Lab6: FSM

107061112 王昊文

Experiments

1 Finish the time display function supporting 24-hour (00-23).

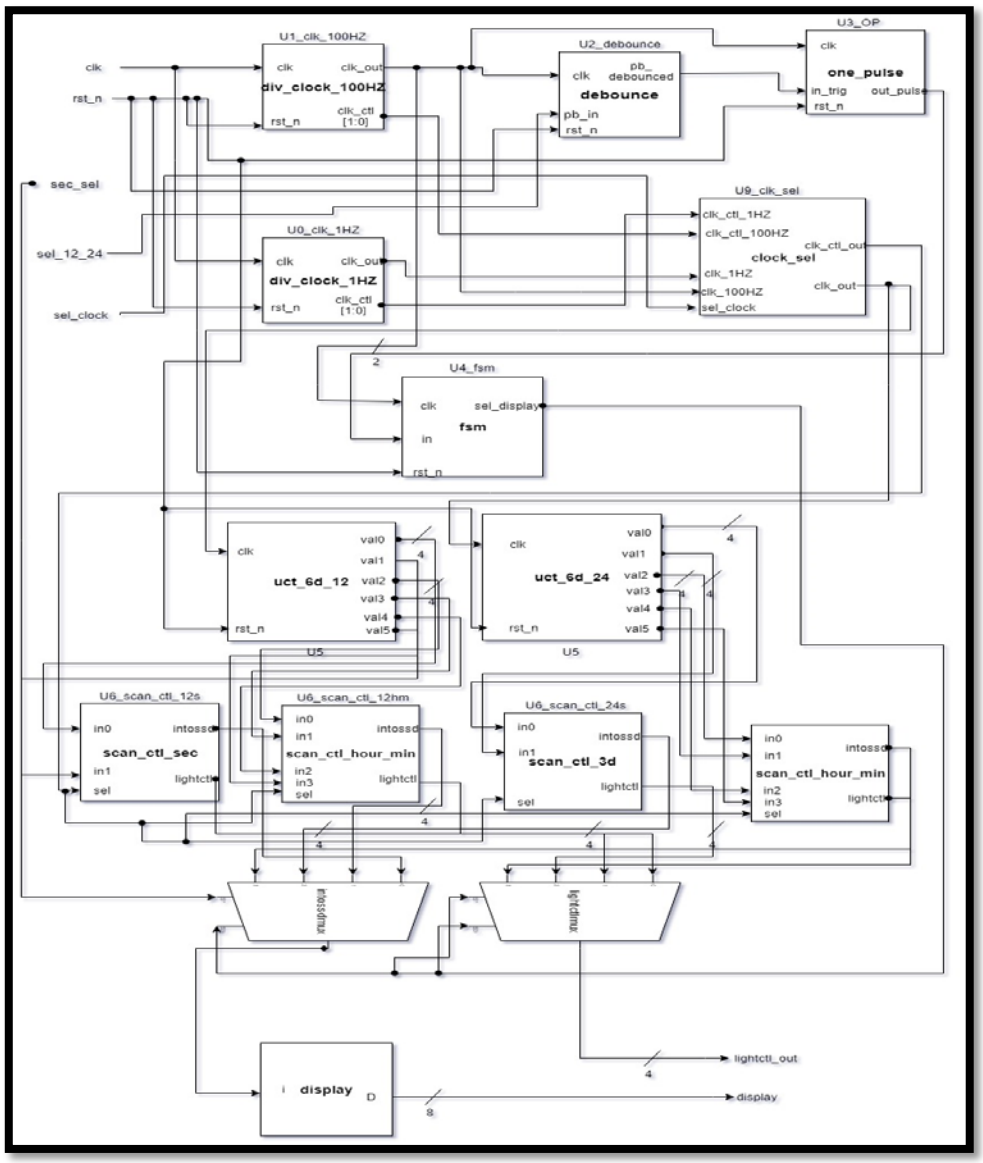
1.1 Support two modes: AM/PM and 24-hour.

Design Specification

✓ I/O

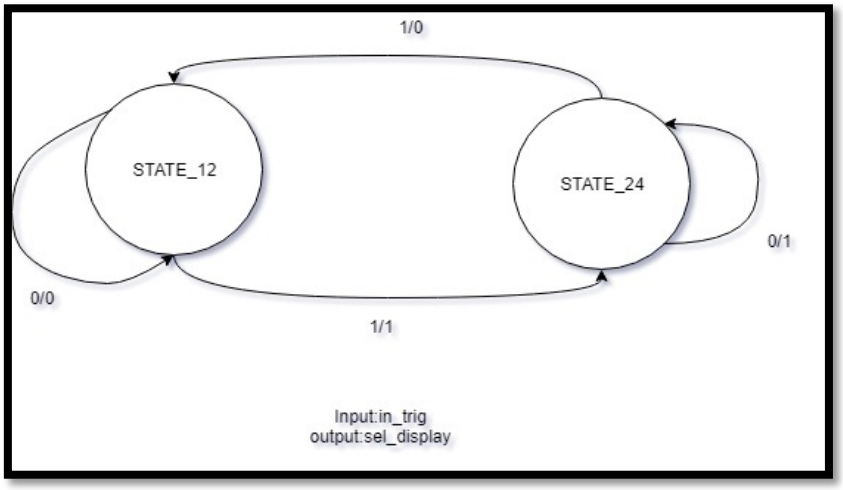
```
Input clk;          // crystal clock
Input rst_n;        // low active reset
Input sec_sel;      // select second mode
Input sel_12_24;    // select 12_24 mode
Input sel_clock;    // select fast or slow clock (for demo uses)
Output [3:0]lightctl_out;
Output [7:0]display;
```

✓ Logic Block:



這一題必須展現24hr與12hr的clock，我使用了兩個counter去數，將其limit與重數的設定一個改成24，另外一個設成12即可。而fsm可以根據目前使用者的輸入來選擇顯示的display，有秒，12/24等選擇，fsm會有選擇指標進入mux來做選擇。Sec_sel為秒功能選擇，而每個counter當中都包含著秒的計數，所以不管選擇12/24，sec_sel按下去都會顯示秒。Clock_sel則為demo的時候選擇較快的clock所使用。基本上其他模組皆為沿用之前的lab。最後面的兩個mux為選擇輸出使用。

Fsm：



以上input為使用者選擇，output則為之後mux選擇要顯示的模式的selector。

✓ Design Implementation

Input				
clk	*rst_n*	*sec_sel*	*sel_12_24*	*sel_clk*
W5	R2	V17	U18	V16

Output ssd							
display[0]	***display[1]***	***display[2]***	***display[3]***	***display[4]***	***display[5]***	***display[6]***	***display[7]***
V7	U7	V5	U5	V8	U8	W6	W7

Output lightctl			
lightctl[0]	*lightctl[1]*	*lightctl[2]*	*lightctl[3]*
U2	U4	V4	W4

✓ Discussion

這次的lab與上次差不多，只需透過兩個counter去計數即可。然而因為12/24小時兩個起始值不同，後來發現只要在數到某個數字的時候輸出想要的輸出即可，強制變換他的輸出。因此並沒有花費太多的時間。後來我發現，基本上遇到多功能顯示counter的時候，就用幾個counter去計數，最後再使用mux來選取就好。如過設計時透過使用者改變輸入而去改動counter的initial value，容易造成錯誤。有幾個功能就用幾個counter去數，一來好控制，二來也不容易出錯，只需要focus在每一個counter功能正確就好。

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 7-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 (4SSDs).

Design Specification

✓ I/O

Input clk; // crystal clock

Input rst_n; // low active reset

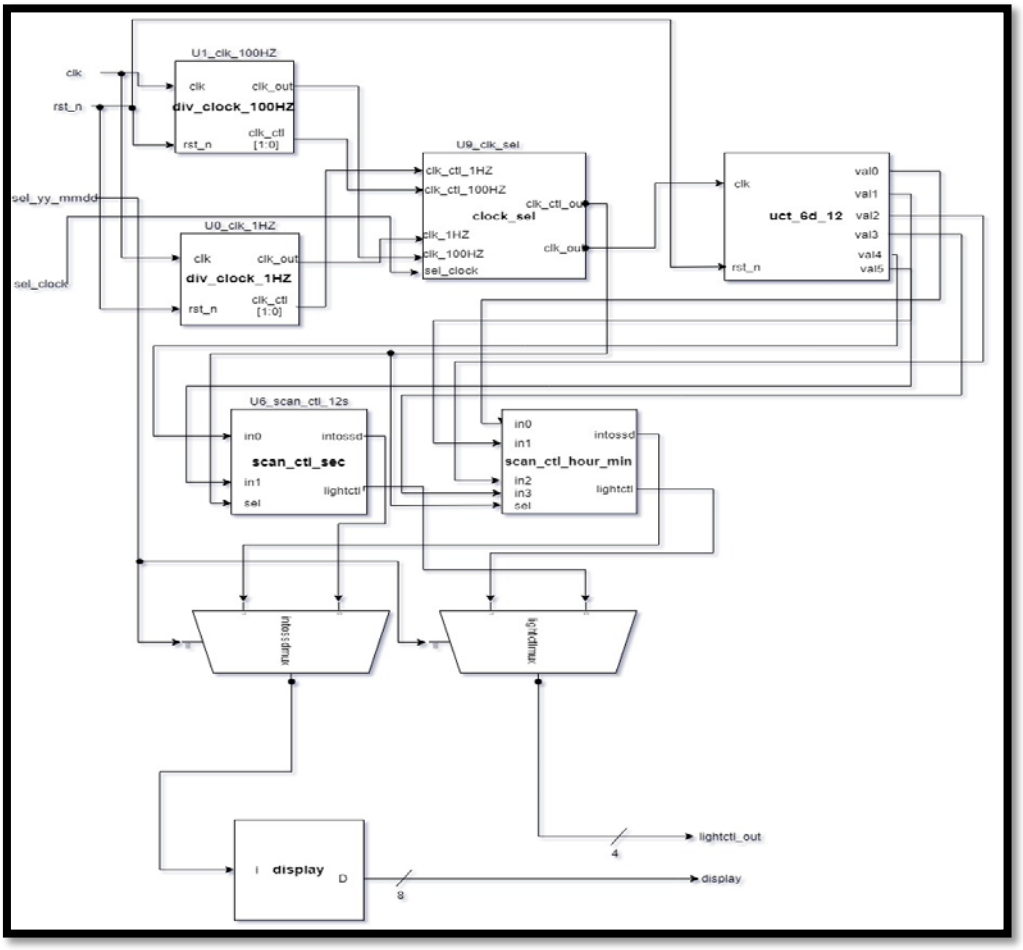
Input sel_yy_mmdd; // select year or month / date

Input sel_clock; // select fast or slow clock (for demo uses)

Output [3:0]lightctl_out;

Output [7:0]display;

✓ Logic Block:



這一題比上一題更加簡單，由於題目說可以使用dip switch，因此我使用dip switch來切換年月與日，不需要用到finite state machine來做選擇。然後這一題我只使用了一個counter，裡面就包含了年月日的計數。然後這些counter裡面設定條件，譬如大月小月，如過月的部分數到大月，日的counter數到31就必須近位。2月則是數到28進位。而這些條件我使用了一個指標->recur reset來當作另外一個reset counter的條件，recur reset時會將計數值回到initial value。
這一題我選擇使用一個counter來做，是因為顯示單位之間彼此都有進位關係。儘管是來自同一個counter，我仍然送到不同的scan裡面，這樣到後棉還是可敵到選擇的效果。
Sel_mmdd_yy，這個input就是後面mux做選擇時的select。

✓ Design Implementation

Input			
clk	*rst_n*	*sec_sel*	*sel_clk*
W5	R2	V17	V16

Output ssd							
display[0]	***display[1]***	***display[2]***	***display[3]***	***display[4]***	***display[5]***	***display[6]***	***display[7]***
V7	U7	V5	U5	V8	U8	W6	W7

Output lightctl			
lightctl[0]	*lightctl[1]*	*lightctl[2]*	*lightctl[3]*
U2	U4	V4	W4

✓ Discussion

這一題比前面那一題更簡單了一些，想法都是一模一樣，只需要在counter的部分動一些手腳即可。我仍然保持一個觀念，就是幾個功能就用幾個counter來做，才不會錯亂。到後面再使用counter來做選擇即可。
3 (Bonus) Add the time display support of both AM/PM and 24-hour, and the leap year support. (The year will start from 2000 to 2200 and use 4 SSDs to display.)

Design Specification

✓ I/O

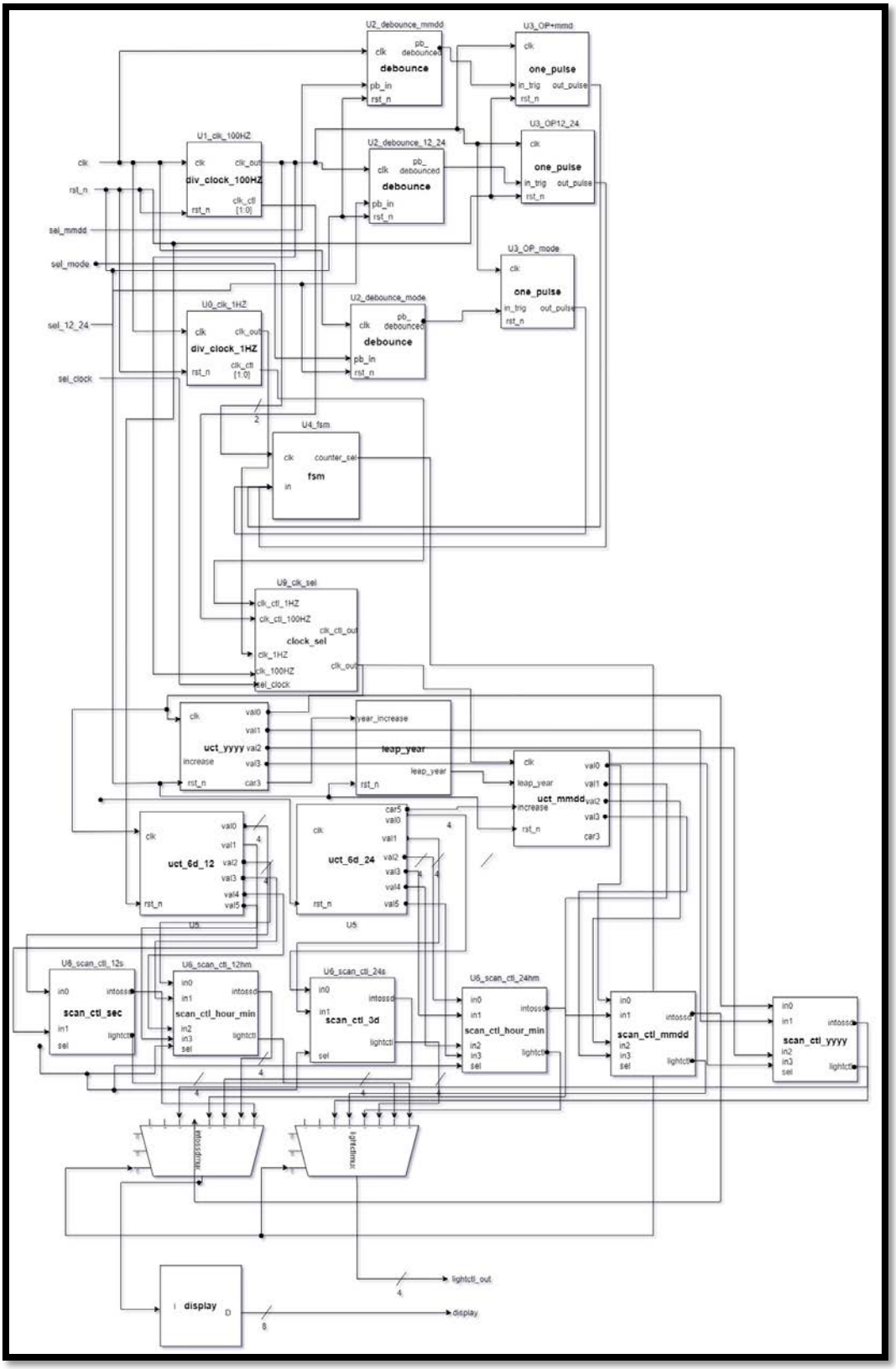
```
Input clk;              // crystal clock
Input rst_n;            // low active reset
Input sel_mmdd_time;    // select year or month / date
Input sel_mode;         // time mode or mmdd mode
Input sel_12_24;        // in time mode-> select 12/24 or sec, mmdd mode -> yy or
mmdd
```

Input sel_clock; // select fast or slow clock (for demo uses)

Output [3:0]lightctl_out;

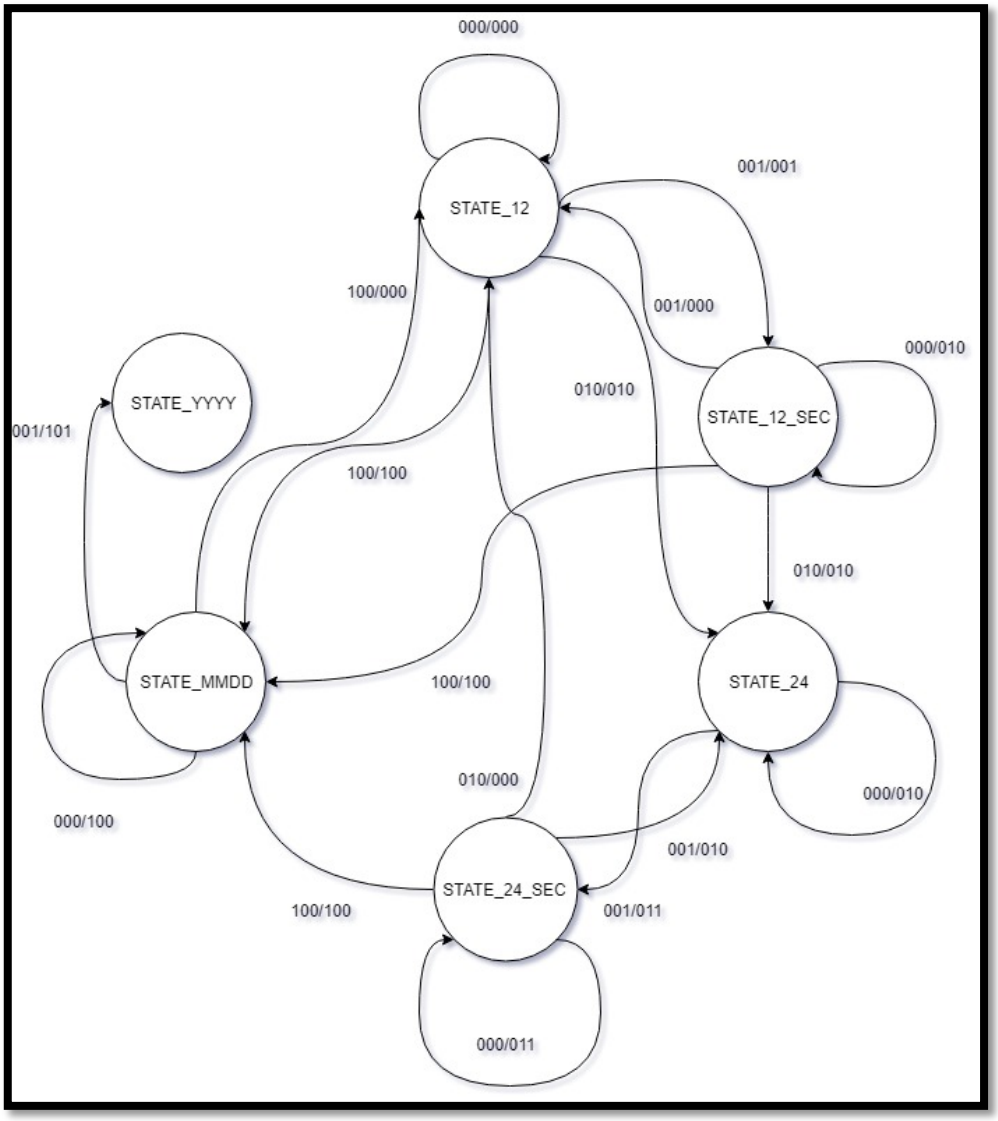
Output [7:0]display;

✓ Logic Block:



這一題由於老師說將input用成push button會加分，因此我將fsm重新設計過，後面會附上fsm設計圖。這一題因為要有year，時間，秒的顯示，以及12/24小時，總共使用了四個counter來計數。一樣fsm會根據使用者目前的輸入來選擇後面mux的輸出。整體設計與前面的都雷同，一樣前面做訊號處理，後面進到fsm，然後到後面輸出。唯一不同的點是需要支援leap year功能，我採用了多一個leap year模組，判斷年份的末兩位，是否為4的倍數，若是的話輸出給counter做recur reset的判斷。

Fsm:



Input共有三個，暫時以in作為input的代號。In[0]為模式選擇，在12/24功能下按下in[0]，可以選擇顯示sec / hour_min。在年月功能下按下in[0]可以切換年或月日。In[1]為小時制選擇，在時間功能下按才會有反應，可以選擇12/24。In[2]為功能選擇，可以選擇時間功能或年月日功能。在預設情況下，如果想從年月日功能切換到12/24，預設會直接切換到12小時制模式，想到24小時制必須再按一次in[1]。反之，如果想從時間功能切換回年月日功能，預設會進入年月功能，

如果想察看年份，必須再按一次in[0]。

In[2], in[1], in[0]分別為{sel_mmdd_time, sel_12_24, sel_mode}。

✓ Design Implementation

Input					
clk	*rst_n*	*sel_mmdd_time*	*sel_12_24*	*sel_mode*	*sel_clock*
W5	R2	W19	U18	T17	V17

Output ssd							
display[0]	***display[1]***	***display[2]***	***display[3]***	***display[4]***	***display[5]***	***display[6]***	***display[7]***
V7	U7	V5	U5	V8	U8	W6	W7

Output lightctl			
lightctl[0]	*lightctl[1]*	*lightctl[2]*	*lightctl[3]*
U2	U4	V4	W4

✓ Discussion

其實做完這一題之後我覺得我的設計並不是非常理想。剛開始看到題目就開始修改fsm，fsm其實變的很複雜。幸好我運氣很好，fsm並沒有出現什麼error，fsm變得複雜的壞處就是很難debug。後來我想想，能不能直接將第一題，第二題做好的top module直接丟進第三題裡面，最後用一個mux取選擇想要顯示哪個module，這樣聽起來做起來感覺會簡單很多。這一題其實難度不難，只是設計起來非常繁複。

不過在製作第三題的時候我曾經遇上大麻煩，就是leap year的設計。我使永一個counter，每四年數一次，output 一次leap year的訊號。可是不管怎麼做就是會有bug，也找不出來在哪邊。後來我的猜想是因為我使用了gated clock，畢竟我的counter需要有年counter的計數訊號當成我的clock，這樣或許會導致一部份訊號不正常。後來聽從同學建議還是多使用combinational logic來做一些簡單的判斷，避免一些debug上面的麻煩。

✓ Conclusion

這一次的lab其實性質與上一次類似，上次做覺得複雜，到了這次就已經很熟練那個套路。其實大一上學習counter時還感覺不出來counter的強大，沒想到可以應用這麼多，在這堂課程真的有滿滿應用所學的成就感。