 的值輸入到 scan_ctl 裡。

(ii) 因為這裡只有兩個 state (顯示月日或者年)而且判斷的輸入的是 DIP switch，明確的 1 或 0。所以我是直接在 scan_ctl 做判斷與選擇。
判斷的方法為　如果 in == 1 則選擇顯示月跟日　若 in==0 則選擇輸出年。

✓ **Logic Functions**

- **day(查表方式決定 limit)**

```
case(month)
        1:limit= 31;
        2:limit= 28;
        3:limit= 31;
        4:limit= 30;
        5:limit= 31;
        6:limit= 30;
        7:limit= 31;
        8:limit= 31;
        9:limit= 30;
        10:limit= 31;
        11:limit=30;
        12:limit= 31;
```

- **scan_ctl（決定輸出月跟日還是年）**

```
 if (in)
ssd_in = in0;
else
ssd_in = in4;
if (in)
ssd_in = in1;
else
ssd_in = in5;
```

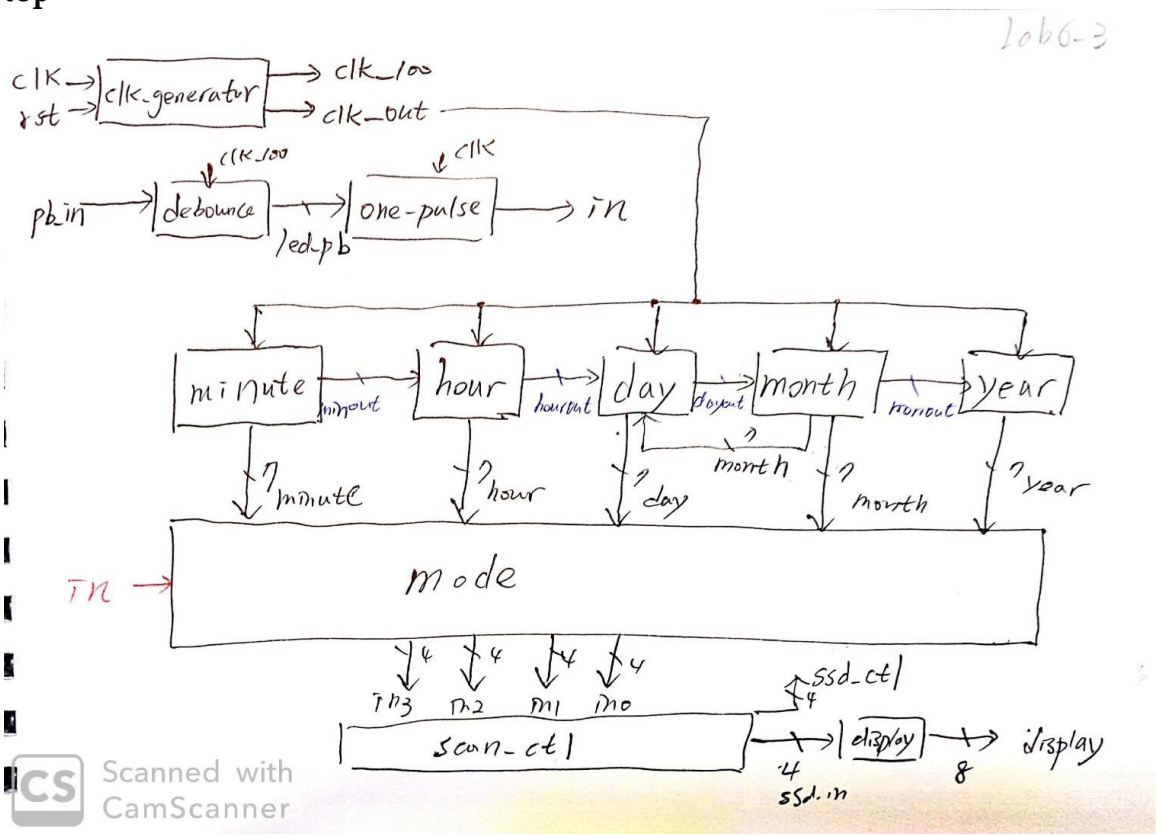
✓ **I/O pin**

I/O	clk	rst	in	ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]	
LOC	W15	U18	V17	U2	U4	V4	W4	
I/O	D_ssd[7]	D_ssd[6]	D_ssd[5]	D_ssd[4]	D_ssd[3]	D_ssd[2]	D_ssd[1]	D_ssd[0]
LOC	W7	W6	U8	V8	U5	V5	U7	V7

3.
Design Specification

- ✓ **Input / Output**
  For an electric watch with YEAR-MONTH-DAY and AM/PM transfer 24-hour, and the leap year support:
  Input: clk, rst, pb_in.
  Output: display[7:0], ssd_ctl[3:0].

- ✓ **Block Diagram**
  - **top**



Design Implementation

- ✓ **Function Descriptions**

  這題的功能為能夠顯示年月日小時分鐘並且有 12/24 小時制轉換與閏年的計時器。

  (i) 計數器的部分有 5 種 ：minute hour day month year， day 的 limit 需靠 month 與 year 來判斷，因此 month, year 有輸出到 day 中。

  此處判斷 day limit 的方法是用查表但因為有閏年的差別，這裡的做法為若判斷 year 為閏年　判斷式 logic funciton 中 (ps.在硬體實驗時最好不要用到跟除法相關的運算子因為會影響速度或出錯，但在這裡我沒有為了這個做一個除法器) 則輸出的 month 當作原為 2 改為 13 照樣查表。

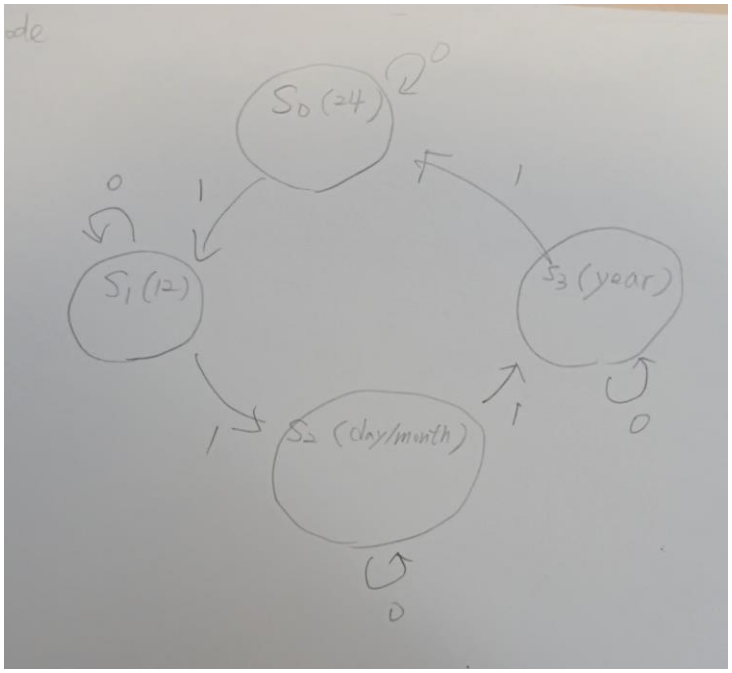
  把 minute cout 當作 hour carry_in，把 hour cout 當作 day carry_in,　把 day cout 當作 month carry_in , 把 month cout 當作 year carry_in。這五個 module 各自輸出 hour, minute, day, month, year 的值輸入到 mode 裡。

  (ii) mode 中　輸入以上的值加上經過 debounce one pulse 的 in 來判斷誰該輸出到顯示元件。

  總共有 4 個 state 分別為　(1)顯示時分 24 小時制　(2)顯示時分 12 小時制　(3)顯示月日　(4)顯示年

  與前兩題相同的作法在做 state 的轉換。

- ✓ **Logic Diagram**
  - **mode**



- ✓ **Logic Functions**

- **day**

leap year :

if (((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0 && year % 3200 != 0))

月份查表：

```
case(month)
    1:limit= 31;
    2:limit= 28;
    3:limit= 31;
    4:limit= 30;
    5:limit= 31;
    6:limit= 30;
    7:limit= 31;
    8:limit= 31;
    9:limit= 30;
    10:limit= 31;
    11:limit=30;
    12:limit= 31;
```

✓ **I/O pin**

I/O	clk	rst	pb_in	pm	ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]
LOC	W5	U18	T17	L1	U2	U4	V4	W4
I/O	display[7]	display[6]	display[5]	display[4]	display[3]	display[2]	display[1]	display[0]
LOC	W7	W6	U8	V8	U5	V5	U7	V7

Discussion

跟上次相比，這次的實驗比較熟練，也比較順利。但是因為太多種 counter 了，要同時檢查輸出有沒有對很困難，所以要不斷的調整 clock 輸出的頻率。
這次 demo 前突然發現我 lab6_1 少加了 second 的顯示，但因為 counter 的相連比較清楚，所以可以很快地就串上多一種顯示了！
像現在有越來越多的 module 要處理，我最常出錯的就是 bit 數打錯。例如有次燒板子時發現為什麼秒走到 57 就會直接跳 0 想來想去後來發現是 bit 數少了一個，只有 3bit 難怪無法顯示 5’8’ 5’9’。

Conclusion

這次實驗大概都在碰 counter 與 mode 的控制。mode 是用 FSM 做成的。
其實各種 counter 的原理都很像，一個 cin 一個 cout，一個接一個串起來就變成多種顯示的 watch 了。而且這些 module 都可以 reuse 可以做到規格化。
另外在 state 的轉換也很重要，先畫好 state diagram 就會清楚現在在哪個 state 要做甚麼判斷與輸出。