**Lab6\_1**

**Design Specification**

* For a clock with second, minute, and hour:

Input:

switch // 時分與秒切換

rst\_n // control rst\_n button

clk

mode // 24小時制與12小時制切換

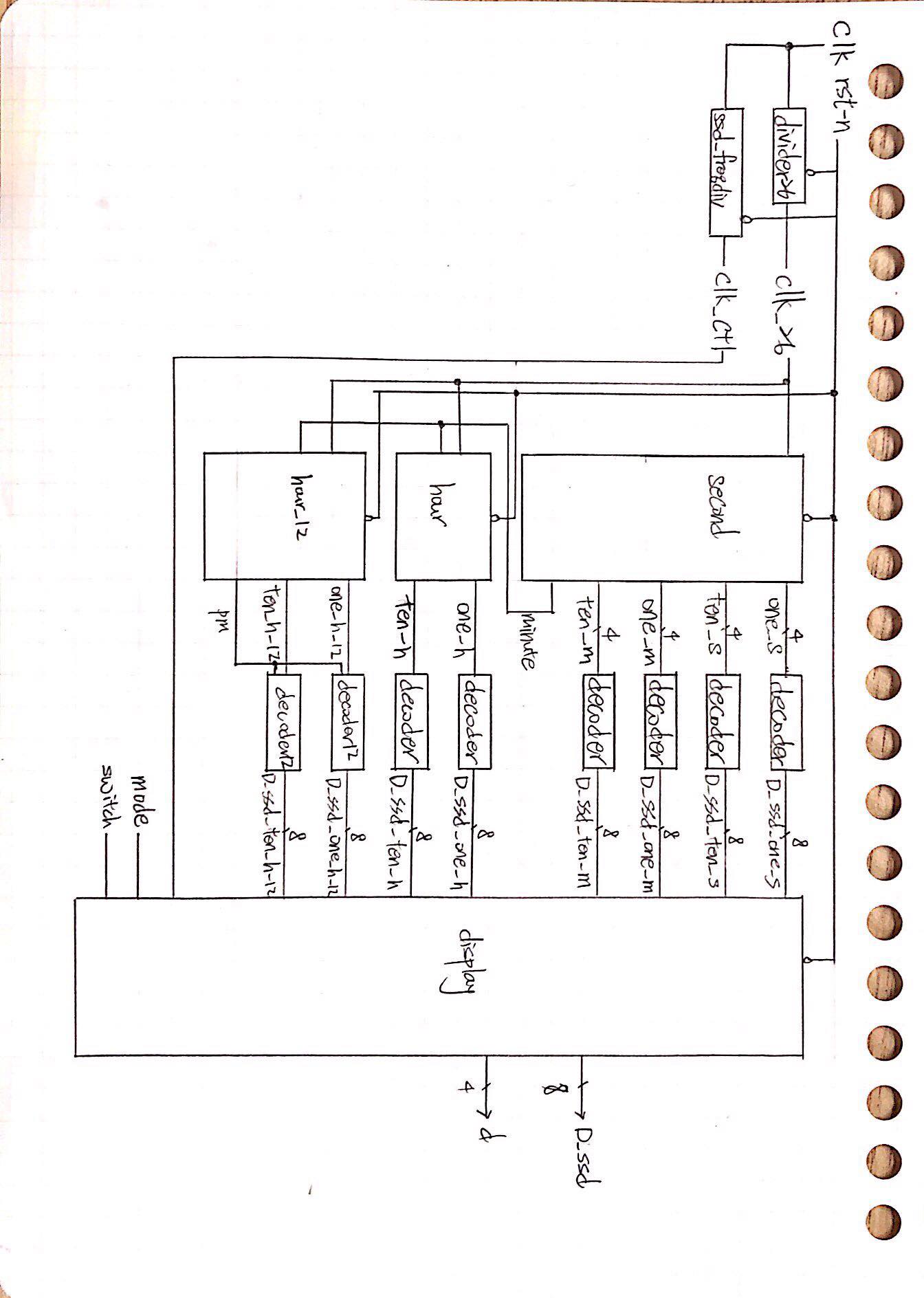
clk\_c // 控制秒的頻率(非必要，因此無在block diagram中顯示)

Output:

D\_ssd[7:0] // 7-segment display

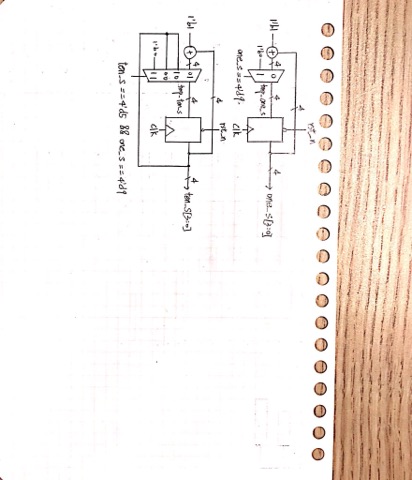
d[3:0]

* Draw the block diagram of the design.

****

**Design Implementation**

* 本題由ssd\_freqdiv、divider26、second、hour、hour\_12、decoder、decoder12、display八個module組成，其中second、hour、hour\_12皆為upcounter，其差異進位條件不同；decoder與decoder12的差異在於7-segment display的點有沒有亮。以下將分別說明second、hour、hour\_12、display的功能，而ssd\_freqdiv、divider26、decoder則在先前的lab即使用過，將不在贅述。
* Second、hour、hour\_12



Second的功能為計算秒與分，主要組成為BCD upcounter。上圖為秒的部份的示意圖。而分的部分由於結構類似，因此將以文字說明。

* 分的個位數部分

|  |  |  |  |
| --- | --- | --- | --- |
| MUX條件 | | | MUX output |
| one\_s == 4'd9 | ten\_s == 4'd5 | one\_m == 4'd9 | one\_m |
| default | | | one\_m |
| 1 | 1 | 0 | tmp\_one\_m |
| 1 | 1 | 1 | 4'd0 |

* 分的十位數部分

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MUX條件 | | | | MUX output | |
| one\_s == 4'd9 | ten\_s == 4'd5 | one\_m == 4'd9 | ten\_m == 4'd5 | ten\_m | minute |
| 1 | 1 | 1 | 0 | tmp\_ten\_m | 1'b0 |
| 1 | 1 | 1 | 1 | 4'd0 | 1'b1 |
| default | | | | ten\_m | 1'b0 |

十位數部分多加了一個minute作為整個second的output，其作用為判斷hour要不要進位。

Hour的部份其結構亦與上圖相似，差異在於MUX條件。

* 時的個位數部分

|  |  |  |  |
| --- | --- | --- | --- |
| MUX條件 | | | MUX output |
| ten\_h == 4'd2 && one\_h == 4'd3 | one\_h == 4'd9 | minute | one\_h |
| 0 | 1 | 1 | 4'd0 |
| 1 | 0 | 1 | 4'd0 |
| 0 | 0 | 1 | tmp\_one\_h |
| default | | | one\_h |

其中(ten\_h == 4'd2 && one\_h == 4'd3)與one\_h == 4'd9不可能同時成立。

* 時的十位數部分

|  |  |  |  |
| --- | --- | --- | --- |
| MUX條件 | | | MUX output |
| ten\_h == 4'd2 && one\_h == 4'd3 | one\_h == 4'd9 | minute | ten\_h |
| 0 | 1 | 1 | tmp\_ten\_h |
| 1 | 0 | 1 | 4'd0 |
| default | | | ten\_h |

Hour\_12為12小時制的模式，在我的設計中凌晨12點為00。

* 時的個位數部分
  + 若minute == 1
    - 若為凌晨00點或中午12點，則one\_h\_12為1
    - 若為早上09點或晚上11點，則one\_h\_12為0
    - 其餘one\_h\_12加1
  + 若minute == 0，one\_h\_12維持不變
* 時的十位數部分
  + 其中加入pm表示下午
  + 若minute == 1
    - 若為晚上11點或中午12點，則ten\_h\_12為0；pm <= (~pm)
    - 若為早上09點，則ten\_h\_12為1；pm <= pm
  + 若minute == 0，ten\_h\_12維持不變；pm <= pm

Input & Output

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| second | | hour | | hour\_12 | |
| input | output | input | output | input | output |
| clk | one\_s | clk | one\_h | clk | one\_h\_12 |
| rst\_n | ten\_s | rst\_n | ten\_h | rst\_n | ten\_h\_12 |
|  | one\_m | minute |  | minute | pm |
| ten\_m |  | |  | |
| minute |

* Display

此為控制螢幕顯示時間的裝置，其中switch為控制顯示時分或秒，mode為控制時制。表中的c隨頻率改變，每個頻率所顯示的時間位數不同，透過視覺暫留讓我們同時看到不同數字；d則是控制4個7-segment display亮。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| switch | mode | c | D\_ssd | d |
| 1 | X | 00 | D\_ssd\_one\_s | 4'd1110 |
| 1 | X | 01 | D\_ssd\_ten\_s | 4'd1101 |
| 1 | X | default | 8'b11111111 | 4'd0000 |
| 0 | 1 | 00 | D\_ssd\_one\_m | 4'd1110 |
| 0 | 1 | 01 | D\_ssd\_ten\_m | 4'd1101 |
| 0 | 1 | 10 | D\_ssd\_one\_h | 4'd1011 |
| 0 | 1 | 11 | D\_ssd\_ten\_h | 4'd0111 |
| 0 | 1 | default | 8'b11111111 | 4'd0000 |
| 0 | 0 | 00 | D\_ssd\_one\_m | 4'd1110 |
| 0 | 0 | 01 | D\_ssd\_ten\_m | 4'd1101 |
| 0 | 0 | 10 | D\_ssd\_one\_h\_12 | 4'd1011 |
| 0 | 0 | 11 | D\_ssd\_ten\_h\_12 | 4'd0111 |
| 0 | 0 | default | 8'b11111111 | 4'd0000 |

* I/O pin

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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] |
| VOC | W7 | W6 | U8 | V8 | U5 | V5 | U7 | V7 |
| I/O | d[3] | d[2] | d[1] | d[0] | clk | rst\_n | mode | switch |
| VOC | W4 | V4 | U4 | U2 | W5 | V17 | V16 | W16 |

**Lab6\_2**

**Design Specification**

* For a calendar:

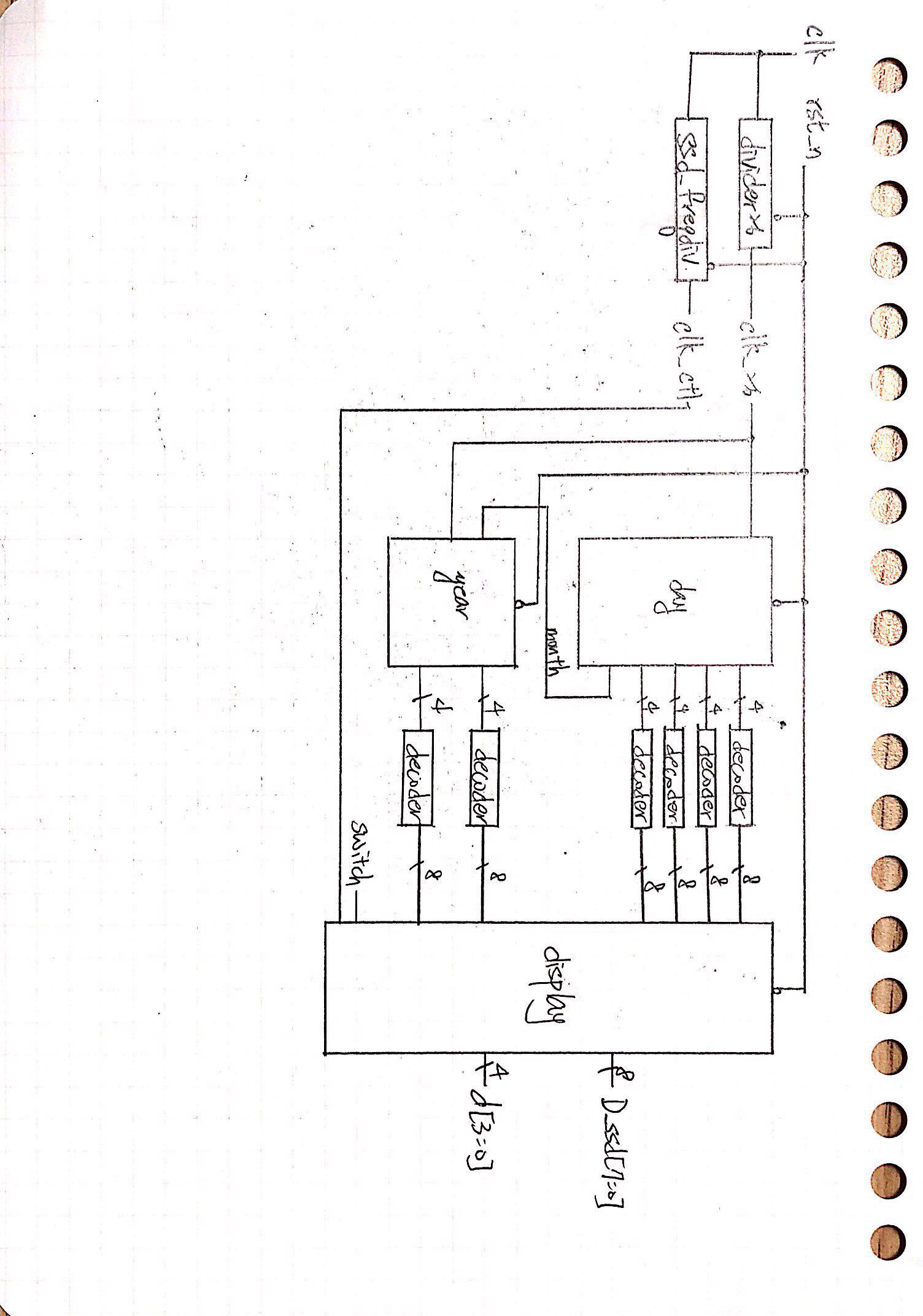
Input: clk,

rst\_n,

switch // 切換年與月日

Output: D\_ssd[7:0], d[3:0]

* Draw the block diagram of the design.



**Design Implementation**

* 此設計與lab6-1相似，由ssd\_freqdiv、divider26、day、year、decoder、display六個module組成，其中day與year皆為upcounter，其差異進位條件不同。以下將分別說明day與year的功能，其餘與lab6-1相同，將不再贅述。
* day與year

兩者與第一題的second結構相似，因此在此不再作圖，將以文字敘述MUX的判斷。

day：

在這裡我加入了month\_ctl\_one與month\_ctl\_ten作為月份天數的判斷，用以判斷天數是否進位或改變。當one\_d == month\_ctl\_one且ten\_d == month\_ctl\_ten時，代表過了一個月。

* 日的個位數部分

|  |  |  |  |
| --- | --- | --- | --- |
| MUX條件 | | | MUX output |
| one\_d == month\_ctl\_one | ten\_d == month\_ctl\_ten | one\_d == 4'd9 | one\_d |
| 1 | 1 | X | 4'd1 |
| X | 0 | 1 | 4'd0 |
| 0 | X | 1 | 4'd0 |
| default | | | tmp\_one\_d |

* 日的十位數部分

|  |  |  |  |
| --- | --- | --- | --- |
| MUX條件 | | | MUX output |
| one\_d == month\_ctl\_one | ten\_d == month\_ctl\_ten | one\_d == 4'd9 | ten\_d |
| 1 | 1 | X | 4'd0 |
| X | 0 | 1 | tmp\_ten\_d |
| 0 | X | 1 | tmp\_ten\_d |
| default | | | ten\_d |

* 月的個位數部分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MUX條件 | | | | MUX output |
| one\_d == month\_ctl\_one | ten\_d == month\_ctl\_ten | one\_m == 4'd2 &&  ten\_m == 4'd1 | one\_d == 4'd9 | one\_m |
| 1 | 1 | 1 | X | 4'd1 |
| 1 | 1 | 0 | 1 | 4'd0 |
| 1 | 1 | 0 | 1 | 4'd0 |
| 1 | 1 | 0 | 0 | tmp\_one\_m |
| default | | | | one\_m |

* 月的十位數部分
  + 十位數部分多加了一個month作為整個day的output，其作用為判斷year要不要進位。
  + 若為12月且過了一個月，ten\_m為0，month為1
  + 否則若為9月且過了一個月，ten\_m為1，month為0
  + 其餘狀況ten\_m維持不變，month為0

year：

* 年的個位數部分

|  |  |  |
| --- | --- | --- |
| MUX條件 | | MUX output |
| one\_y == 4'd9 | month | one\_y |
| 0 | 1 | tmp\_one\_y |
| 1 | 1 | 4'd0 |
| X | 0 | one\_y |

* 年的十位數部分

|  |  |  |  |
| --- | --- | --- | --- |
| MUX條件 | | | MUX output |
| ten\_y == 4'd9 | one\_y == 4'd9 | month | ten\_y |
| 1 | 1 | 1 | 4'd0 |
| 0 | 1 | 1 | tmp\_ten\_y |
| X | X | 0 | ten\_y |

年的部份較簡單，只要藉由month判斷是否進位即可，其餘為BCD counter的應用。

* I/O pin

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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] |
| VOC | W7 | W6 | U8 | V8 | U5 | V5 | U7 | V7 |
| I/O | d[3] | d[2] | d[1] | d[0] | clk | rst\_n | switch |  |
| VOC | W4 | V4 | U4 | U2 | W5 | V17 | V16 |  |

**Lab6\_3**

**Design Specification**

* Input: clk,

rst\_n,

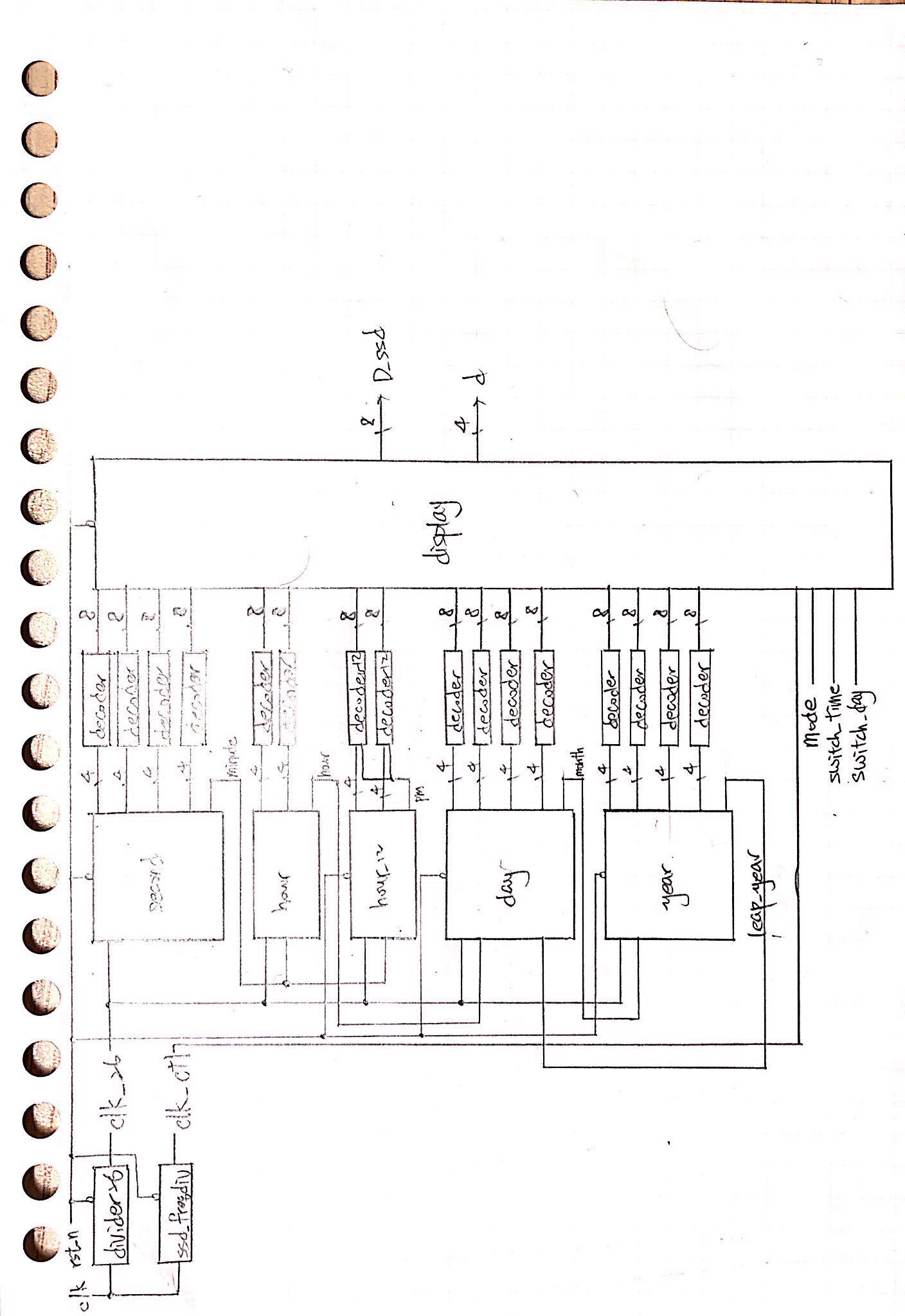
switch\_day // 切換年與月日

switch\_time // 切換秒與時分

mode // 切換時制

Output: D\_ssd[7:0], d[3:0], leap\_year

* Draw the block diagram of the design.



leap\_year

**Design Implementation**

* 第三題為第一題與第二題的結合，其中有二差別。一為日期需要利用小時去判斷是否加一，我將兩個之間利用hour連結作為hour的output與day的input，藉此作為判斷依據。另一差別為第三題的年為2000~2200，且需判斷閏年。以下將說明我的作法
* Year

Input：clk, rst\_n, month

Ouput：one\_y, ten\_y, hun\_y, thu\_y, leap\_year

* 其中年的千為數恆為2，百位數、十位數、個位數為BCD counter，跟第二題部分相似，並在2200年時回到2000年重新計算。
* Leap\_year作為閏年的判斷。其中我設了一個11 bits 的變數leap\_cnt作為計數器，隨著年份增加而增加(即為年的binary形式)。判斷閏年的條件為

**leap\_year = leap\_cnt % 4 == 0 && (leap\_cnt % 400 == 0 || leap\_cnt % 100 != 0)**

利用and與or的性質，加上很厲害的邏輯，就可以利用一條式子就判斷是否為閏年。

* I/O pin

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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] |
| VOC | W7 | W6 | U8 | V8 | U5 | V5 | U7 | V7 |
| I/O | d[3] | d[2] | d[1] | d[0] | clk | rst\_n | leap\_year | mode |
| VOC | W4 | V4 | U4 | U2 | W5 | V17 | U16 | V16 |
| I/O | switch\_time | switch\_day |  |  |  |  |  |  |
| VOC | W16 | W17 |  |  |  |  |  |  |

**Discussion**

　　這次的lab都是divider、BCD counter、decoder、display由組成。其中BCD counter的MUX判斷為三題中的主要差異。我認為除了MUX的條件判斷為本次實驗重點外，還有另外一個重點為進位的條件判斷。

　　在第一題的時候，我一開始將分與秒分開做，但發現會遇到無法處理兩者的連結，所以之後將二者放在同一個module。但如果將時分秒都放在一起感覺很複雜，很難debug，所以之後又想到可以藉由output一個進位的enable來作為判斷，因此在hour的時候就分開做了。

　　在做第二題時，原本我並沒有加入判斷每個月天數的變數，因此在寫日期進位條件的時候遇到了很大的麻煩，很容易出錯。因此我加入了month\_ctl\_one與month\_ctl\_ten幫助判斷是否進位，有效解決了判斷式複雜容易出錯的問題。另外，我原本想利用上題的方式，利用enable判斷月份是否加一，但卻發生了月份改變會慢一個clk的問題。就其原因為一開始的設計為在月份要改變的當下enable才會為1，因此會到下個clk時月份才改變。因此最後還是利用原本的將月份與日期做在同一個module，並利用month\_ctl\_one與month\_ctl\_ten判斷月份是否改變。

　　最後是第三題閏年的判斷。其判斷方式有很多種，例如先判斷是否為4的倍數，再判斷是否為100的倍數，最後判斷是否為400的倍數；藉由三次判斷得出答案。此方法雖然很容易理解，但需要經過三次判斷，較沒效率。因此我利用邏輯結構，藉由一個條件即可判斷結果，雖然較不直觀，但卻可以練習邏輯判斷，真的很方便。另外一個差異是，是否需要多設置一個變數紀錄年份用來判斷。我認為多加一個變數會讓程式看得比較清楚，缺點是須注意其值什麼時候要改變，整個程式也會比較冗長。

**Conclusion**

這次的實驗著重於細節的判斷，若沒注意年月日時分秒的進位，就很容易出問題，因此需要很仔細的思考。另外一個細節是年月日時分秒的連結，須注意是否會快或慢一個clock，需要清楚自己enable的設計以免出錯。