Lab9: Keyboard (Calculator)

107061112 王昊文

Experiment

1. Implement Key Board

Design Specification

✓ I/O

Inout PS2_DATA, PS2_CLK; // keyboard data

Input clk; // crystal clock

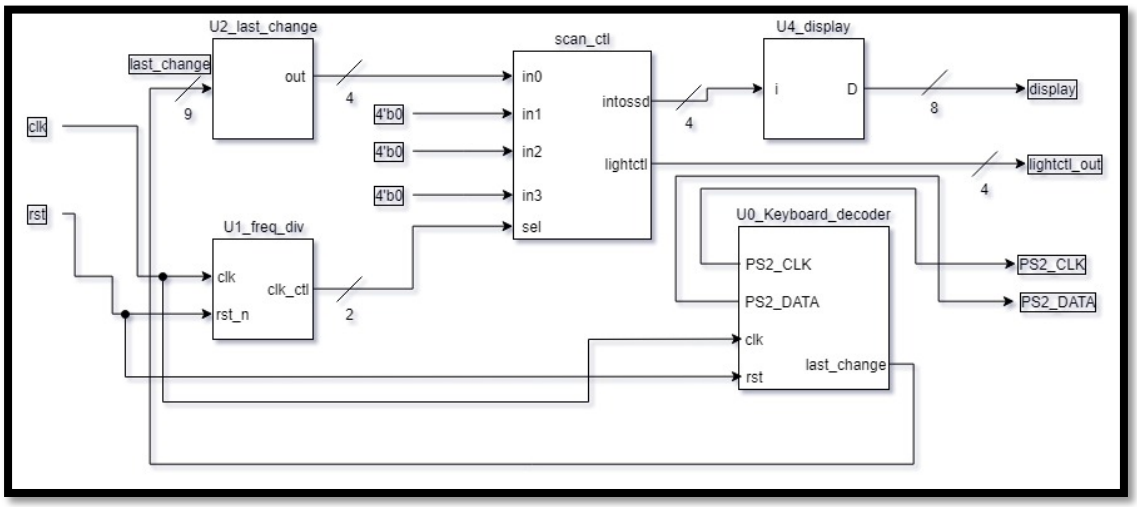
Input rst; // high active reset

// display outputs

Output [3:0]lightctl_out;

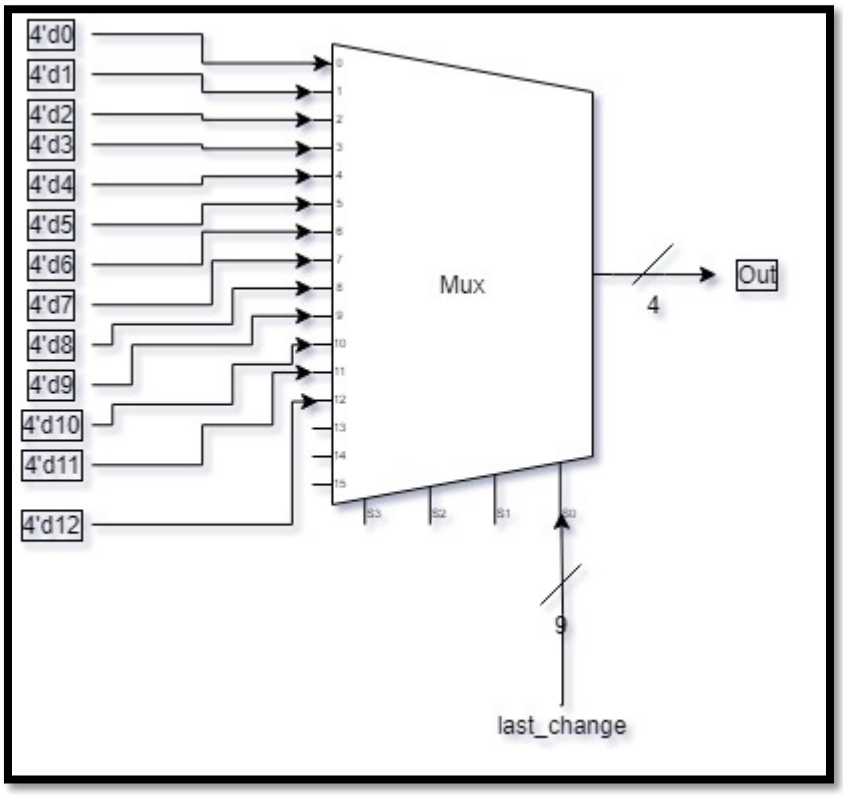
Output [7:0]display;

✓ Logic Block:



第一題的部分只需要判斷keyboard decoder送出來的訊號，經過last_change這個模組來判斷，然後將結果送進最後的顯示模組即可。

✓ Last change模組



Last_change這個模組的input訊號為keyboard decoder送出來的訊號，包含make code以及extend code的部分，唯一組16進位碼。根據現在送進來的訊號可以知道目前input為什麼數字，還有a, s, m哪一個按鍵。

✓ Design Implementation

Inout	
PS2_CLK	*PS2_DATA*
C17	B17

Input	
clk	*rst*
W5	R2

Output ssd							
display[0]	***display[1]***	***display[2]***	***display[3]***	***display[4]***	***display[5]***	***display[6]***	***display[7]***
V7	U7	V5	U5	V8	U8	W6	W7

Output lightctl			
lightctl[0]	*lightctl[1]*	*lightctl[2]*	*lightctl[3]*
U2	U4	V4	W4

✓ Discussion

這一題算是裡面的最入門題，只要清楚老師給予的keyboard decoder的基本運作原理，知道怎麼抓取key_down，key_valid，last_change即可輕鬆顯示在七段顯示器上面。

✓ Experiment

2. Implement a single digit decimal adder using the key board as the input and display the results on the 14-segment display (The first two digit are the addend/augend, and the last two digits are the sum).

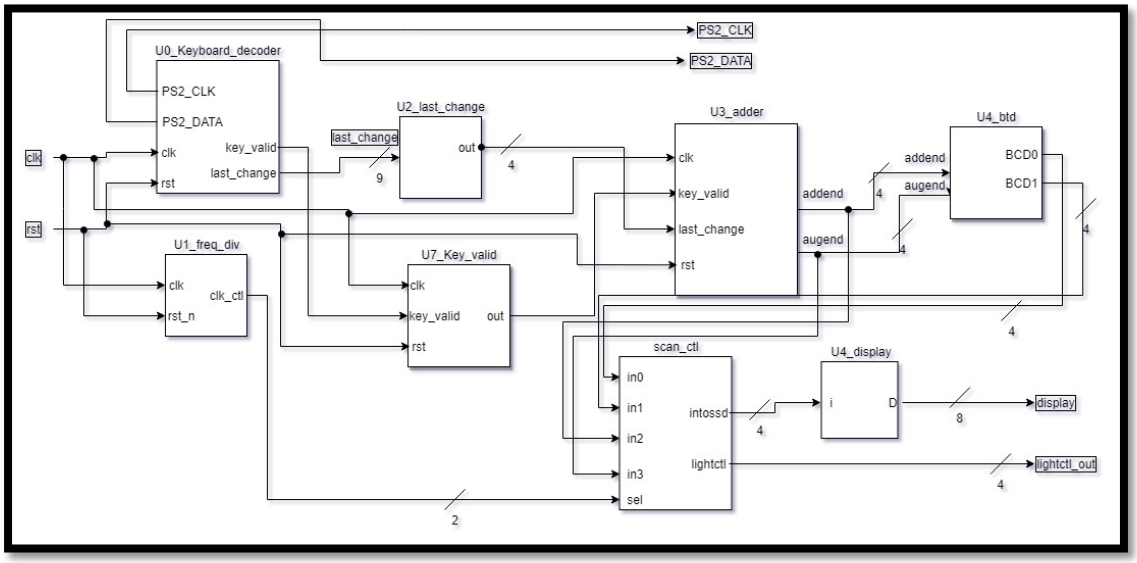
Design Specification

✓ I/O

Inout PS2_DATA, PS2_CLK; // keyboard data
Input clk; // crystal clock
Input rst; // high active reset

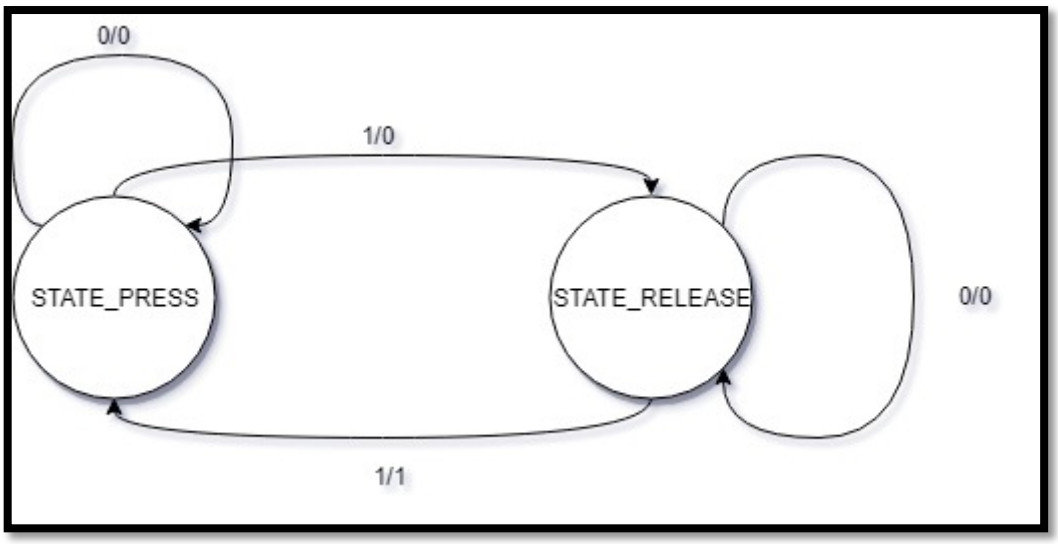
// display outputs
Output [3:0]lightctl_out;
Output [7:0]display;

✓ Logic Block:



這一題必須要在輸入的時候直接顯示被加數，加數跟答案，因此last change送進adder模組作運算的時候，直接將addend augend輸出，然後addend augend會到btd(binary to decimal，lab01使用過)模組作答案的運算。相加之後將值轉換為10進位制送進顯示模組，就完成了這一題。

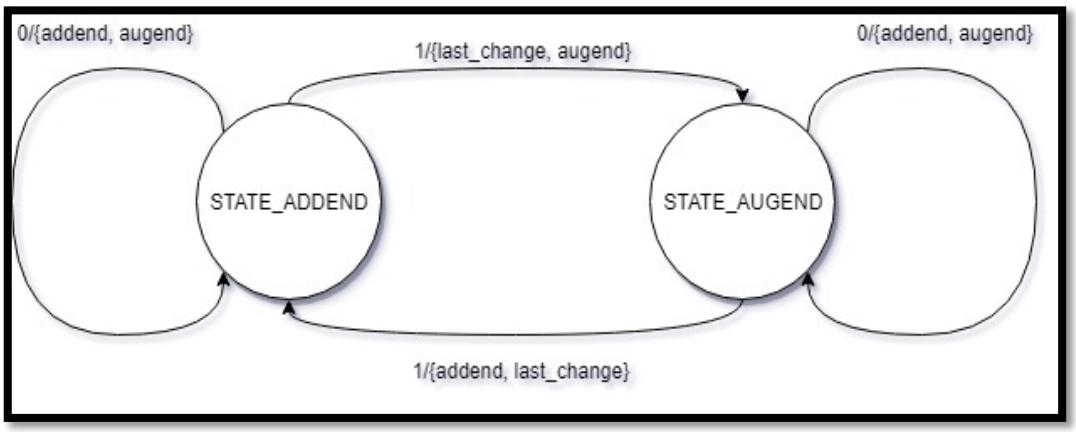
✓ Key_valid module



Input: key_valid
Output:處理過的key_valid

Input為keyboard decoder送出來的key_valid，由於按鍵release的時候也會送出訊號1，因此我們要對這個訊號作一點修正，讓他只會在按下的時候輸出1，比較實用，然後其他時候輸出0。

✓ adder module



Input: key_valid
Output: {addend_tmp, augend_tmp}

Input為前一個模組輸出的key_valid，當有一個按鍵按下去的時候input為1，然後維持一個clock cycle(100MHz)。我的設定是他會不斷在這兩個state變動，當被加數輸入完，就會進入加數的state，然後輸入完會再進入加數的state，不斷循環。

✓ Design Implementation

Inout	
PS2_CLK	*PS2_DATA*
C17	B17

Input	
clk	*rst*
W5	R2

Output ssd							
display[0]	***display[1]***	***display[2]***	***display[3]***	***display[4]***	***display[5]***	***display[6]***	***display[7]***
V7	U7	V5	U5	V8	U8	W6	W7

Output lightctl			
lightctl[0]	*lightctl[1]*	*lightctl[2]*	*lightctl[3]*
U2	U4	V4	W4

✓ Discussion

這一題也算是相當簡單，只要將前兩個顯示器接到使用者輸入的數字，然後後面兩個顯示答案即可。只是在做的時候會有overflow的問題，而改動bit數目的時候可能會有error的情形，只要將超過14的case特別拿出來討論就好。

✓ Experiment

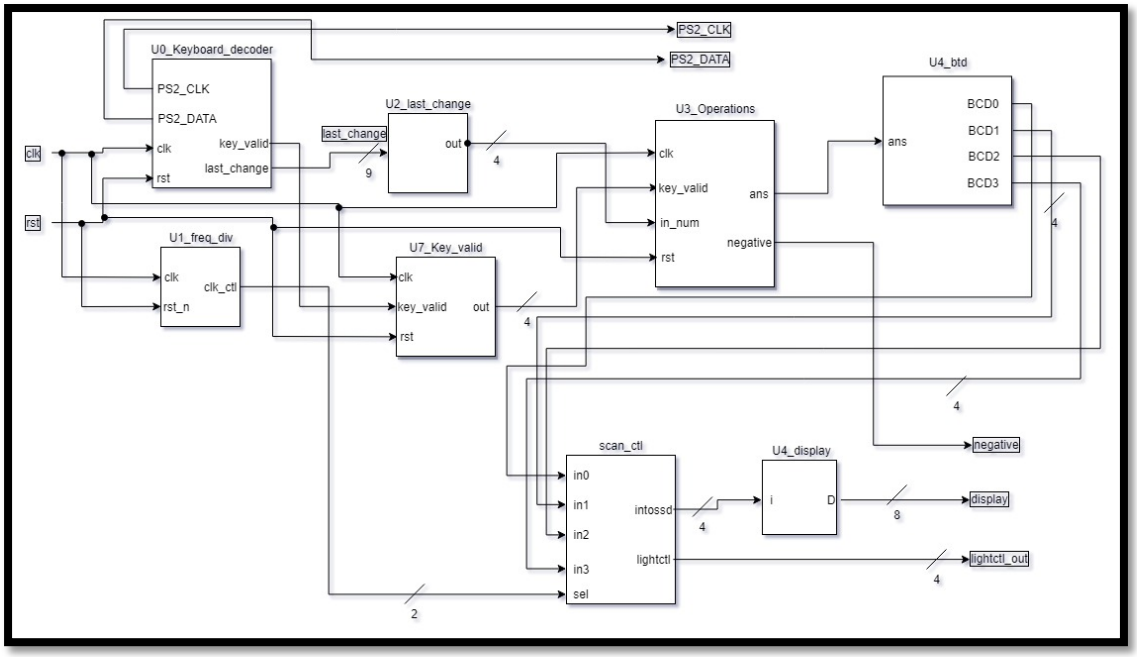
3. Implement a two-digit decimal adder/subtractor/multiplier using the right-hand-side keyboard (inside the red block). You don't need to show all inputs and outputs at the same time in the 7-segment display. You just need to show inputs when they are pressed and show the results after "Enter" is pressed.

Design Specification

✓ I/O

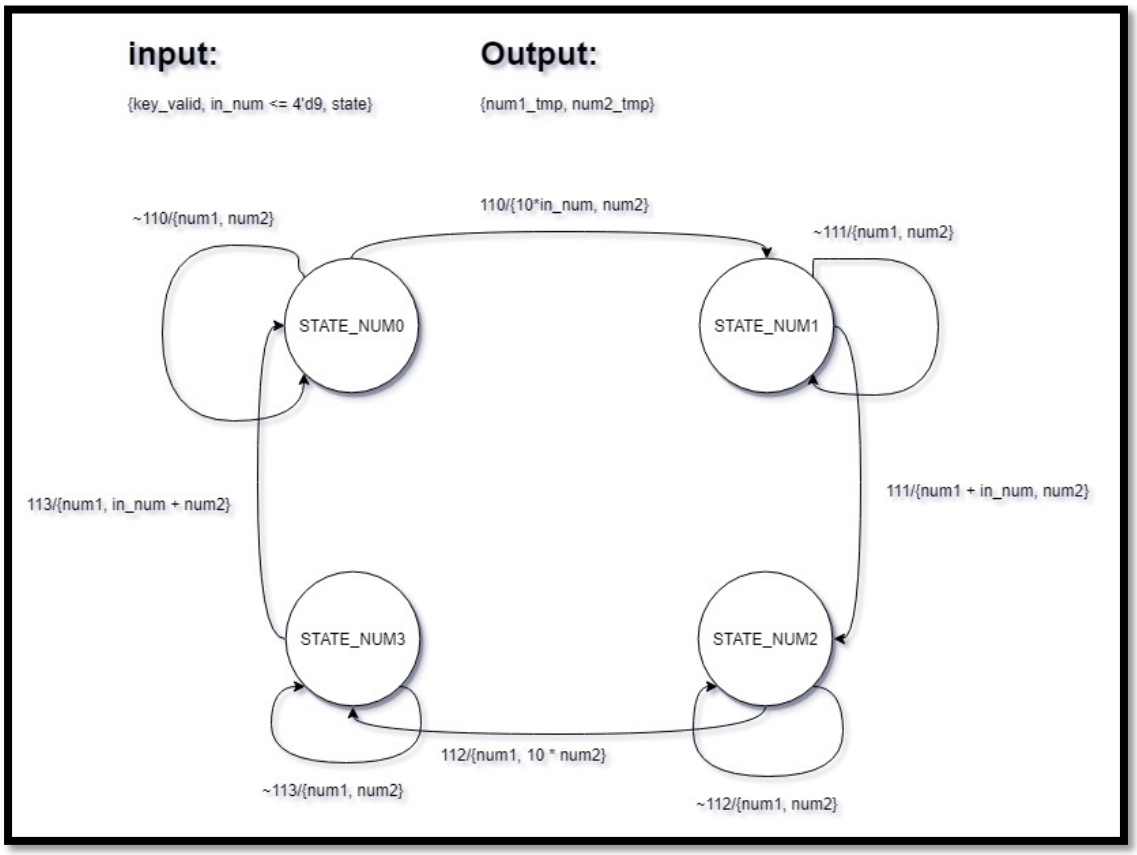
```
Inout PS2_DATA, PS2_CLK;          // keyboard data
Input clk;                        // crystal clock
Input rst;                        // high active reset

// display outputs
Output [3:0]lightctl_out;
Output [7:0]display;
Output negative;                  // led output, if the answer is negative, negative = 1
```

✓ Logic Block:

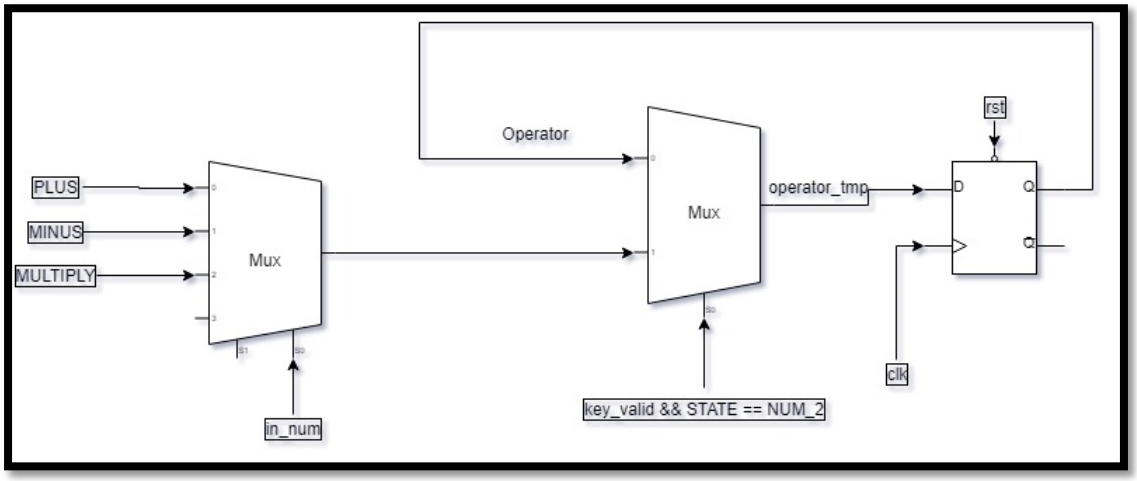


總架構與上一題類似，都是一樣經過key_valid訊號處理，然後進入last change來判斷目前輸入的東西，然後會進入複雜的Operation模組來進行題目要求的運算，最後進入Btd將答案overflow、轉換為十進位，最後接到輸出。

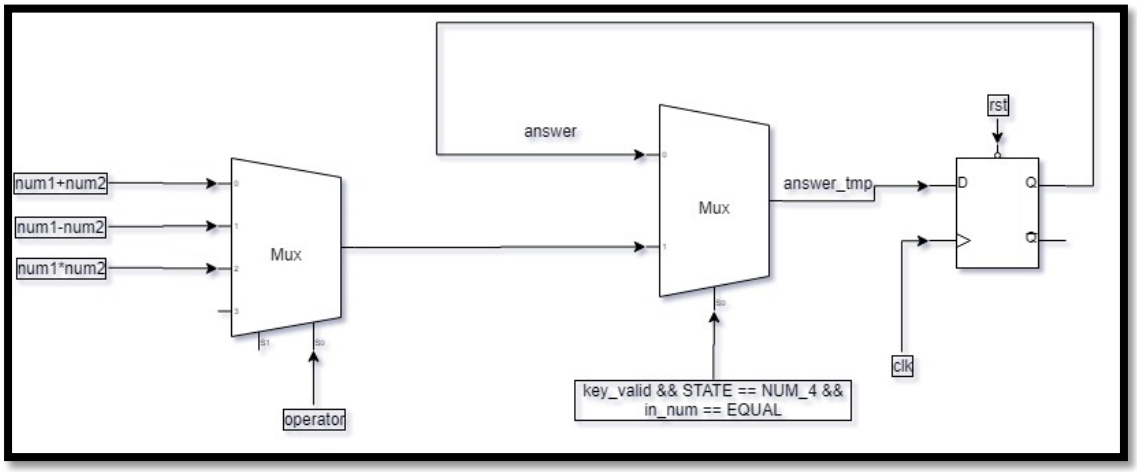
✓ Operations Module



利用key_valid, in_num <= 4’d9(亦即目前輸入不為enter或加減乘，僅為數字)，還有state(用目前為輸入第幾個數字當作state，共要輸入四個數字，所以有state 0, 1, 2, 3)當作state轉換的標準。這樣設定條件就可以保證fsm只在輸入數字的情況下做state的轉換，其餘狀況都保持在同一個state。當輸入第一個數字的時候，state會跳到輸入第二個數字的state，然後將input進來的數字assign到第一個數字的十位數，故要乘以十。當第二個數字進來的時候會跳到輸入第三個數字的state，然後input進來的數字assign給num1，會加上剛剛的assign的十位數，然後num2也是以此類推。然後輸入完最後一個數字，fsm就會進到輸入第一個數字的狀態，為使用者下一次輸入做準備。



當使用者輸入完第二個數字，才能開始輸入想要的運算。所以當state不是num2的時候，運算會一直保持使用使用者上一次設定。如果目前在輸入第二個數字的state，使用者就可以輸入想要的運算子進來，供之後系統作運算。



使用者輸入完想要的運算子，然後當系統到達輸入完所有數字的state，這時就會判斷使用者是否按下enter。當沒有按下enter，answer會一直保持在系統reset之下的零。如果使用者按下enter，系統就會根據剛剛設定operator輸出使用者的答案，將數字輸出到下一個處理二進位到時進位的模組裡面。

在這模組裡面還有一個邏輯就是判斷如果目前operation為minus，而且第二個數字大於第一個數字，系統會將negative輸出為一，到後面的led做輸出。

✓ BTD：

答案算出來會是一長串的二進位數字，我把他轉換為四格顯示的方法為，BCD3，也就是千位數，就是直接將answer除以1000所得到的值，百位數就除以100檢掉10*BCD3，其餘位數作法類似。

✓ Design Implementation

Inout	
PS2_CLK	*PS2_DATA*
C17	B17

Input	
clk	*rst*
W5	R2

Output ssd							
display[0]	***display[1]***	***display[2]***	***display[3]***	***display[4]***	***display[5]***	***display[6]***	***display[7]***
V7	U7	V5	U5	V8	U8	W6	W7

Output lightctl			
lightctl[0]	*lightctl[1]*	*lightctl[2]*	*lightctl[3]*
U2	U4	V4	W4

Negative: L1

✓ Discussion

這一題複雜度就提高很多，剛開始在設計fsm的時候我卡了很久，其實要設計的很有系統其實很困難，尤其是小細節很多。後來決定把state切的細一點，可以在美輸入一個數字的時候做很細部的調整。然後再處理答案顯示的部分，我想不到好的方法，因此使用了很軟體式的作法，利用除法將其中的位數一個一個抽出來，然後去做處理。雖然verilog讓我合成電路成功，但是實際情形我思考到底要用什麼樣的邏輯去拼湊出一個除法器呢？？

4. Implement the “Caps” control in the keyboard. When you press A-Z and a-z in the keyboard, the ASCII code of the pressed key (letter) is shown on 7-bit LEDs.

Design Specification

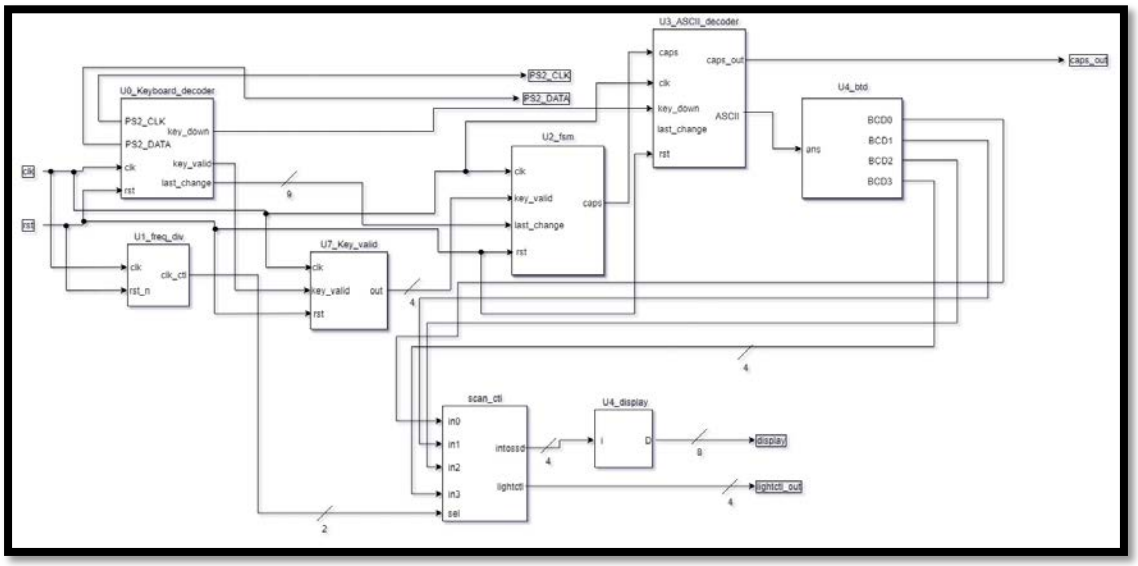
✓ I/O

```
Inout PS2_DATA, PS2_CLK;          // keyboard data
Input clk;                        // crystal clock
```

Input rst; // high active reset

// display outputs
Output [3:0]lightctl_out;
Output [7:0]display;
Output caps_out; // led signal, if it is in capitol mode
Output [6:0] // ASCII led output

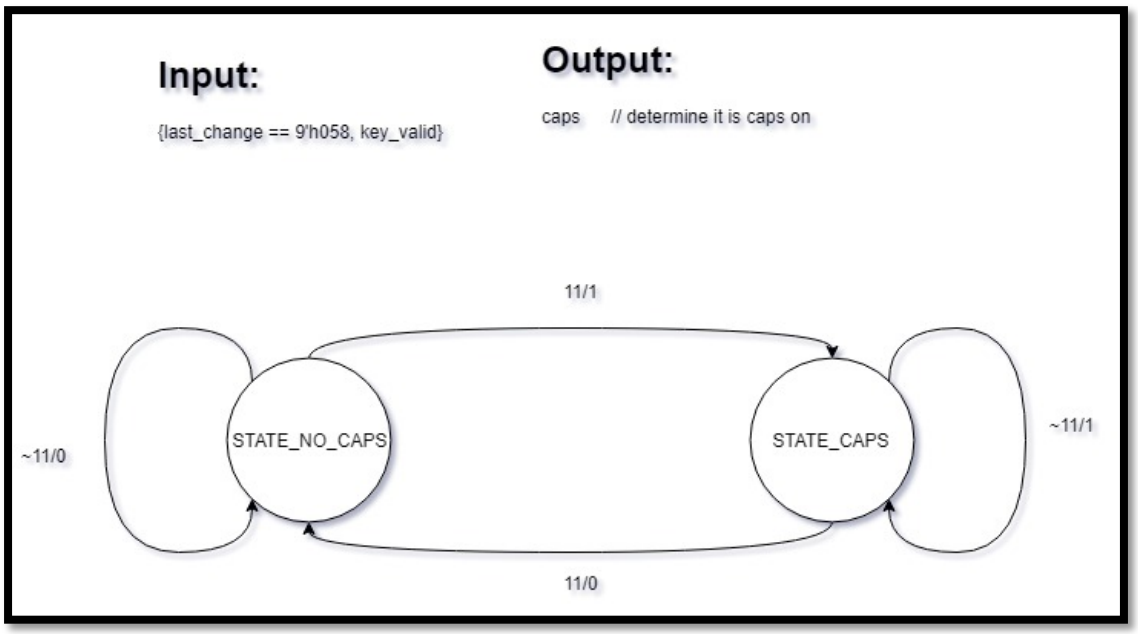
✓ Logic Block:



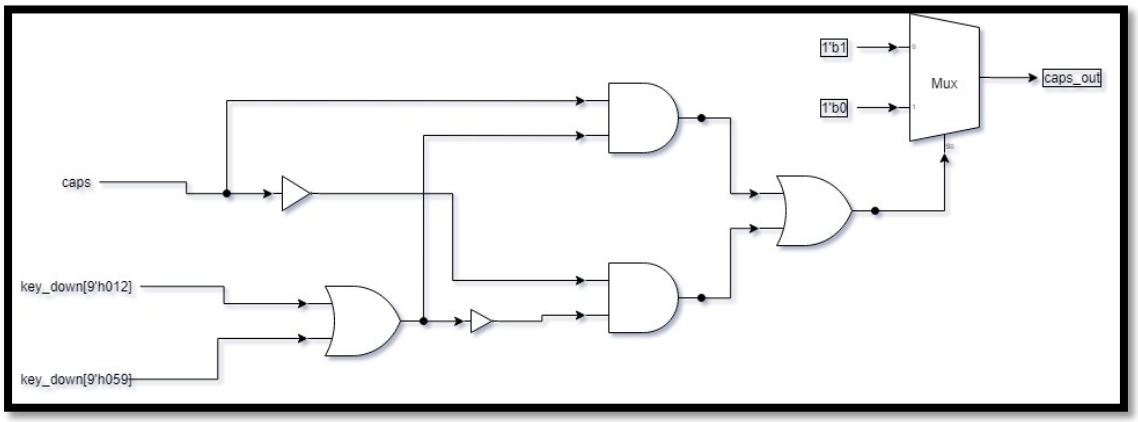
一樣，與前一題架構類似，只是需要顯示字母的ASCII CODE，因此fsm與處理輸出的部分會小有改變。一樣系統收到新的input的時候就會去判斷使用者輸入的字母，由於字母眾多，我製作一個global.v檔來轉換keyboard decoder的16進位檔轉換成字母。然後由於題目要求要有大寫了寫兩種模式，FSM主要就是在處理這部分的事情。

最後題目要求要將答案顯示在led上面，為了閱讀方便，我還製作了到七段顯示器上面顯示。

✓ FSM



如果last_change是9’h058，對應到鍵盤上的caps鍵，還有判斷當下是否有按下去，則會進行大小寫state轉換。



判斷輸出為大寫還是小寫，其中key_down[9’h012]以及key_down[9’h059]為兩個shift鍵透過與caps的state來做判斷可以檢驗當下是否輸出大寫。（若caps_out為1為輸出大寫，反之則輸出小寫）。當caps，兩個shift都為零的時候輸出小寫，還有caps被按下shift也被按下輸出小寫，反之全部輸出大寫。

✓ Design Implementation

Inout	
PS2_CLK	*PS2_DATA*
C17	B17

Input	
clk	*rst*

W5	R2

Output ssd							
display[0]	***display[1]***	***display[2]***	***display[3]***	***display[4]***	***display[5]***	***display[6]***	***display[7]***
V7	U7	V5	U5	V8	U8	W6	W7

Output lightctl			
lightctl[0]	*lightctl[1]*	*lightctl[2]*	*lightctl[3]*
U2	U4	V4	W4

Output led						
ASCII[0]	*ASCII[1]*	*ASCII[2]*	*ASCII[3]*	*ASCII[4]*	*ASCII[5]*	*ASCII[6]*
U16	E19	U19	V19	W18	U15	U14

Caps_out: L1

✓ Discussion

這一題比較討厭的是要處理複合鍵的問題，就必須使用到key down來看同時還有什麼東西按下去了。剛開始key down的使用方法我還研究了很久，後來發現會是make code的那個位置發生改變，因此只要判斷如果兩個為字同時被按下即可。這個實驗讓我瞭解到了真正見盤勢如何設定複合鍵，感覺會在期末專題中有很多的應用。

✓ Conclusion

這次的lab接觸了新的層面，讓我看見了verilog再見盤方面的應用。當然scan那個部分的原理我還是不太懂，但至少我可以利用他來製作我想要的功能。期待之後學會vga的部分可以拿來製作遊戲，感覺會真的很有趣。