**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

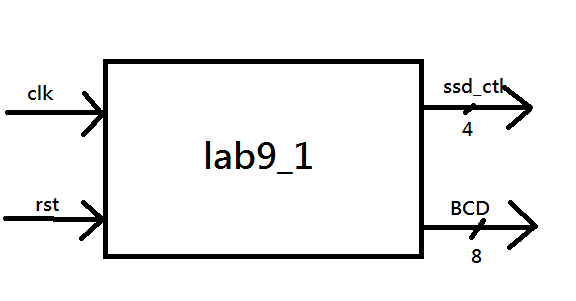
PS2\_CLK

(3) Output:

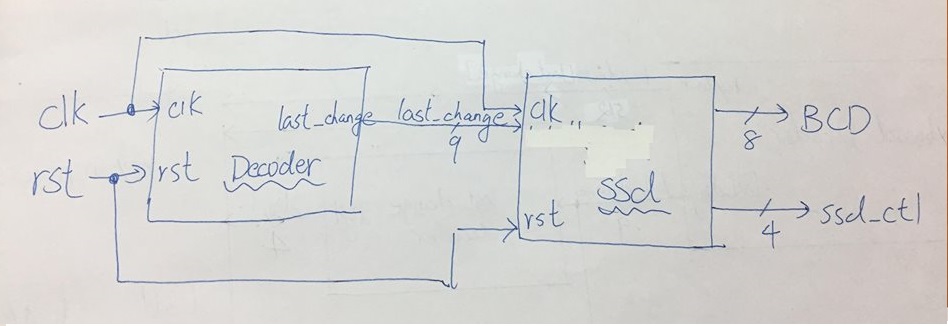
[7:0] BCD: 七段顯示器顯示的數字

[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

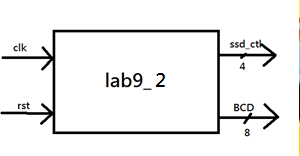
PS2\_CLK

(3) Output:

[7:0] BCD: 七段顯示器顯示的數字

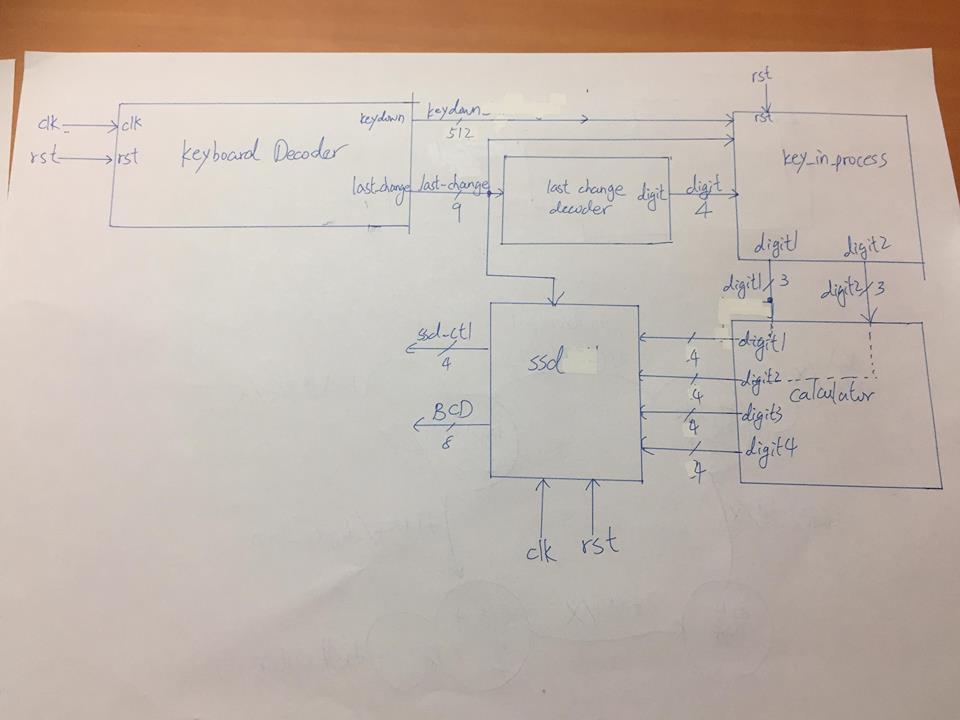
[3:0] ssd\_ctl: 控制哪個數字要亮

(4) Block diagram:

****

這部分跟第一題一模一樣

**Design Implementation**

****

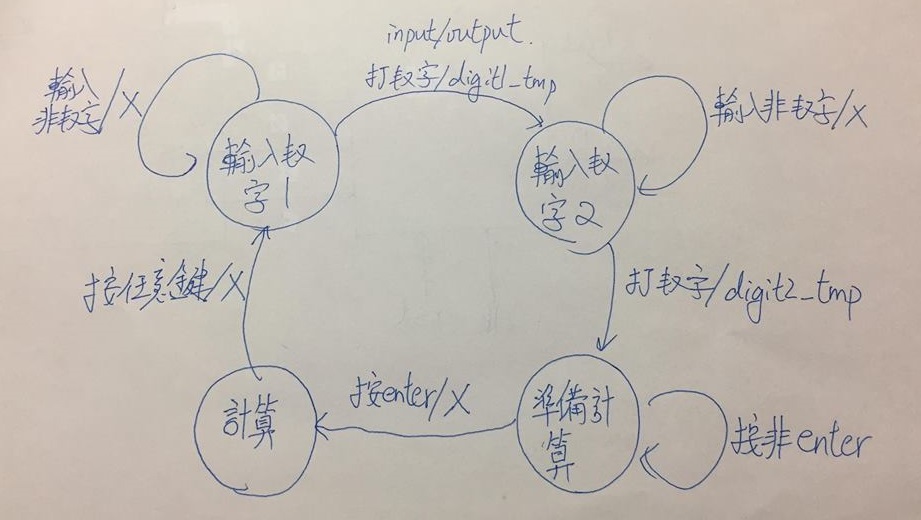
這一題所使用的block如下：

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

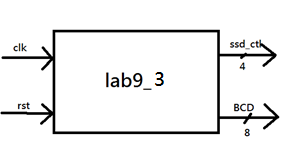
PS2\_CLK

(3) Output:

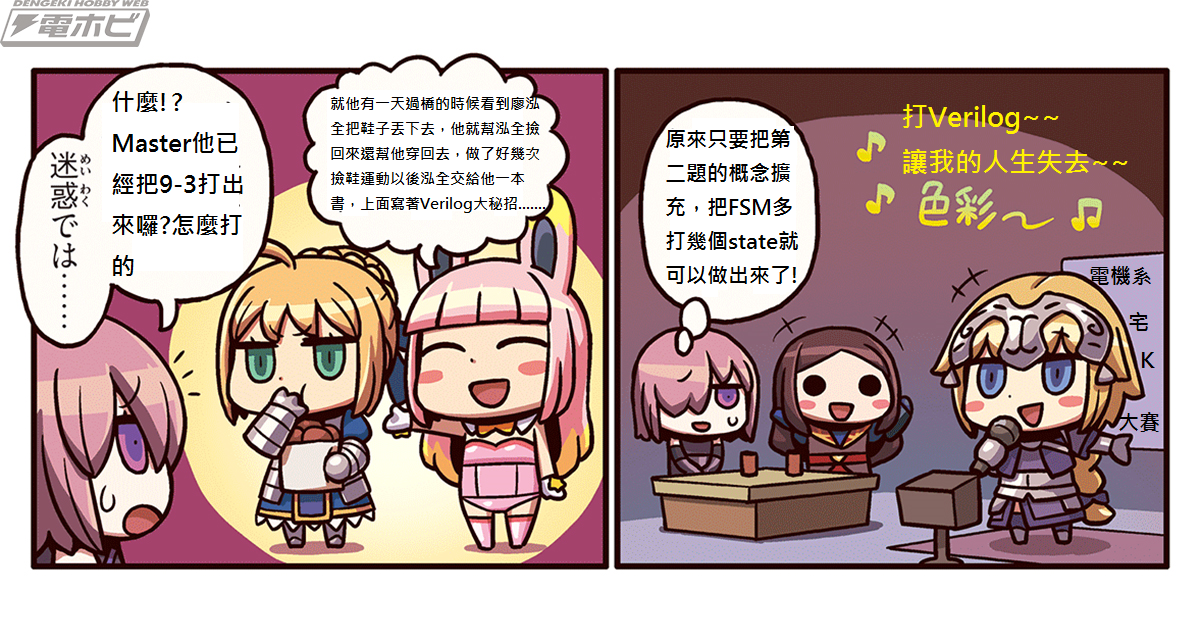
[7:0] BCD: 七段顯示器顯示的數字

[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

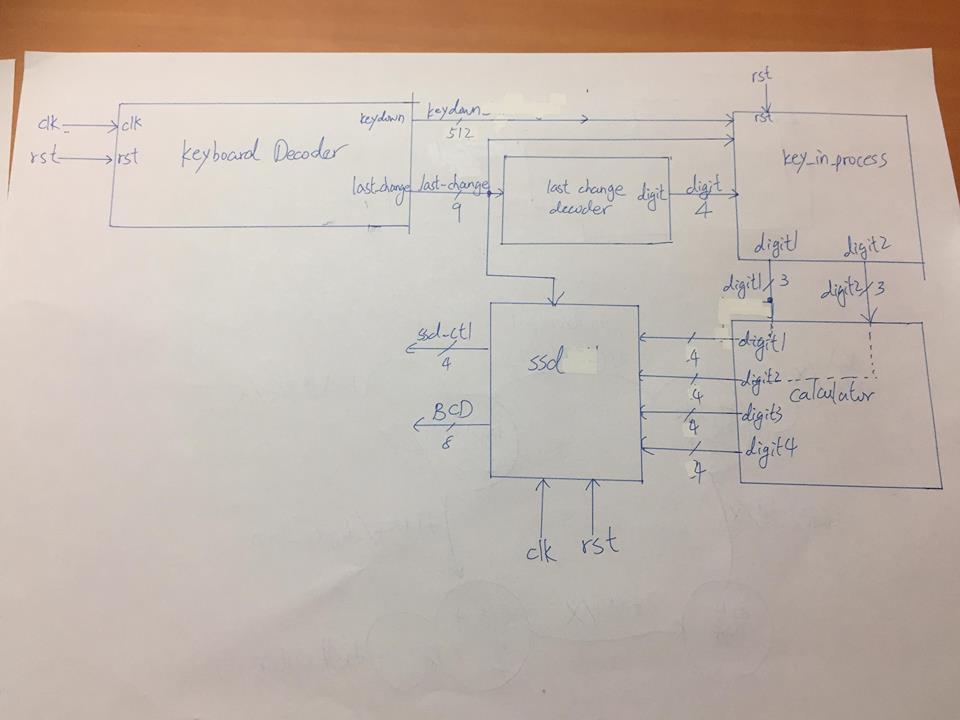
6. 輸入Enter，告訴電腦輸入完成準備運算

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

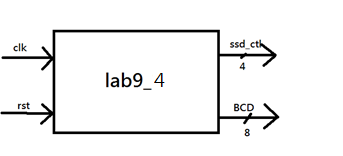
PS2\_CLK

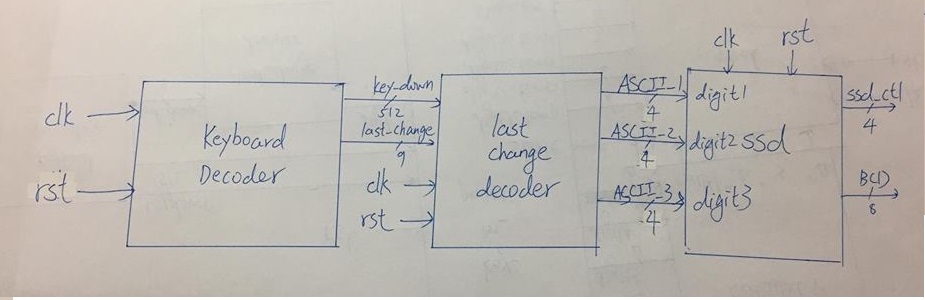
(3) Output:

[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:

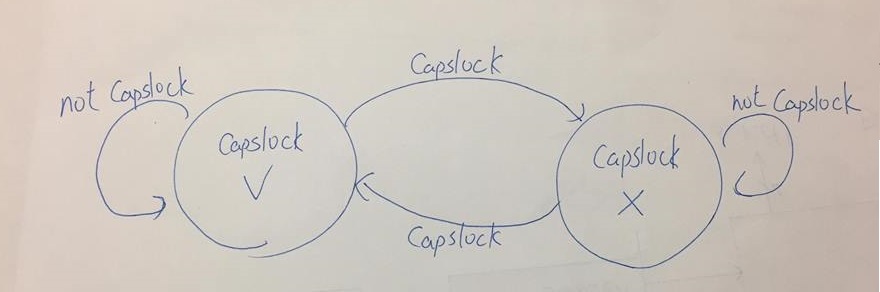


**Design Implementation**

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。