Lab08 (109061256 陳立萍)

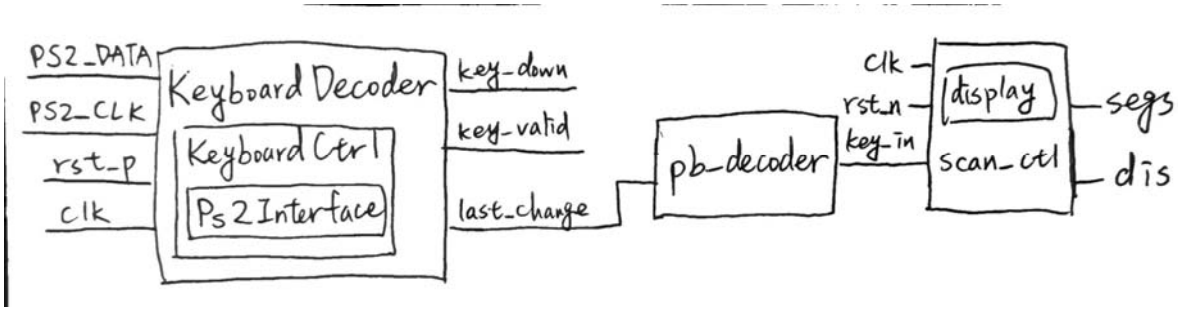
Lab8_1: Implement key board using the left-hand-side keyboard (inside the black blocks).

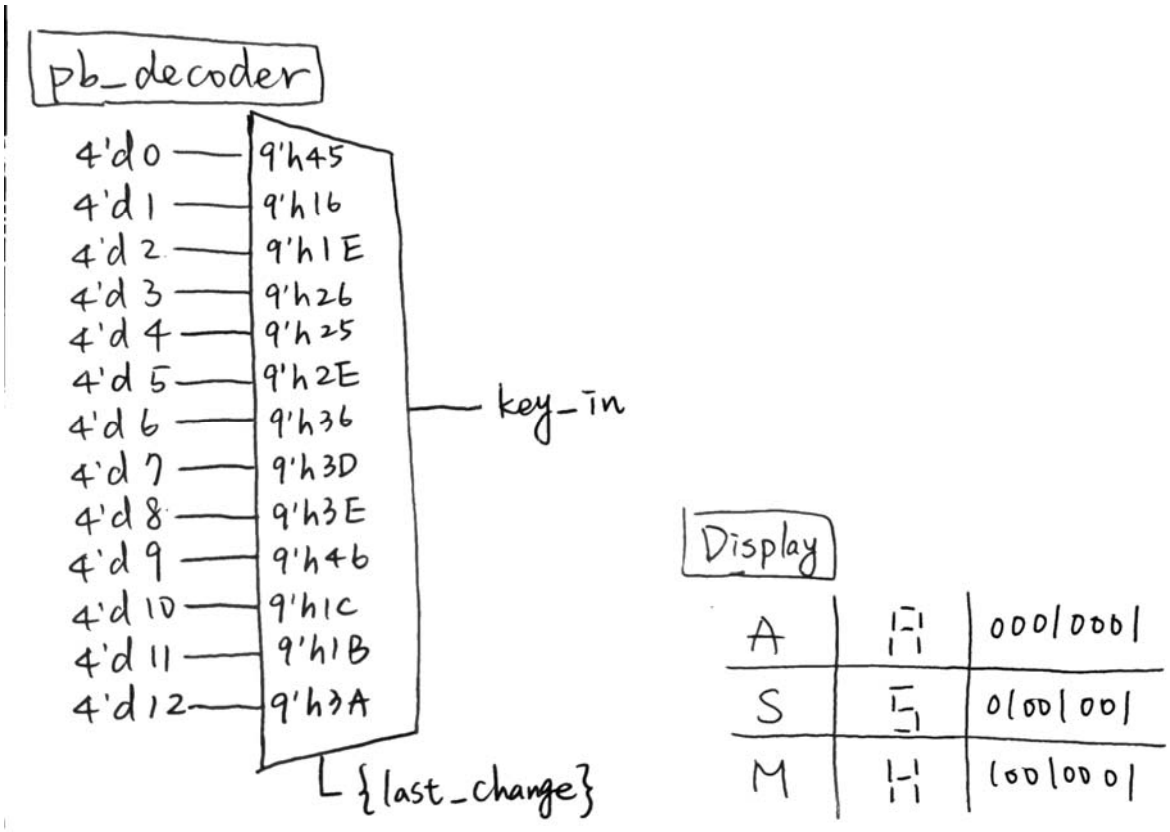
1.1 Press 0/1/2/3/4/5/6/7/8/9 and show them in the seven-segment display. When a new number is pressed, the previous number is refreshed and overwritten.
1.2 Press a/s/m (addition/subtraction/multiplication) and show them in the seven-segment display as your own defined A/S/M pattern. When you press "Enter", refresh (turn off) the seven-segment display.

Design Specification

inout : PS2_DATA, PS2_CLK
input : clk (100Mhz), rst_n (reset)
output : [7:0]segs(七段段顯示器圖形), [3:0]dis(四個七段顯示器)

logic diagram





I/O pin assignment:

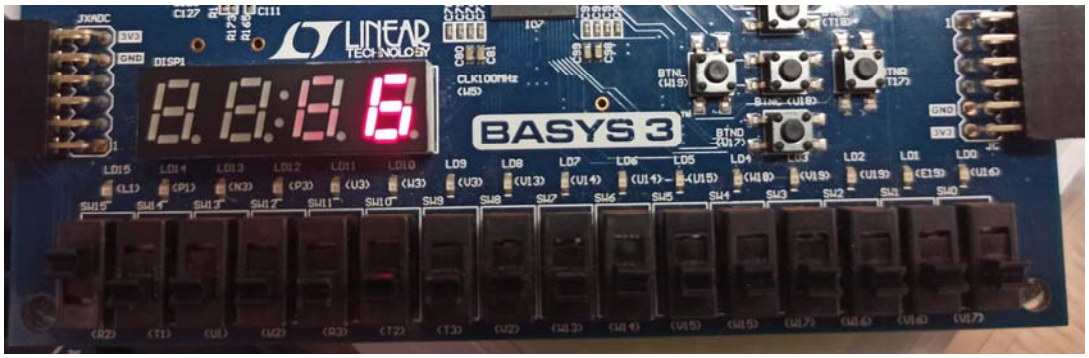
PS2_DATA	PS2_CLK	rst_n	clk
B17	C17	R2	W5

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

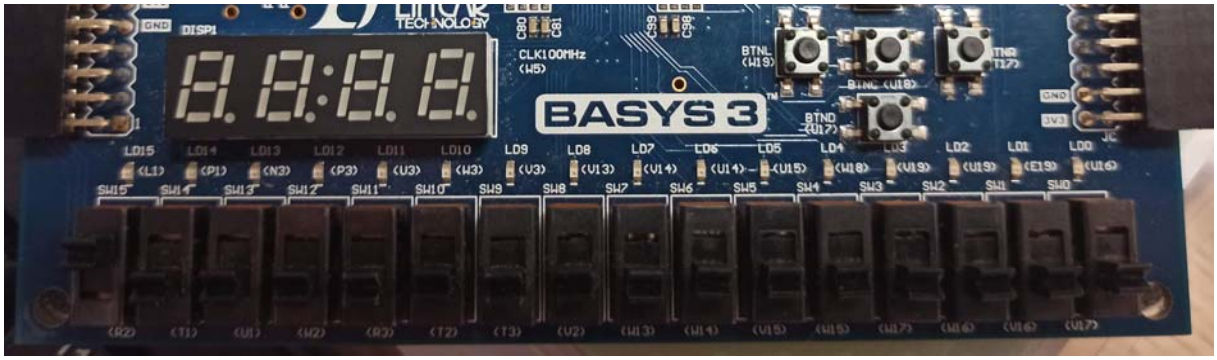
dis[3]	dis[2]	dis[1]	dis[0]
W4	V4	U4	U2

Design Implementation

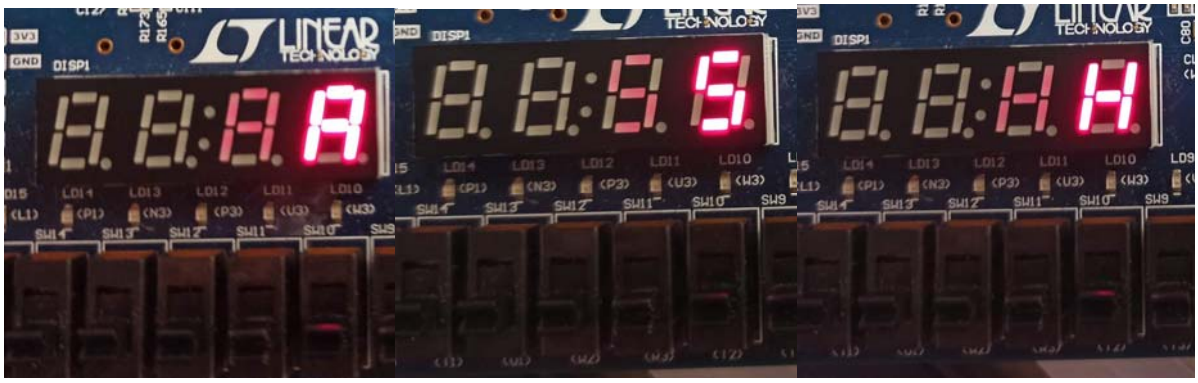
press 6



press “enter”



press a/s/m



設計方法：

把 Keyboard.v, KeyboardCtrl.v, Ps2Interface.v include 進來，再利用 Keyboard.v 的 output 來做我們要的處理。另外 scan_ctl.v：製造視覺暫留，display.v：七段顯示器輸出，之前已介紹過，不再重複介紹。

＊pb_decoder.v：利用一個多工器將.v 的 last_change 輸出接進來，按照每一個 last_change 的值對應鍵盤的編碼，下去解碼為 0~9 和 A,S,M，再將解碼後的值給 key_in, key_in 會接到 scan_ctl 去做七段顯示器的輸出。

最後再用 lab8_1_top.v 把上述的.v 結合。

Lab8_2: Implement a single digit decimal adder using the keyboard as the input and display the results on the 7-segment display (The first two digits are the addend/augend, and the last two digits are

the sum).

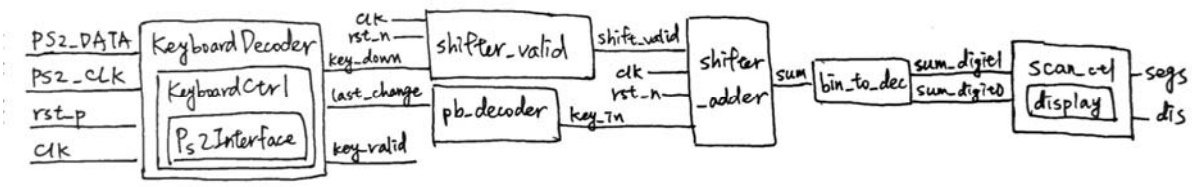
Design Specification

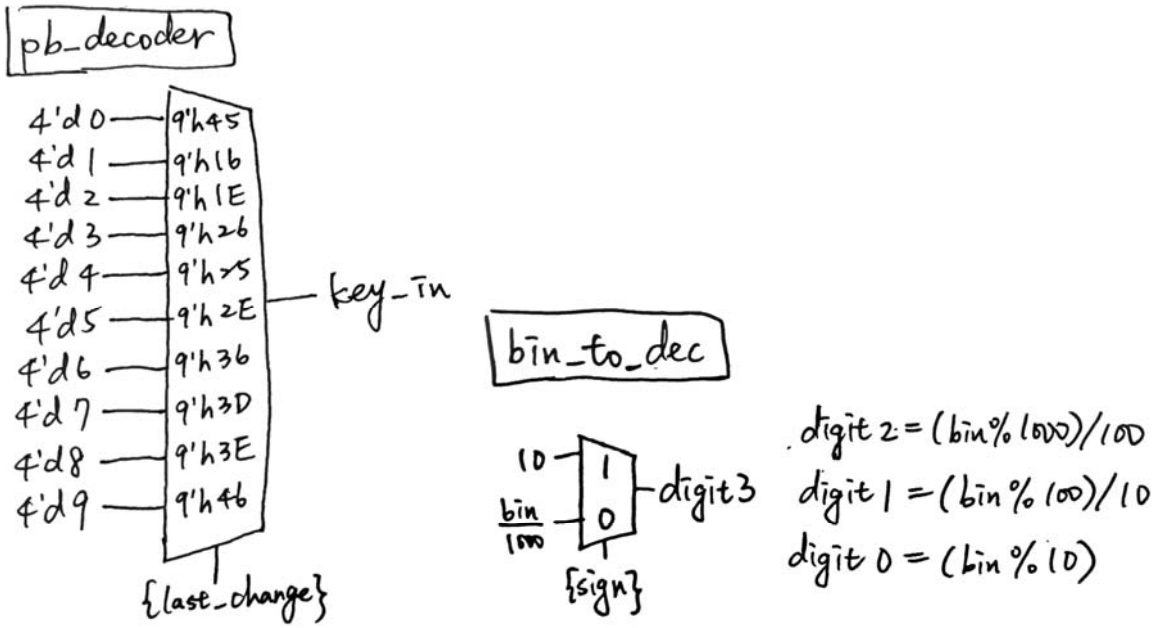
inout : PS2_DATA, PS2_CLK

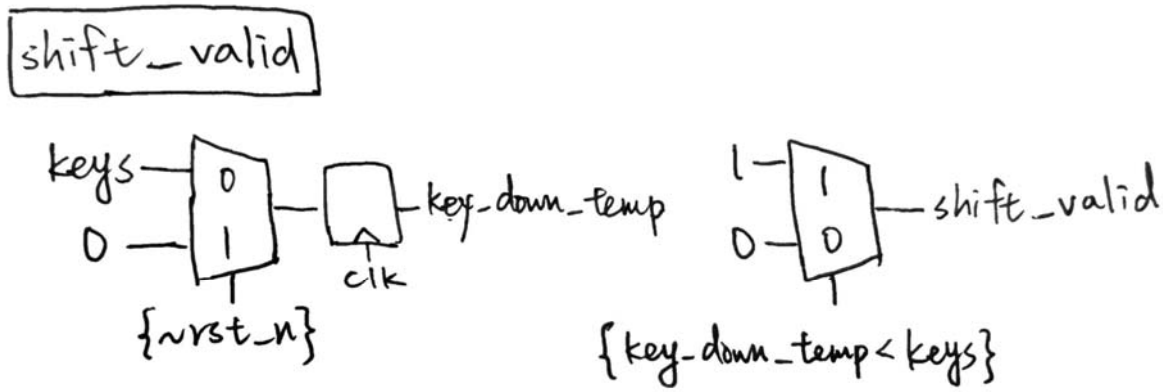
input : clk (100Mhz), rst_n (reset)

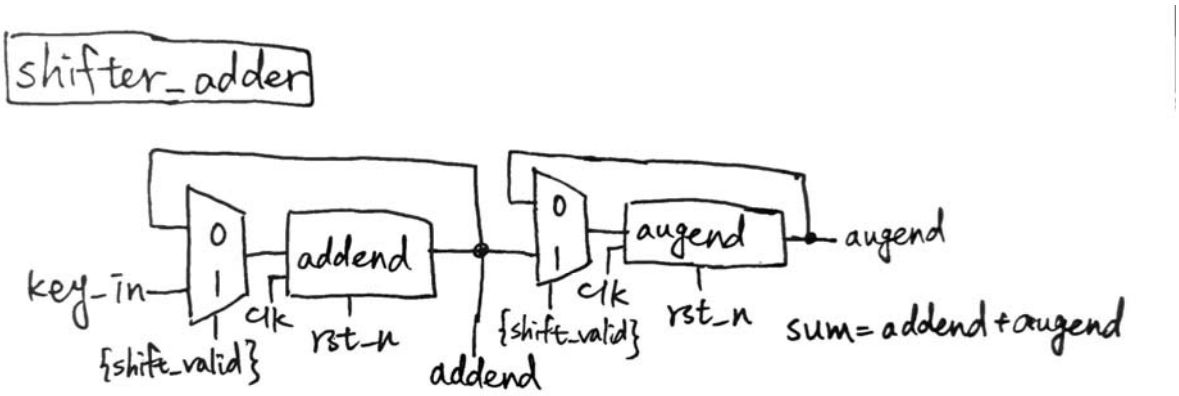
output : [7:0]segs(七段段顯示器圖形), [3:0]dis(四個七段顯示器)

logic diagram









I/O pin assignment:

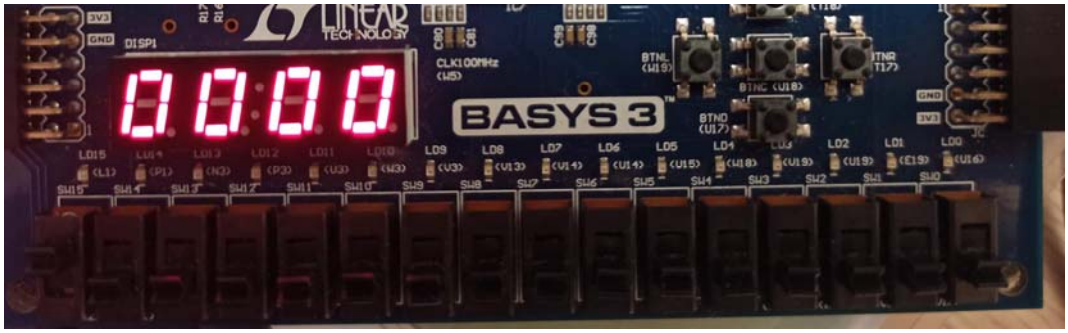
PS2_DATA	PS2_CLK	rst_n	clk
B17	C17	R2	W5

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

dis[3]	dis[2]	dis[1]	dis[0]
W4	V4	U4	U2

Design Implementation

reset



6+7=13

7+2=9

設計方法：

把 Keyboard.v, KeyboardCtrl.v, Ps2Interface.v include 進來，再利用 Keyboard.v 的 output 來做我們要的處理。另外 scan_ctl.v：製造視覺暫留，display.v：七段顯示器輸出，pb_decoder.v：由多工器做按鍵輸入解碼，之前已介紹過，不再重複介紹。

＊shift_valid.v:接收一個組合的訊號，此訊號為[511:0]中的 0~9 數字代表的組合，傳進來後會判斷這一個組合訊號是否和前一個時刻暫存器(key_down_temp)的值相同（暫存器預設為 0）。如果相同可能代表 1.一直按下同一個按鍵沒有放開，因此 shift_valid 是 0，或是 2.沒有按鍵被按下，則暫存器和 input(keys)都是 0，shift_valid 也是 0，以上兩種狀況七段顯示器都不能 shift 一個位置。而組合訊號和前一個時刻暫存器的值不相同時，則代表重新按下某按鍵的瞬間，因此 shift_valid 為 1。

＊shifter_adder.v:整合了 shift 和 adder 的功能。首先，shifter 的部分，會先接收 shift_valid 的輸出訊號進來，利用此訊號來作要不要前兩個七段顯示器往右 shift 的判斷，如果 shift 為 1，則 addend 和 augend 都會往右 shift 一個位置，反之 shift_valid 為 0 時則不 shift 維持原樣。Adder 的部分，就是簡單的加法運算，把 addend 和 augend 加在一起給 sum 做 output。

＊bin_to decv:用來做二進位轉十進位的運算，十進位的個位數即為輸入值用 10 取餘數，十位數則為輸入值處以 10。

最後再用 lab8_2_top.v 把上述的.v 結合。

3.Lab8_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.

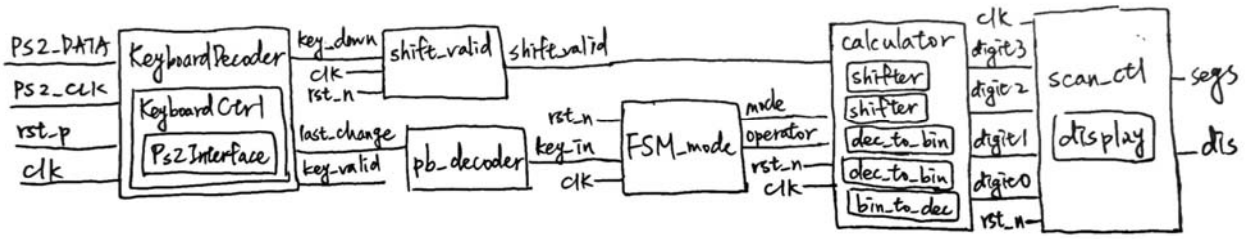
Design Specification

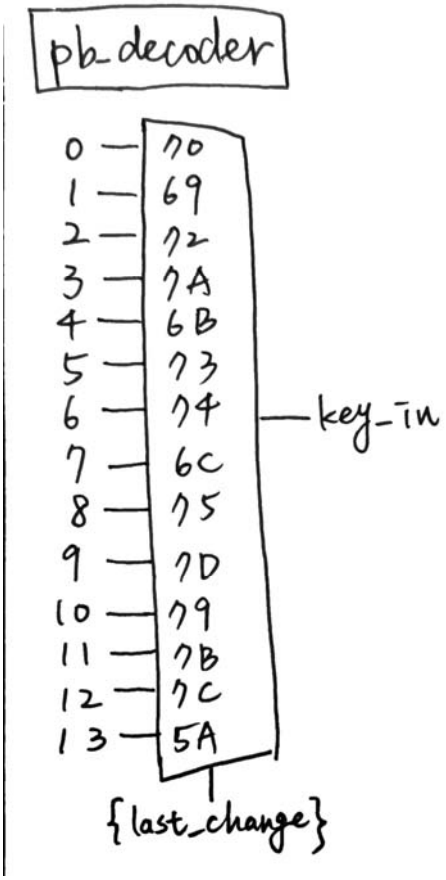
inout : PS2_DATA, PS2_CLK

input : clk (100Mhz), rst_n (reset)

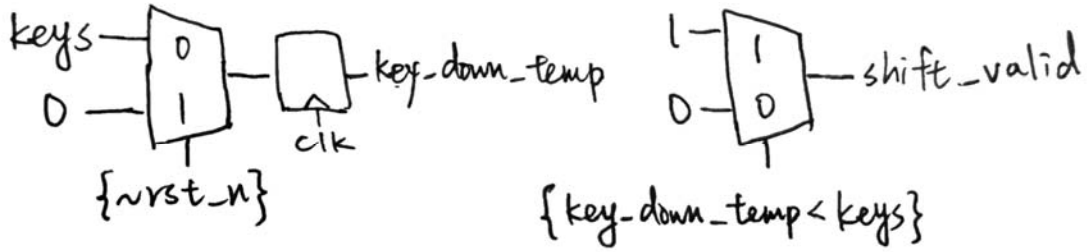
output : [7:0]segs(七段段顯示器圖形), [3:0]dis(四個七段顯示器)

logic diagram



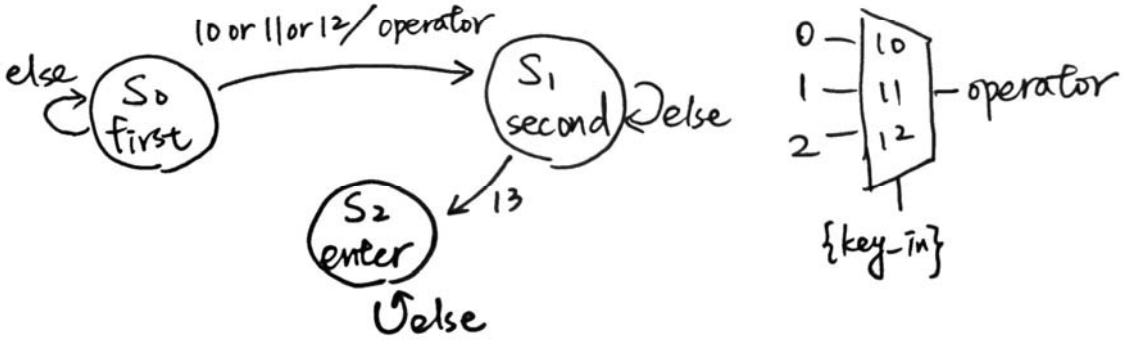


shift_valid

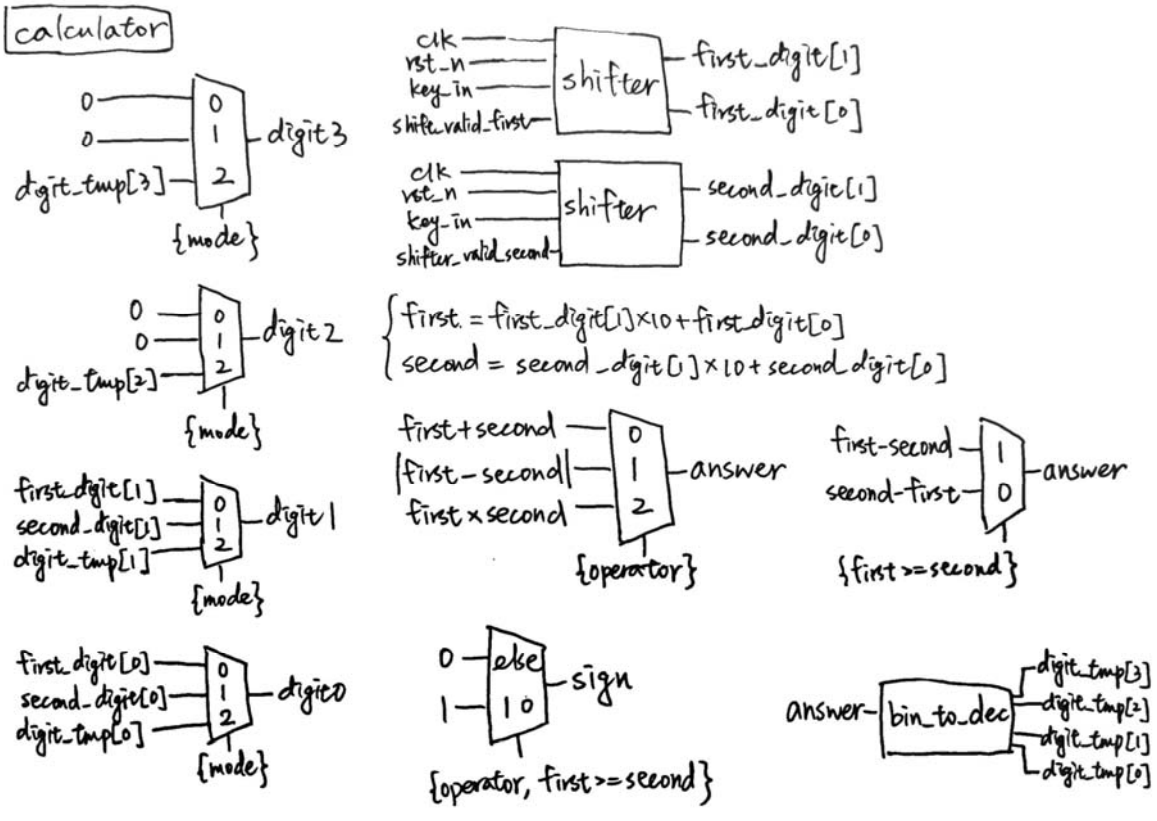


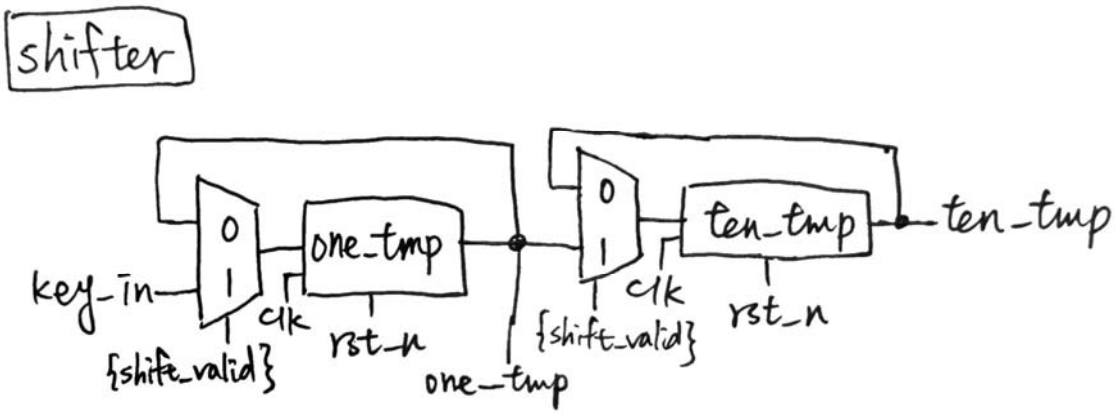
FSM_mode

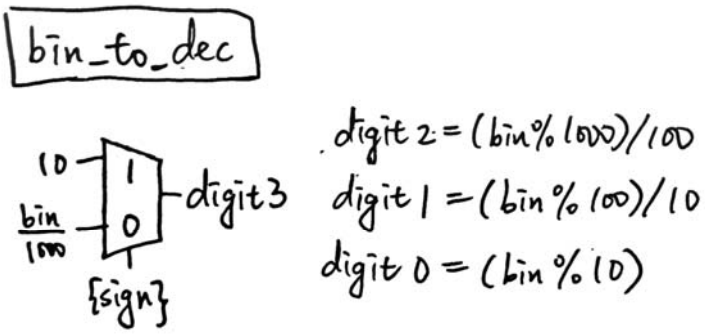
Input = key_in



calculator







I/O pin assignment:

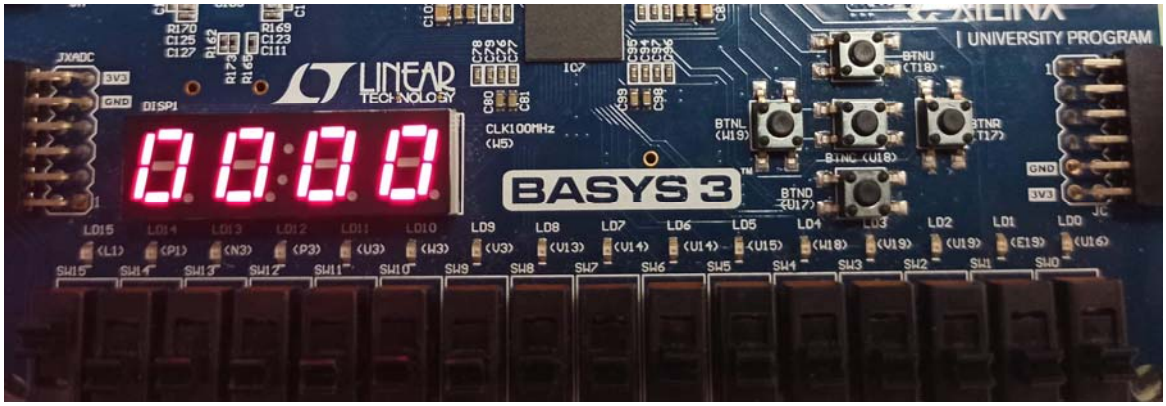
PS2_DATA	PS2_CLK	rst_n	clk
B17	C17	R2	W5

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

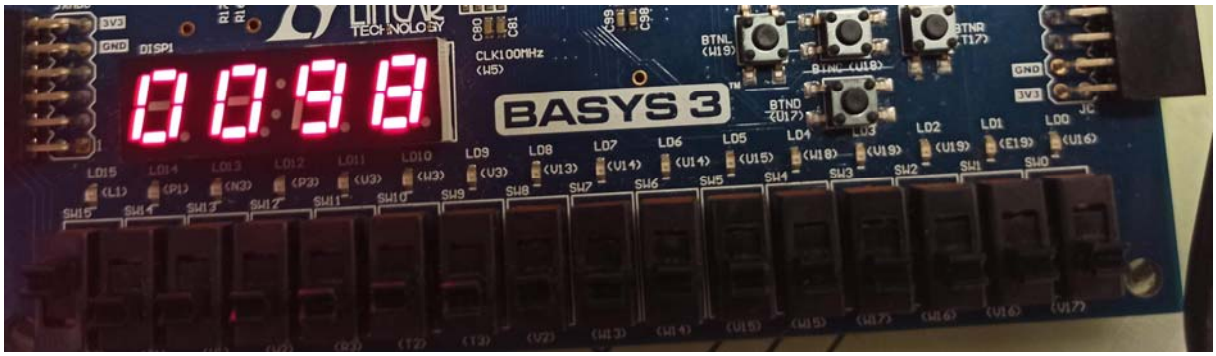
dis[3]	dis[2]	dis[1]	dis[0]
W4	V4	U4	U2

Design Implementation

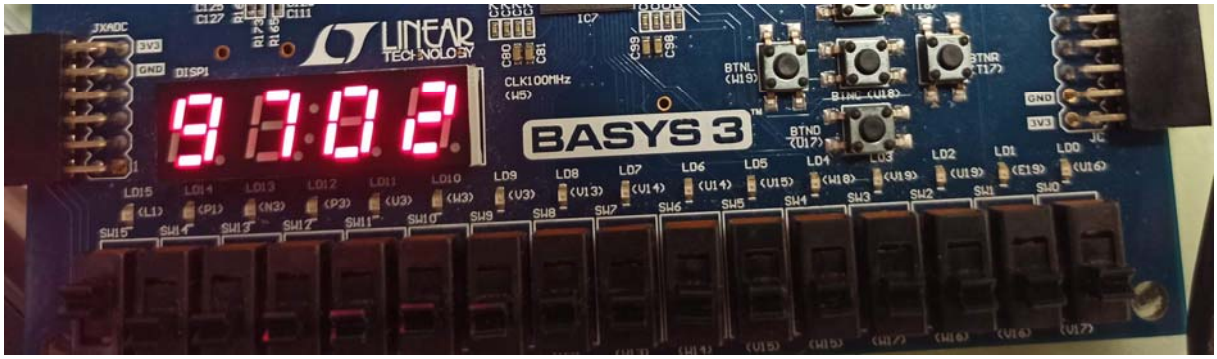
reset / press + or – or *



key in 99, and press *, then key in 98



press “enter”, 99*98 = 9702



設計方法：

把 Keyboard.v, KeyboardCtrl.v, Ps2Interface.v include 進來，再利用 Keyboard.v 的 output 來做我們要的處理。

另外 scan_ctl.v：製造視覺暫留，display.v：七段顯示器輸出，pb_decoder.v：由多工器做按鍵輸入解碼，shift_valid.v：判斷是否重新按下按鍵，之前已介紹過，不再重複介紹。

＊FSM_mode.v：分為 3 種狀態，first：第一個數輸入， second：第二個數輸入， enter：輸出答案。這裡會接 pb_decoder 的輸出進來，判斷是否按下 + / - / * / enter。reset 後會在 first 狀態，如果有按下+或-或*則會跳到 second 狀態，輸出 mode 訊號為 0，同時也輸出 operator 訊號，operator 訊號由一個多工器產生 ，輸入+ 則 operator=0，輸入 - 則

operator=1，輸入 * 則 operator=2。跳到 second 狀態後，會輸出 mode=1 訊號，如果按下 enter 鍵則會跳到 enter 狀態，輸出 mode=2 訊號，之後便會一直維持在 enter 狀態。
＊shifter.v：由 shift_valid 訊號和十位數是否為 0 來做兩個判斷來控制七段顯示器是否往左 shift，因為只有十位數為零的時候才可以再輸入個位數。因此如果 shift_valid = 1 且十位數為零時，則十位數和個位數都會往左 shift 一個位置，反之則不 shift 維持原樣。
＊dec_to_bin.v：把十位數的輸入x10 再加上個位數後輸出。
＊bin_to_dec.v：由一個多工器輸出 digit3，當輸入 sign 訊號為 1，代表負號，則 digit3 = 4'd10(display.v 定義 10 是負號)，digit2 就是(bin % 1000) / 100，digit1 為(bin % 100) / 10，digit0 等於 bin % 10。
＊calculator.v：由 4 個多工器分別輸出 digit3，digit2，digit1，digit0 給 scan.ctl.v 顯示在七段顯示器上的四個數字。多工器的判斷由輸入 mode 來控制，mode=0 則代表第一個數輸入模式，digit3=0，digit2=0，digit1=first_digit[1]，digit0=first_digit[0]，而 mode=1 代表第二個數輸入模式，digit3=0，digit2=0，digit1=second_digit[1]，digit0=second_digit[0]，最後 mode=2 時代表 enter 模式，digit3=digit_tmp[3]，digit2=digit_tmp[2]，digit1=digit_tmp[1]，digit0=digit_tmp[0]。first_digit[1]和 first_digit[0]與 second_digit[1]和 second_digit[0]由 shifter.v 產生，其中兩個不同數字的 shift_valid 分別由 shift_valid 和（mode==0/1）作 and 運算產生。接著 first_digit[1]和 first_digit[0]與 second_digit[1]和 second_digit[0]會放入 dec_to_bin 做十進位轉二進位轉換，first 和 second 會接收 dec_to_bin 的 output。再由一個多工器判斷輸入的 operator 訊號來將 first 和 second 作不同的 + / - / *運算，其中減法會多一個負數判斷，如果第一個數小於第二個數，則 sign=1 表示負號，其他狀況 sign 都是 0，表示正數。運算結果再送給 answer，最後再將 answer 放入 bin_to_dec 輸出 4 個二進位的數字給 scan_ctl.v。
最後再用 lab8_3_top.v 整合上述各個 module。

4.Lab8_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/lower case in the keyboard and show the ASSCII code of the pressed key on 7-bit LEDs.
4.2 Implement the combinational keys. When you press “Shift” and the letter keys at the same time,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.

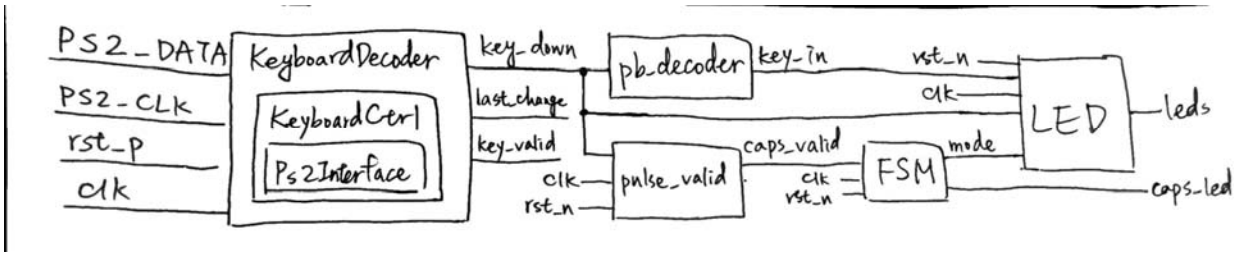
Design Specification

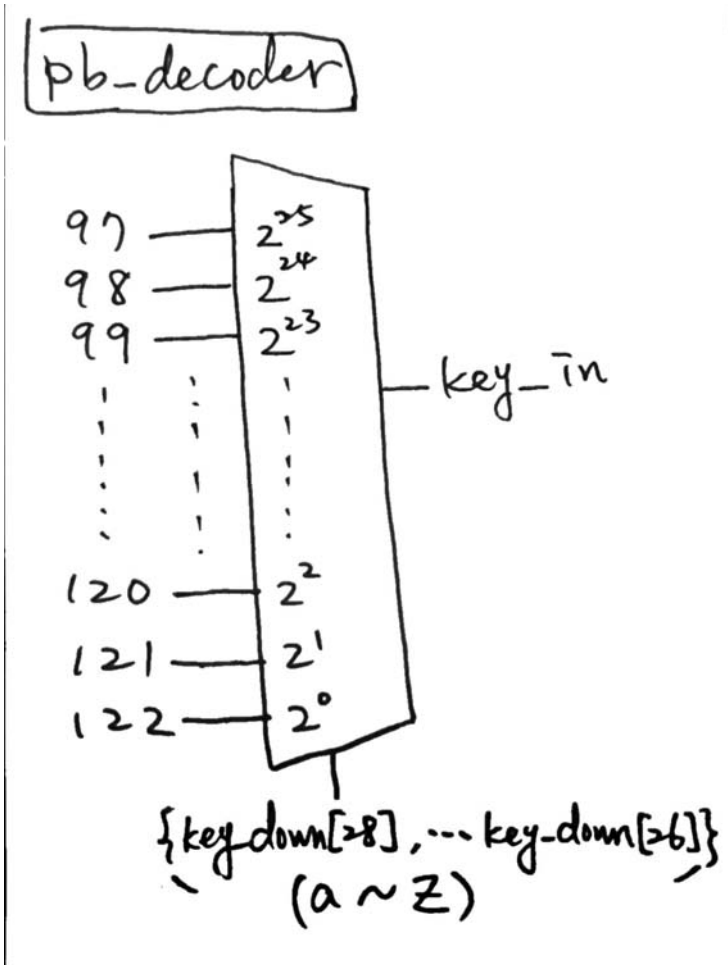
inout : PS2_DATA, PS2_CLK

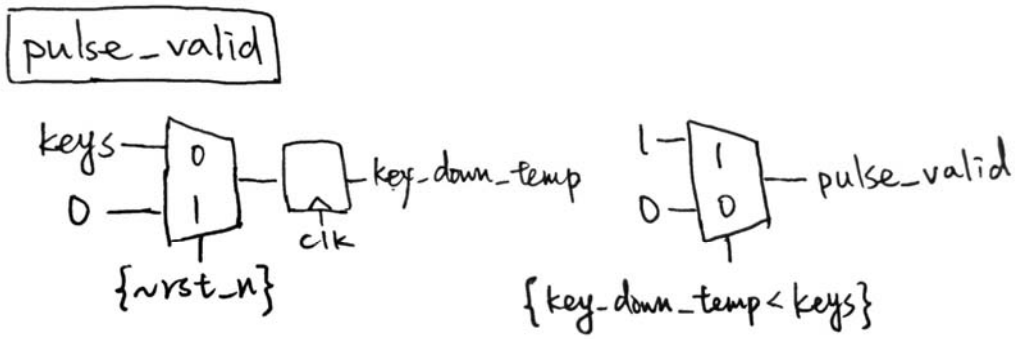
input : clk (100Mhz), rst_n (reset)

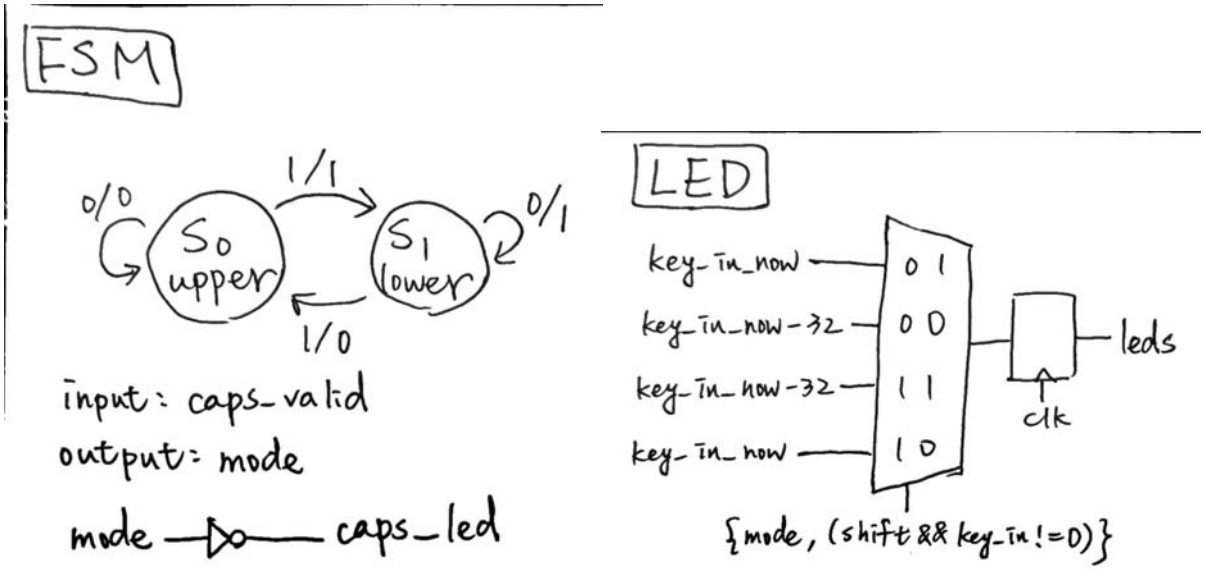
output : caps_led (大小寫 LED 指示燈), [6:0]leds (LED output)

logic diagram









I/O pin assignment:

PS2_DATA	PS2_CLK	caps_led	rst_n	clk
B17	C17	L1	R2	W5

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

Design Implementation

a(97 = 1100001)

A(65 = 1000001)

in lower case, press “shift” and “a” at the same time (1100001 -> 1000001)

設計方法：

把 Keyboard.v, KeyboardCtrl.v, Ps2Interface.v include 進來，再利用 Keyboard.v 的 output 來做我們要的處理。

＊pb_decoder.v：和前 3 題題目的 pb_decoder 一樣用多工器來輸出，不一樣的是，這裡的 input 是接由 a~z 的 key_down 組合成的 26bits 訊號進來，output 是十進位 97~122，也就是小寫字母的 asscii code。

＊pulse_valid.v：和前 3 題的 shift_valid 一模一樣，只是接的訊號只有 caps lock 的 key_down[88]。

＊FSM.v：分為兩種狀態，大寫與小寫狀態，由 pulse_valid.v 產生的 caps_valid 作為在狀態轉換的判斷，只要 caps_valid 是 1 就會跳到另一個狀態。另外，如果在大寫狀態，輸出 mode 就是 0，caps_leds 大小寫指示燈為 1，在板子上燈會亮起，反之小寫狀態輸出 mode 為 1，caps_leds 指示燈為 0，在板子上燈不會亮。

＊LED.v：主要由多個多工器運作。接 FSM.v 的 mode 還有 pb_decoder.v 的 key_in 進來，mode=0 代表大寫模式，輸出 key_in - 32（key_in 是小寫的 code)，接著再由另一個多工器來判斷是否同時按下 shift 和字母，是則輸出小寫(key_in)，否則維持大寫。若輸入 mode=1 則為小寫模式，輸出 key_in，接著再由另一個多工器來判斷是否同時按下 shift 和字母，是則輸出大寫(key_in - 32)，否則維持小寫。

最後再用 lab8_4_top.v 整合上述各個 module。

Discussion

Lab8_1

這一題因為沒有特別的要求，所以只要接 last_change 進來解碼再輸出到七段顯示器就可以了。結果為按下按鍵後會一直顯示該數字或英文字。

Lab8_2

這題難的地方是 shift 判斷與 shift_valid 的產生。因為 last_change 是最後被動到的按鍵的 code 因此按下或放開都會收到該按鍵的 code。一開始我用 last_change 作為 shift_valid 的判斷，結果每按下一個數字，前兩個七段顯示器都是該數字，因為按下該按鍵時會 shift 一次放開又再 shift 一次。因此後來我改成用 key_down，也就是如上面敘述的方法，就排除了放開按鍵的 code 輸入。

Lab8_3

這題是我作最久的一題，我一直卡在 calculator 的地方，因為有 shift 和加減的功能，還有在 reset 後要變成 0000，導致我很混亂。因為我希望在按下+ / - / *後可以跳回 0000，再讓使用者輸入，但是一開始做出來第二個數字都可以被存入，可是第一個數字不管怎麼樣都是 0，無法存住。導致 answer 一直都是第二個數字的值，後來才發現要把「如果按下+ / - / *後就要個位數和十位數歸零」拿掉，因為當沒有按下任何按鍵，shifter 的 input ：key_in 就會是 0000 也就是會自動歸零，如果我再讓他按下+ / - / *後歸零，就會把第一個數的值覆蓋掉。這個部分的 bug 我找了 2 天，好險後來改掉那一行就成功了。

Lab8_4

這一題難度和第三題比起來下降很多，比較有難度的部分是 shift 的部分，因為一開始也是用 last_change 訊號進去 pb_decoder，所以當同時按下 shift 和字母後，先放 shift 的話就會暗掉，然後再放字母又亮起來，後來我也是和第二題一樣改成 key_down 訊號輸入 pb_decoder 就改掉這個 bug 了。

Conclusion

這次的 lab 在我做完後覺得其實和聲音那個 lab 比起來沒有很困難，只是因為我對於老師給的 KeyboardDecoder.v 的 output 不清楚要怎麼使用。起初我一直以為 last_change 就是按下時才會傳送該按鍵的編碼，後來才知道原來按下與放開都是老師所說的「按鍵被更動」，都會傳出該按鍵的編碼。因此後來的題目我大多改用 key_down 下去解碼，只是在找那個按鍵的 key_down 編號要花一點時間。這個 lab 學到了好多，雖然很累，但是真的很有趣，尤其是大小寫的那題，結果輸出真的跟鍵盤的顯示一模一樣。

References

教授授課投影片：語法運用，符號運用，設計觀念，鍵盤解碼觀念。