Lab 5: Timers and Stopwatches

1 Construct a 50-second down counter (timer) with pause function. When the counter goes to 0, all the LEDs will be lighted up. You can use one push button for reset and one other for pause/start function.

Design Specification

IO 輸出入設定:

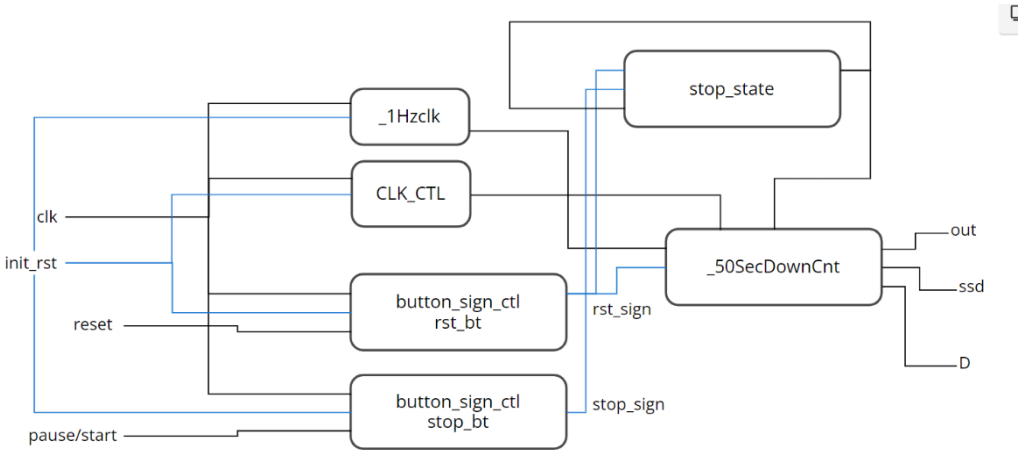
輸入: clk(1bit), rst(1 bit, reset button), stop(1 bit, pause/start button)

輸出: stop_state (1 bit, show the present state is pause or not),

out (15 bits, when time out, all led light up),

ssd (4 bits, control 7 Seg show), D (8 bits, control 7 Seg show)

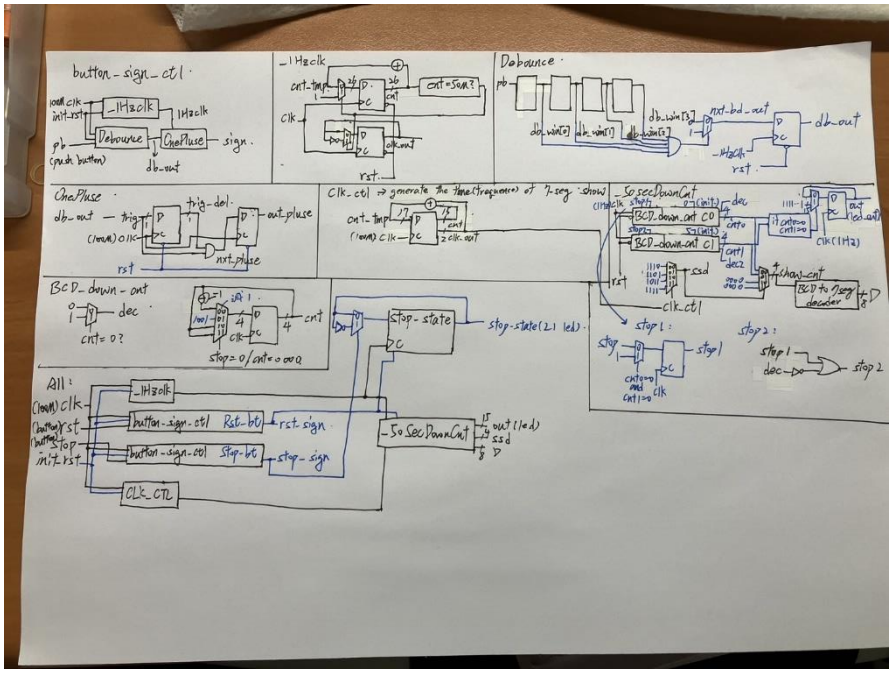
Block Diagram



Design Implementation

PS(圖中 stop_state 還要再接一條線至_50SecDownCnt)

將每個部分模組化後，再用 top module 將每個部分拼裝起來。由於我用到七段顯示器顯示當前秒數，所以多了 CLK_CTL 控制七段顯示器顯示頻率。另外由於 one_pluse 是用 1Hz 的時間去接，所以在實際操作的時候，需要長按(建議長按直至狀態改變(1 秒左右，因為



one pluse 接 1Hz 的訊號)直至 stop_state led(L1)改變時方可鬆手(stop_state == 1 時，代表處於暫停狀態。反之，熄滅就是倒數狀態)。
另外，為了要顯示 50 秒的倒數，我寫了兩位 BCD Down Counter，並加入 stop 參數來控制每個位數何時進行下數。同時在 top module 裡在把這個 stop 接到 stop_state，就可以利用按鈕控制倒數計時器。
最後，由於 button_sign_ctl, 1Hzclk, CLK_CTL 是用正反器寫的。所以他們還是需要有一個 rst 設定，這裡我用 init_rst 去控制，並在 top module 裡用 initial block 來進行初始化。

Pin assignment

IO	clk	rst	stop	Stop_state	Out[14]	Out[13]	Out[12]	Out[11]	Out[10]
Pin	W5	W19	T18	L1	P1	N3	P3	U3	W3
IO	Out[9]	Out[8]	Out[7]	Out[6]	Out[5]	Out[4]	Out[3]	Out[2]	Out[1]
Pin	V3	V13	V14	U14	U15	W18	V19	U19	E19
IO	Out[0]	ssd[3]	ssd[2]	ssd[1]	ssd[0]	D[7]	D[6]	D[5]	D[4]
Pin	U16	W4	V4	U4	U2	W7	W6	U8	V8
IO	D[3]	D[2]	D[1]	D[0]					
Pin	U5	V5	U7	V7					

Discussion

在這次實驗裡，我在寫 BCD Down cnt 時，我設定的變數較多，這樣使得 BCD Counter 彈性大，在寫兩位 BCD 時不需要寫兩個檔案。我認為在這個小題裡遇到最大的問題便是不知道如何對 clk, rst 進行初始化。因為既不能用 switch 設置，若用按鈕設定的話又需要另一個 rst 變數做初始化。所以就使用 initial block 設定。
在實際呈現時，需要注意的是最左邊的 LED 燈是用於顯示當前的 stop state 狀態，否則因為我們有寫 debounce 的原因，我們需要長按一段時間訊後才會出來。但我時常按太久便不確定現在的狀態為何，因此多加了一個 state 輸出，也可告訴使用者按鈕有無反應。

Conclusion

在這個實驗裡，我學到了

- 按鈕的使用

在這個小節裡遇到的另外一個問題便是不知道其他正反器應該要用哪個時脈控制。所以我就先寫成 switch 的形式，這樣就清楚自己該用哪個頻率去處理。之後再轉換成按鈕的方法。這個方法讓我在設計該小題時思路清晰許多。

2 Implement a stopwatch function (00:00-59:59) with the FPGA board.

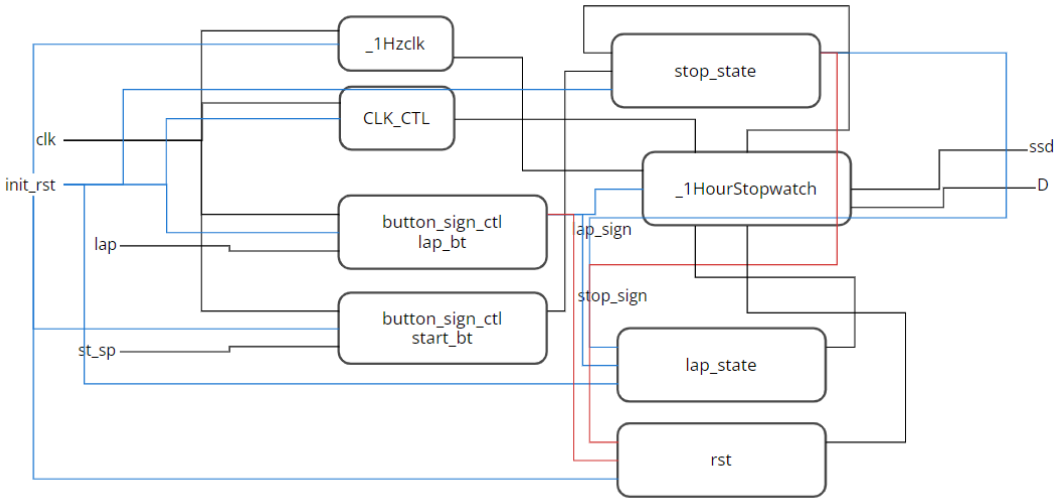
Design Specification

IO 輸出入設定

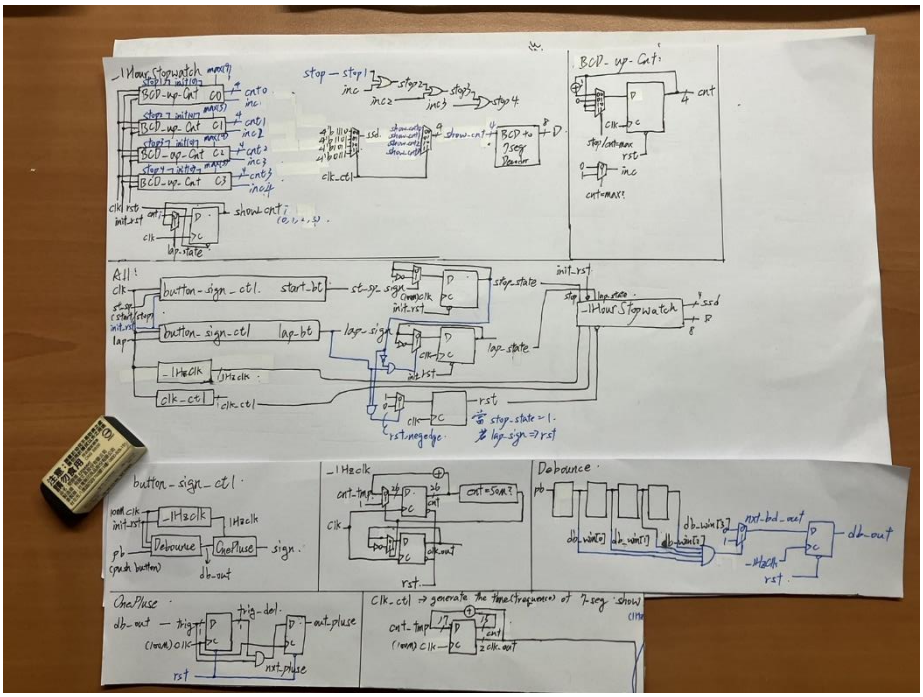
輸入: clk(1 bit), st_sp(1 bit, start/stop button), lap(1 bit, lap button)

輸出: stop_state(1 bit, led output), lap_state(1 bit, led output), ssd(4 bits), D(8 bits)

Block Diagram



Design Implementation



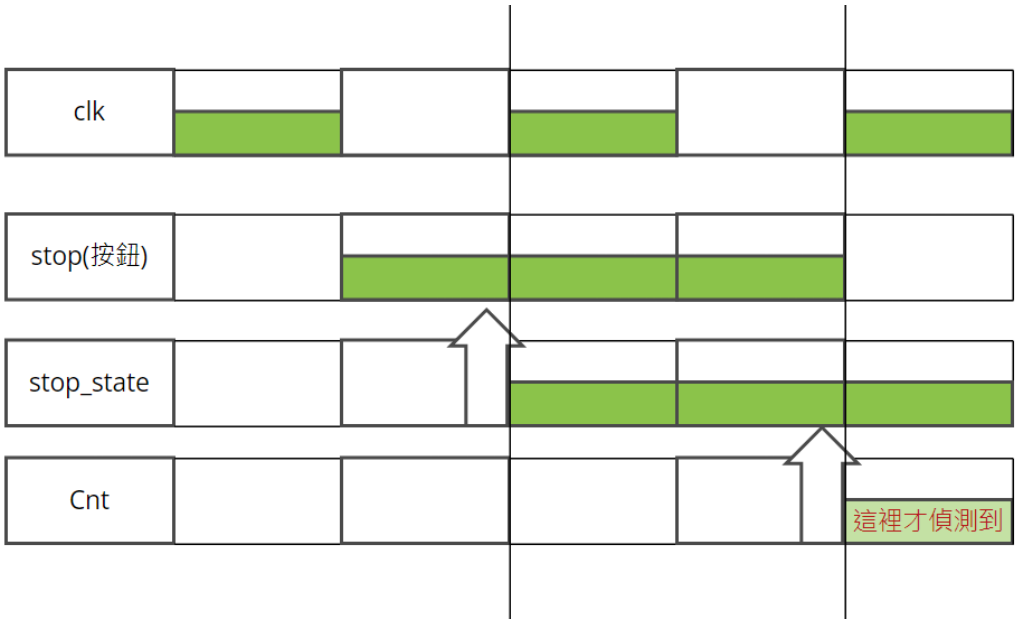
大致上都跟上一題一樣，button_sign_ctl 沒變。唯一變的是 1HourUpCnt，還有多了一些狀態變數來顯示當前的運行狀態。在 _1HourUpCnt 裡，從下數變成上數，並多了兩位數字。為了 lap_state 時凍結數字顯示，我另外宣告了四個 4bits 變數來保存按下 lap 瞬間時的數字。此外 rst 和 lap 是共用一個按鈕，在這裡我設定是當計時器在 stop_state 狀態時，在按一次 lap 會

reset。反之當計時器在計時時按下則會是凍結功能，所以又多了 1 個狀態變數(lap_state)。大致上的運作便是這樣。
要注意的是上數過程中需要用到 60 進位，所以 BCD Up Cnt 多了 max 參數，來達成這樣的目的(因為第二位最大到 6)。一樣有 stop 參數外接到 stop_state 來控制計時器狀態。

Pin assignment

IO	clk	lap	st_sp	lap_state	stop_state	ssd[3]	ssd[2]	ssd[1]	ssd[0]
Pin	W5	T18	W19	P1	L1	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	

Discussion



在實作的時候，我發現我的 stop 按下去時，秒數都會等一秒過後才停。我推測是時脈的問題。由於按下去的瞬間(stop_state 改變之後)，posedge 是看電位升高前的瞬間，而 stop_state 的時脈和計數器都是一赫茲，所以 cnt 看那時的 stop_state 是還沒改變。但 lap_state 不會出現這種狀況，是因為我的 lap 在儲存時間時是隨著 cnt 存的，只有當 lap_state==1 時才暫停儲存，所以 lap 的切換很自然。若 stop_state 改為 always blcok 做，而不用正反器去寫搞不好可以改善這個問題。
另外一個困難的點是 rst 設定，這裡要注意的是 rst 在變成 0 後的一段時間，須改回 1。這部分我就設定在 stop_state == 0 and lap_sign ==1 時才變成 1，其餘為 0。而剛好我們按鈕不會按太久，因此過了那個瞬間(狀態)，rst 又會變回 1。

Conclusion

在這一小題裡，我學到了

- Timewatch 的寫法

在經過一段時間的研究後，我發現延遲時間不只一秒。雖然討論裡有初步了討論為何按鈕和實際顯示有一段時間誤差，但還有另外一個原因。就是因為我為了寫 lap，所以我把輸出數字設為 show_cnt，而在 lap==0 時，show_cnt 會時時紀錄當前的 cnt，但當 cnt 暫停時，和上面相同的原理，show_cnt 會有延遲時間。因為暫停的流程是 stop→stop_state→cnt→show_cnt，而 stop_state 是控制 cnt，而不是控制 show_cnt，使得這一時差變得更明顯。後來的修正是 show_cnt 直接從 cnt 去取，除非遇到 lap_state == 1，才讀取 show_cnt (i), (i = 1, 2, 3..)。

3 Implement a timer (can support as long as 59:59) with the following functions.

Design Specification

IO 輸出入設定:

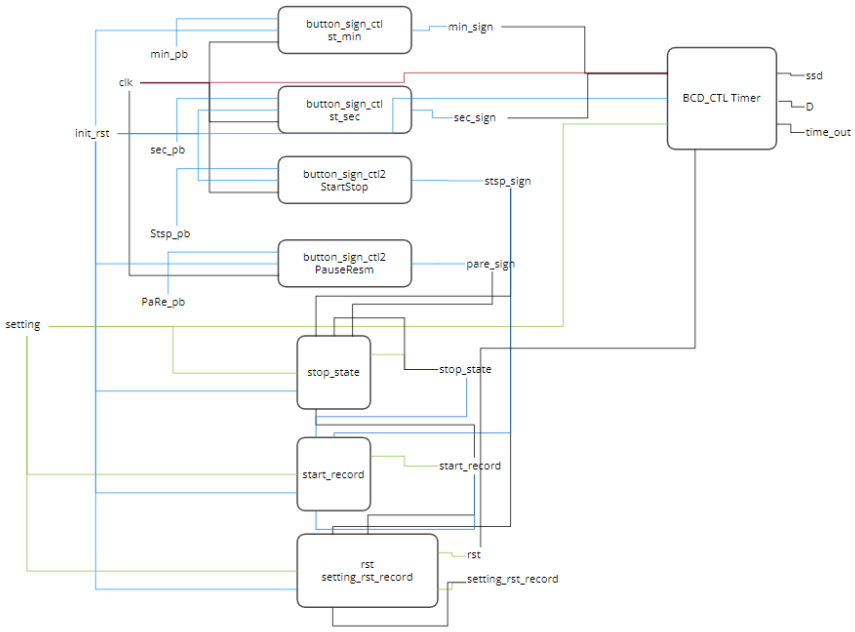
輸入: clk(1 bit), setting(1 bit, V17 switch), min_pb(1 bit,setting min button), sec_pb(1 bit,setting sec button), StSp_pb(1 bit, Start/Stop button), PaRe_pb(1 bit, Pause/Resume button)

輸出: ssd(4 bits, control 7seg. Show),

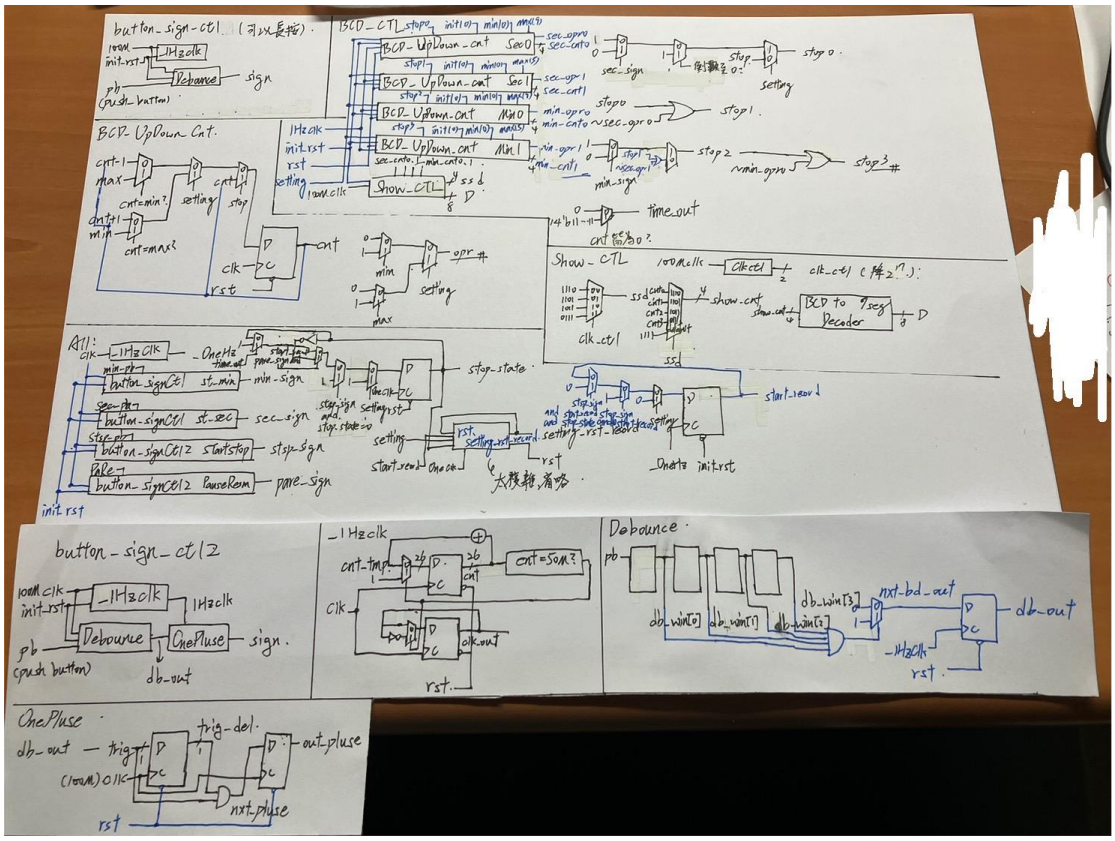
D(8 bits, control 7seg. Show),

stop_state(1 bit, L1 led), start_record(1 bit, R1 led), time_out(14 bits, 14led)

Block diagram



Design Implementation



這一題因為了多了許多狀態，所以設定上變得更複雜。我用四個狀態變數: stop_state, rst, setting_rst_record, start_record。Stop_state 只是記錄當前的數字有沒有在計時，按下 pause/resume()、stop 按鈕都會改變其狀態。而 rst 是用於

當暫停狀況下按下 stop 按鈕讓計時器 reset 時，或是將 setting switch 扳上去時，程式會利用 rst 自動將計時器歸零。但因為不能一直保持 rst==0 狀態(否則會一直維持在歸零的狀態)，所以當我們 reset 過之後，setting_rst_record 會變成 1，來告訴 rst 之後可以回歸到 1(negedge 偵測)，setting_rst_record 在經過 setting 改變後便會重新回到 0，這樣我們才能重新 rst。而 start_record 是用於判斷現在是不是處於已經按下 start 運行的狀態，在 start 按下以前，按任何鍵(除 setting switch 除外)都不會有任何反應。而按下 start 以後到中途 reset 以前，都是處在正常運行的狀態(包含數到 0)。若中間暫停後進行 reset，那 start_record 燈就會熄滅，來告訴使用者現在已經跳出計時的狀態，這時候按 pasue/resume 都沒用。而且當 reset 啟動時，時間會自動歸 0。而 time_out 只有偵測七段顯示器上的數字是否為 0，所以 reset 後，time_out 的 14 個 LED 燈都會亮起(setting ==1 時不會亮)。另外兩個燈為 stop_state, start_state，所以為了讓使用者清楚程式當前的狀態，我用兩個 LED 燈來使狀態可視化。(後來修正成 16 個燈在倒數為 0 時都會亮，狀態變數我已在程式碼中註解掉了，若要使用可以反註解(但 xdc 檔要再另外設定))

另外在 switch up(setting 狀態下)，程式會先自動歸零(須等個一秒左右)，接著就可以設定時間了。而設定秒數和分鐘的按鈕，由於我把 one pluse 拔掉了，所以可以長按直至七段顯示器顯示到使用者想用的數字。另外有個要注意的點就是這時候的秒若超過 60 後不會進位到分。

Pin assignment

IO	clk	setting	min_pb	sec_pb	StSp_pb	PaRe_pb		
Pin	W5	V17	U17	T17	W19	T18		
IO	ssd[3]	ssd[2]	ssd[1]	ssd[0]	D[7]	D[6]	D[5]	D[4]
Pin	W4	V4	U4	U2	W7	W6	U8	V8
IO	D[3]	D[2]	D[1]	D[0]	time_out[15]	time_out[14]		
Pin	U5	V5	U7	V7	L1	P1		

IO(time_out)	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pin	N3	P3	U3	W3	V3	V13	V14	U14	U15	W18	V19	U19	E19	U16

Discussion

最難的過程我覺得是要 rst 後又要讓 rst 回到原本狀態，可能沒有事先畫好 FSM，導致在寫過程中這部分一直卡卡的，所以才設了那麼多的狀態變數。除此之外本題和上一題並無太多不同。另外，為了因應這小題一開始設定時的上數功能，所以我把 BCD 計數器改成可以上下數的計數器，由 setting 狀態來控制現在應該上數還是下數。最後，關於 reset 功能，本來是想做成可以 reset 到 setting 狀態時

的功能，但會跳出 Place 30-574] Poor placement for routing between an IO pin and BUFG.的錯誤訊息。經過上網查詢後發現可以直接從 xdc 檔強制設定來忽略這個錯誤訊息。但我為了防止程式出現更多的錯誤，所以我選擇 reset 到 0。

Conclusion

在這一小題，我學到了

- 先畫好 FSM 的重要性

由於沒有先畫好 FSM，導致在實作過程中出現許多錯誤，所以才宣告了許多狀態變數來處理，如果好好整理這些狀態變數或許就能解決問題。可以感覺到這次的 lab 實驗比前面幾個難度跨了好幾個層次。希望之後可以記取這次的教訓，先畫好圖在實作。

References

https://blog.csdn.net/qq_39507748/article/details/115437791
這是關於前面提到的錯誤訊息的成因和解決方式，但作者強烈建議不要用 xdc 檔來強制跳過錯誤訊息。