Lab 9: Keyboard (Calculator)

106061218 李丞恩

1. Implement Key Board.

Design Specification

(1) Input:

clk: Global clock, 100MHz

rst: Global reset, 接在 dip Switch 上

(2) Inout:

PS2_DATA

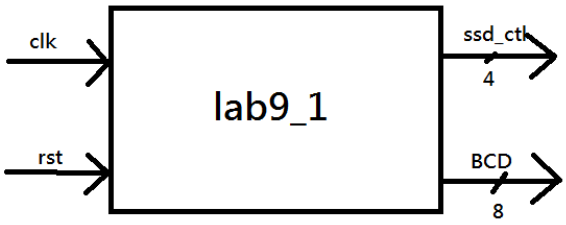
PS2_CLK

(3) Output:

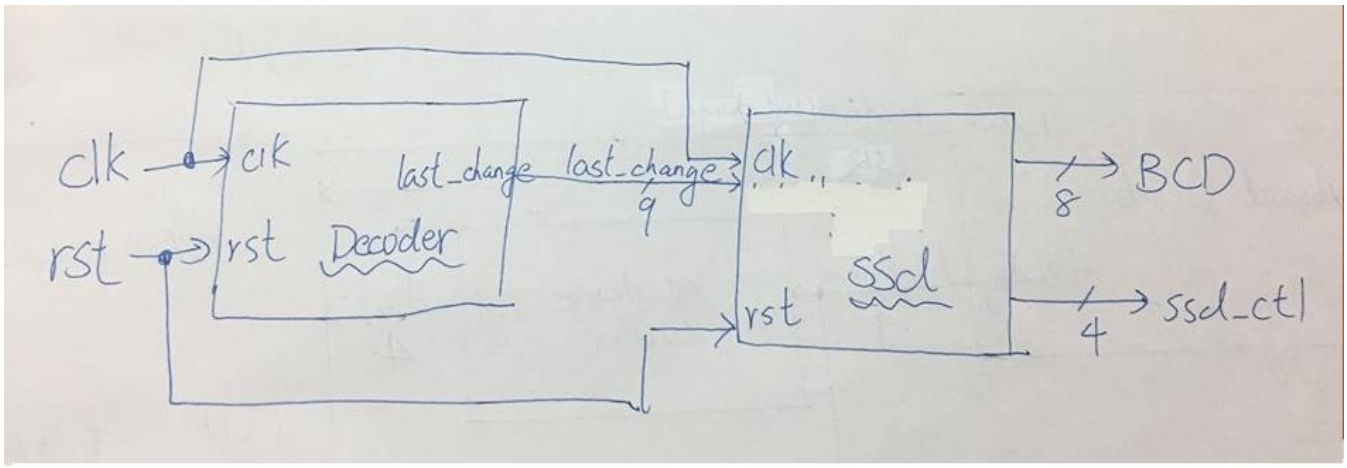
[7:0] BCD: 七段顯示器顯示的數字

[3:0] ssd_ctl: 控制哪個數字要亮

(4) Block diagram:



Design Implementation



每個 block 的功用如下：
Decoder：即 KeyboardDecoder，老師給的範例程式。
ssd：把 last_change 用 case 的語法轉成對應的數字後顯示在七段顯示器上。

Discussion

Eay Easy So Easy!
這一題說按下 Enter 要全暗，那只要 ssd 偵測到 Enter 鍵的 last_change 就把 BCD 全部設為 1 就 OK 了！

2. Implement a single digit decimal adder

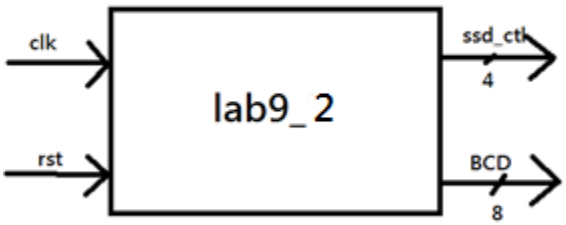
Design Specification

(1) Input:
clk: Global clock, 100MHz
rst: Global reset, 接在 dip Switch 上

(2) Inout:
PS2_DATA
PS2_CLK

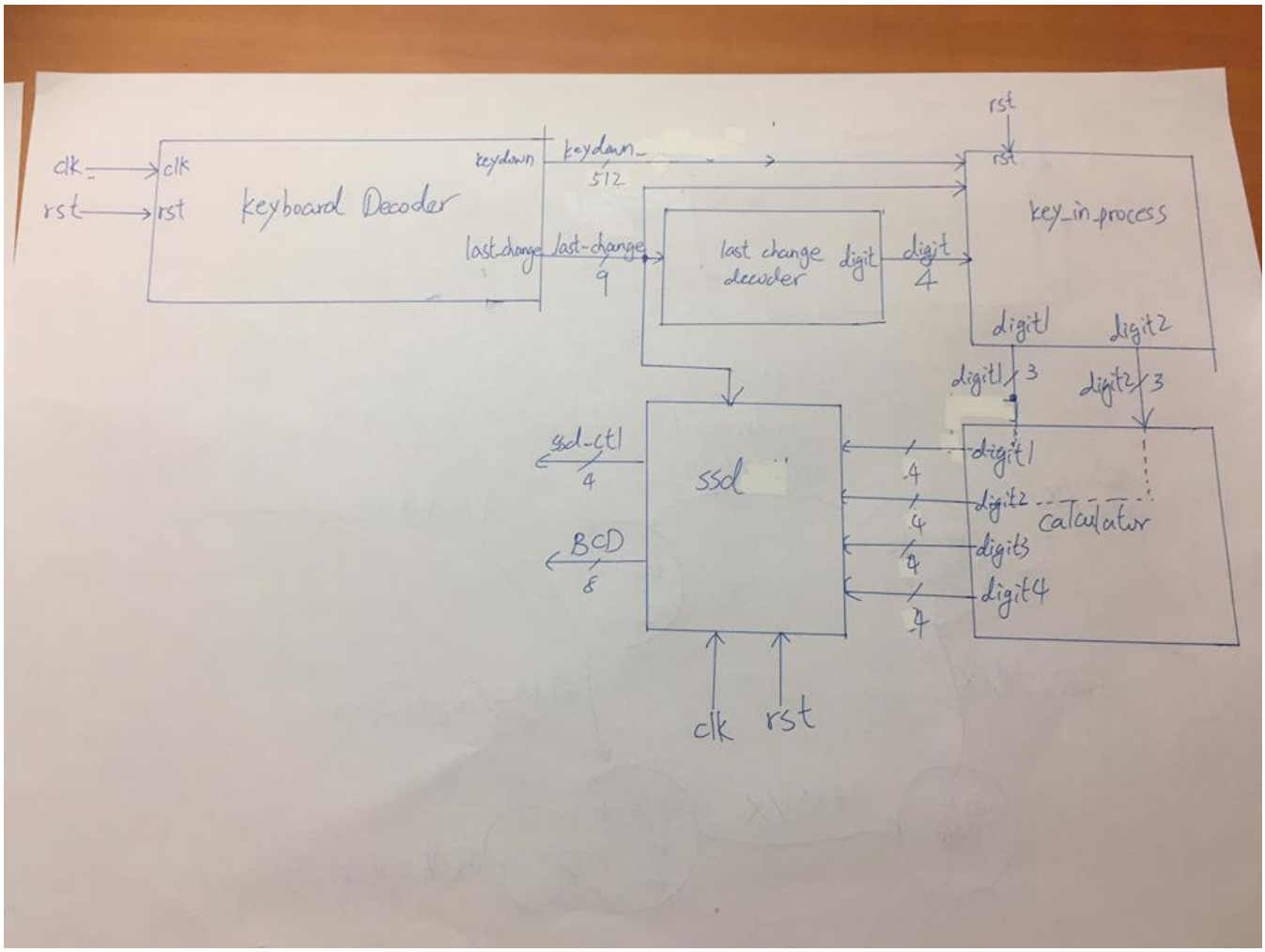
(3) Output:
[7:0] BCD: 七段顯示器顯示的數字
[3:0] ssd_ctl: 控制哪個數字要亮

(4) Block diagram:



這部分跟第一題一模一樣

Design Implementation

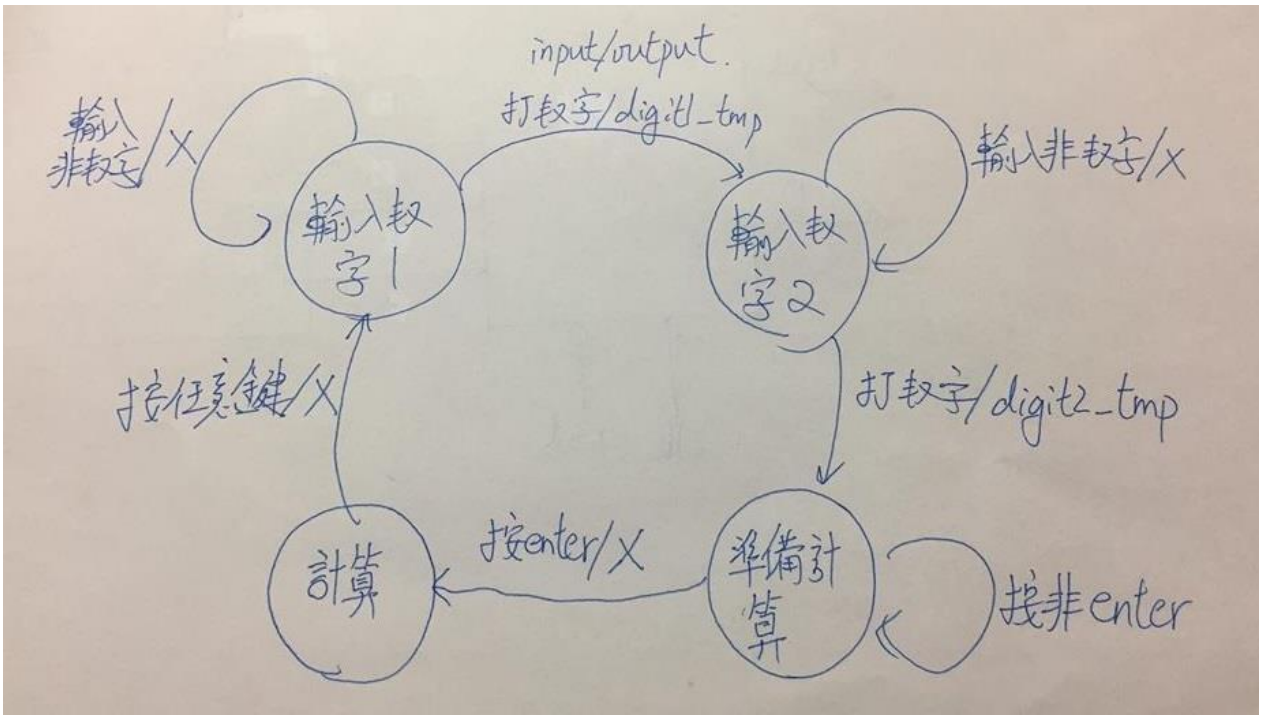


這一題所使用的 block 如下：

KeyboardDecoder: 老師提供的程式，產生 last_change 後傳進 last_change_decoder；產生 key_down 傳進 key_in_process。

last_change_decoder: 將每次輸入的 last_change 解碼成十進位數。

key_in_process: 一台具有 Output 的 FSM，這是它的狀態圖，每按一次對應的按鍵就會跳到下一個 state 並把鍵盤輸入的數字存到對應的暫存器裡。最後在 calculate 這個狀態時把數字傳給 calculator 做計算。



calculator: 計算兩個數字的加法，產生和。

ssd:顯示被加數，加數以及和。

Discussion

這一題我認為可以改得更好的地方有幾個：輸入數字後才讓對應的七段顯示器亮起來。將 state 接出去後寫一個 decoder 生成七段顯示器的 enable 接到 ssd 的 case(sel)裡應該就可以了。

另外一個地方是可以想辦法把再寫一個 FSM，使算出總合後再按一次 Enter 可以清空所有數字並使所有燈都暗下來。這部分我本來想做但頭太痛沒弄出來。

在寫這個 lab 的遇到一個問題，就是一次只能做一次運算，比如算完 8+9=17 後就沒辦法在算執行下一次運算，後來發現是由於 FSM 判斷的標準不夠嚴格造成。

另外一個麻煩是被加數和加數常常會一起輸入，比如說我想算 5+6，但我按下 5 後板子就判定我是在算 5+5。後來發現原因是我在 state 的 DFF 中利用 key_valid 當成類似把 next_state 傳過去的標準，而 key_valid 每按一次按鍵就會有兩次變化才會造成這樣。改用同學的寫法，也就是把 key_valid 換成 key_down[last_change]就解決 bug 了。

3. Implement a two-digit decimal adder/subtractor/multiplier

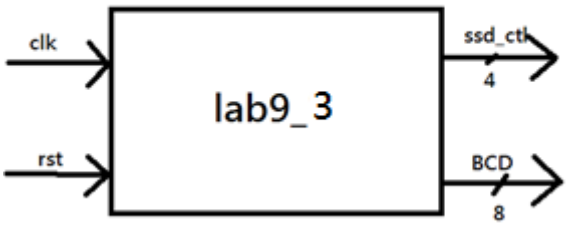
Design Specification

(1) Input:
clk: Global clock, 100MHz
rst: Global reset, 接在 dip Switch 上

(2) Inout:
PS2_DATA
PS2_CLK

(3) Output:
[7:0] BCD: 七段顯示器顯示的數字
[3:0] ssd_ctl: 控制哪個數字要亮

(4) Block diagram:



Design Implementation

跟瑪修學妹講得一模一樣！這一題只需要把第二題的 code 擴充，把 FSM 多寫幾個 state 就可以了，所使用的 block 如下：

KeyboardDecoder: 老師提供的程式，產生 last_change 後傳進 last_change_decoder；產生 key_down 傳進 key_in_process。

last_change_decoder: 將每次輸入的 last_change 給予編號，1~9 的數字給相同編號，運算符號(加、減、乘)就給予大於 10 的編號，兩者傳進 key_in_process 與 calculator。

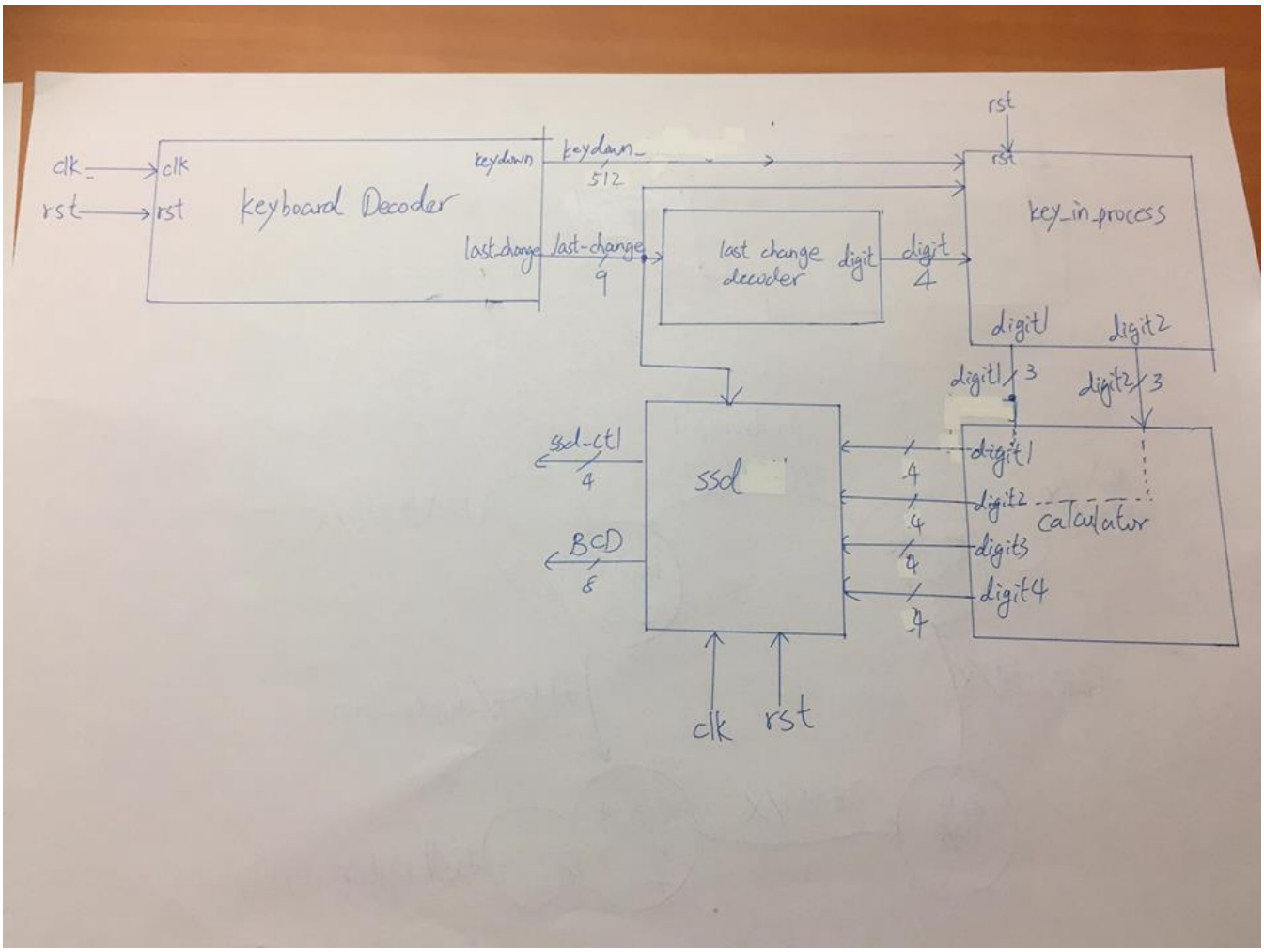
key_in_process: 一台具有 Output 的 FSM，state 總共有八個：

1. 輸入第一個數字的十位數
2. 輸入第一個數字的個位數
3. 輸入運算符號，存在一個暫存器裡。
4. 輸入第二個數字的十位數
5. 輸入第二個數字的個位數
6. 輸入 Enter，告訴電腦輸入完成準備運算
7. 再按一次 Enter 輸出運算結果

這些 state 中每按一次對應的按鍵就會跳到下一個 state，但狀態圖太複雜就不畫了，不過假設你該輸入數字時輸入的不是數字，就會停留在原本的 state。假設輸入的是數字，就會把鍵盤輸入的數字存到對應的暫存器裡。輸入 Enter(狀態 7)時會把 state 丟出去當 calculator 的致能，使之開始運作，並把數字傳給 calculator 做計算。

calculator: 在 state 是運算狀態時，同時計算加、減、乘法。再根據 key_in_process 中記錄到的運算符號決定輸出是和、差或是積。

ssd: 顯示兩個數字還有運算結果。利用 state 在 1~6 時會顯示所輸入的兩個數字，state 在 7 時改顯示運算結果。



Discussion

這一題可以改進的地方跟第二題一樣，輸入數字後才讓對應的七段顯示器亮起來。將 state 接出去後寫一個 decoder 生成七段顯示器的 enable 接到 ssd 的 case(sel)裡應該就可以了。

另外一個點是兩數相減如果結果是負數的話，我的計算機只會以 10 進位顯示兩數差的 2 補數，而不是顯示一個負數。不過我想 2 補數在科技普及的今天應該算國民基本常識吧！所以應該不打緊。當然可以用上學期交的方法在 calculator 把兩數差的 2 補數加回 256 後再讓它顯示一個負號。不過這牽涉到要顯示 3 或 4 個七段顯示器，在七段顯示器每個數字的致能生成上會有點麻煩。

寫這一題還有遇到一個問題就是我輸入數字與顯示不同步，後來把接到 ssd 的從 digit1~4 改成 digit_temp1~4 就沒問題了。因為前者需要等一個 key_down 才會讓後者傳進去。

4. Implement the “Caps” control in the keyboard.

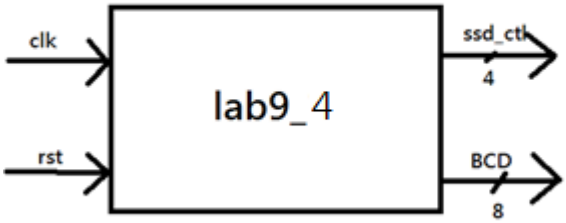
Design Specification

(1) Input:
clk: Global clock, 100MHz
rst: Global reset, 接在 dip Switch 上

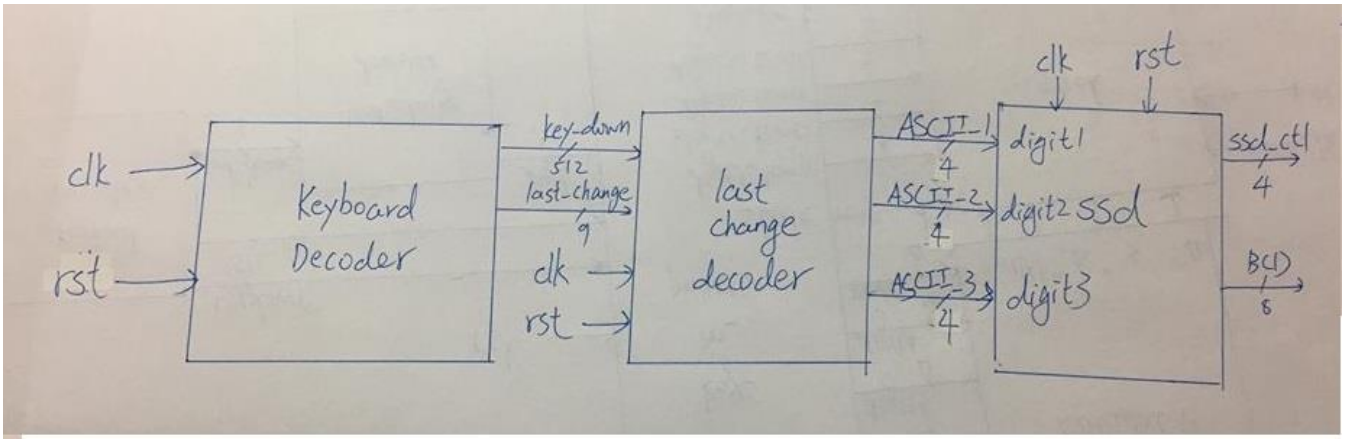
(2) Inout:
PS2_DATA
PS2_CLK

(3) Output:
[7:0] BCD: 七段顯示器顯示的數字
[3:0] ssd_ctl: 控制哪個數字要亮

(4) Block diagram:



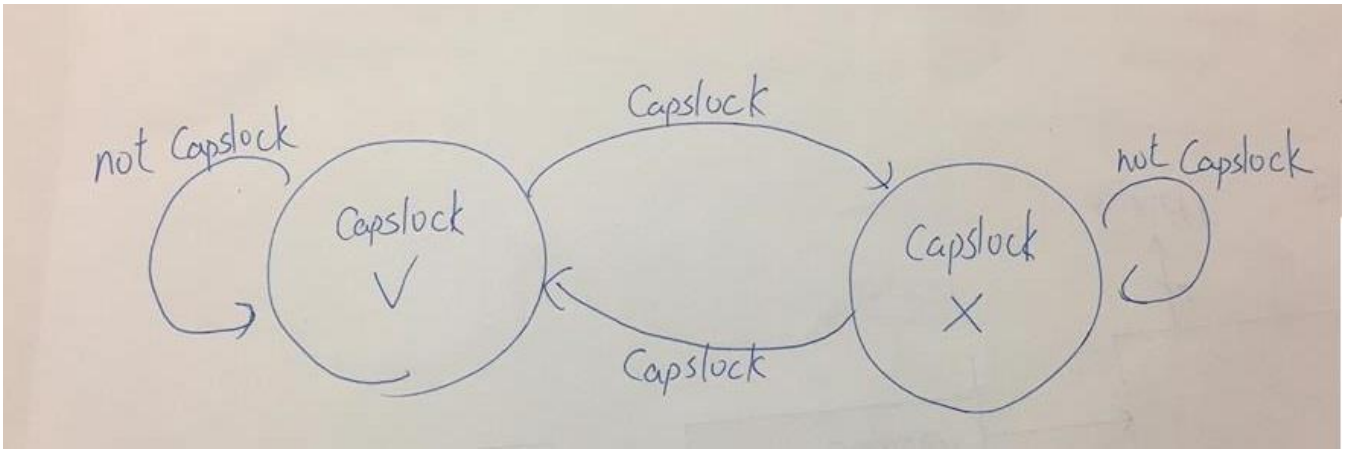
Design Implementation



這題相對簡單。只要三個 module

KeyboardDecoder: 老師提供的程式，產生 last_change 後傳進 last_change_decoder。

last_change_decoder: 將 26 個字母的 last_change 解碼成對應的 ASCII code。並帶有一只有兩個 state 的 FSM，也就是判斷現在 Capslock 有沒有作用。



ssd: 顯示 ASCII code

Discussion

所以 ASCII 到底要怎麼唸？

Conclusion

藉由這次 lab 我學到鍵盤與 FSM 是強大的組合，而且鍵盤沒有 push button 需要 debounce 的問題，因此我的期末專題決定以 keyboard 為主。

另外，關於我的 lab5、7、8

開玩笑的，我大致上已經做完了，請助教同意讓我補交報告，感恩 QQ

References

上課的講義，馬席彬 教授著，2018。

感謝非常熱心助人的廖泓全同學與黃友廷同學，教我用 key_down[last_change]的寫法解決 bug。