106010006 黃詩瑜

Lab 9: Keyboard (Calculator)

1. Implement Key Board

1.1 Press 0/1/2/3/4/5/6/7/8/9 and show them in the seven-segment display. When a new number is pressed, the previous number is refreshed and over written.

1.2 Press a/s/m (addition/subtraction/multiplication) and show them in the seven- segment display as your own defined A/S/M pattern. When you press “Enter”, refresh (turn off) the seven-segment display.

Specification:

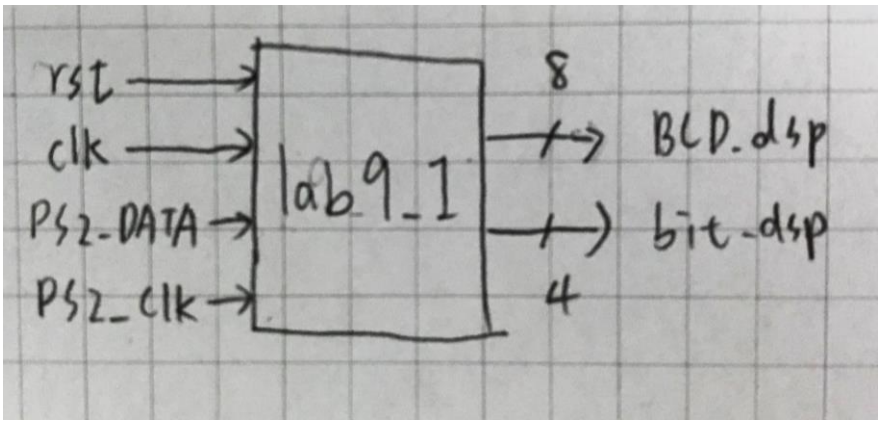
Input: clk, rst

Inout: PS2_DATA, PS2_CLK

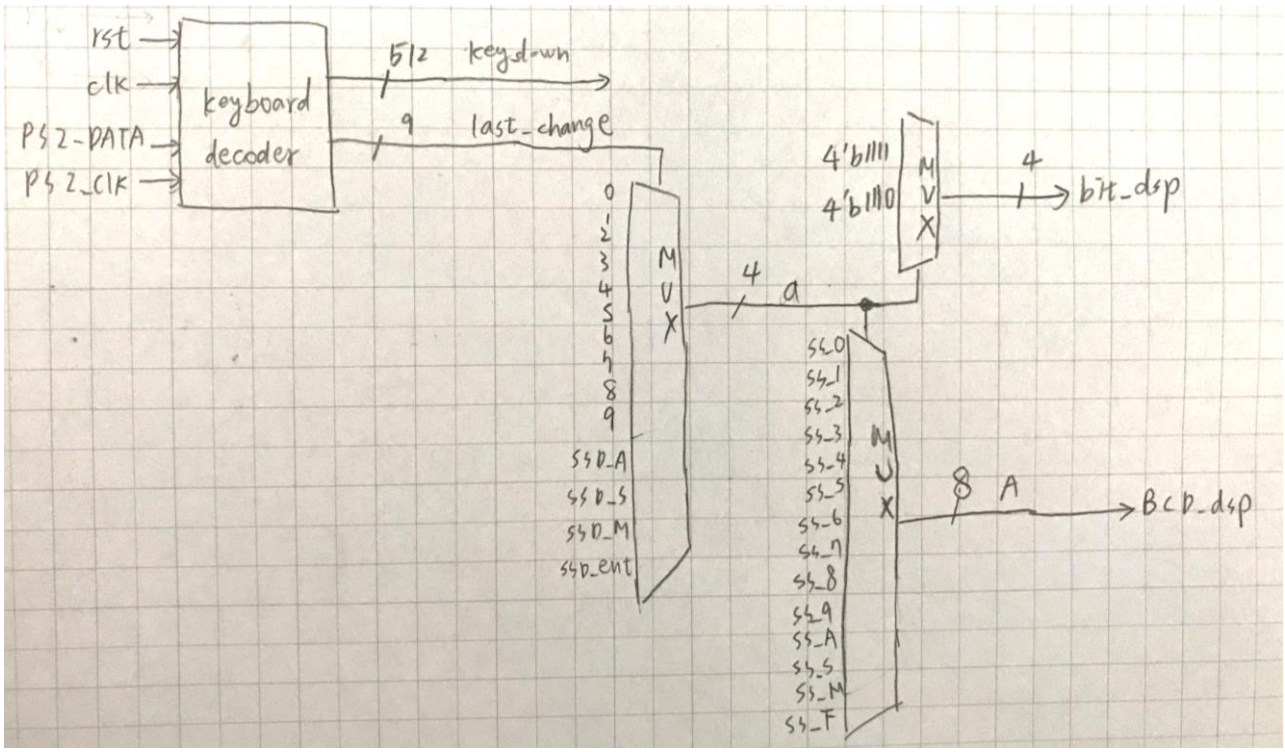
Output: [3:0]bit_dsp, [7:0]BCD_dsp

I/O	clk	rst	PS2_CLK	PS2_DATA	bit_dsp[0]	bit_dsp[1]	bit_dsp[2]	bit_dsp[3]
LOC	W5	T18	C17	B17	U2	U4	V4	W4

BCD_dsp [0]	BCD_dsp [1]	BCD_dsp [2]	BCD_dsp [3]	BCD_dsp [4]	BCD_dsp [5]	BCD_dsp [6]	BCD_dsp [7]
V7	U7	V5	U5	V8	U8	W6	W7



Implementation:



先定義鍵盤上的按鍵 KEY_0~9/A/S/M/ENT，和數字、英文對應的 7-seg LED 燈 SS_0~9/F/S/M/A

```
`define KEY_0 {1'b0 , 8'h45}
`define KEY_1 {1'b0 , 8'h16}
`define KEY_2 {1'b0 , 8'h1E}
`define KEY_3 {1'b0 , 8'h26}
`define KEY_4 {1'b0 , 8'h25}
`define KEY_5 {1'b0 , 8'h2E}
`define KEY_6 {1'b0 , 8'h36}
`define KEY_7 {1'b0 , 8'h3D}
`define KEY_8 {1'b0 , 8'h3E}
`define KEY_9 {1'b0 , 8'h46}
`define KEY_A {1'b0 , 8'h1C}
`define KEY_S {1'b0 , 8'h1B}
`define KEY_M {1'b0 , 8'h3A}
`define KEY_ENT {1'b0 , 8'h5A}
`define SS_0 8'b00000011
`define SS_1 8'b10011111
`define SS_2 8'b00100101
`define SS_3 8'b00001101
`define SS_4 8'b10011001
`define SS_5 8'b01001001
`define SS_6 8'b01000001
`define SS_7 8'b00011111
`define SS_8 8'b00000001
`define SS_9 8'b00001001
`define SS_F 8'b01110001
`define SS_S 8'b01001001
`define SS_M 8'b00010011
`define SS_A 8'b00010001
```

偵測當按下任何鍵時，依照 last_change 的值選擇 KEY_0~9/A/S/M/ENT，再將 3-bit a 選為代表的數字與英文，接下來依照 a 的值選擇對應的 8-bit SS_0~9/A/S/M 給 8-bit A，用來顯示在 7-

seg LED 燈上，將 BCD_dsp 設為Ａ，按下 enter 時，4-bit bit_dsp 為 4'b1111，讓 LED 全暗，若按下其他鍵，bit_dsp 為 4'b1110，就能顯示按下的數字與英文。

討論：

在聽老師上課講解原理的時候不大懂，以為超級難，結果設計其實還好，了解重要且會用到的功能就可以了。

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).

Specification:

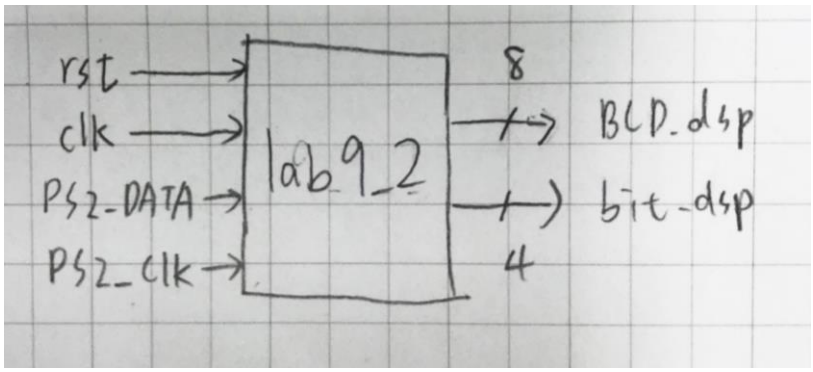
Input: clk, rst

Inout: PS2_DATA, PS2_CLK

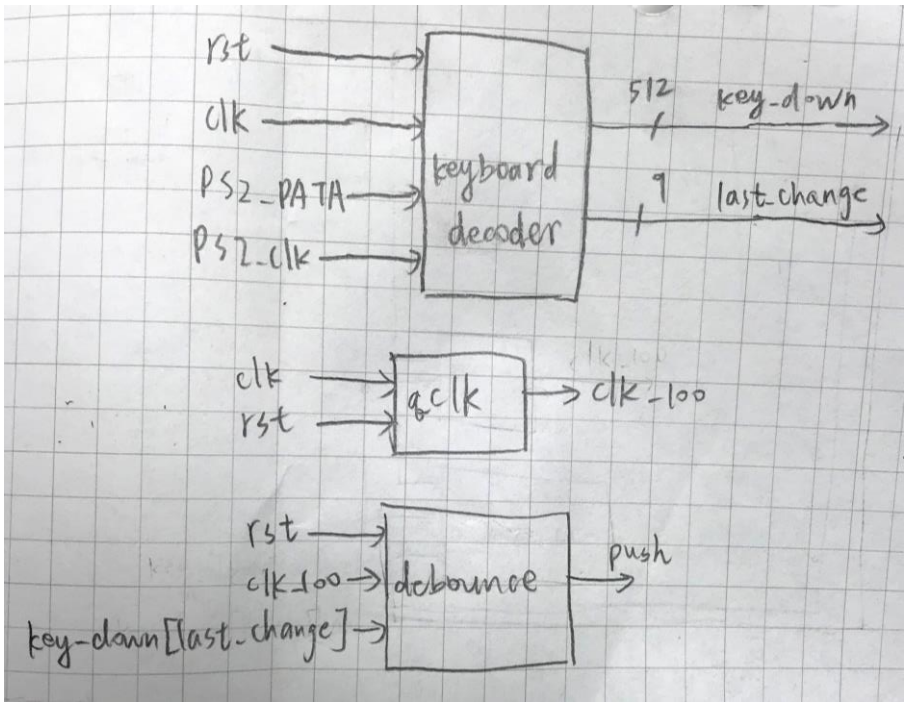
Output: [3:0]bit_dsp, [7:0]BCD_dsp

I/O	clk	rst	PS2_CLK	PS2_DATA	bit_dsp[0]	bit_dsp[1]	bit_dsp[2]	bit_dsp[3]
LOC	W5	T18	C17	B17	U2	U4	V4	W4

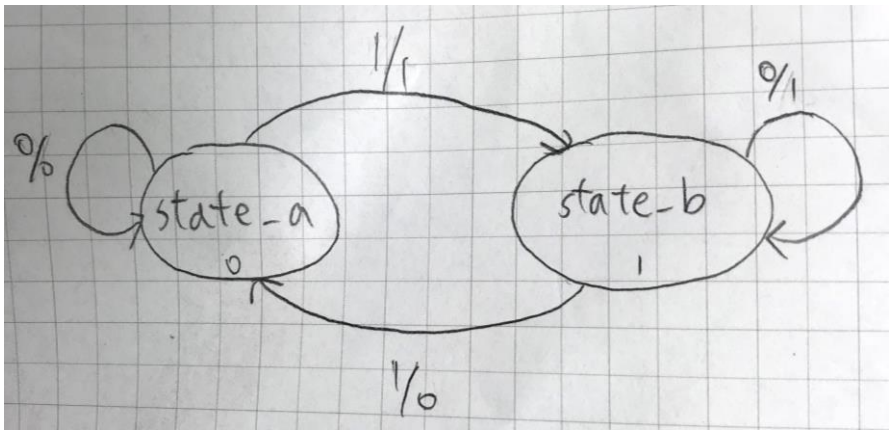
BCD_dsp [0]	BCD_dsp [1]	BCD_dsp [2]	BCD_dsp [3]	BCD_dsp [4]	BCD_dsp [5]	BCD_dsp [6]	BCD_dsp [7]
V7	U7	V5	U5	V8	U8	W6	W7



Implementation:



第二題增加了 debounce module，input 為 key_down[last_change]，按下按鍵就會經過 debounce 和 onepulse 的處理，產生 output push，也增加了 qclk module，產生 output clk_100，是 100HZ 的 clock，用於 debounce 和 onepulse 中。



首先依照 last_change 的值將 5-bit in 選為對應的數字，設兩個 state，在 state_a 按下 0~9 的按鍵且 push 為 1 時，就會跳到 state_b，且 2-bit give 值為 1，在 state_a 按下其他按鍵，就會保持在 state_a 且 give 為 0，當在 state_b 按下 0~9 的按鍵且 push 為 1 時，就會跳到 state_a，且 give 值為 2，反之則保持在 state_b 且 give 為 0。

設一組 flip-flop，5-bit a,b 一開始都是 0，當 give 為 0，a_tmp 為 a，b_tmp 為 b，當 give 為 1，a_tmp 為 in，b_tmp 為 0，當 give 為 2，a_tmp 為 a，b_tmp 為 in，所以當按下 0~9 按鍵，就會將數字給 a，b 就是 0，再按另一個數字，a 值不變，b 值為按下的數字。

設 5-bit c 為 a+b，5-bit c1 為 c/10，5-bit c2 為 c%10，這樣 a+b 的值就能顯示在 7-seg LED 燈上了

討論：

這題我用了 state 來設計，後來做到第三題時才發現其實不用那麼複雜，就有使用另一種方法。

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.

Specification:

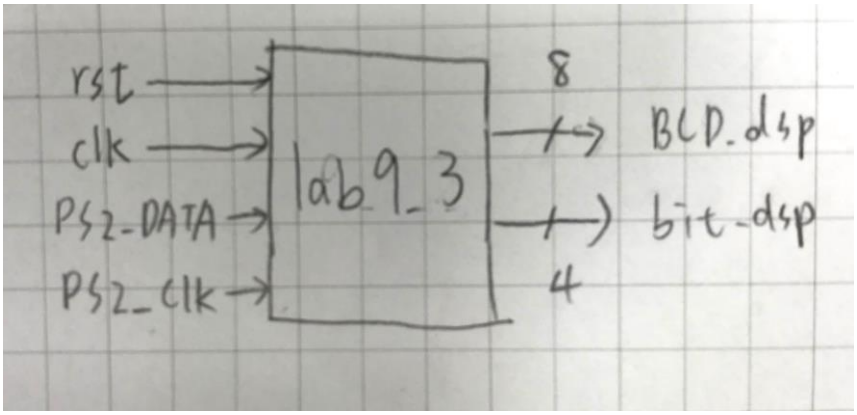
Input: clk, rst

Inout: PS2_DATA, PS2_CLK

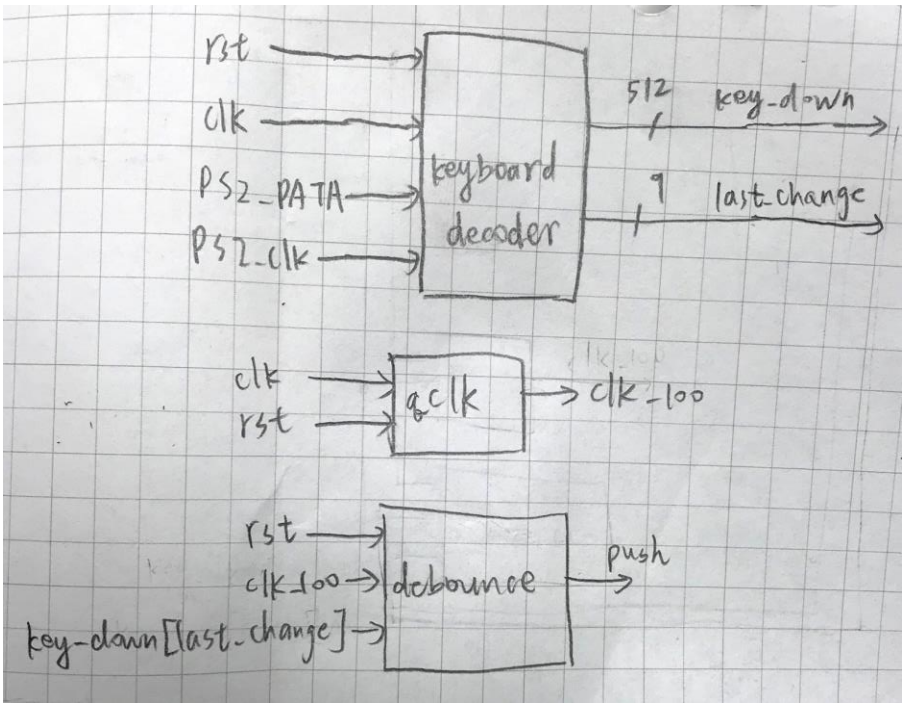
Output: [3:0]bit_dsp, [7:0]BCD_dsp

I/O	clk	rst	PS2_CLK	PS2_DATA	bit_dsp[0]	bit_dsp[1]	bit_dsp[2]	bit_dsp[3]
LOC	W5	T17	C17	B17	U2	U4	V4	W4

BCD_dsp [0]	BCD_dsp [1]	BCD_dsp [2]	BCD_dsp [3]	BCD_dsp [4]	BCD_dsp [5]	BCD_dsp [6]	BCD_dsp [7]
V7	U7	V5	U5	V8	U8	W6	W7



Implementation:



這題也是有 debounce 和 qclk modules，首先依照 last_change 的值將 4-bit in 選為對應的數字，設定一組 flip-flop，5-bit en 一開始為 0，當按下 A/S/M 鍵就會加 1，另一組 flip-flop，一開始 7-bit a1,a2 為 0，當 en 為 0 且按下 0~9 的鍵，a2 的值為 a1，a1 的值為 in，其他則 a2 的值為 a2，a1 的值為 a1，所以當按下 0~9 鍵，會將數字給 a1，a2 是 0，這時按下 A/S/M 時 en 變 1，a1,a2 的值就會固定，若這時繼續按下 0~9，a1 原本是 in 就會傳給 a2，然後 a1 為新的 in 值，還有另一組 flip-flop 是 7-bit b1,b2，差別只有當 en 為 1 且按下 0~9 的鍵，b2 的值為 b1，b1 的值為 in，所以當按下 A/S/M 之後再按下的 0~9 會傳給 b1,b2 這組。

一組 FF，7-bit a,b 一開始為 0，之後 a 為 10*a2+a1，b 為 10*b2+b1，按下的 1 位/2 位數字就傳給了 a,b，一組 FF，一開始 2-bit op 為 0，當按下 M 鍵，op_tmp 為 1，按下 A，op_tmp 為 2，按下 S，op_tmp 為 3，按下其他鍵則 op_tmp 為 op，當 op 為 1 時，15-bit sum 為 a*b，op 為 2，sum 為 a+b，op 為 3 且 a>b，sum 為 a-b，若 op 為 3 但 a<b，則 sum 為 b-a。

設 4-bit s1,s2,s3,s4，s4 為 sum/1000 代表千位數，s3 為 sum/100%10 代表百位數，s2 為 sum/10%10 代表十位數，s1 為 sum%10 代表個位數。

顯示的部分，當 in 為 SSD_A/S/M 選擇 8-bit ASM 為 SS_A/S/M，a1,a2,b1,b2,s1,s2,s3,s4 也選擇對應的 A1,A2,B1,B2,S1,S2,S3,S4，當按下 enter 鍵且為減運算，當 a>b 顯示的 4 個數字是 S4,S3,S2,S1，當 a<b 則為一個負號,S3,S2,S1，當按下 enter 鍵但為其他運算則顯示的 4 個數字是 S4,S3,S2,S1，在按下 A/S/M 顯示為 ASM，當 en 為 0 時顯示為 A2,A1，當 en 為 1，顯示為 B2,B1。

討論：

其實一開始我打算設計能夠連續運算，而不是做完一次運算後要按 reset 才能重新算，我原本分開成很多 module，結果按下按鍵完全沒反應，也找不到錯誤，就把全部的 code 打在一個 module，按下按鍵就有反應了，但是按下 enter 結果還是沒跑出來，我就把多的 code 刪掉，設計成只做一次運算就好，結果到最後才發現是 enter 鍵定義錯誤，沒有設到右邊那一區的鍵盤，我下次要多加細心才行。

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.
4.1 Press “Caps Lock” key to change the status of capital/lower case on the keyboard. Use a led to indicate the status of capital/lowercase in the keyboard and show the ASSCII code of the pressed key one 7-bit LEDS.
4.2 Implement the combinational keys. When you press “Shift” and the letter keys at the same time. The 7-bit LEDs will show the ASCII code of the uppercase/lowercase of the pressed letter when the “Caps Lock” is at the lowercase/uppercase status.

Specification:

Input: f_cst, rst

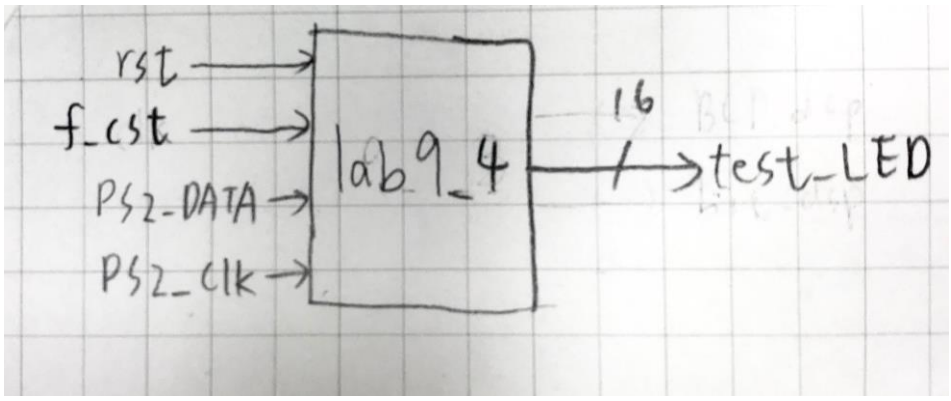
Inout: PS2_DATA, PS2_CLK

Output: [15:0]test_LED

I/O	f_cst	rst	PS2_CLK	PS2_DATA
LOC	W5	T18	C17	B17

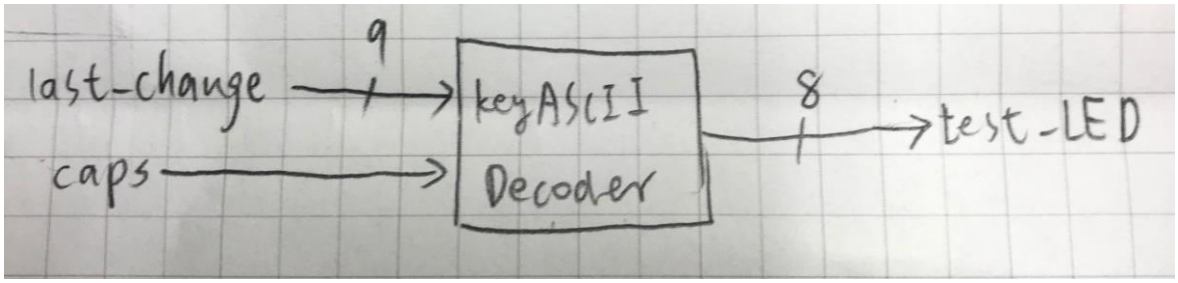
test_LED [0]	test_LED [1]	test_LED [2]	test_LED [3]	test_LED [4]	test_LED [5]	test_LED [6]	test_LED [7]
U16	E19	U19	V19	W18	U15	U14	V14

test_LED [8]	test_LED [9]	test_LED [10]	test_LED [11]	test_LED [12]	test_LED [13]	test_LED [14]	test_LED [15]
V13	V3	W3	U3	P3	N3	P!	L1



Implementation:

當按下 caps lock 鍵後 1-bit caps_lock，會由 0 變 1，再按一次就由 1 變 0，當 caps_lock 為 1 且 key_down[`KEY_SFT]為 0，代表沒按 shift 鍵，則 caps 為 1，當 caps_lock 為 0 且 key_down[`KEY_SFT]為 1，代表有按 shift 鍵，則 caps 也是 1，其他情況則 caps 為 0。



另一個 module input last_change 會偵測按下的按鍵給 8-bit ASCII_對應的 ASCII_A~Z 的值，當 input caps 為 1 output 8-bit ASCII 為 ASCII_，當 caps 為 0，ASCII 為 ASCII_加 32。

將 ASCII 給 test_LED[7:0]，caps_lock 給 test_LED[15]，有沒有按下 caps lock 就可以由一個 LED 燈看出來了。

討論：

第 4 題比第 3 題相對簡單，只是在定義鍵盤上的 A~Z 26 個字母花了比較長的時間。

結論：

只要了解 last_change, key_down 等意思，其實設計並不難，只是要注意要改成 posedge rst，我有些 module 是複製之前的檔案，忘記檢查結果就跑不出來了。