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。

1.1 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

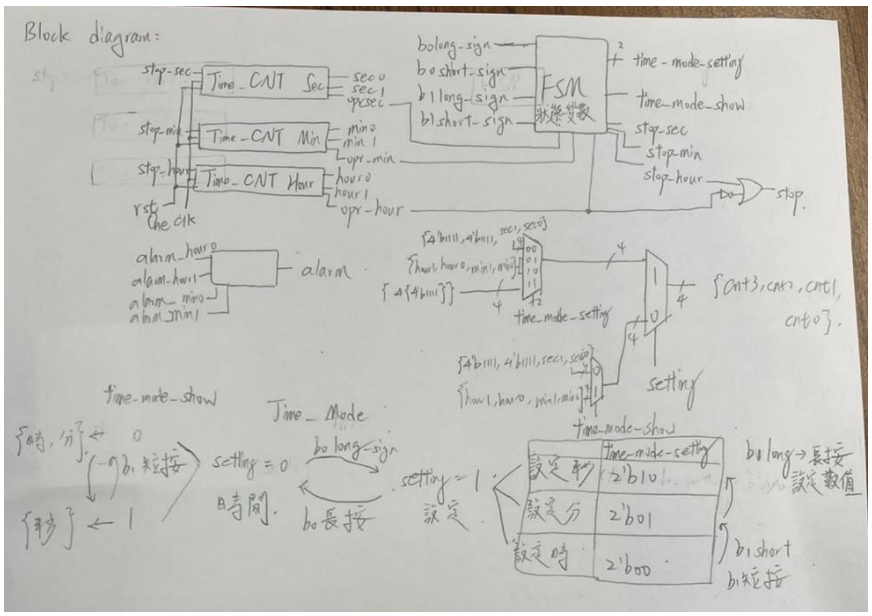
輸入:

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

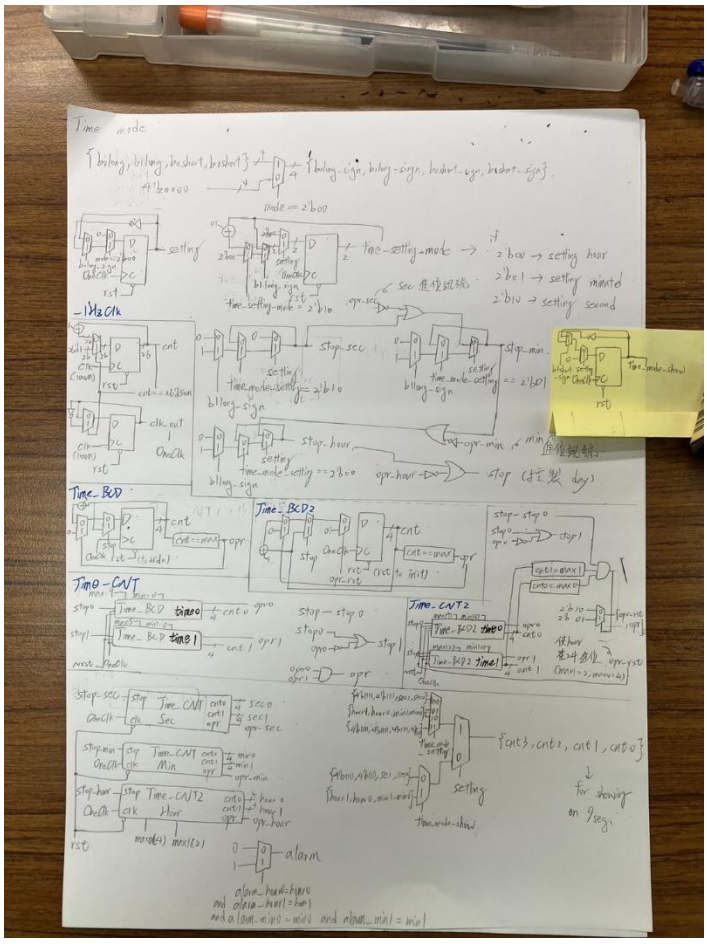
輸出:

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。

1.2 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

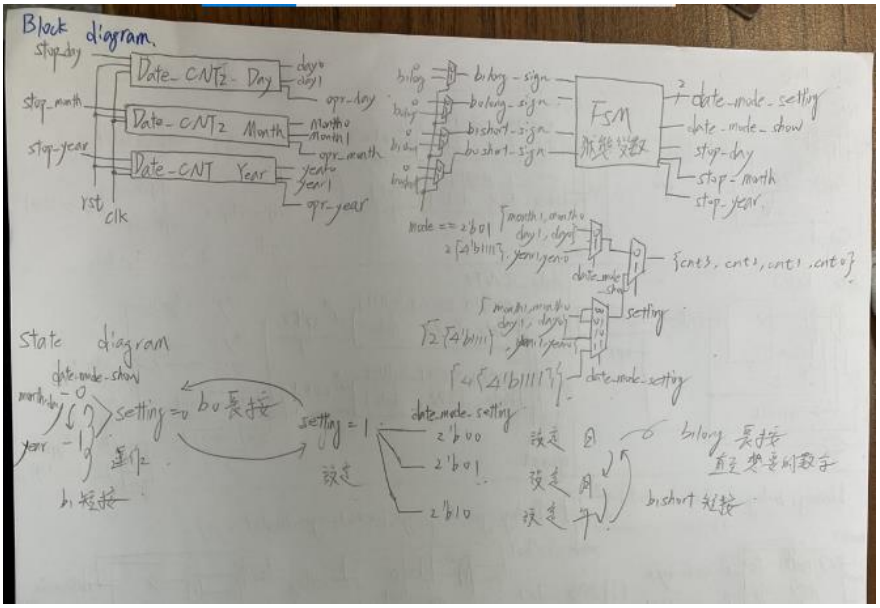
輸入:

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

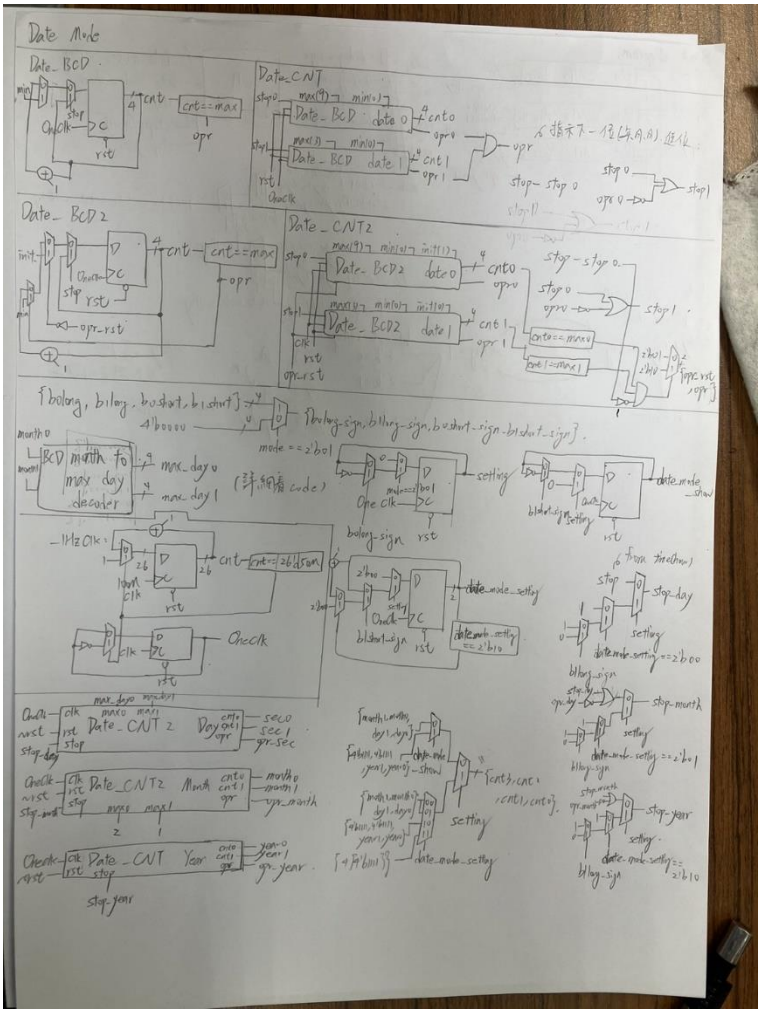
輸出:

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)。

1.3 Support one alarm function.

模組名:Alarm_Mode

Design Specification

輸出入設定

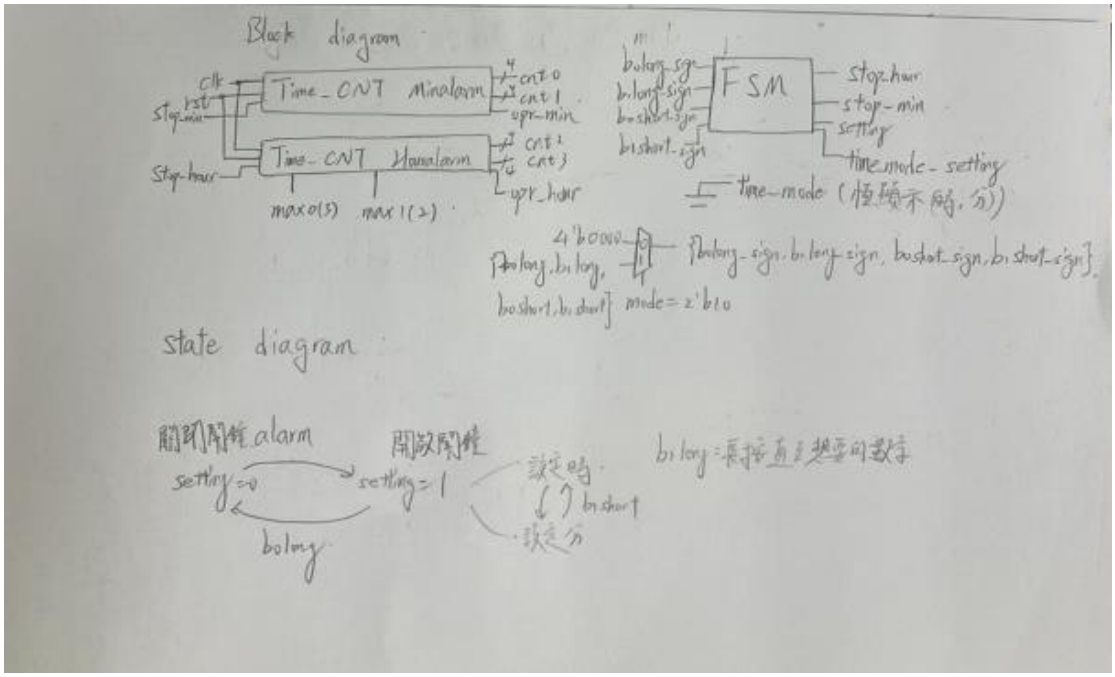
輸入:

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

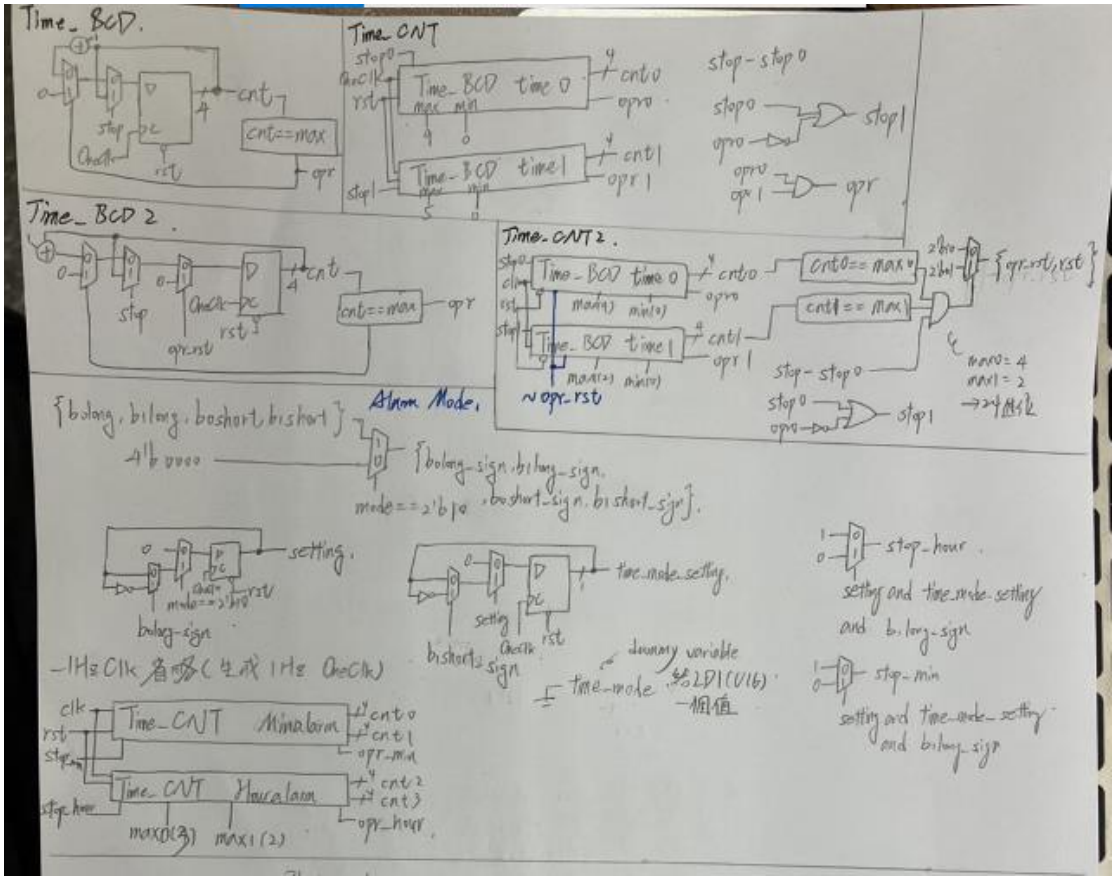
輸出:

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。

1.4 Support a stopwatch function (Use lab5 exp2's codes).

模組名: STOPWATCH_Mode

Design Specification

輸出入設定

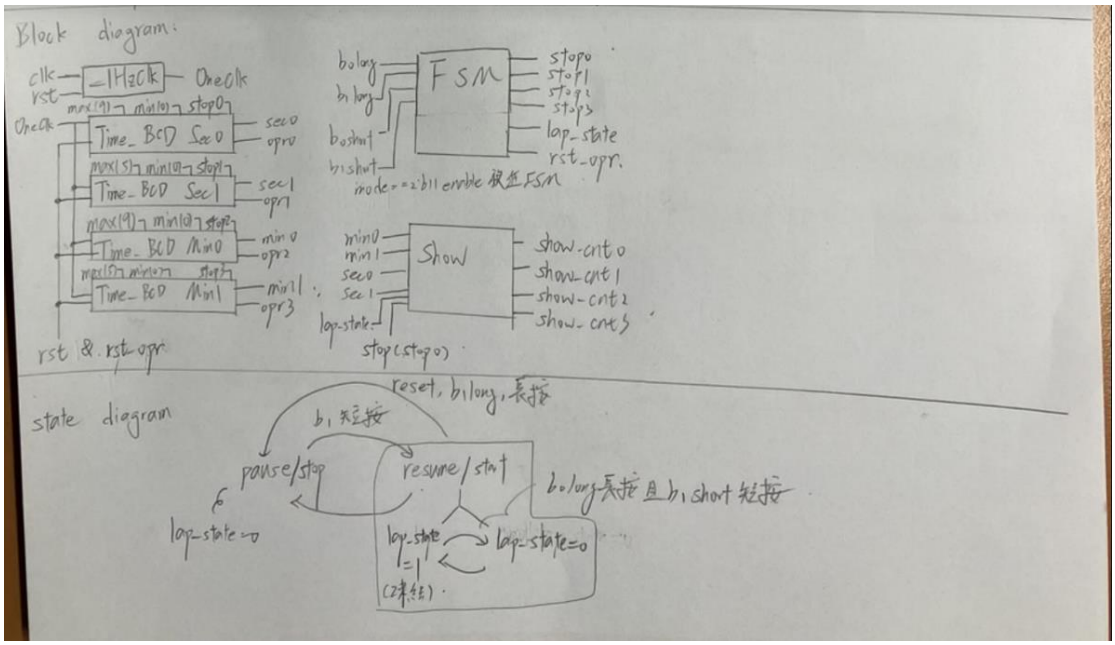
輸入:

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

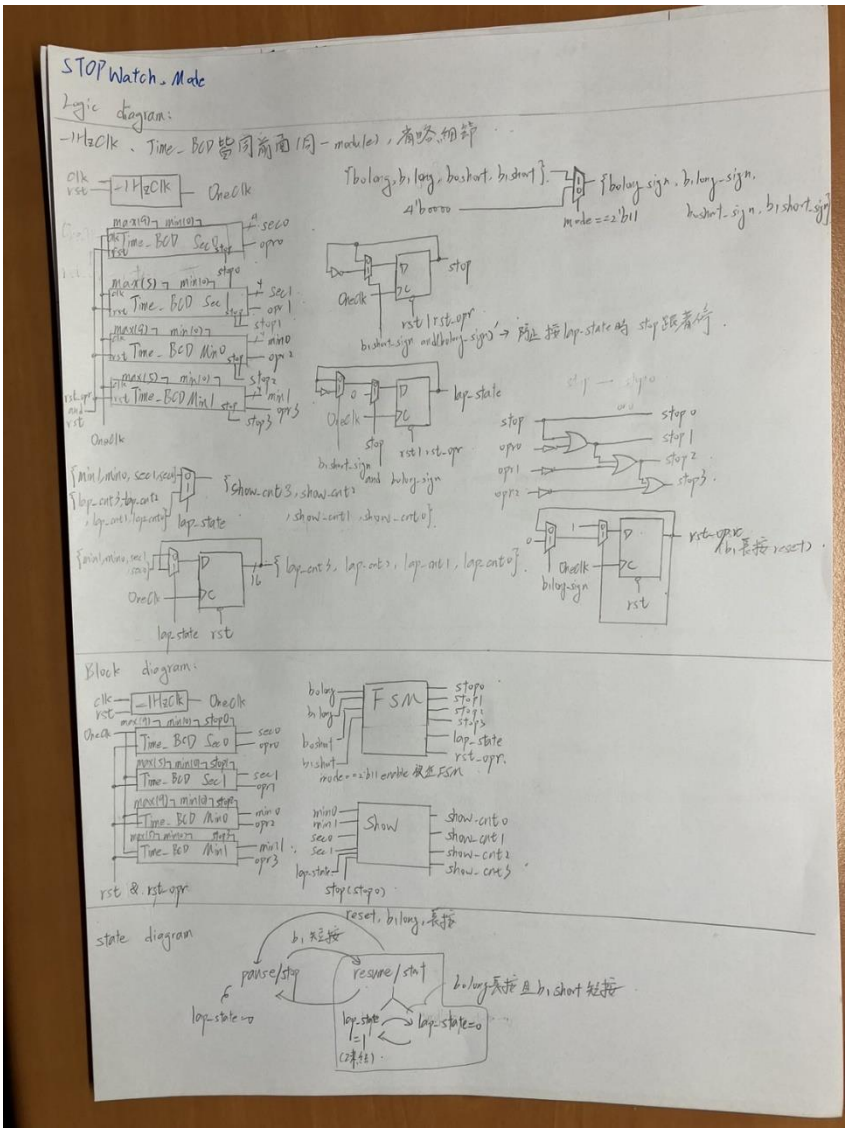
輸出:

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

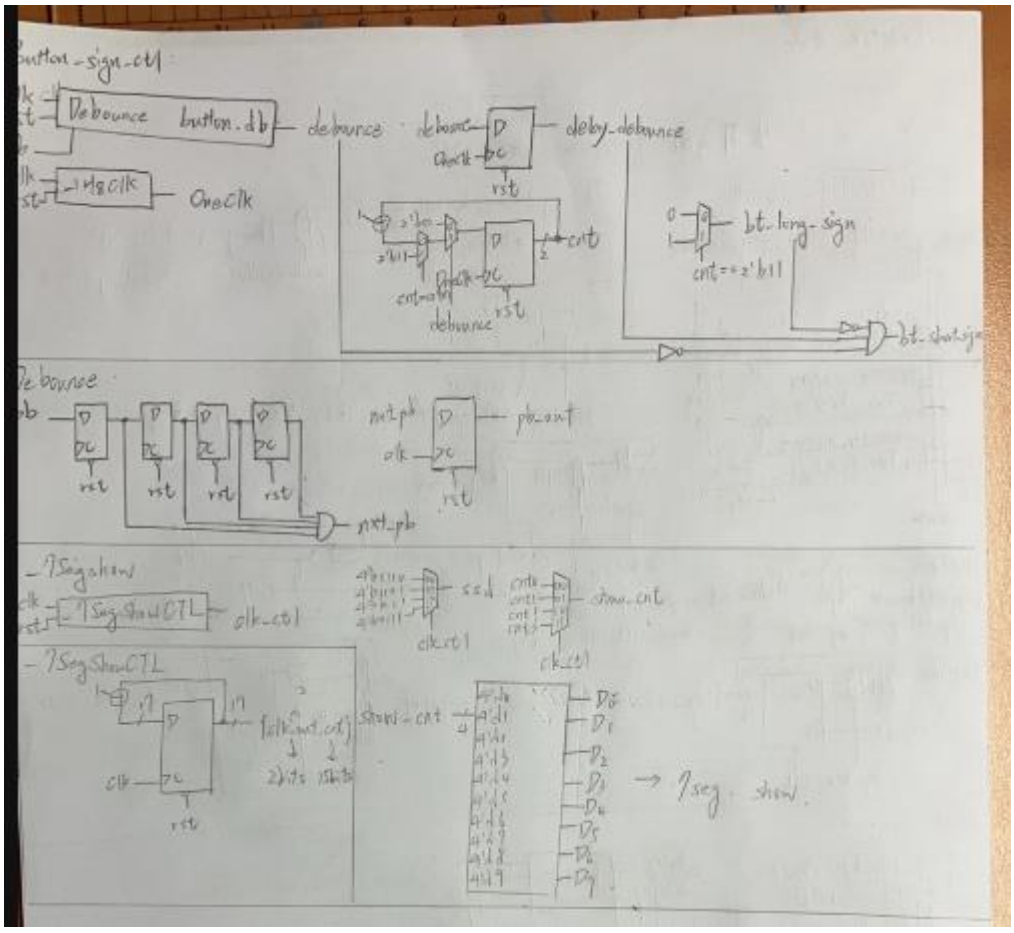
輸出入設定

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

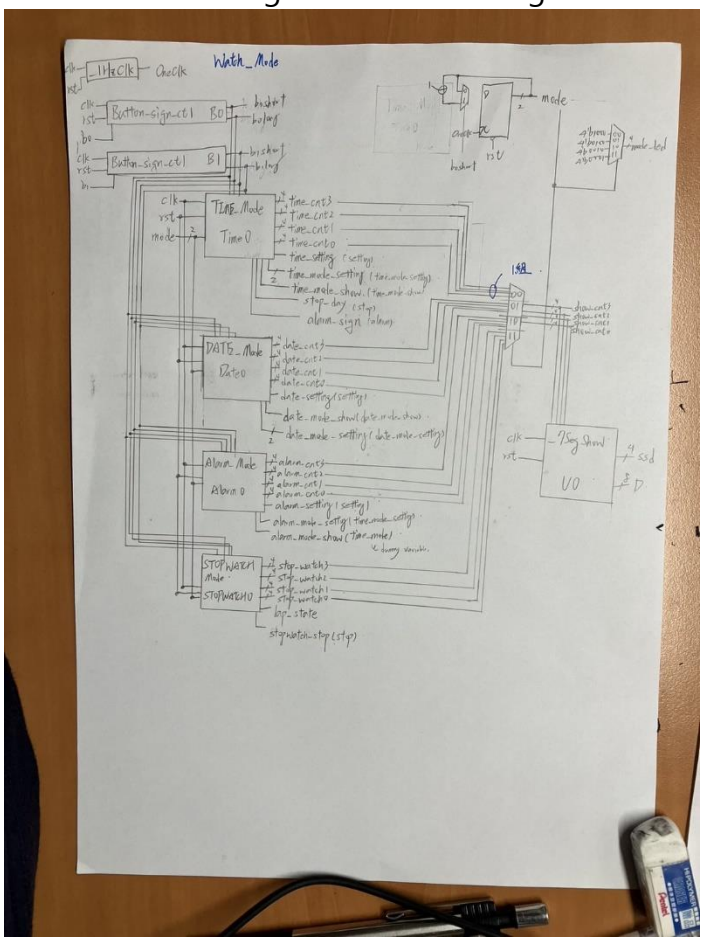
輸入:

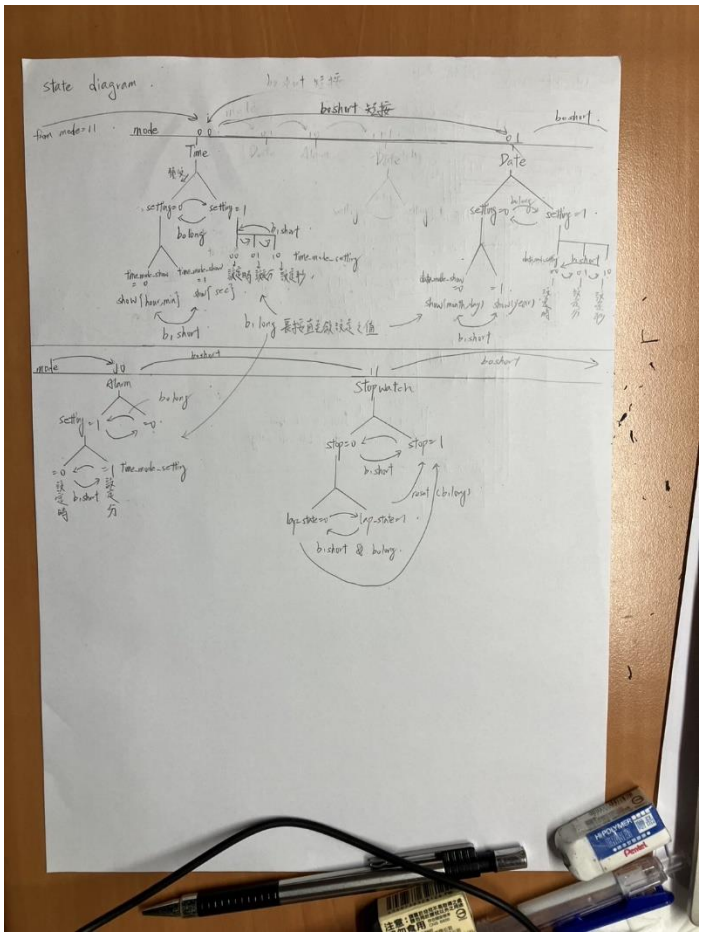
clk	1 bit	時脈(100M)
rst	1 bit	Reset(外接 SW15(R2))
b0	1 bit	BTNL 按鈕(W19)
b1	1 bit	BTNR 按鈕(T17)

2　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