**Lab 6 110060027朱豐蔚**

1. **exp\_1 (electronic clock)**

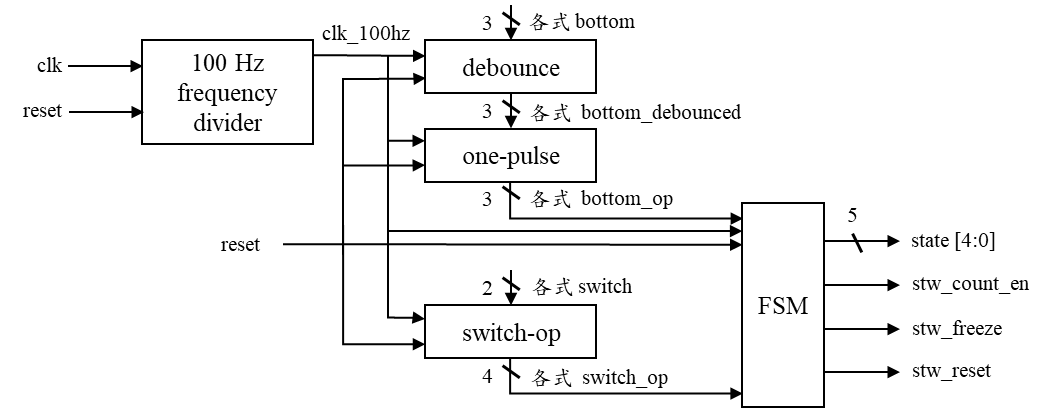
**Design Specification**

Input: buttom\_left, buttom\_mid, buttom\_right, clk, reset, switch\_alarm, switch\_mode\_left,

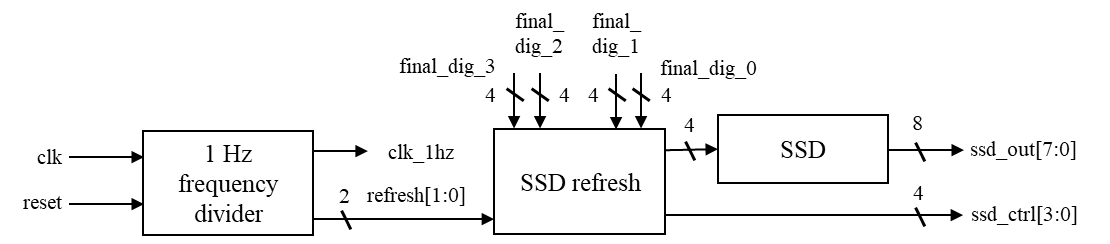
switch\_mode\_right.

Output: stw\_work, led\_alarm[7:0], ssd\_ctrl[3:0], ssd\_out[7:0], led\_display[2:0], led\_set [3:0].

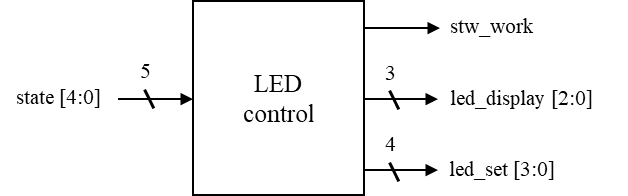
**Design Implementation**

 Block diagram:

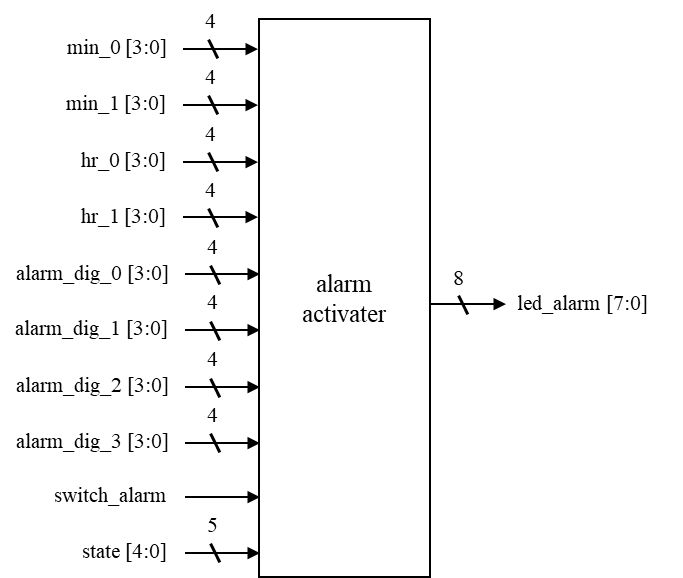
|  |  |
| --- | --- |
| 100 Hz  frequency divider | 將1億hz 的clk，透過設立500K的counter將其轉成100 hz的clk\_100hz訊號，以供後續debounce、one-pulse使用 |
| debounce | 使用windows的設計處理雜訊，以防止按鈕失靈或誤觸 |
| one-pulse | 將debounce訊號轉成一clk週期的脈衝訊號，以提供給FSM使用 |
| switch-op | 將switch的turn on、turn down以類似one-pulse的延誤設計，將其轉成一clk週期的脈衝訊號，以提供給FSM使用 |
| FSM | 控制所有state彼此之間的切換，以輸出state狀態給後續進行各功能的切分，並輸出有關與控制stopwatch的訊號stw\_count\_en、stw\_freeze、stw\_reset |



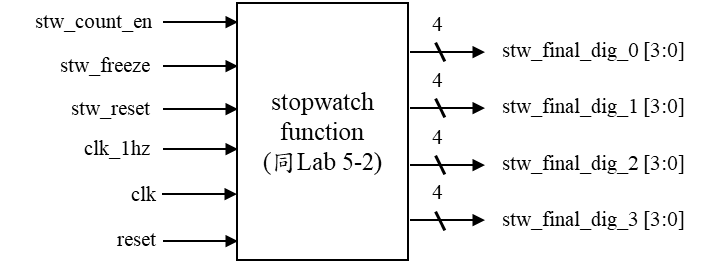
|  |  |
| --- | --- |
| 1 Hz  frequency divider | 將1億hz 的clk，透過設立50M的counter將其轉成1hz的clk\_1hz訊號，並取其counter的第17、16位元作為refresh [1:0]，以提供SSD四個數字的輪轉更新用 |
| SSD refresh | 透過refresh訊號，輪替輸出四個輸入數字與控制SSD顯示哪個位置的ssd\_ctrl訊號 |
| SSD | 將BCD轉成七段顯示器的輸出，使其能輸出供肉眼辨別的羅馬數字 |



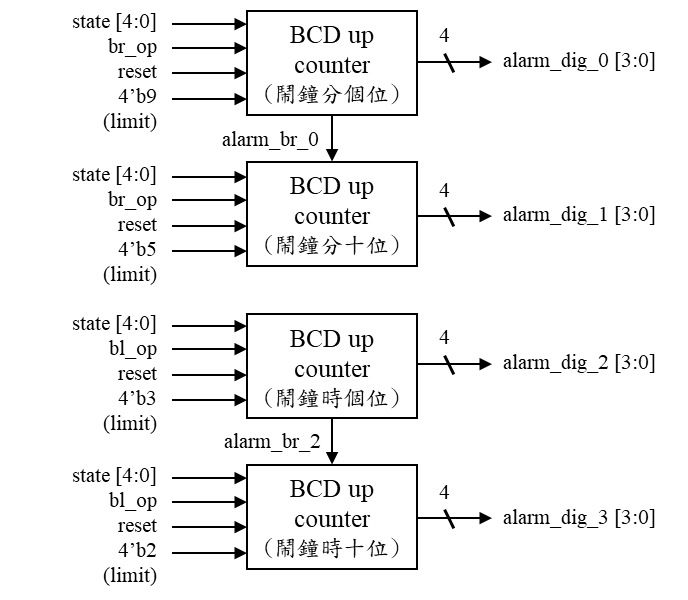
|  |  |
| --- | --- |
| LED control | 依據state的狀態，控制該亮那些燈(不包含鬧鐘的燈)，在設定模式下(state[4:3] == 2’b01)，就讓led\_set根據設定的項目分別亮起；在顯示模式下(state[4:3] == 2’b00)，就讓led\_display根據顯示的項目分別亮起；在stopwatch模式下(state[4:3] == 2’b10)，就讓stw\_work亮起 |



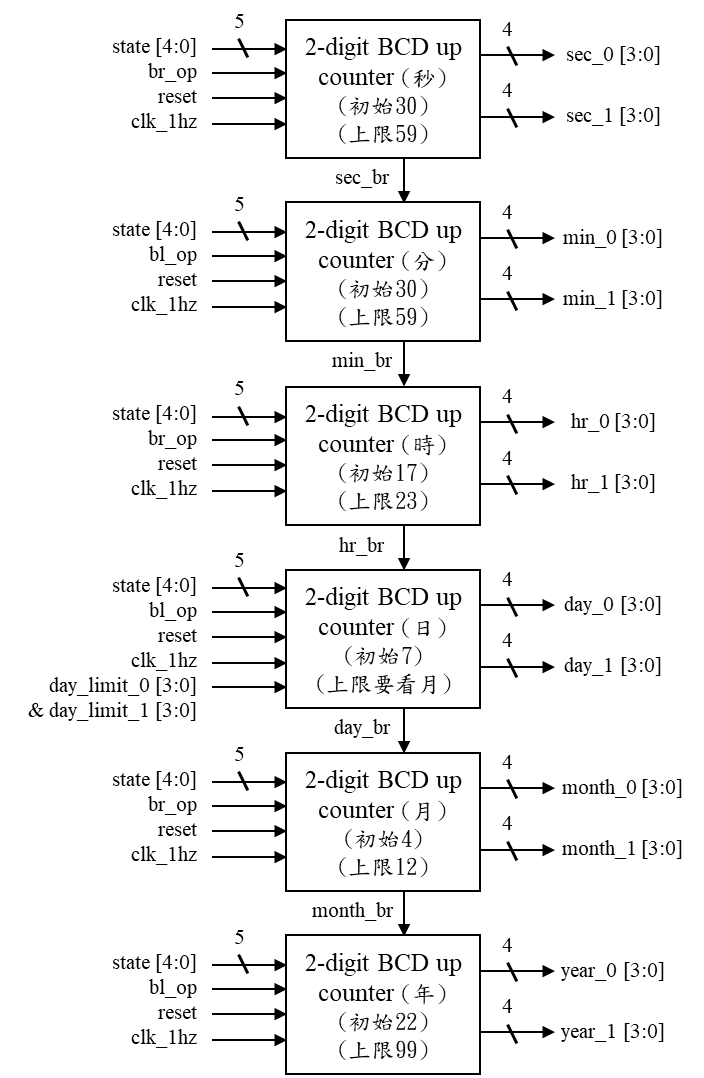
|  |  |
| --- | --- |
| alarm activater | 若小時的十位數、個位數與鬧鐘小時的十位數、個位數相同，且分鐘的十位數、個位數與鬧鐘分鐘的十位數、個位數相同，且switch\_alarm是打開的狀態，就使鬧鐘的燈亮起  注意：在stopwatch模式下(state[4:3] == 2’b10)，鬧鐘功能要關閉 |



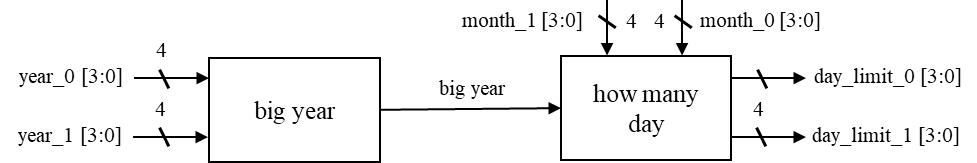
|  |  |
| --- | --- |
| Stopwatch function  (同Lab 5-2) | 同Lab 5-2，設立4個counter處理分鐘、秒的十位數、個位數，並另外用4個flip flops處理freeze功能，以達到stopwatch的目的 |



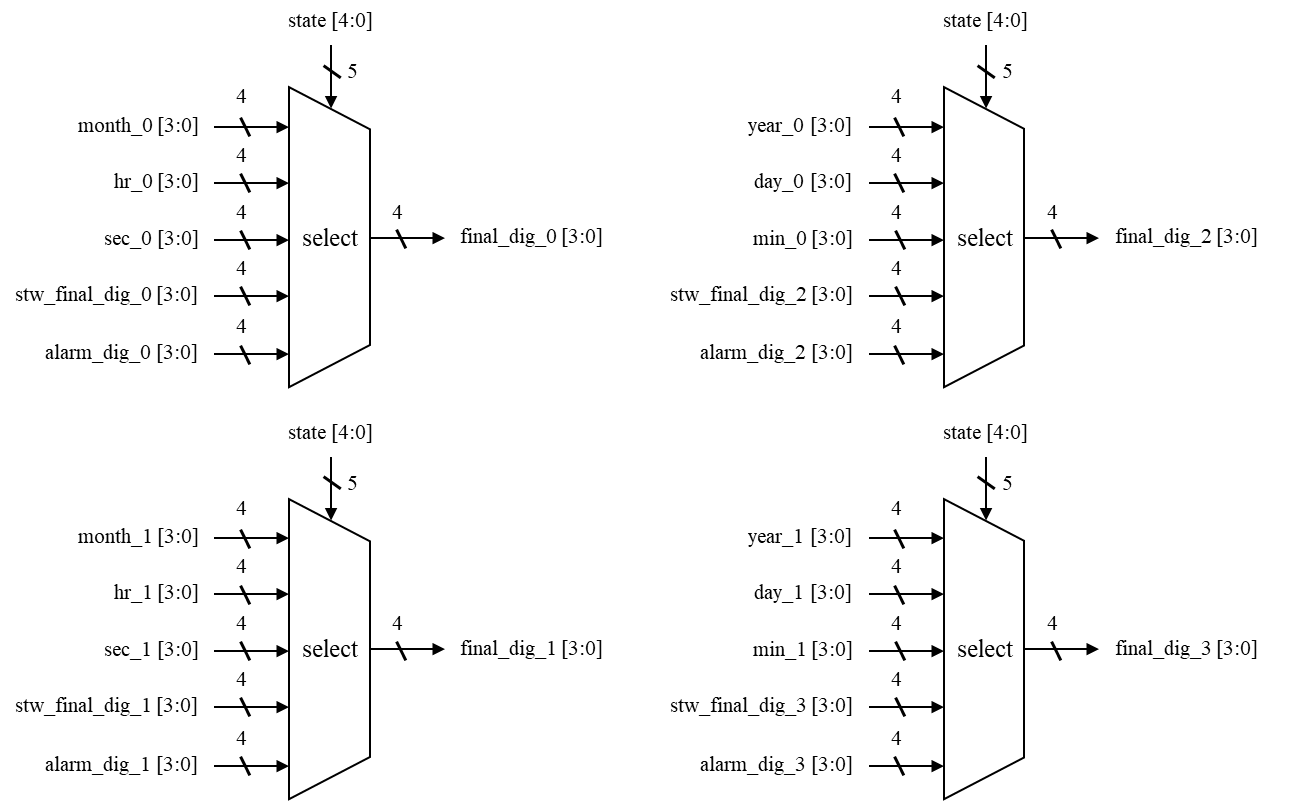
|  |  |
| --- | --- |
| BCD up counter | 在counter中，需要定義初始值、上限、enable訊號、與進位訊號，當在鬧鐘的設定模式(state[4:0] == 5’b01000)下，counter才會被enable，此時能夠透過右邊按鍵去控制鬧鐘的分鐘數、用左邊按鍵去控制鬧鐘的小時數  注意：鬧鐘的初始值為17：31，故要讓4位數字的初始值由上至下分別設定為1、3、7、1。上限值如圖中所示 |



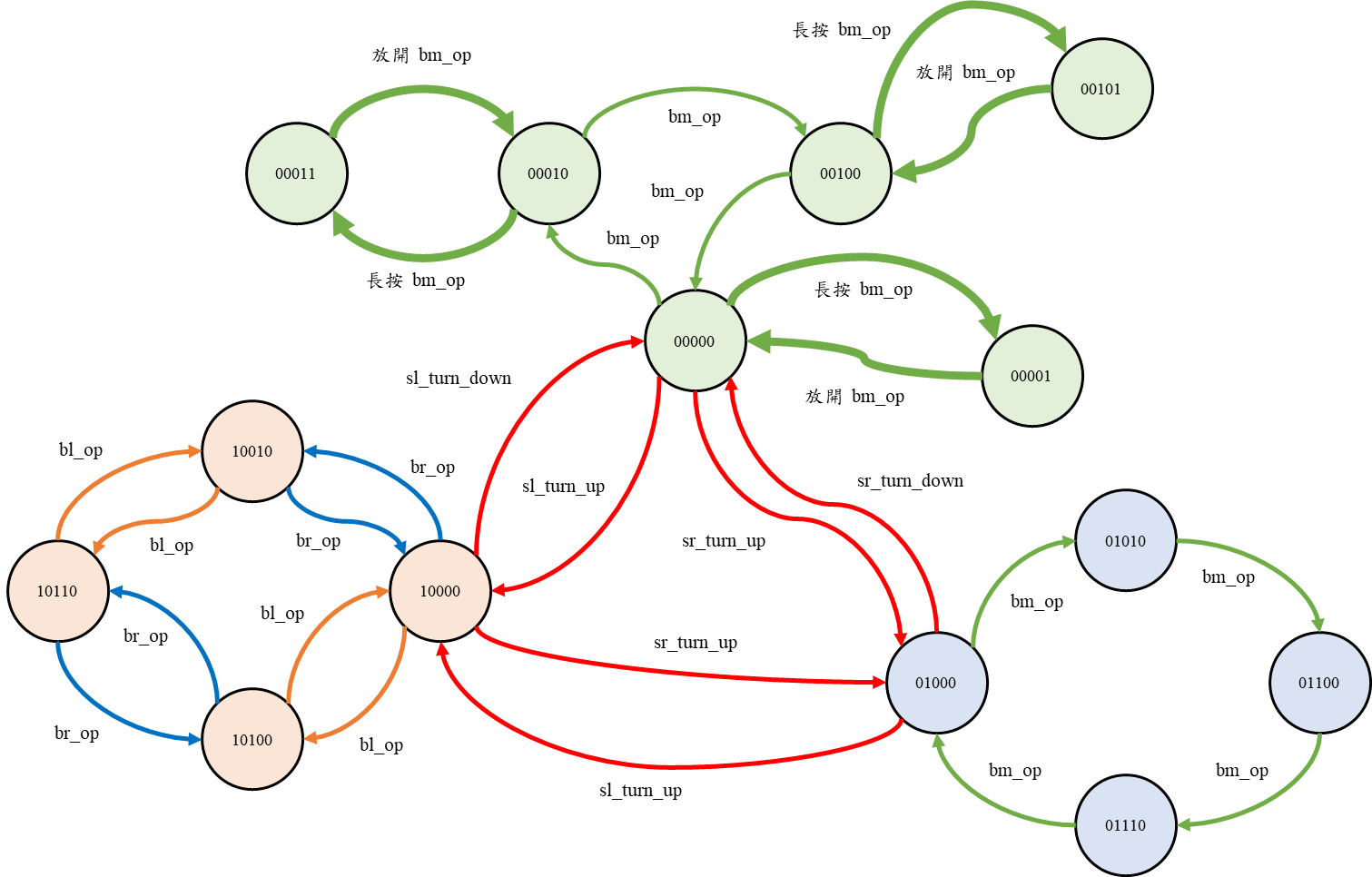
|  |  |
| --- | --- |
| 2-digit BCD  up counter | 類似上述counter，但在此使用2 digit的版本，以方便進行初始值、上限值的設定，當在鬧鐘的顯示模式(state[4:3] == 2’b00)下，此時clk訊號設定為clk\_1hz，而在鬧鐘的設定模式(state[4:3] == 2’b01)下，此時clk訊號設定為右邊按鍵與左邊按鍵，即在設定模式下，時間是不會繼續數的，此點符合日常的使用習慣  注意：時鐘的初始值為22年04月07日17時30分30秒，故要讓6個counter的初始值由上至下分別設定為30、30、17、07、04、22。  上限值除了日期的上限會隨月份、閏年有所改變(由後續模組進行設定)，其餘皆與圖中所示相同 |



|  |  |
| --- | --- |
| big year | 觀察年的個位數與十位數的關係，若十位數的最後一位元(year\_1[0])是1，代表十位數為奇數，此時個位數為2或6 (year\_0[1:0] == 2’b10 )即滿足閏年條件；若十位數的最後一位元(year\_1[0])是0，代表十位數為偶數，此時個位數為0或4或8 (year\_0[1:0] == 2’b00 )即滿足閏年條件，在閏年時，使輸出big\_year = 1  注意：在本次Lab設計中，並未考慮逢百不閏、逢400又閏的情況 |
| how many day | 透過case與閏年與否，判斷當月有幾天，作為天數的上限控制 |



|  |  |
| --- | --- |
| select | 設計一個5選1的select模組，select的訊號由state決定，當為顯示年月或設定年月模式時(state[4:0] == 5’b00000 or 5’b01010)，使最終輸出為年與月的counter訊號；當為顯示日時或設定日時模式時(state[4:0] == 5’b00010 or 5’b01110)，使最終輸出為日與時的counter訊號；當為顯示分秒或設定分秒模式時(state[4:0] == 5’b00100 or 5’b01100)，使最終輸出為分與秒的counter訊號；當為設定鬧鐘模式時(state[4:0] == 5’b01000)，使最終輸出為鬧鐘的counter訊號；當為stopwatch模式時(state[4:3] == 2’b10)，使最終輸出為stopwatch的counter訊號 |

 Design flow:

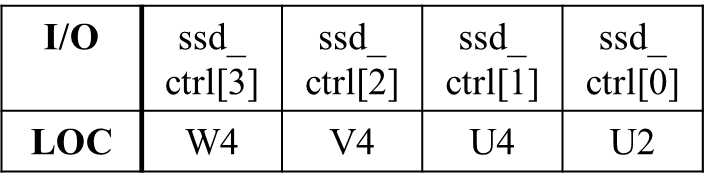
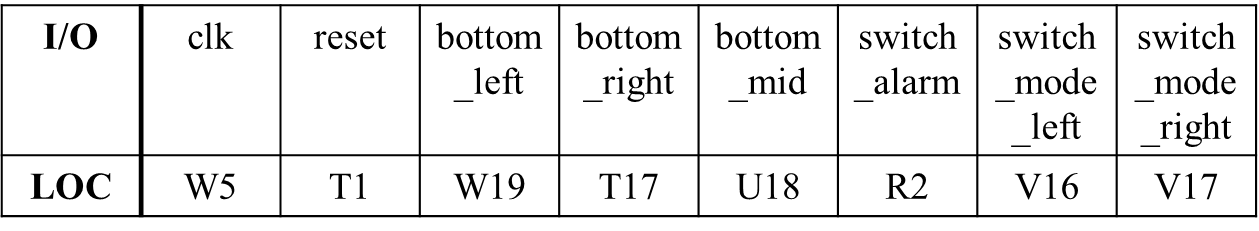
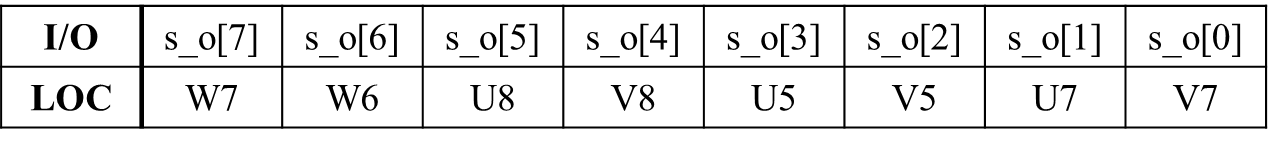
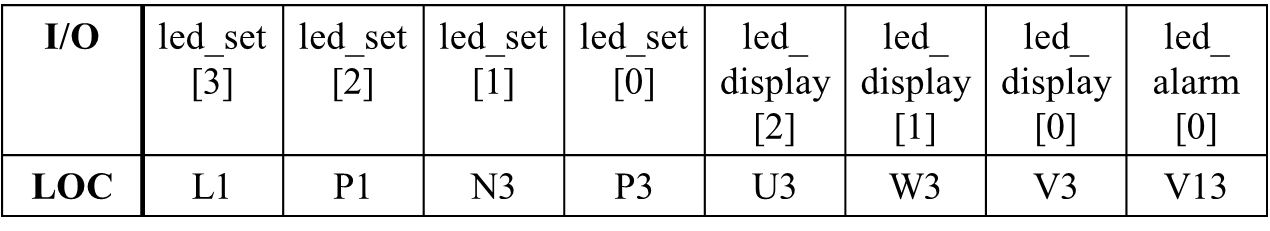
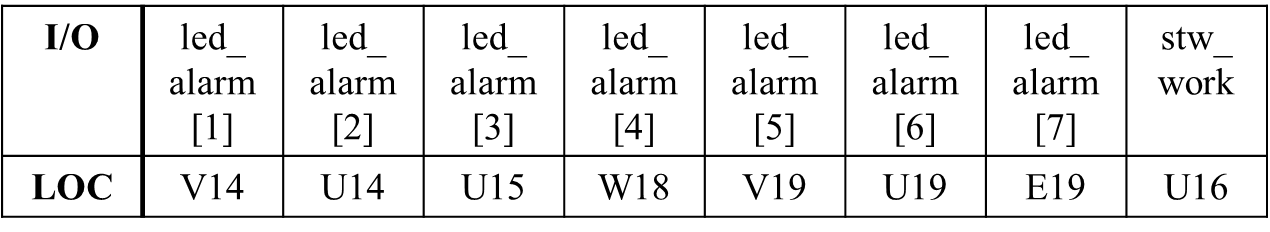
考慮此題所需要的state，如上圖設計，淺綠色背景的state為顯示模式，其中以三個state(5’b00000、5’b00010、5’b00100)分別作為顯示年月、顯示日時、顯示分秒的主state，彼此透過中央按鍵切換，而我另外創立三個副state(5’b00001、5’b00011、5’b00101)作為各自的長按顯示鬧鐘功能。在長按顯示鬧鐘、放開又回到原狀態的功能上我下了很大的心思，我設想了兩種方式，一是如上述再多設立三個state，二是設立record變數去紀錄長按前的模式，使放開後能透過if來決定state的變化方向。在本次Lab中，我並未嘗試過第二種寫法，有待未來嘗試。

回到FSM上，淺藍色背景的state為設定模式，其中以四個state(5’b01000、5’b01010、5’b01100、5’b01110)分別作為設定鬧鐘、設定年月、設定日時、設定分秒的state，彼此透過中央按鍵切換。

最後，淺橘色背景的state為stopwatch模式，同Lab 5-2的FSM，此功能需要4個state與右邊按鍵、左邊按鍵鍵來分別控制是否數、是否lap的切換，而長按左邊按鍵要有reset的功能，如前述module中FSM的描述，我透過stw\_count\_en、stw\_freeze、stw\_reset的設計來完成stopwatch功能。

而顯示模式、設定模式、stopwatch模式彼此的切換則是使用switch\_op(分為turn on 與turn down)的訊號來完成。

而除了FSM以外的其餘設計細節，皆於上述block diagram的module介紹中提及。

I/O pins assignment :

**Discussion:**

這次的Lab對我來說是一次很大的挑戰，也是第一次讓我們著手進行那麼大量的模組牽線與撰寫，除了考驗我們對每個模組的使用掌握度外，更考驗著我們對於那麼多模組彼此的串聯使否邏輯清晰與了解透徹，才能達成這次的實驗。

在設計中，有兩個點是我思考最久的，第一是2 digit counter的初始值、上限值撰寫問題、第二是閏年與天數上限的問題。

在原先，其實我是使用2個單位元的BCD counter 作為2 digit counter的撰寫方式，但由於在日與月的撰寫中，必須考慮到沒有0月或0日存在，故需要在進位後的值上做考慮，除此之外，更要考量reset後的初始值，此兩者彼此更可能會互相干擾，於是在後期，我便改為直接設立2 digit counter來統一化各式counter以解決counter互相不統一初始值、上限值、進位後值的問題，而這樣的寫法也幫助了我在進行進位的寫法上更為簡潔與快速。這次的經驗也讓我了解到了統整化的重要性，透過小模組更進一步的推疊，就能夠形成大模組，以應付更複雜、更多樣的各種情況。

而關於閏年與天數上限的問題，我在如何判斷閏年上設想了許久，在原先，我本是打算使用case硬處理各式年份對應到的閏年情況，但後來，透過思考後，我認為4年一閏的規律應該會與2的平方有關係，才進而發現十位數與各位數與閏年之間的關係，進而解決掉閏年的問題。這次經驗也讓我思考，若今日遇到不是4的規律的問題，例如3次、5次，是否也有通解，但我認為當不是2的次方時，找到其規律的可能性就極為低。

**======================================================================**

**Conclusion**

我認為這次Lab最大的收穫，是有了在很複雜的情況、很多的模組下，能夠控制並串聯各個模組的經驗，我想未來一定會遇到更多同樣複雜，或更複雜的情況，但只要按部就班，先從FSM下手，再分別進行各項功能的撰寫，就能省去許多debug的時間與問題，也能更有計畫性的完成project。

此外，在進行撰寫時所遇到的各種問題，如初始值、上限值，或是閏年的判斷，也都是在訓練我們遇到問題時的解決思考靈活性，這點是最為重要的，在遇到困難時，從各個方面下手，尋找解決方式，我喜歡這樣的挑戰與嘗試，也熱愛於其中。

感謝老師與助教安排這次的Lab，讓我能在經驗與創意上都有明顯的突破，並在完成Lab的剎那收穫滿滿的成就感與感動。

**References**

FSM編撰方式: (from 第六週上課講義)

Freeze功能編撰方式: (from 第五週上課講義)