**Lab 6: Electronic Clock**

1 Implement an electronic clock with the following functions:

雖然這次只有一個問題，但是由多個子問題(模組)構成，過於複雜。因此我想先把每個部份的Design Specification和Design Implementation寫完，方便使用者可以看懂每個細節在做什麼。最後將所有模組(功能)整合在一起，Discussion, Conclusion和Reference最後再一起寫。

* 1. Finish the time display function supporting 24-hour (00-23). The time display can show hour:minute and second, and use one push button to switch the display

模組名: Time\_Mode

**Design Specification**

IO輸出入設定

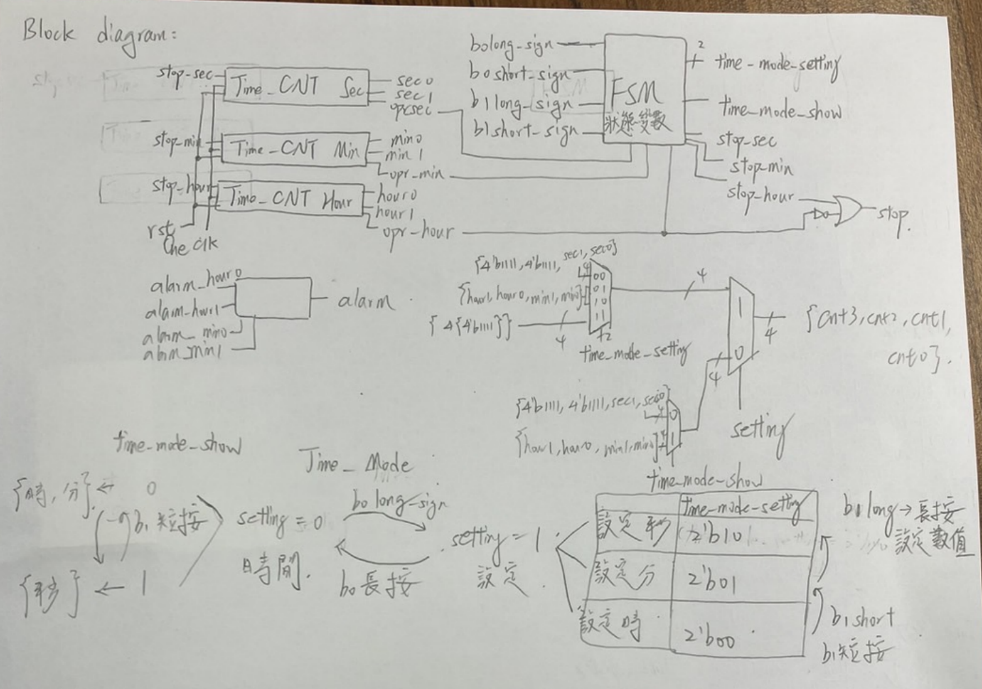
輸入:

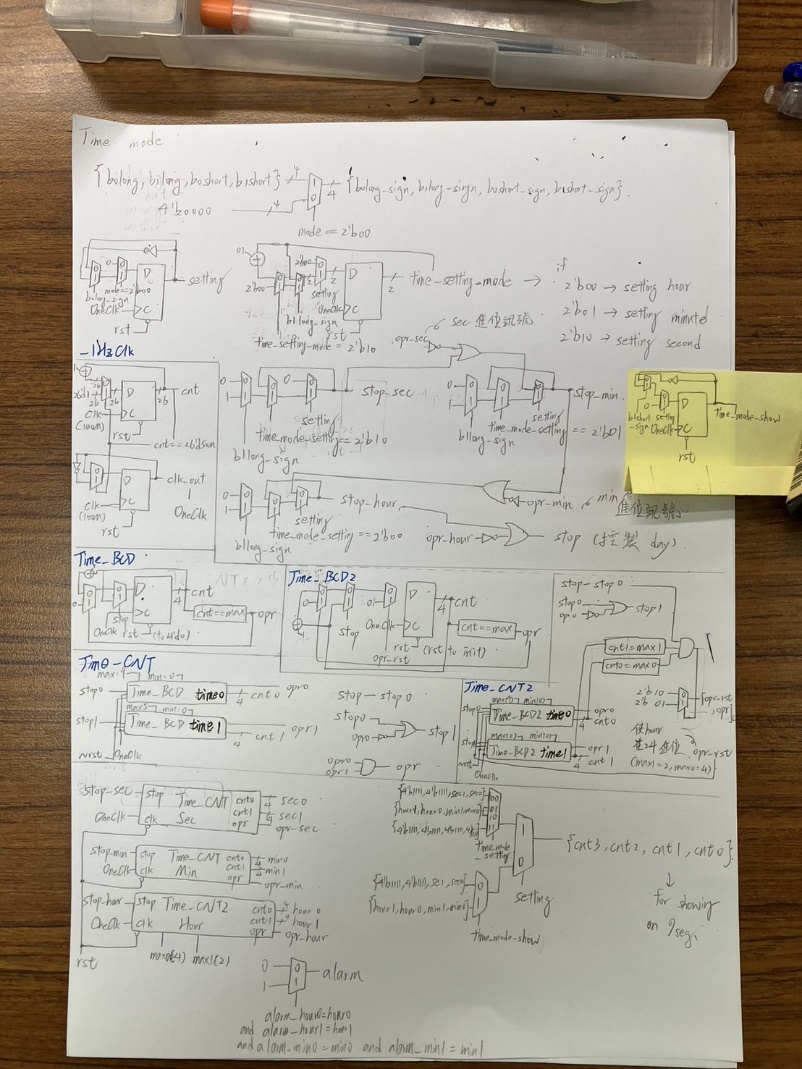
|  |  |  |
| --- | --- | --- |
| clk | 1 bit | 時脈(100M) |
| rst | 1 bit | Reset(外接SW15(R2)) |
| mode | 2 bits | 紀錄現在在哪種功能 |
| b0long | 1 bit | b0(W19)按鈕 長按 |
| b0short | 1 bit | b0(W19)按鈕 短按 |
| b1long | 1 bit | b1(T17)按鈕 長按 |
| b1short | 1 bit | b1(T17)按鈕 短按 |

輸出:

|  |  |  |
| --- | --- | --- |
| cnt3 | 4 bits | 顯示於7Seg左邊數來第一個 |
| cnt2 | 4 bits | 顯示於7Seg左邊數來第二個 |
| cnt1 | 4 bits | 顯示於7Seg左邊數來第三個 |
| cnt0 | 4 bits | 顯示於7Seg左邊數來第四個 |
| setting | 1 bit | LD15(L1)告訴使用者是否在設定狀態 |
| time\_mode\_setting | 2 bits | LD1, 2(E19, U19)告訴使用者現在在設定哪一個位數(時分秒) |
| time\_mode\_show | 1 bit | 告訴使用者現在在顯示(時分or 秒)(LD1 U16) |
| stop | 1 bit | 控制Date功能裡的day的上數 |
| alarm | 1 bit | 當時間和鬧鐘相等時，亮起(LD14)(P1) |
| alarm\_min0 | 4 bits | 鬧鐘所設定時間的分(個位數) |
| alarm\_min1 | 4 bits | 鬧鐘所設定時間的分(十位數) |
| alarm\_hour0 | 4 bits | 鬧鐘所設定時間的時(個位數) |
| alarm\_hour1 | 4 bits | 鬧鐘所設定時間的時(十位數) |

Logic block and state diagram



**Design Implementation**

我的設計是先設定一個狀態變數setting，當setting == 1時為設定時間的狀態。反之，setting == 0為正常手錶的狀態。當setting==1時，有一個狀態變數time\_mode\_setting表示當前設定哪一時間(時分秒)。setting == 0時也有一個time\_mode\_show表示當前顯示的是(時分)還是(秒)。Setting狀態由b0長按來切換，子狀態變數(time\_mode\_setting/show)由b1短按來控制。而在設定時，只需長按b1到使用者想設定的數字再停下即可。同時，我也利用這些狀態變數來控制當前輸出到七段顯示器的數字是哪些。

基於上面的規則，當setting=0時，所有的時分秒BCD的stop皆為0。而當setting = 1，所有BCD的stop為1除非我們按下b1long且time\_mode\_setting符合該條件時。

這裡為了方便進行程式管理，我寫了Time\_CNT(2)來進行兩位BCD的上數。要注意的是小時是在24小時進位，有別於一般的BCD計數器，我額外拉了兩條max0, max1並多了opr\_rst。max0, max1來偵測是否達到23(圖中logic diagram筆誤，在此更正)，若此刻的時為23且要進位(上數)時，opr\_rst = 1使得兩位BCD自動回到0。

此外，為了防止我們在其他功能按下b0, b1按鈕而不影響time的狀態。我用mode來代表當前我們正在使用的功能，也藉此利用它來當作enable防止我們在其他功能下影響到time的狀態。

最後，為了達成接下來的alarm功能，我將使用者設定的鬧鐘時間接進Time\_Mode裡，來偵測何時該亮燈。

* 1. For the date functions in clock (no leap year), we have the following functions: § Day (Jan/March/May/July/Aug/Oct/Dec: 1-31, Feb: 28, Apr/June/Sept/Nov: 30), § Month (1-12), § Year (00-99). Month-Day function can display in the 4 7-segment displays, and use one push button to select the display of Year (2 Seven-Segment Displays, SSDs) or Month-Day (4 SSDs).

模組名: Date\_Mode

**Design Specification**

輸出入設定

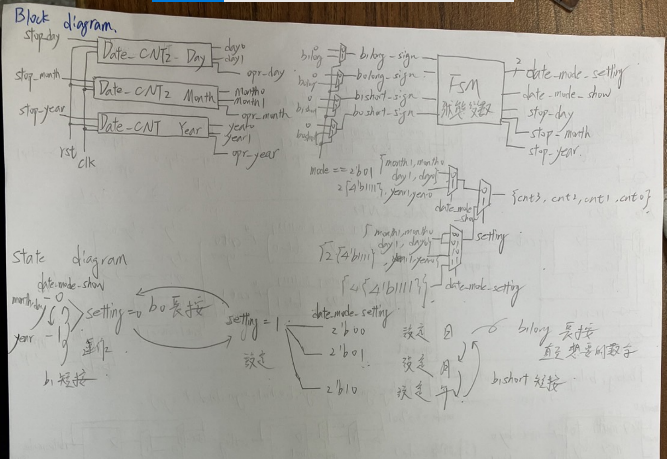
輸入:

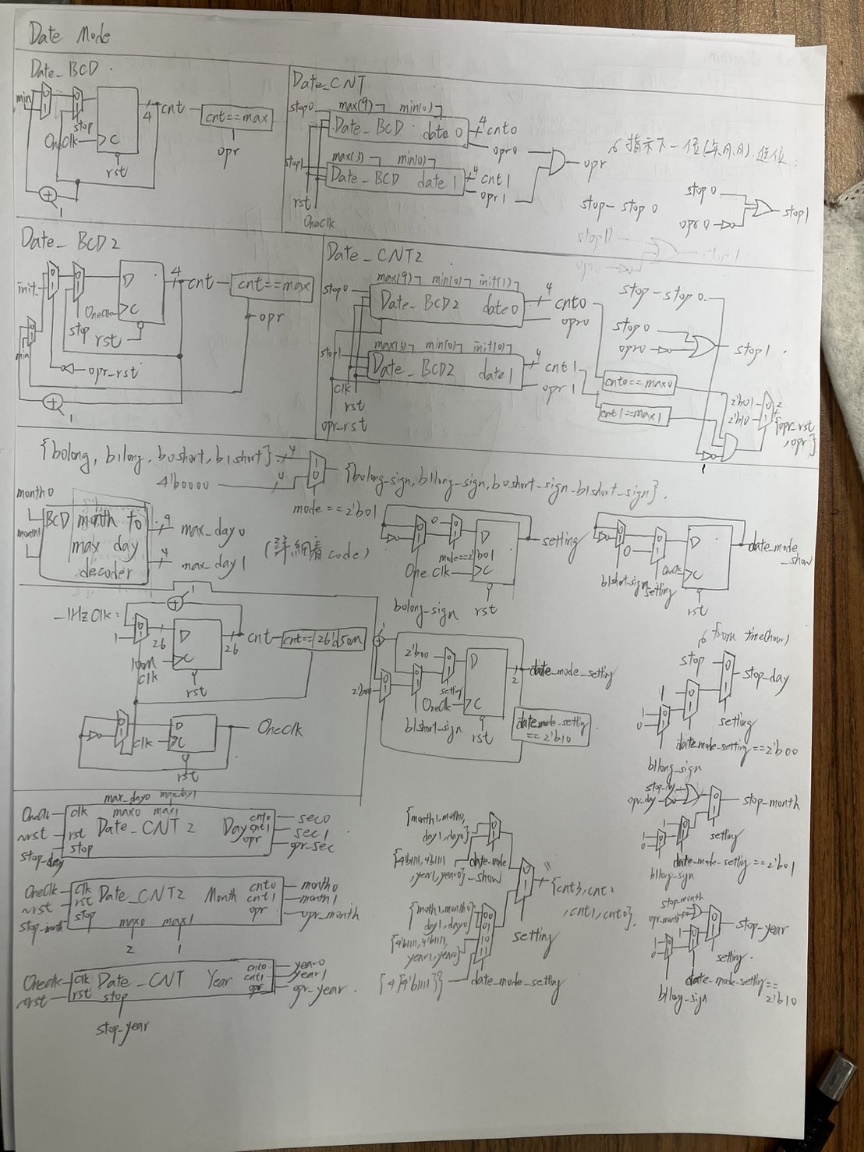
|  |  |  |
| --- | --- | --- |
| clk | 1 bit | 100M時脈 |
| rst | 1 bit | reset(外接SW15(R2)) |
| mode | 2 bits | 顯示現在在哪個功能 |
| stop | 1 bit | 從time接出來，控制day的進位 |
| b0long | 1 bit | b0按鈕長按 |
| b0short | 1 bit | b0按鈕短按 |
| b1long | 1 bit | b1按鈕長按 |
| b1short | 1 bit | b1按鈕短按 |

輸出:

|  |  |  |
| --- | --- | --- |
| cnt3 | 4 bits | 顯示於7Seg左邊數來第一個 |
| cnt2 | 4 bits | 顯示於7Seg左邊數來第二個 |
| cnt1 | 4 bits | 顯示於7Seg左邊數來第三個 |
| cnt0 | 4 bits | 顯示於7Seg左邊數來第四個 |
| date\_mode\_setting | 2 bits | LD1, 2(E19, U19)告訴使用者現在在設定哪一個位數(年月日) |
| date\_mode\_show | 1 bit | 告訴使用者現在在顯示(時分)or (秒)(LD1 U16) |
| setting | 1 bit | LD15(L1)告訴使用者是否在設定狀態 |

Block diagram and state diagram



**Design Implementation**

和Time\_Mode相同的運作原理，先有個mode==2’b01作為enable，控制按鈕訊號，防止當系統在其他狀態時，按下按鈕會影響到Date\_Mode的狀態。若要進入setting狀態時，長按b0long進入，讓setting = 1，接著再利用b1short短按來切換要設定的位數(年月日)。接著，當要設定實際數值時，長按b1long直至七段顯示器顯示出使用者想要的數字。這裡建議使用者當看到七段顯示器顯示數字為想要數字的前一個時，可以放開按鈕，因為長按會使得長按訊號多延遲一段時間。最後，若要結束設定狀態，再長按b0long回到一般狀態。

大致上所有的程式碼跟Time\_Mode一樣，大同小異。唯一不一樣的是每月的天數不一樣，需要用一個case去決定每月的最大值。剩下的方法和上面的小時Hour一樣。這裡要另外寫一個Date\_CNT和Date\_BCD是因為日月的起始是由01開始而非00。因此需要多兩個BCD init0, init1，來控制(當然init也可以直接寫在BCD計數器裡)。同樣地，在一般模式時短按b1short切換顯示模式(月日)和(年)。

* 1. Support one alarm function.

模組名:Alarm\_Mode

**Design Specification**

輸出入設定

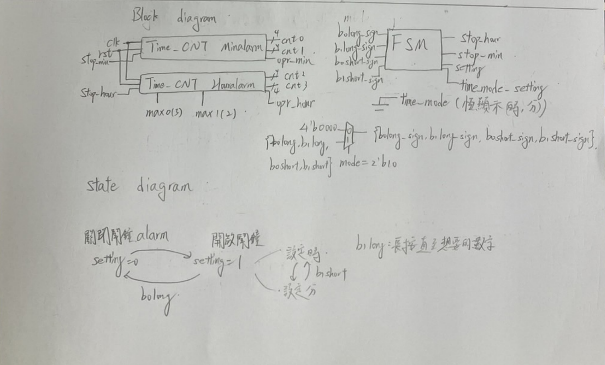
輸入:

|  |  |  |
| --- | --- | --- |
| clk | 1 bit | 100M時脈 |
| rst | 1 bit | reset(外接SW15(R2)) |
| mode | 2 bits | 顯示現在在哪個功能 |
| b0long | 1 bit | b0按鈕長按 |
| b0short | 1 bit | b0按鈕短按 |
| b1long | 1 bit | b1按鈕長按 |
| b1short | 1 bit | b1按鈕短按 |

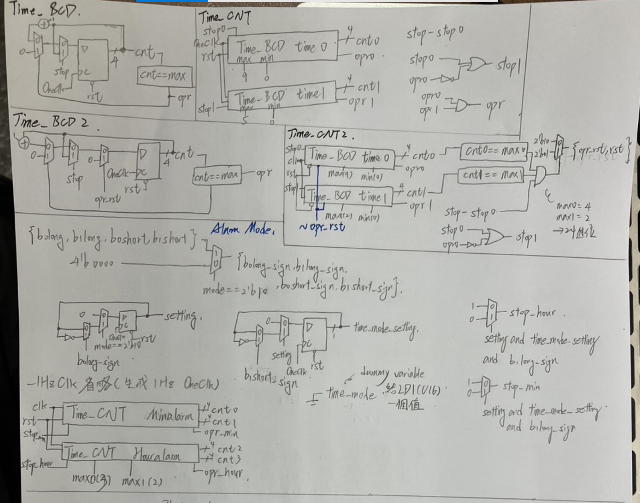
輸出:

|  |  |  |
| --- | --- | --- |
| cnt3 | 4 bits | 顯示於7Seg左邊數來第一個 |
| cnt2 | 4 bits | 顯示於7Seg左邊數來第二個 |
| cnt1 | 4 bits | 顯示於7Seg左邊數來第三個 |
| cnt0 | 4 bits | 顯示於7Seg左邊數來第四個 |
| time\_mode\_setting | 1 bit | LD1, 2(E19, U19)告訴使用者現在在設定哪一個位數(時，分) |
| time\_mode | 1 bit | 0(dummy variable)無意義 |
| setting | 1 bit | LD15(L1)告訴使用者是否在設定狀態(鬧鐘開啟狀態) |

Block diagram and state diagram



**Design Implementation**

****

在這裡我們直接挪用在Time\_Mode裡所用到的Time\_CNT和Time\_CNT2。我的alarm偵測我寫在Time\_Mode(判斷alarm是否等於當前的time)和top\_module(決定alarm是開還是關)裡，利用alarm\_setting控制(alarm\_setting為Alarm\_Mode的setting)鬧鐘是否是開或是關。其餘沒有什麼特別另外寫的東西，功能操作都和前面兩者類似。

長按b0long可改變setting狀態，若setting=1，代表鬧鐘是打開的，並可以藉由長按b1來設定數值;短按切換設定時或分。若setting=0，則鬧鐘是關閉的狀態，而且也無法調整。

* 1. Support a stopwatch function (Use lab5 exp2’s codes).

模組名: STOPWATCH\_Mode

**Design Specification**

輸出入設定

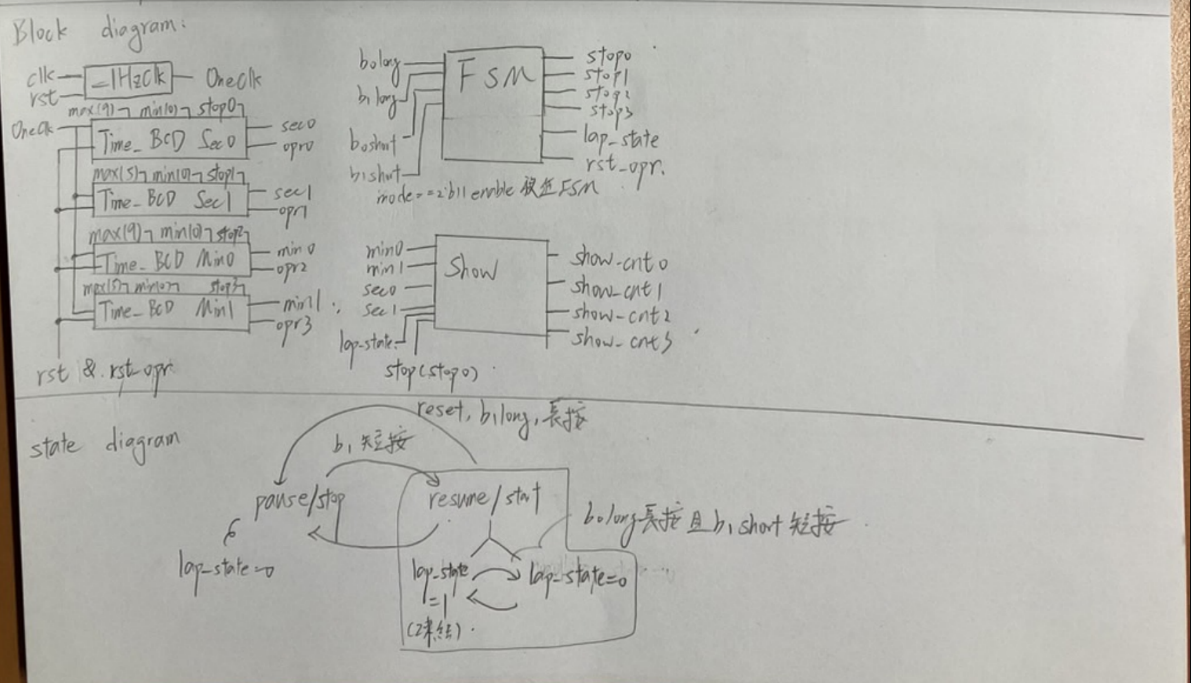
輸入:

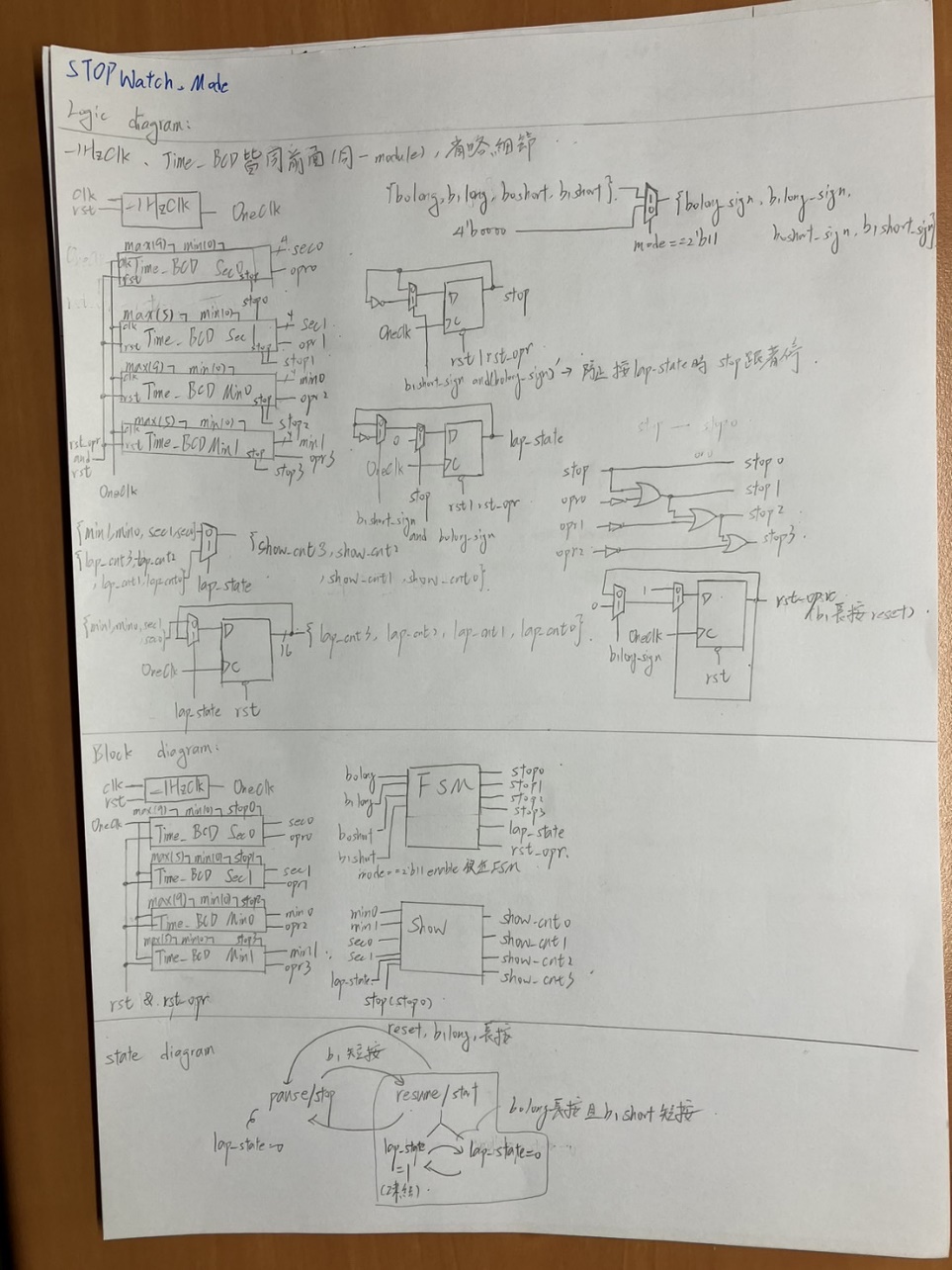
|  |  |  |
| --- | --- | --- |
| clk | 1 bit | 100M時脈 |
| rst | 1 bit | reset(外接SW15(R2)) |
| mode | 2 bits | 顯示現在在哪個功能 |
| b0long | 1 bit | b0按鈕長按 |
| b0short | 1 bit | b0按鈕短按 |
| b1long | 1 bit | b1按鈕長按 |
| b1short | 1 bit | b1按鈕短按 |

輸出:

|  |  |  |
| --- | --- | --- |
| show\_cnt3 | 4 bits | 顯示於7Seg左邊數來第一個 |
| show\_cnt2 | 4 bits | 顯示於7Seg左邊數來第二個 |
| show\_cnt1 | 4 bits | 顯示於7Seg左邊數來第三個 |
| show\_cnt0 | 4 bits | 顯示於7Seg左邊數來第四個 |
| stop | 1 bit | 顯示計時器是否於暫停狀態(和setting同led) |
| lap\_state | 1 bit | 顯示計時器是否於lap狀態(LD3, V19) |

Block diagram and state diagram



**Design Implementation**

在這裡，狀態變數化簡到剩stop, lap\_state。b1short短按可以控制stop/pause和start/resume的切換。計時途中(stop=0)才可以使用lap，且當lap狀態下按下b1short，會進入暫停pause(stop=1)並取消lap\_state回到當前數到的數字。lap\_state的切換比較麻煩，需要提前長按b0long，接著在要lap的瞬間按下b1short。這樣設計的原因是想要用一個短按控制，但2個按鈕的情況下所有短按都已被利用，所以才選擇這種方式。若要reset，長按b1long即可，計時器會自動回歸到0並停留在stop狀態。

Other module

Module1: Button\_sign\_ctl

Design Specification

輸出入設定

輸入:

|  |  |  |
| --- | --- | --- |
| clk | 1 bit | 100M時脈 |
| rst | 1 bit | reset(外接SW15(R2)) |
| pb | 1 bit | 外接到想偵測的按鈕 |

輸出:

|  |  |  |
| --- | --- | --- |
| bt\_short\_sign | 1 bit | 代表該按鈕的短按訊號 |
| bt\_long\_sign | 1 bit | 代表該按鈕的長按訊號 |

Module2: \_7SegShow

Design Specification

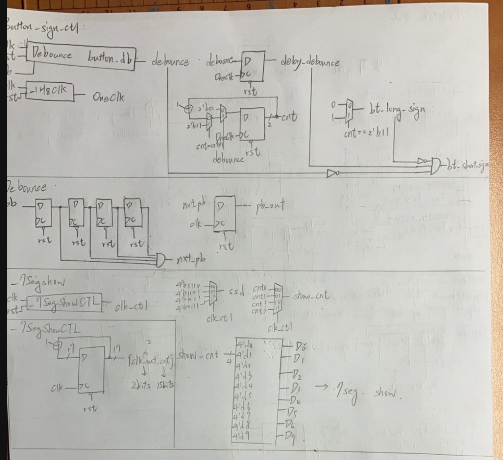
輸出入設定

輸入:

|  |  |  |
| --- | --- | --- |
| clk | 1 bit | 100M時脈 |
| rst | 1 bit | reset(外接SW15(R2)) |
| cnt3 | 4 bits | 顯示於7Seg左邊數來第一個 |
| cnt2 | 4 bits | 顯示於7Seg左邊數來第二個 |
| cnt1 | 4 bits | 顯示於7Seg左邊數來第三個 |
| cnt0 | 4 bits | 顯示於7Seg左邊數來第四個 |

輸出:

|  |  |  |
| --- | --- | --- |
| ssd | 4 bits | 控制四個七段顯示器哪幾個要亮 |
| D | 8 bits | 控制七段顯示器應顯示什麼數字(符號) |



1.5 Support the setting mode to set the time, date, and alarm.

**Combination—Watch Module**

**Design Specification**

IO輸出入設定

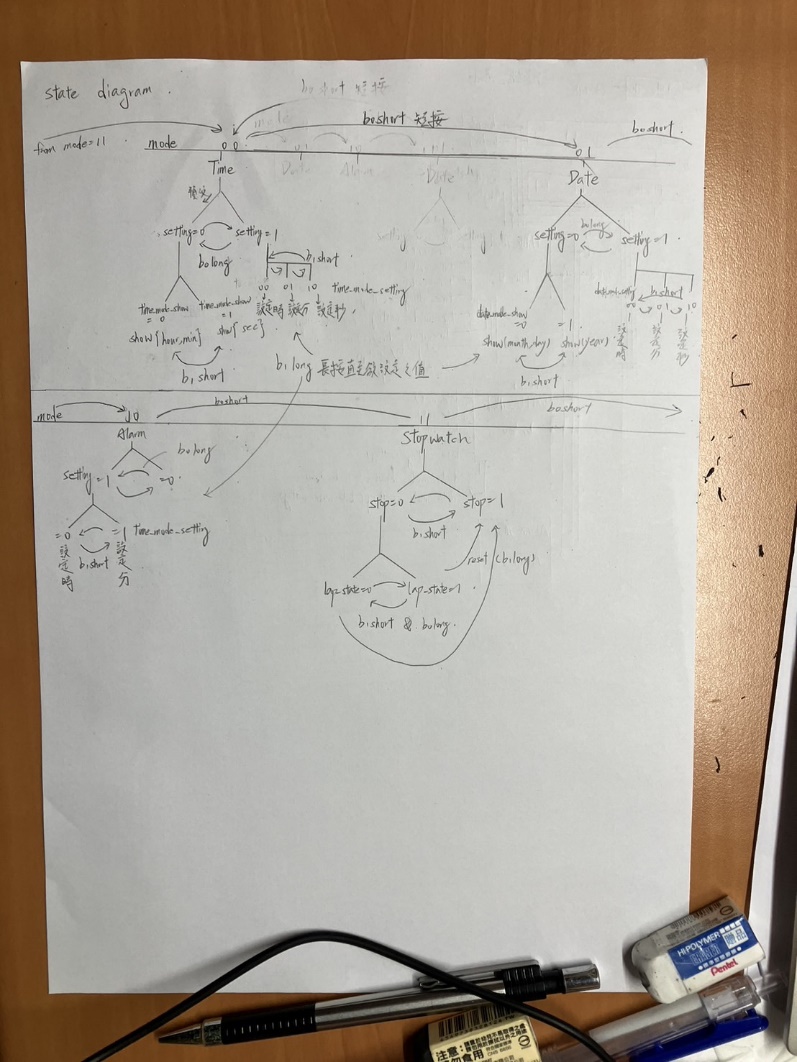
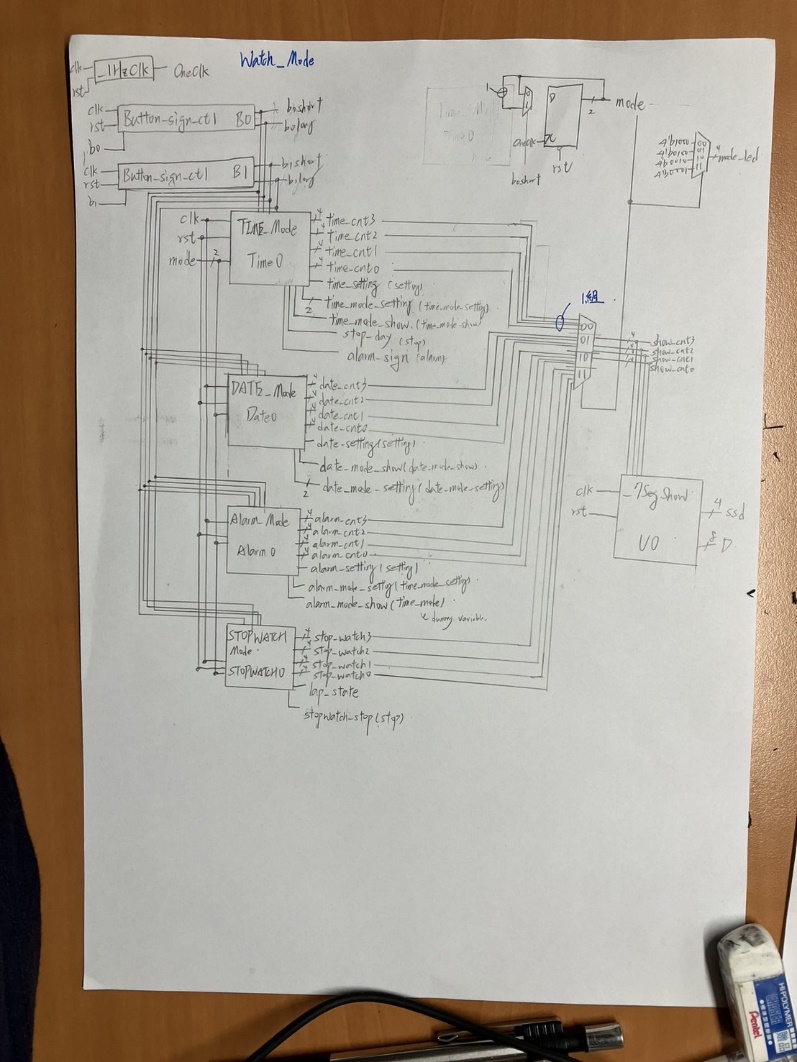
輸入:

|  |  |  |
| --- | --- | --- |
| clk | 1 bit | 時脈(100M) |
| rst | 1 bit | Reset(外接SW15(R2)) |
| b0 | 1 bit | BTNL按鈕(W19) |
| b1 | 1 bit | BTNR按鈕(T17) |

1. 輸出:

|  |  |  |
| --- | --- | --- |
| mode\_led | 4 bits | 顯示現在在哪種功能(LD13, 12, 11,10)(N3,P3,U3,W3) |
| b0long | 1 bit | LD7(V14)顯示是否觸發此訊號 |
| b0short | 1 bit | LD6(U14)顯示是否觸發此訊號 |
| b1long | 1 bit | LD9(V3)顯示是否觸發此訊號 |
| b1short | 1 bit | LD8(V13)顯示是否觸發此訊號 |
| setting | 1 bit | 顯示該功能狀態下的setting狀態(LD15, L1) |
| mode\_show | 1 bit | 顯示各功能在一般狀態下(顯示功能)，七段顯示器所顯示的數值意義 |
| mode\_setting | 1 bit | 顯示各功能在設定狀態下，正在設定哪一位 |
| ssd | 4 bits | 控制四個七段顯示器該顯示哪一個 |
| D | 8 bits | 控制七段顯示器應顯示什麼數字、符號 |
| lap\_state | 1 bit | 顯示在Stopwatch模式下，是否在lap狀態 |
| alarm | 1 bit | 當時間達到鬧鐘所設時間亮起(LD14, P1) |

Block diagram and state diagram



Pin assignment

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IO | clk | rst | b0 | b1 | b0long | b0short | b1long | b1short | setting |
| Pin | W5 | R2 | W19 | T17 | V14 | U14 | V3 | V13 | L1 |
| IO | mode\_show |  |  |  | alarm | lap\_state |  |  |  |
| Pin | U16 |  |  |  | P1 | V19 |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| IO | mode\_setting[0] | mode\_setting[1] | mode\_led[3] | mode\_led[2] | mode\_led[1] |
| Pin | E19 | U19 | N3 | P3 | U3 |
| IO | mode\_led[0] | ssd[3] | ssd[2] | ssd[1] | ssd[0] |
| Pin | W3 | W4 | V4 | U4 | U2 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IO | D[7] | D[6] | D[5] | D[4] | D[3] | D[2] | D[1] | D[0] |
| Pin | W7 | W6 | U8 | V8 | U5 | V5 | U7 | V7 |

**Design Implementation**

使用說明:

b0short在這裡只有一個作用，就是切換手錶功能(模式)。接下來依序講解各個模式如何使用

!!!在長按b1(b1long)時可以設定數字直至使用者想設定之數字，由於延遲問題，建議在目標數字前一位時放開。

**Time 模式:** mode = 2’b00, mode\_led = 4’b1000

setting切換: 長按b0(觸發b0long)

設定模式(setting = 1，LD15亮起)

mode\_setting = 2’b00 🡪 設定 時

mode\_setting = 2’b01 🡪 設定 分

mode\_setting = 2’b10 🡪 設定 秒

切換mode\_setting(LD2(U19), LD1(E19))方法: b1short，依序切換為00🡪01🡪10(循環)

長按b1(觸發b1long)，直到七段顯示器顯示之數字達到使用者想設定的數字。

一般模式(setting = 0，LD15熄滅)

mode\_show = 0 🡪 顯示{時，分}

mode\_show = 1 🡪 顯示{秒}

切換mode\_show(LD0，U16)方法: b1short，依序切換為0 🡪 1(循環)

PS: 長按b1(b1long)並不會觸發任何功能

==================================================

**Date 模式:** Mode = 2’b10, mode\_led = 4’b0100

setting切換: 長按b0(觸發b0long)

設定模式(setting = 1，LD15亮起)

mode\_setting = 2’b00 🡪 設定 日

mode\_setting = 2’b01 🡪 設定 月

// 在設定月、日時間時，建議先設定月在設定日(因為這樣會先設定日的最大值)

mode\_setting = 2’b10 🡪 設定 年

切換mode\_setting(LD2(U19), LD1(E19))方法: b1short，依序切換為00🡪01🡪10(循環)

長按b1(觸發b1long)，直到七段顯示器顯示之數字達到使用者想設定的數字。

一般模式(setting = 0，LD15熄滅)

mode\_show = 0 🡪 顯示{月，日}

mode\_show = 1 🡪 顯示{年}

切換mode\_show(LD0，U16)方法: b1short，依序切換為0 🡪 1(循環)

PS: 長按b1(b1long)並不會觸發任何功能

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

**Alarm 模式:** Mode = 2’b10, mode\_led = 4’b0010

Setting切換: 長按b0(觸發b0long)

開啟鬧鐘模式(setting = 1，LD15亮起)並設定

mode\_setting = 2’b00 🡪 設定 時

mode\_setting = 2’b01 🡪 設定 分

切換mode\_setting(LD2(U19), LD1(E19))方法: b1short，依序切換為00🡪01(循環)

長按b1(觸發b1long)，直到七段顯示器顯示之數字達到使用者想設定的數字。

設定完後便會儲存，不須按任何按鍵。若要關閉鬧鐘就再按一次b0long切換為setting=0。

關閉鬧鐘模式(setting = 0，LD15熄滅)

PS: 這時候按b1，不論短或長按皆沒有任何作用

!!!使用時盡量平放在桌子，若中途出現接觸不良或其他錯誤，可以試試reset後在試一次!!!

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

**Stopwatch模式:** Mode = 2’b11, mode\_led = 4’b0001

stop切換(Stop/pause 🡨🡪Start/resume切換): 短按b1(觸發b1short)

計時模式(stop = 0，LD15熄滅)，這時可以自由切換lap\_state來使用lap功能

lap\_state切換: 在長按b0(觸發b0long)的狀態下短按b1(觸發b1short)

lap\_state = 1(LD3, V19亮起)，七段顯示器之數字凍結

lap\_state = 0(LD3, V19熄滅)，七段顯示器之數字正常計時(回歸到實際計時時間)

非計時模式(stop = 1，LD15亮起)，這時lap\_state一律關閉，回到實際數字時間。

**Discussion**

在實作手錶Watch時，我根據曾經帶過的電子手錶的經驗來設計。切換功能都用短按，而進入設定模式都用長按。但在計時器的模式我遇到了困難，因為短按都幾乎被其他功能綁定了，但我又覺得lap屬於瞬間性的動作，所以應該還是要一個短按，所以我才選擇用一個長按來輔助。因此要切換lap時，可以事先長按b0，然後在瞬間按下b1。(不過由於我接的時脈，短按都需要2秒左右，而長按需要5秒以上)

另外，我還是保留一個switch來處理reset問題。由於我的傳輸線有時會接觸不良，導致程式在接觸不良時發生錯誤。這時就可以利用reset switch來重新設定，不需用在重新program一次。

最後，在一開始寫的時候，我發現當檔名設成Time時，不知為何top module無法成功讀取，而會出現一些錯誤。但在我把檔名設為其他名子時就恢復正常。這種錯誤真的很難找，希望下次不會再遇到類似問題。

**Conclusion**

在這次lab裡，我學到了:

* 如何寫大型複雜的project

這次的Watch手錶的寫法我還是跟馬老師的不太一樣，設了許多狀態變數來處理複雜的狀態。不過我這次有畫了state diagram，雖然我的state diagram很不正式，但我認為直觀很多。且只要有做好enable，就不用怕狀態變數間互相影響。但我認為老師的寫法優點在於只要建立一個BCD計數器就好，不像我每個功能都建立一個，在成本或消耗上可能浪費許多。

**References**

過程中我發現我有時在if裡加入一些冗餘條件時，卻沒有執行一些東西，所以我上網查詢後發現if的條件判斷裡不建議放入太多的邏輯式，因為會生成太長的邏輯鍊造成時脈上的誤差。

<https://community.intel.com/t5/Intel-Quartus-Prime-Software/Verilog-multiple-conditions-inside-an-if-statement/td-p/33719>