Lab6 report

107061218 謝霖泳

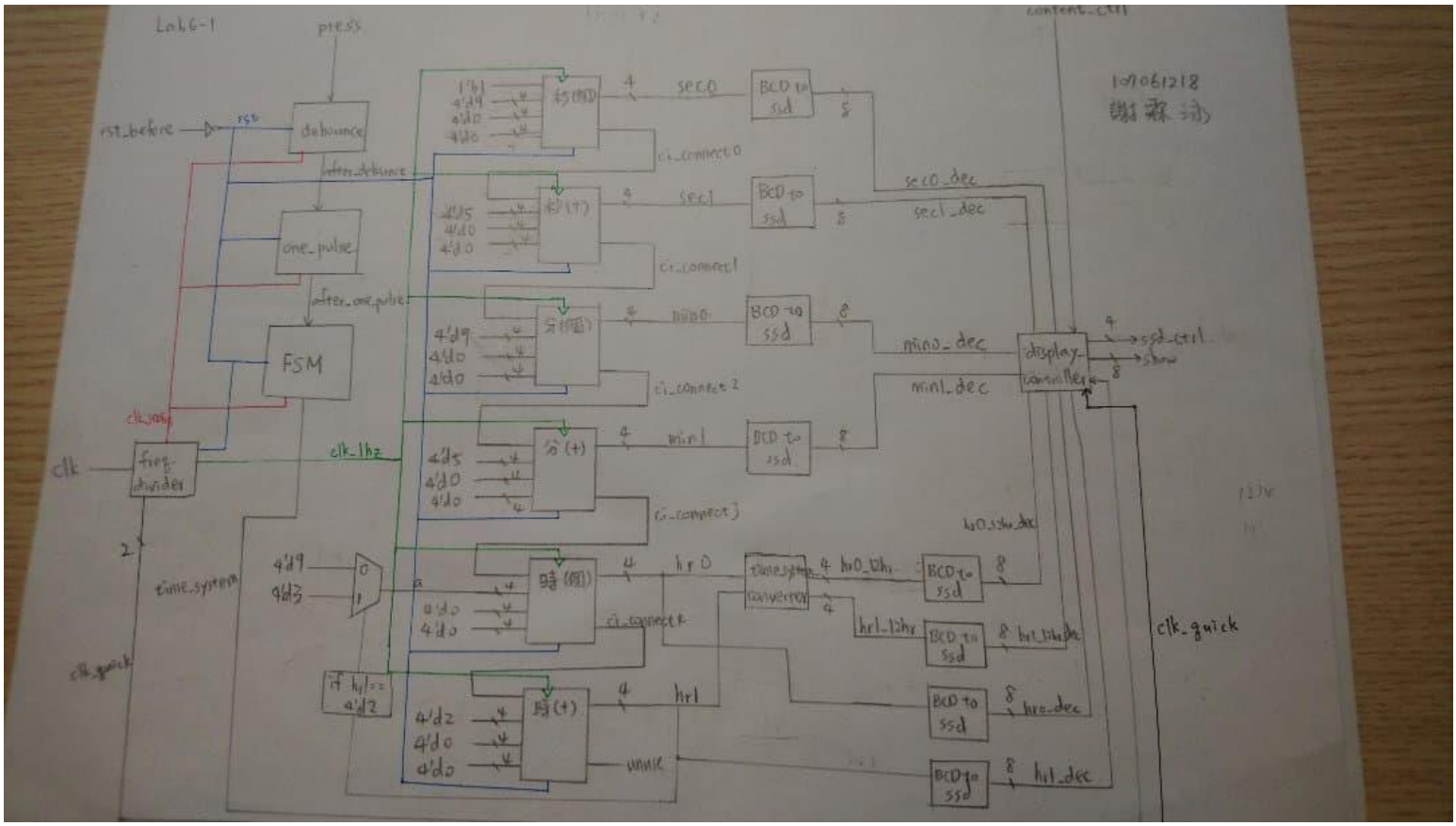
1.

- Design Specification

(1) Top module

Input: press, rst_before, clk, content_ctrl

Output: [3:0] ssd_ctrl, [6:0] show, am_or_not



(2) Frequency divider

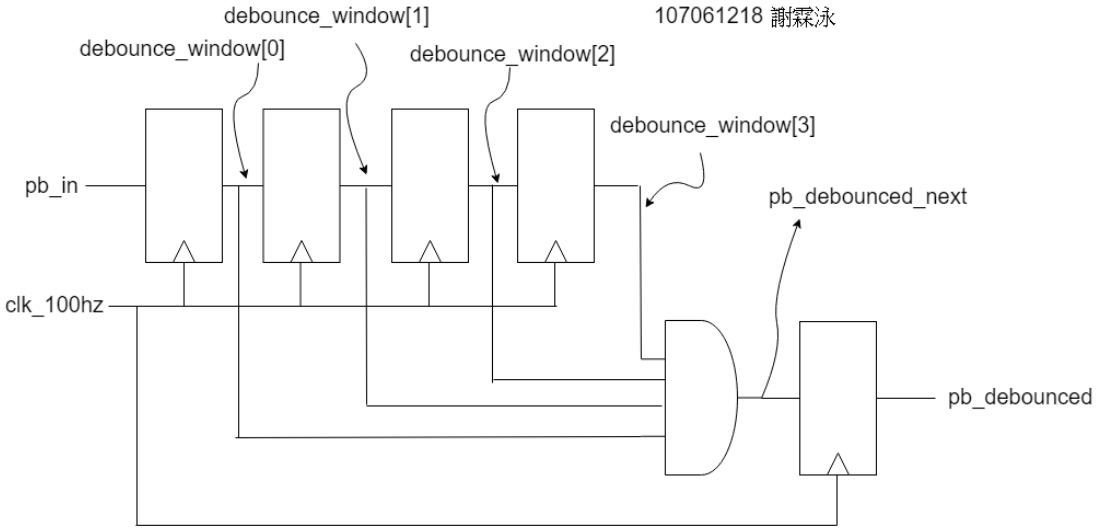
Input: clk_in, rst

Output: clk_out_1hz, clk_out_100hz, [1:0] clk_for_ssd

(3) Debounce

Input: clk_100hz, rst, pb_in

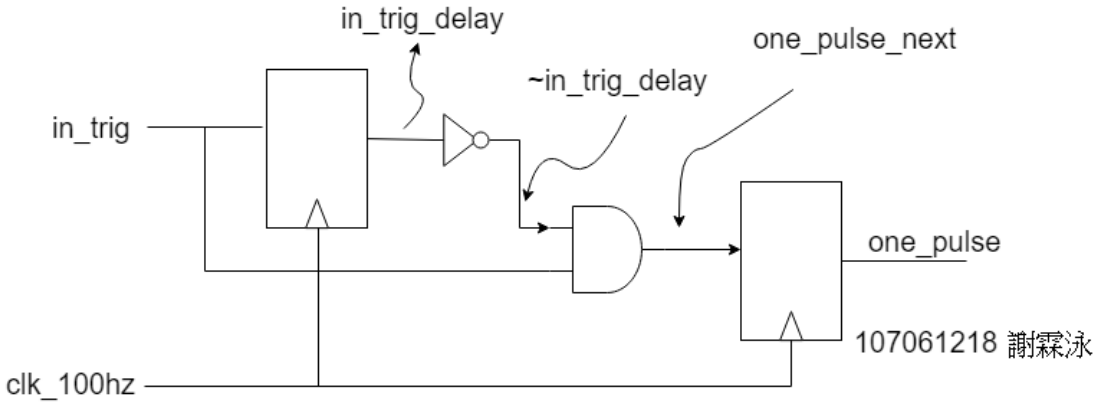
Output: pb_debounced



(4) One pulse

Input: in_trig, clk_100hz, rst

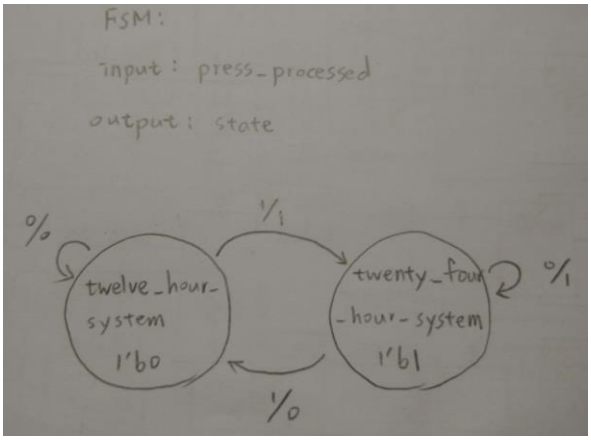
Output: out_pulse



(5) FSM

Input: clk_100hz, rst, pb_processed

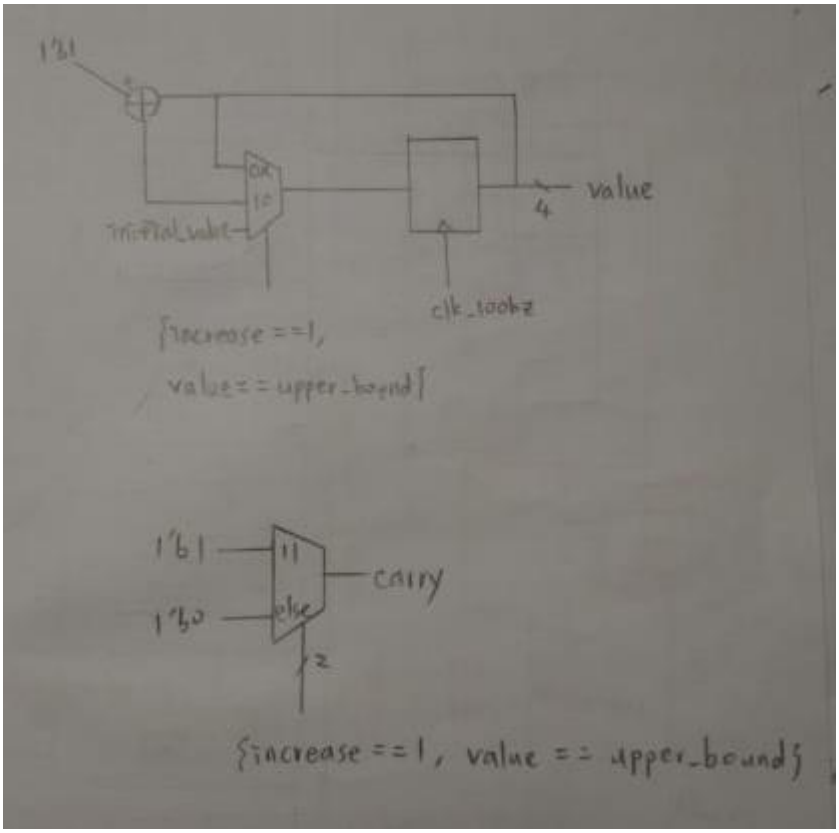
Output: state



(6) Single digit up counter

Input: increase, clk_1hz, rst, [3:0] upper_bound, [3:0] initial_value, [3:0] rst_value

Output: [3:0] value, carry



(7) BCD to 7 segment decoder

Input: i

Output: D_ssd

(8) Time system converter

Input: [3:0] hr0, [3:0] hr1

Output: [3:0] hr0_12hr, [3:0] hr1_12hr, am_or_not_temp

24 小時制	12 小時制	am_or_not_temp	24 小時制	12 小時制	am_or_not_temp
00	12	1'b1	12	12	1'b0
01	01	1'b1	13	01	1'b0
02	02	1'b1	14	02	1'b0
03	03	1'b1	15	03	1'b0
04	04	1'b1	16	04	1'b0

05	05	1'b1	17	05	1'b0
06	06	1'b1	18	06	1'b0
07	07	1'b1	19	07	1'b0
08	08	1'b1	20	08	1'b0
09	09	1'b1	21	09	1'b0
10	10	1'b1	22	10	1'b0
11	11	1'b1	23	11	1'b0

(9) Display controller

Input: [6:0] sec0_dec, [6:0] sec1_dec, [6:0] min0_dec, [6:0] min1_dec, [6:0] hr0_dec, [6:0] hr1_dec, [6:0] hr0_12hr_dec, [6:0] hr1_12hr_dec, [1:0] clk_quick, content, time_system

Output: [3:0] ssd_ctrl, [6:0] show

- Design Implementation

這個 design 總共有 4 個輸入，分別為切換時制的按鈕 press、用來重置裝置的按鈕 rst、晶體振動時脈 clk 與控制要顯示時分還是顯示秒的 switch 輸入名為 content_ctrl。

將切換時制的按鈕輸入名為 press，傳入 debounce 與 one pulse 後進入 FSM 中，FSM 會傳出 1-bit 的訊號。若為 0 表示輸出 12 小時制，若為 1 表示輸出 24 小時制。

接下來進入這個 design 的核心功能，就是 single digit 的 up counter，由 top module 的 block diagram 可發現，除了 1 赫茲的 clock 與 reset 訊號之外，每個 single digit up couonter 都有左側 4 個輸入與右側 2 個輸出。第一個輸入為 increase，就是來自上一位的 carry，告訴我這個 clock trigger 我有沒有需要加 1；第二個輸入為 upper bound，這是引用微積分裡的名詞，就是「上界」的意思，告訴我加到多少是我的極限；第三個輸入為 initial value，告訴我當我到達我的上屆之後，如果還需要繼續上數，我要變成多少；最後一個輸入為 reset value，顧名思義，當我偵測到 low active 的 reset 訊號時，我應該重置為多少。

值得一提的是，小時個位數 up counter 的 upper bound。因為小時個位數的上界並非定值，需要由小時的十位數做判斷，意即當我小時的十位數為 0 或 1 時，個位數的上屆為 9，也就是我們可以數到……18、19 才跳 20，但當小時的十位數已經為 2 時，也就是已經到了 20 幾點，這時候小時個位數的 upper bound 就只能為 3，因為一天最多只有 23 小時。總的來

說，當小時的十位數為 2，此 upper bound 為 3，其他情況下則為 9。

至於 BCD to 7 segment decoder 與 frequency divider 的部分，已於之前 lab 出現極多次，因此不再贅述。

因為我的 design 算出來的結果為 24 小時制，因此，要多一個時制的轉換器換算出 12 小時制所對應的結果，因此，我多寫了一個 time system converter 的 module，該 module 內部為 MUX，依照所傳入的小時十位數與小時個位數作為 selection，詳細換算在 Design Implementation 中可見，輸出 12 小時制的對應結果與 am_or_not_temp 訊號，至於 am_or_not 訊號為何將在下段仔細說明。

我的 12 小時制設定是 am 的時候，七段顯示器的四個點就會亮，所以，在 time system converter 中，輸出一個 am_or_not_temp，在 top module 中判斷，若現在為 24 小時制，則不用顯示任何小數點，因此輸出 am_or_not 為 1'b0，若現在顯示為 12 小時制，才有必要判斷現在的時間是 am 還是 pm，因此輸出的 am_or_not 即為 am_or_not_temp，也就是在 time system converter 中判斷出來的結果。

至於最後的 display controller 的部分，內部亦為 MUX 的結構，若現在欲輸出的內容為秒，也就是 switch 是被撥起來的狀態下，那就顯示秒，反之，就顯示時分，當然，此部分的顯示必須配合除頻器得到的 2-bit 快速變化的頻率以達成視覺暫留之效果，亦於先前的 lab 出現多次，於此不再贅述。

- Discussion

  可以觀察到，時間正常的顯示在七段顯示器上。不過為了 demo 方便，我將理論上應為 1Hz clock 的頻率接為 100Hz，不然 demo 就要跑到天荒地老才能看到位數進位時是否出錯。

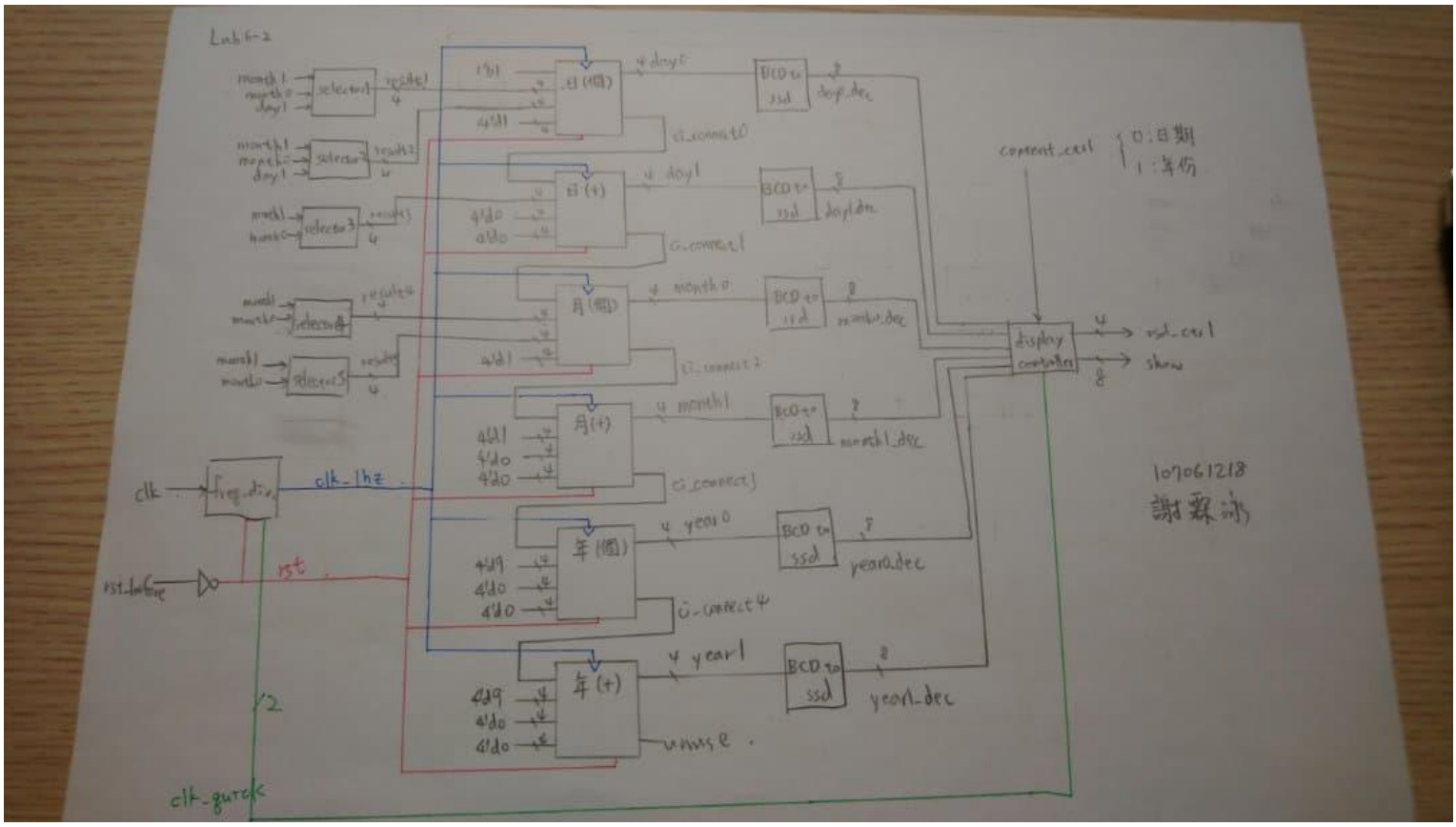
2.

- Design Specification

(1) Top module

Input: clk, rst_before, content_ctrl

Output: [7:0] show, [3:0] ssd_ctrl



(2) Single digit up counter

與上題相同。

(3) Frequency

與上題相同。

(4) Selectors

	input	output
selector1	month1, month0, day1	result1
selector2	month1, month0, day1	result2
selector3	month1, month0	result3
selector4	month1, month0	result4
selector5	month1, month0	result5

(5) Display controller

Input: [7:0] day0_dec, [7:0] day1_dec, [7:0] month0_dec, [7:0] month1_dec, [7:0] year0_dec, [7:0] year0_dec

Output: [3:0] ssd_ctrl, [7:0] show

- Design Implementation

  此題的基本概念與第一題相同，只是運算變成了日月年。在上一題的小時個位數 upper bound 的例子中，我們可以發現，有些位數的 upper bound 抑或 initial value 並非固定不變，而此現象在本題中將會更加顯著，比如日的十位數 upper bound 會與現在幾月相關，如果是 2 月，那 upper bound 就是 2，其他月則為 3。諸如此類需要先經過其他條件判斷才能輸入 single digit up counter 的，在本題中總共有 5 處，分別為日個位數的 upper bound 與 initial value、日的十位數的 upper bound、月的個位數的 upper bound 與 initial value。因此，我分別做了 5 個 module 來處理這些不定的值，分別名為 selector1~selector5。

  再來的動作與上題相同，也就是將各個位數 decode 完的結果，傳入 display controller 中，並配合我現在的 content_ctrl，也就是某個 switch 輸入，撥起來則顯示年，放下來則顯示月日。

- Discussion

  可以看到這個日月年的裝置正常運作，但是這議題還沒考慮閏年的情況，會在下一題做更深入的討論。

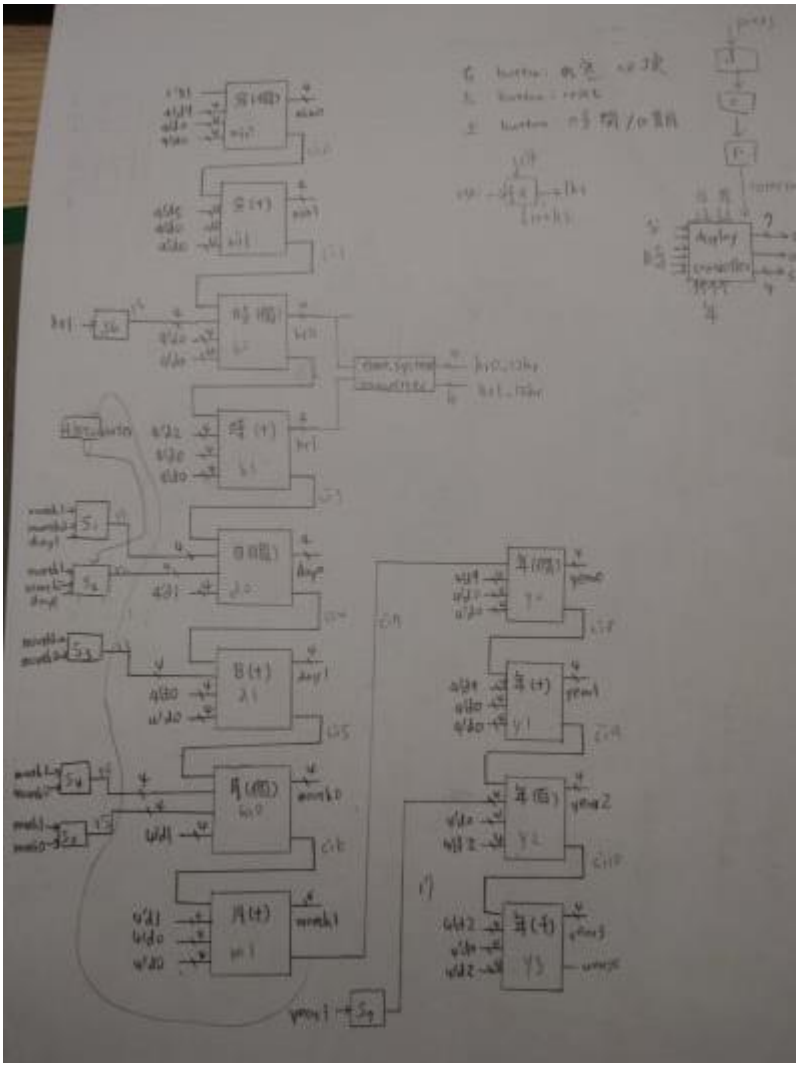
3.

- Design Specification

(1) Top module

Input: clk, rst_before, content_ctrl

Output: [7:0] show, [3:0] ssd_ctrl



(2) Selectors

	input	output
selector1	month1, month0, day1, q	result1
selector2	month1, month0, day1	result2
selector3	month1, month0	result3
selector4	month1, month0	result4
selector5	month1, month0	result5
selector6	hr1	result6
selector7	year3	result7

- Design Implementation

這議題是前兩個小題的整合，要把分時日月年全部都串在一起，不過還是一樣的觀念，一級一級接好就沒事了。此外，因為要多考慮閏年的問題，所以月份個位數的 upper bound 會多出一個選擇依據，就是我多另一個

2-bit binary up counter，每當我的年增加 1 的時候，這個 2-bit 值就加 1，每當這個 up counter 出來的結果 q 為 2'b00 時，就代表今年是閏年，所以 2 月的個位數 upper bound 就會為 4'd9。

- Discussion

  這個裝置可以正常顯示出閏年結果，但是還沒有考慮到每 100 年不會閏年的這個細節，之後可以再多加一層 MUX 去考慮。

4. conclusion

這次 demo 忘記做一個快一點的 clock，導致 demo 花費了比較多時間，下次會嘗試做出可多段變速的 design，以節省大家寶貴的時間。