**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.**

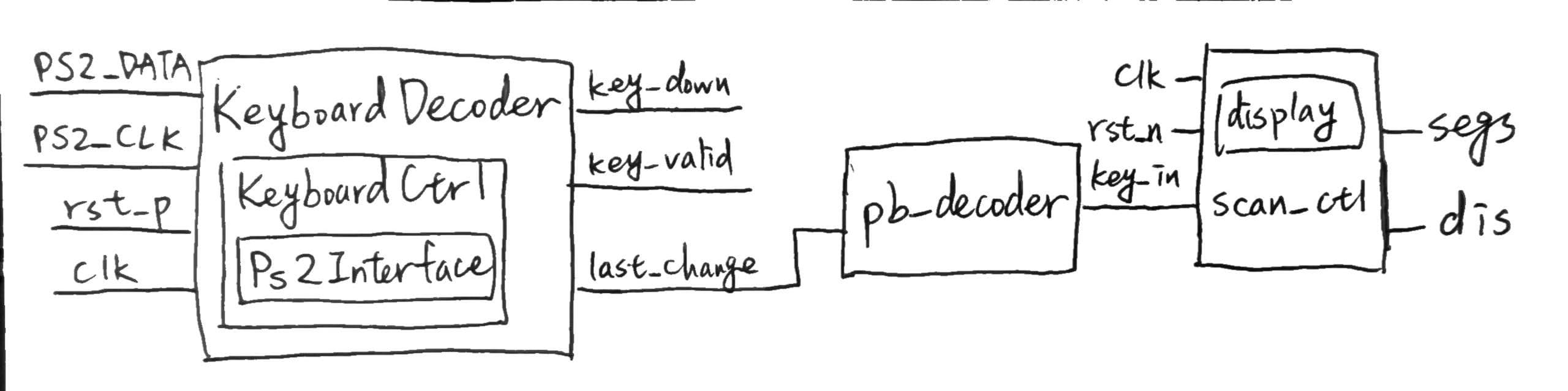
**Design Specification**

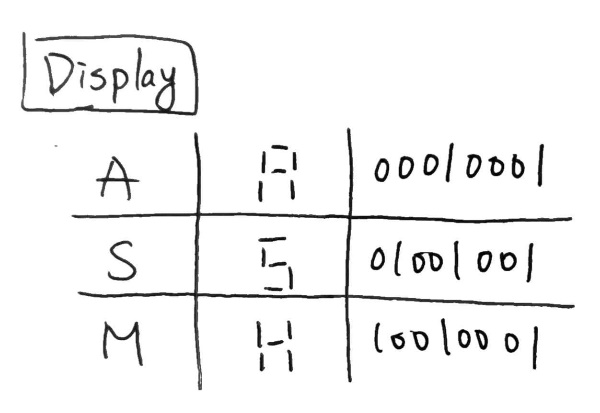
inout : PS2\_DATA, PS2\_CLK

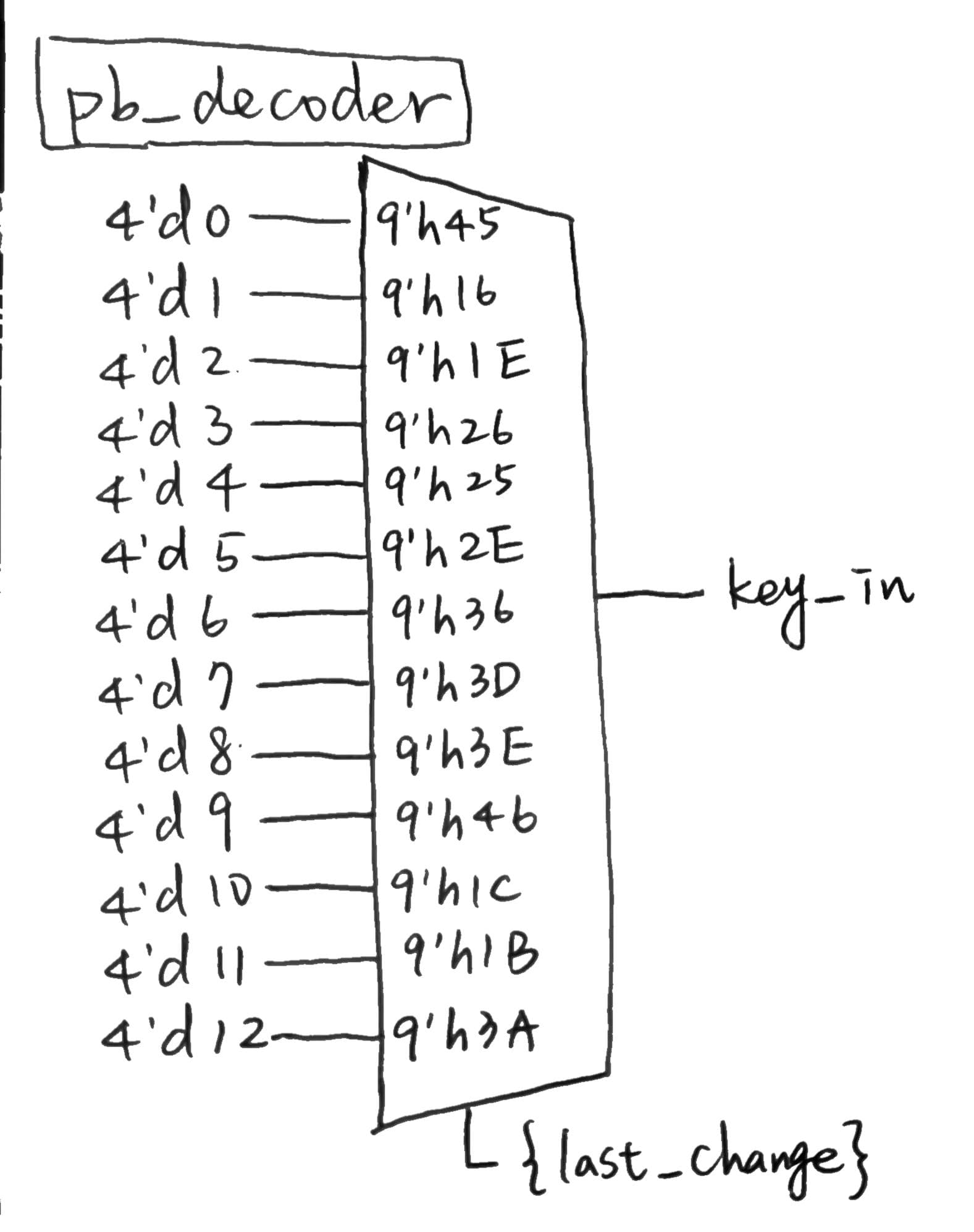
input : clk (100Mhz), rst\_n (reset)

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:

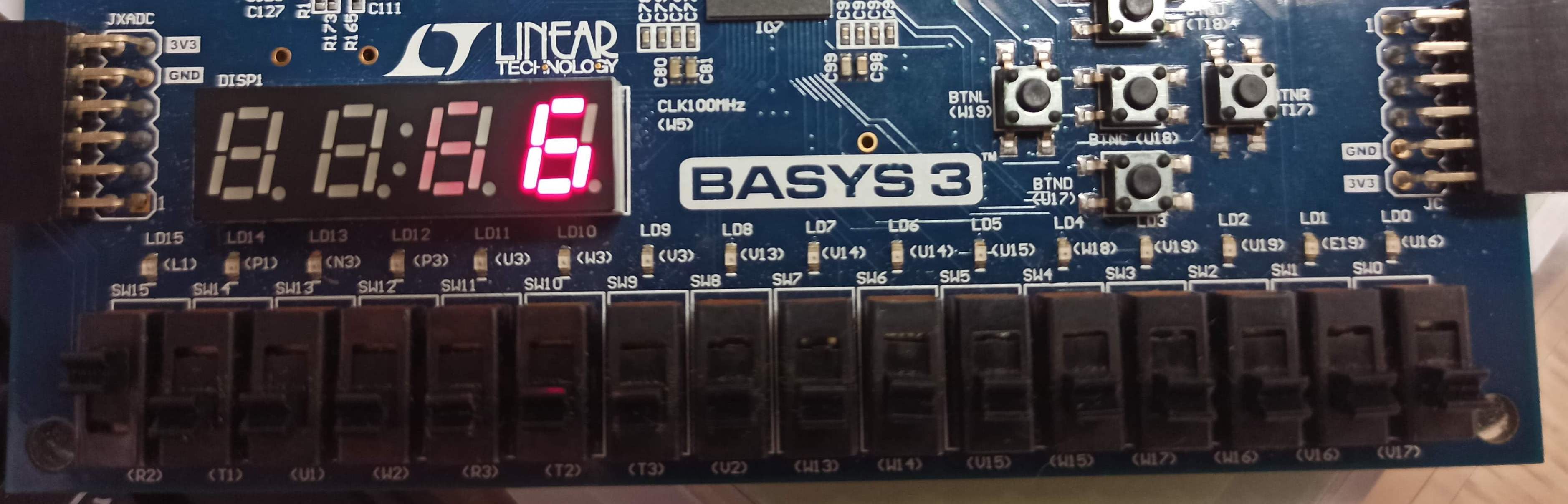
|  |  |  |  |
| --- | --- | --- | --- |
| PS2\_DATA | PS2\_CLK | rst\_n | clk |
| B17 | C17 | R2 | W5 |

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

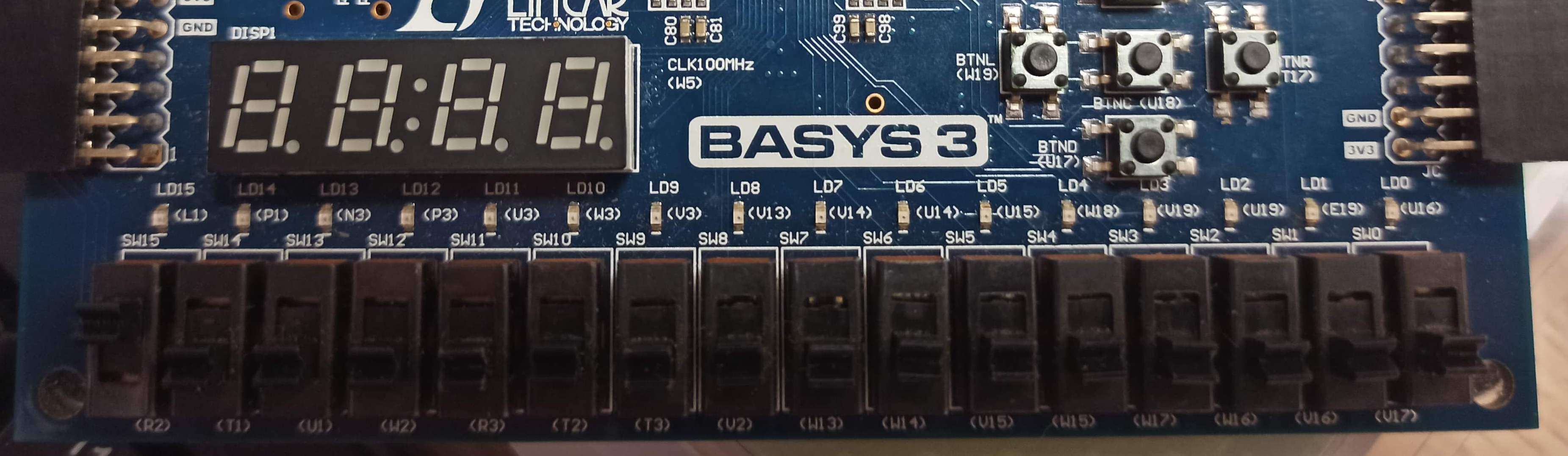
|  |  |  |  |
| --- | --- | --- | --- |
| dis[3] | dis[2] | dis[1] | dis[0] |
| W4 | V4 | U4 | U2 |

**Design Implementation**

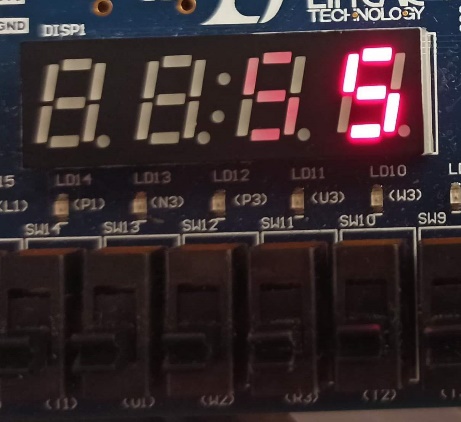
**press 6**



**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).**

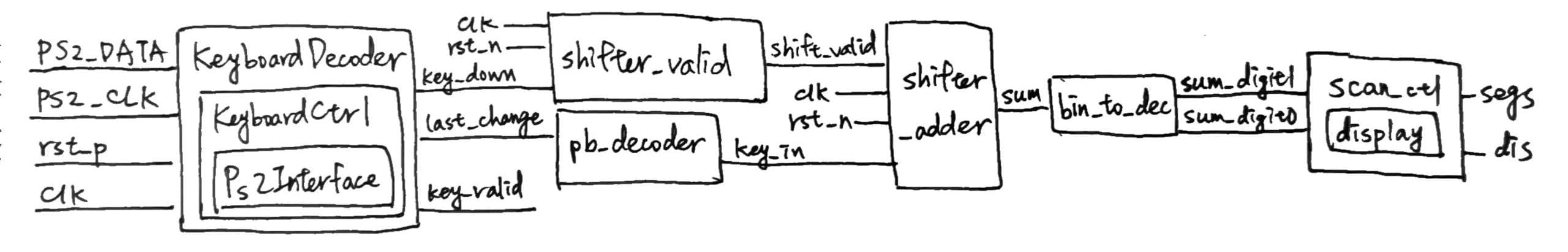
**Design Specification**

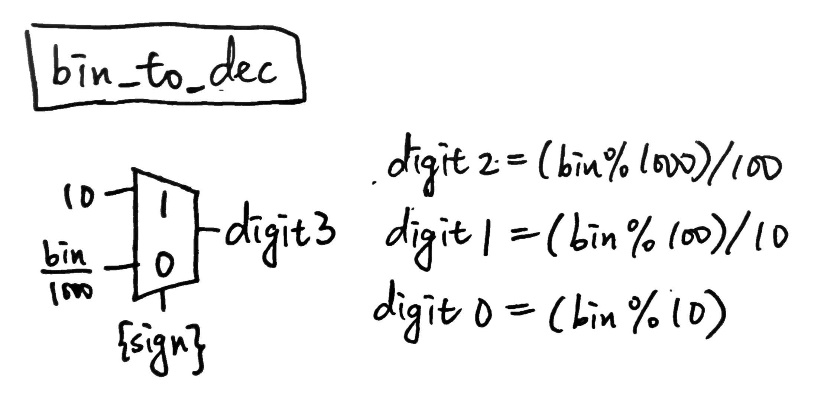
inout : PS2\_DATA, PS2\_CLK

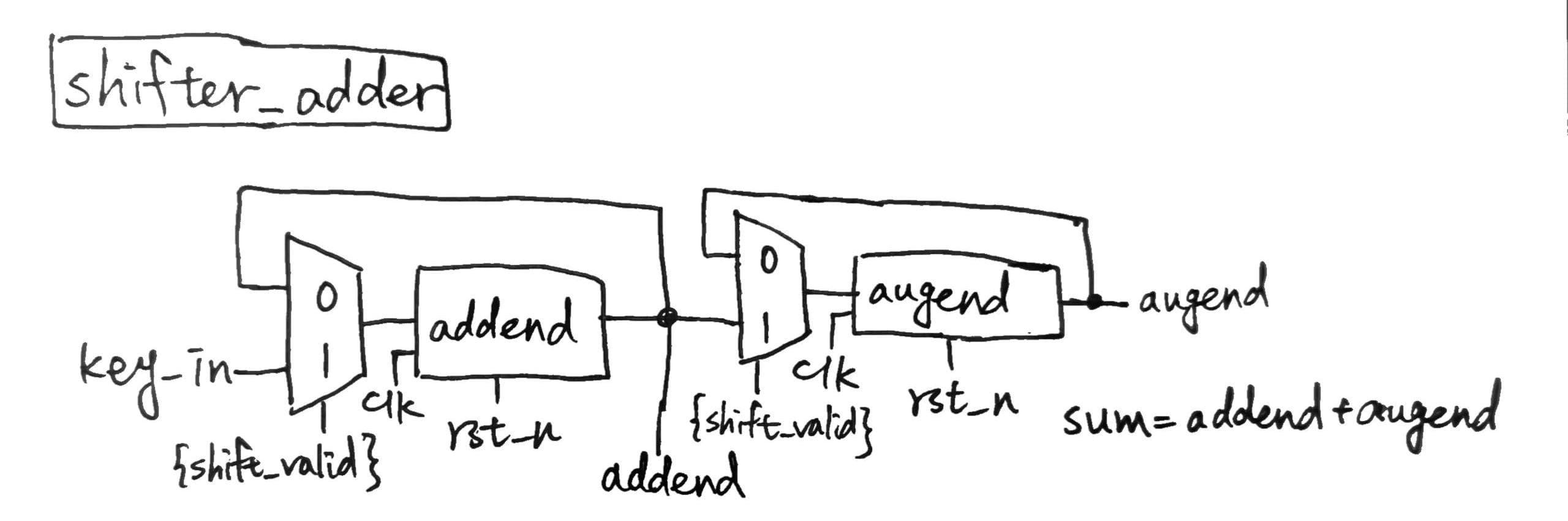
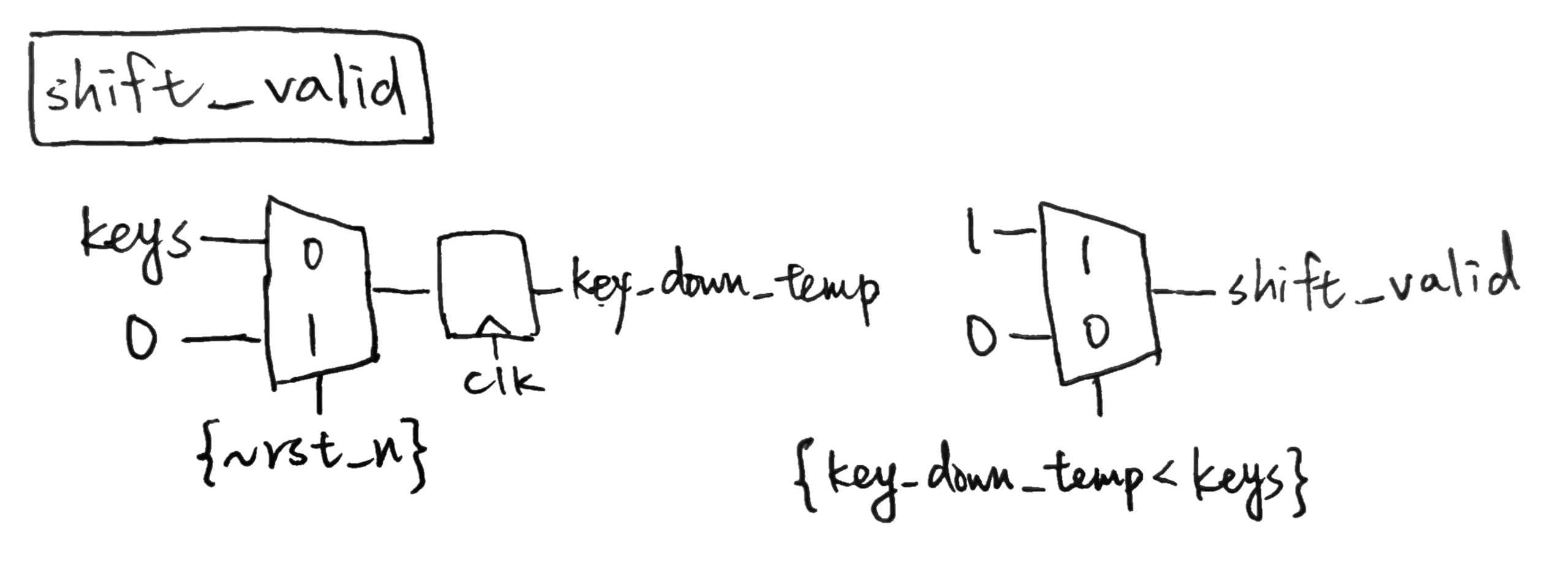
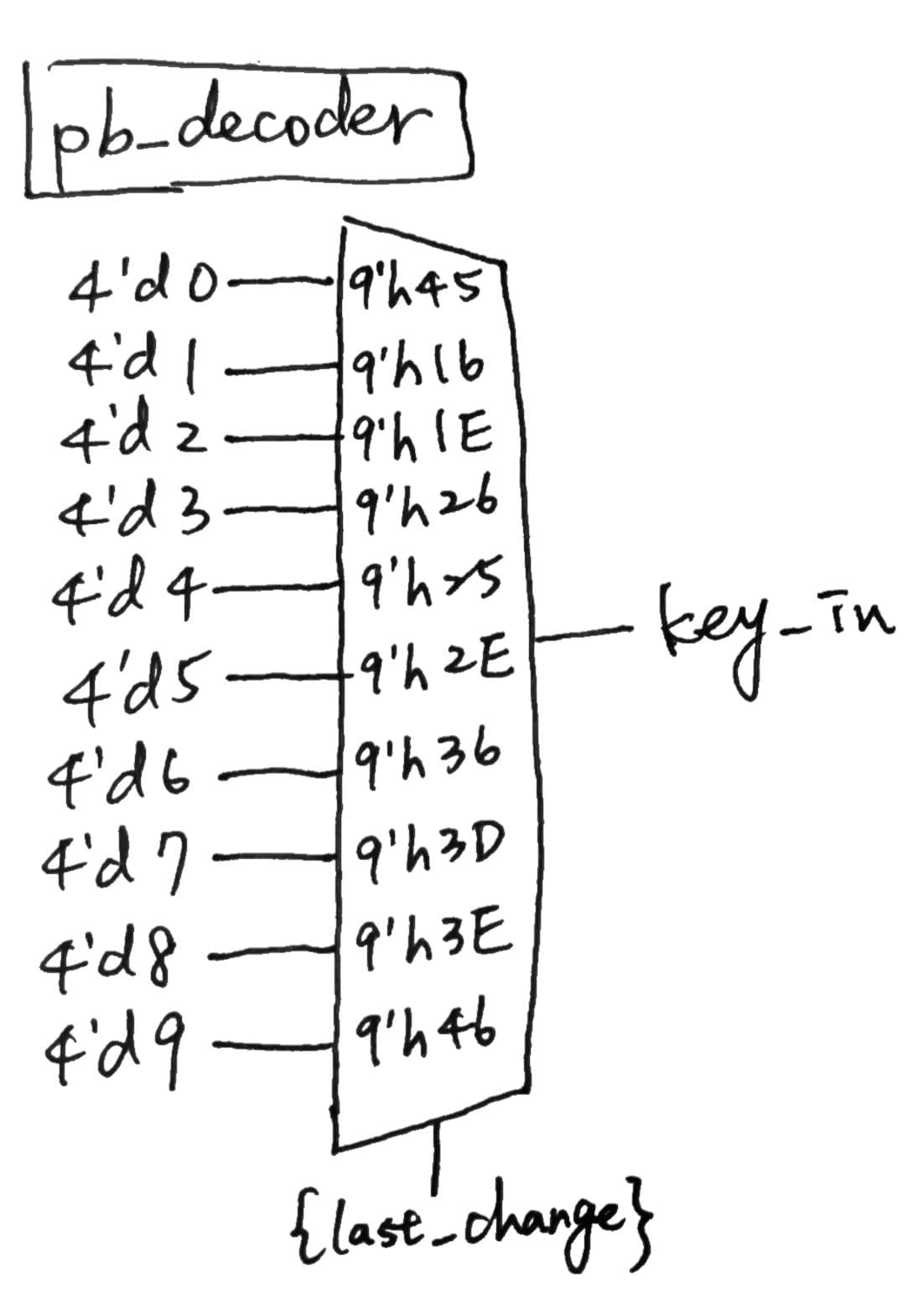
input : clk (100Mhz), rst\_n (reset)

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:

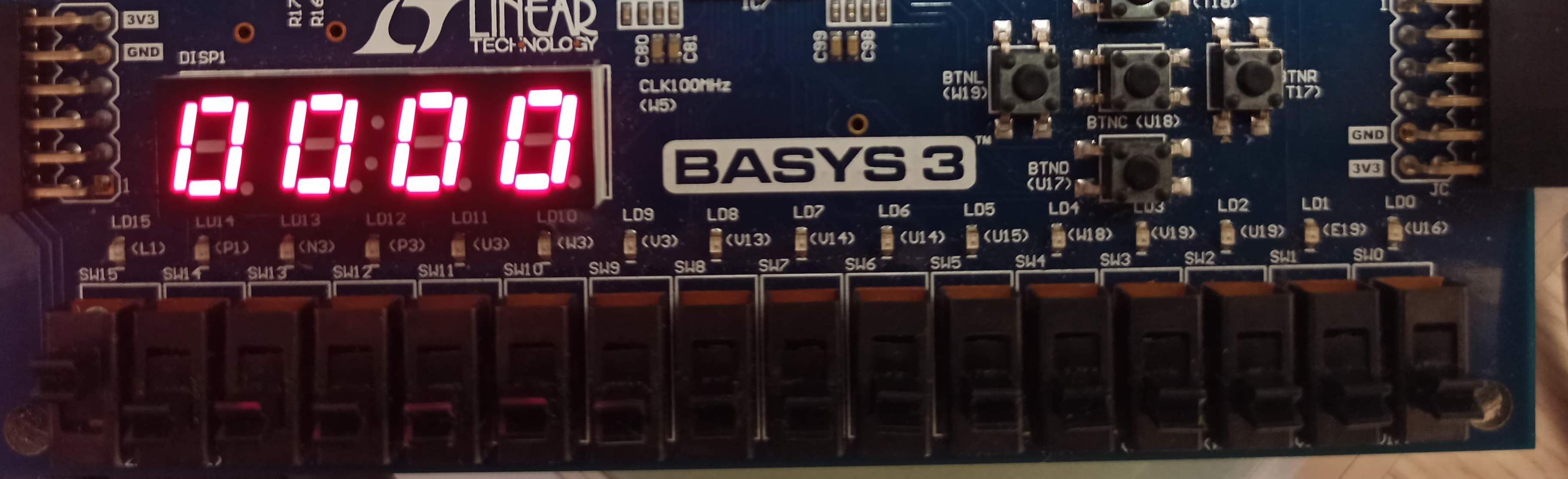
|  |  |  |  |
| --- | --- | --- | --- |
| PS2\_DATA | PS2\_CLK | rst\_n | clk |
| B17 | C17 | R2 | W5 |

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

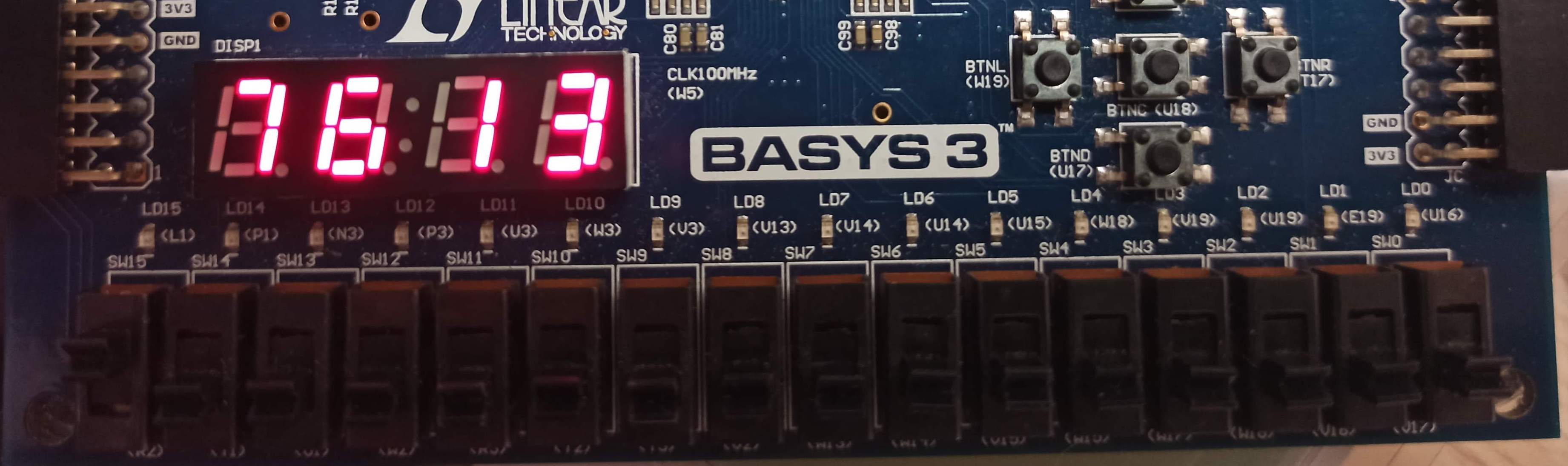
|  |  |  |  |
| --- | --- | --- | --- |
| dis[3] | dis[2] | dis[1] | dis[0] |
| W4 | V4 | U4 | U2 |

**Design Implementation**

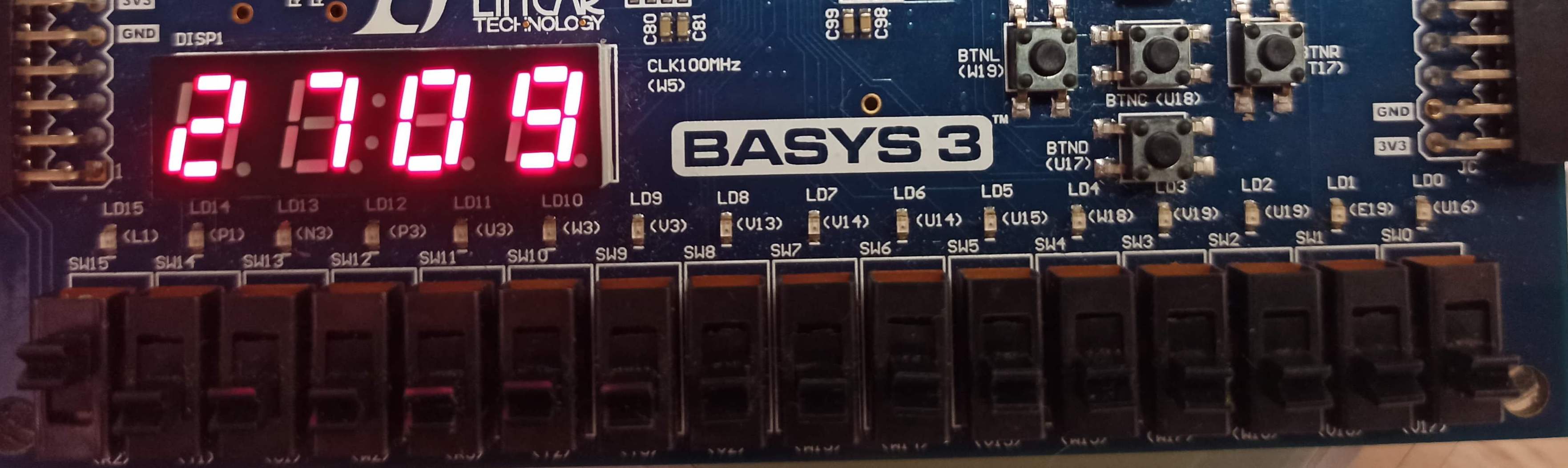
**reset**



**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.**

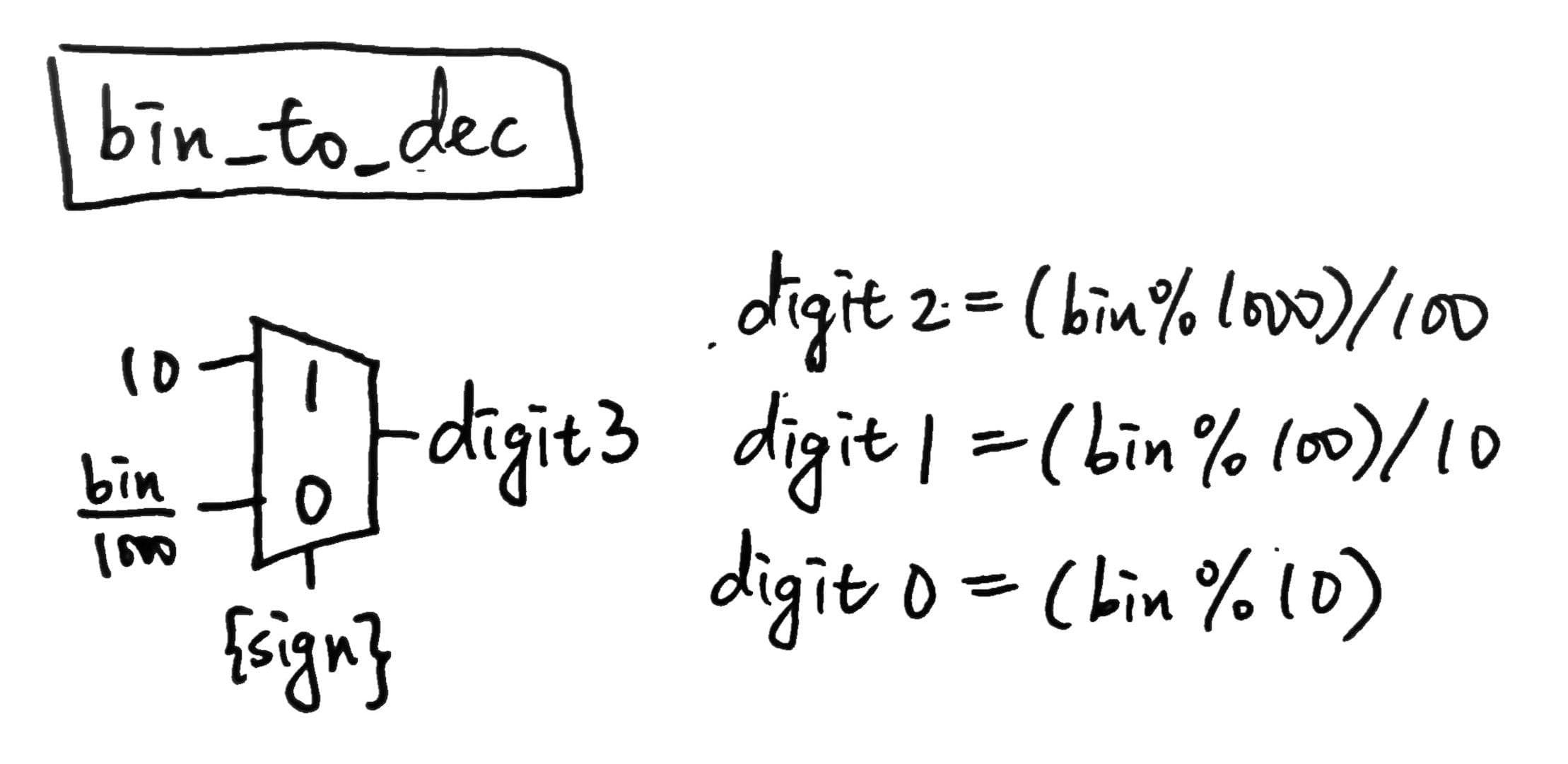
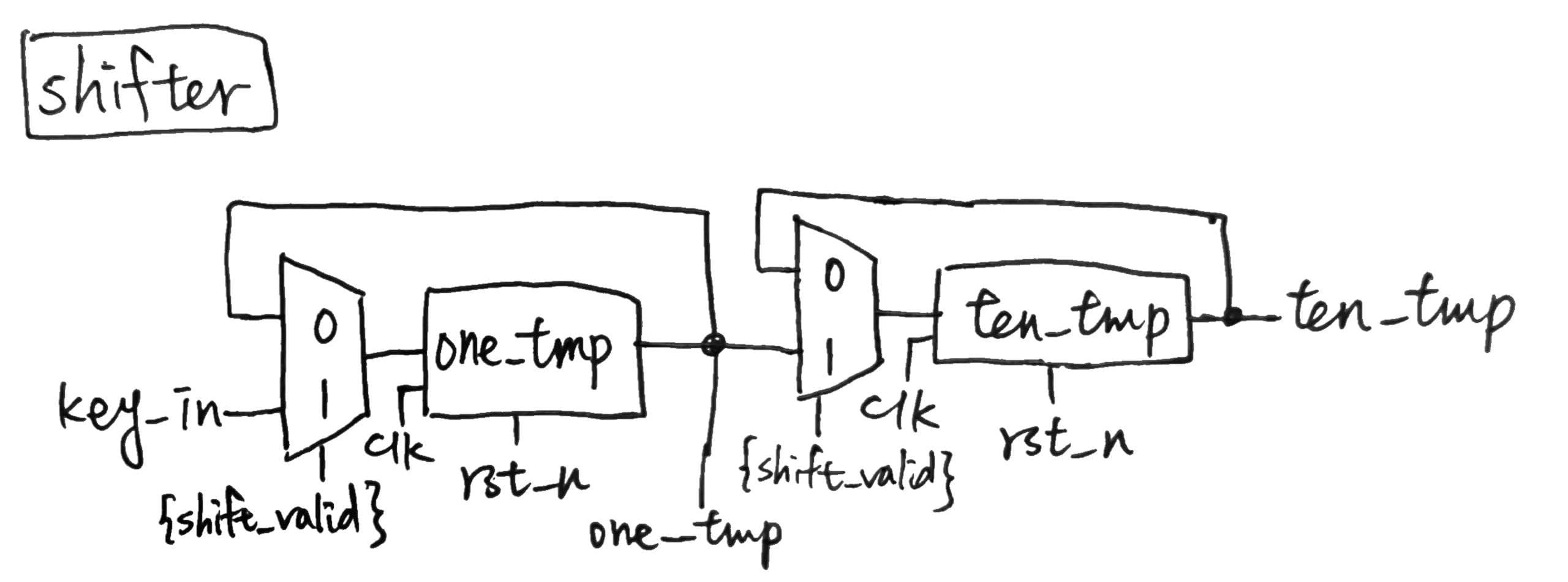
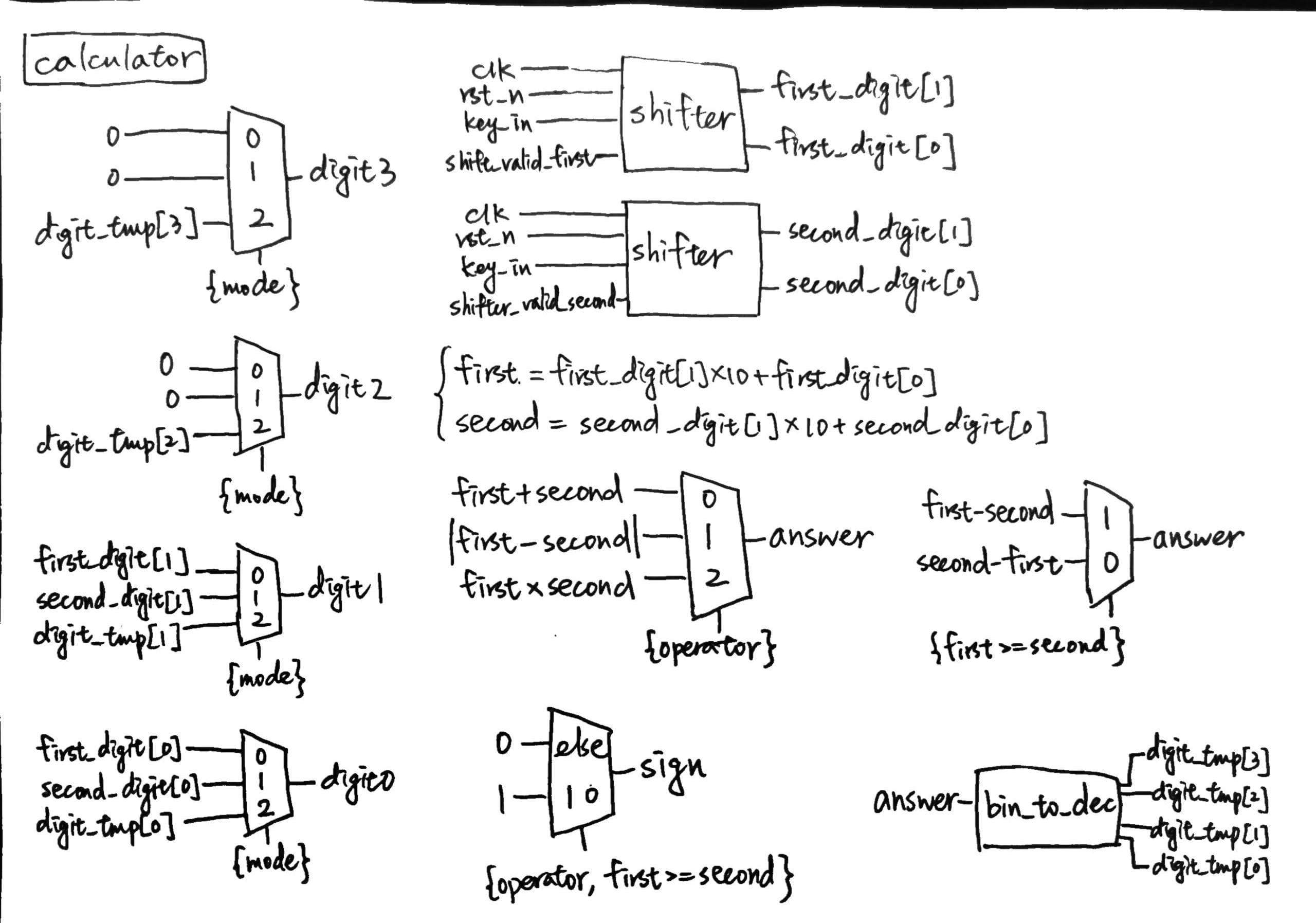
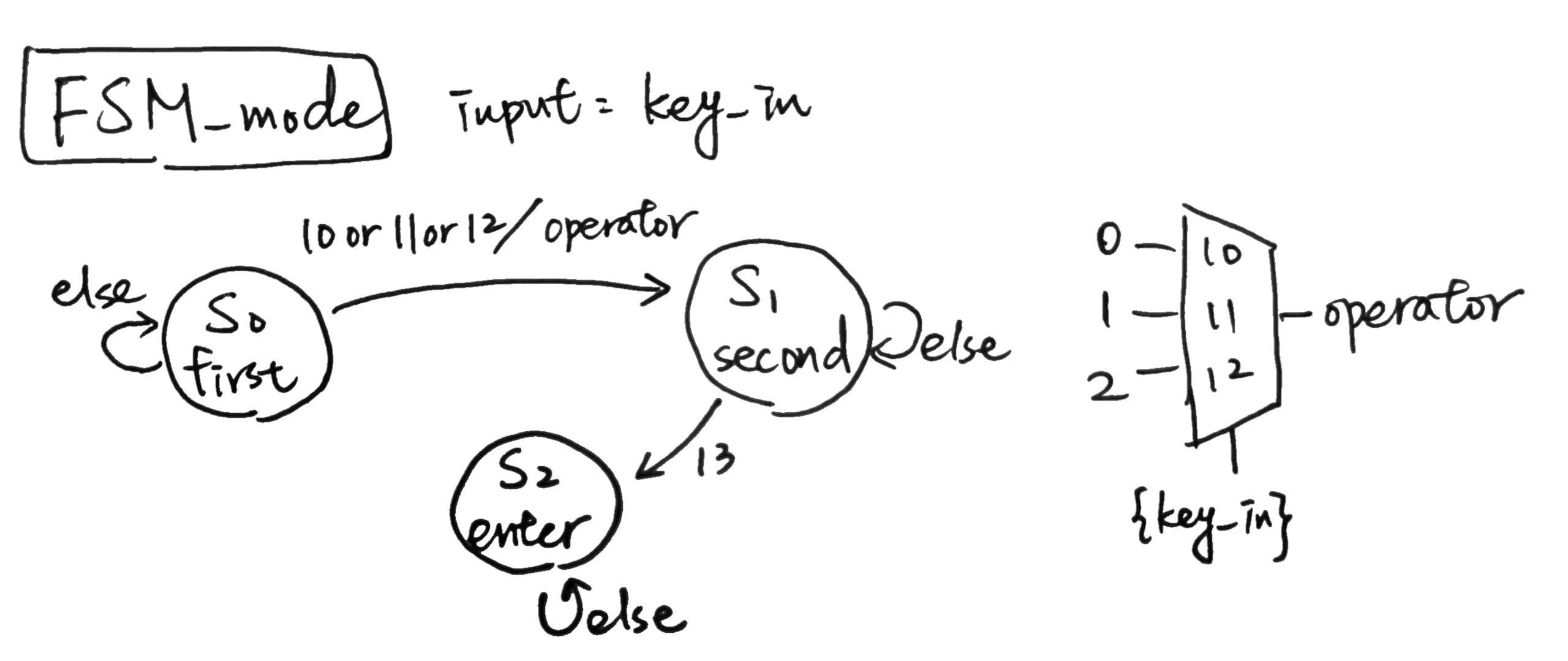
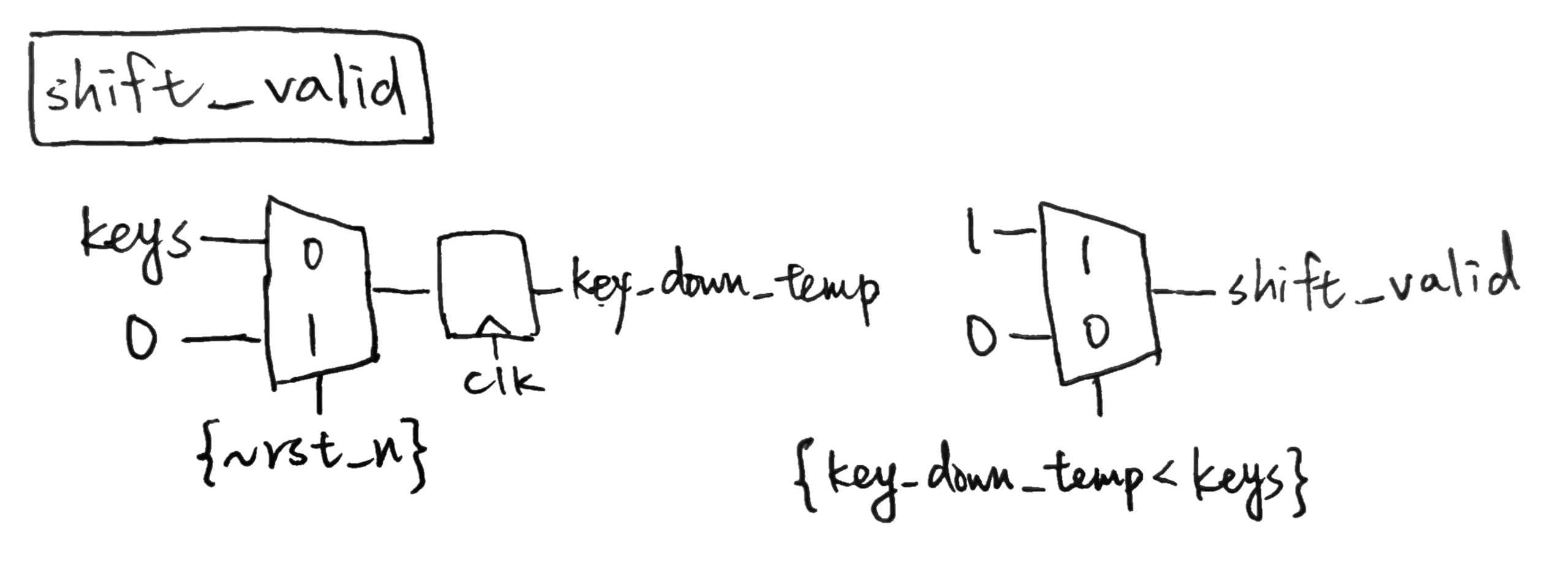
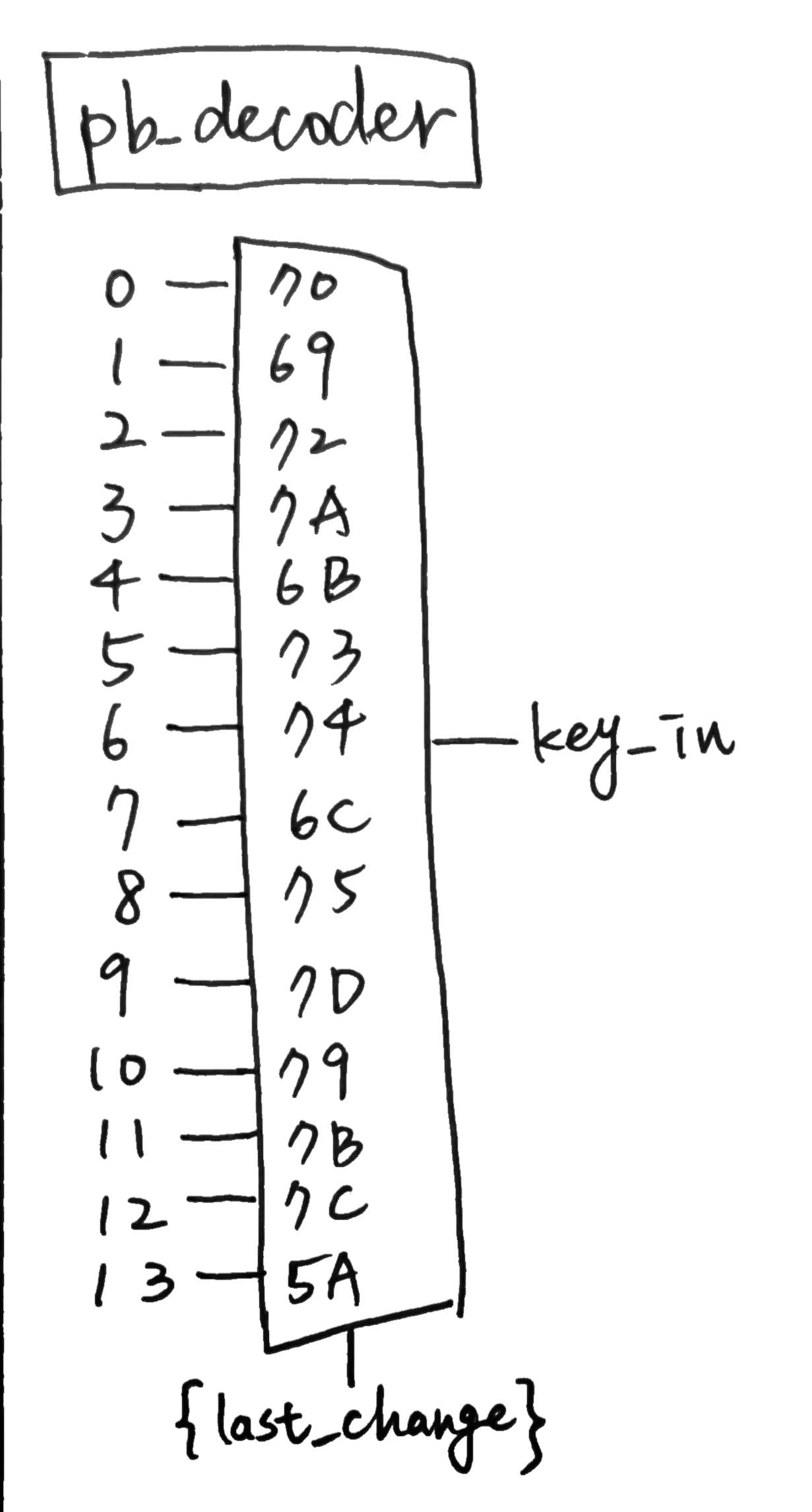
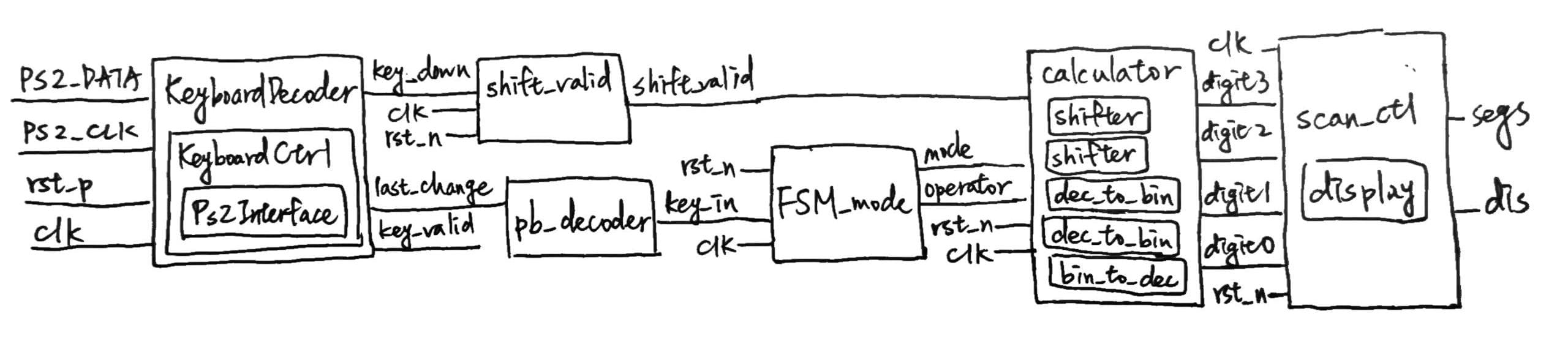
**Design Specification**

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



I/O pin assignment:

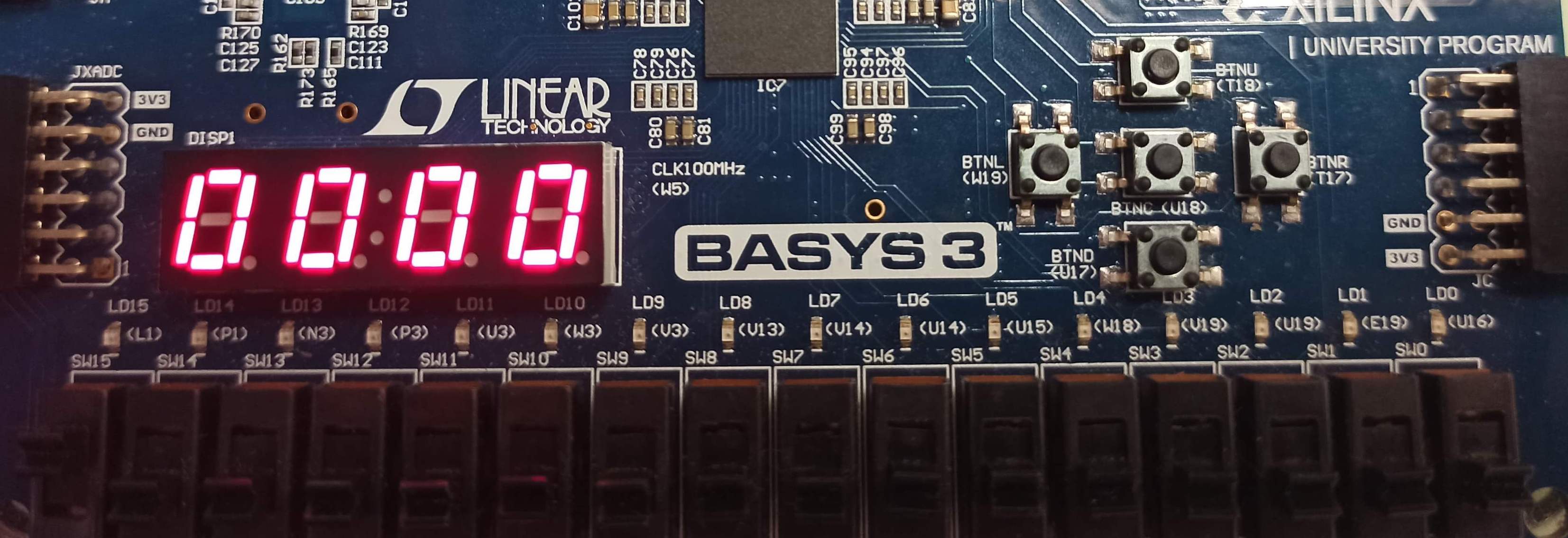
|  |  |  |  |
| --- | --- | --- | --- |
| PS2\_DATA | PS2\_CLK | rst\_n | clk |
| B17 | C17 | R2 | W5 |

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

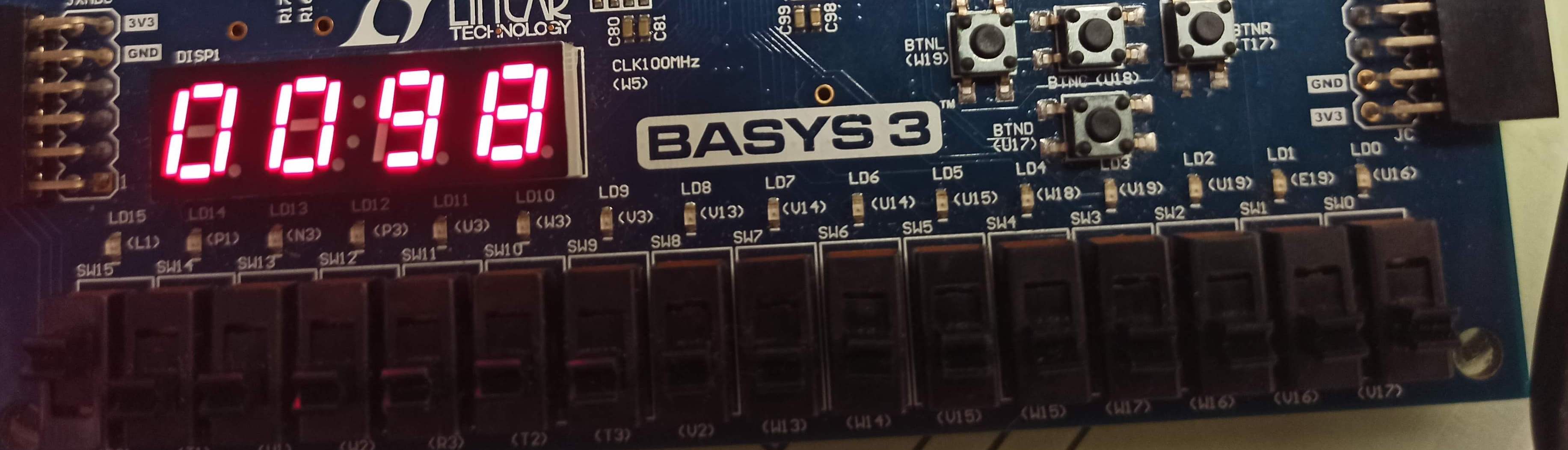
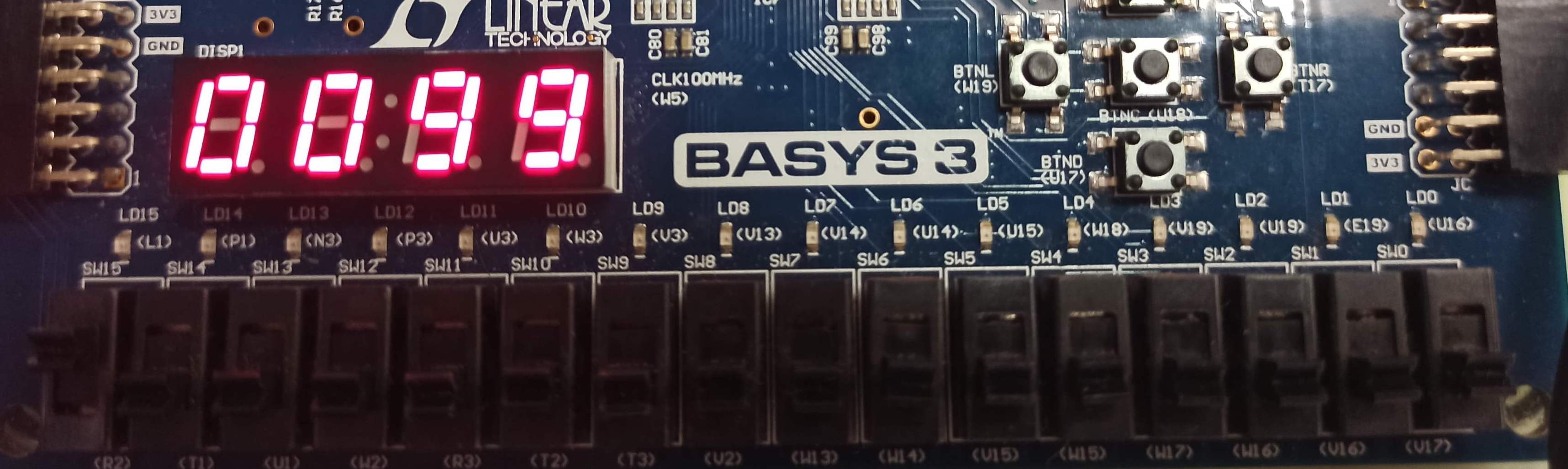
|  |  |  |  |
| --- | --- | --- | --- |
| dis[3] | dis[2] | dis[1] | dis[0] |
| W4 | V4 | U4 | U2 |

**Design Implementation**

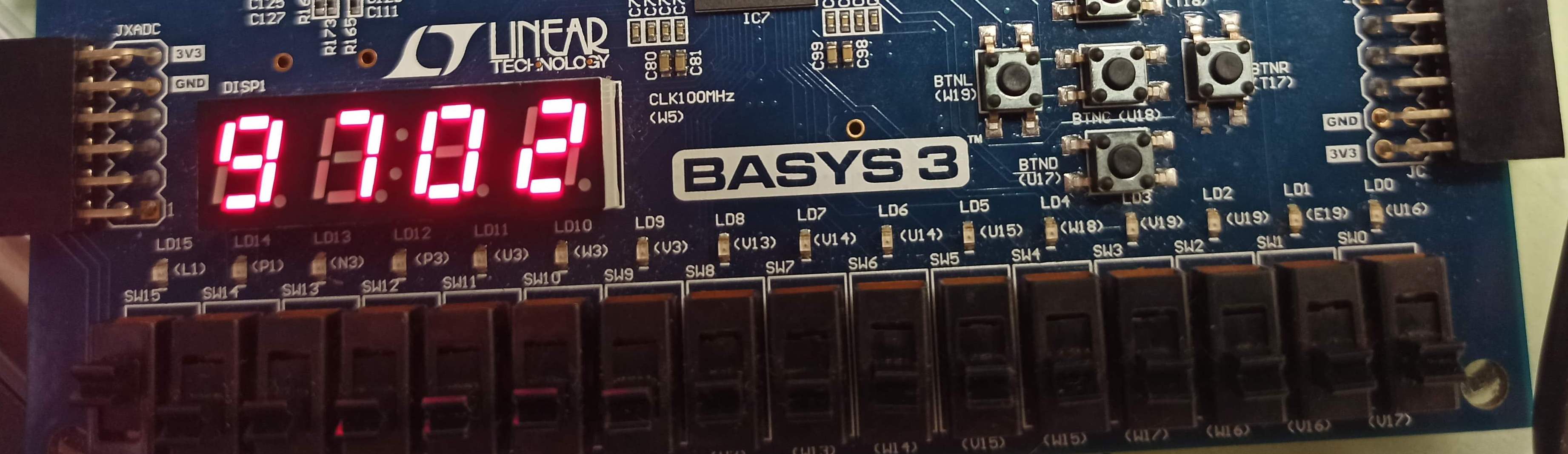
**reset / press + or – or \***



**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×10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.**

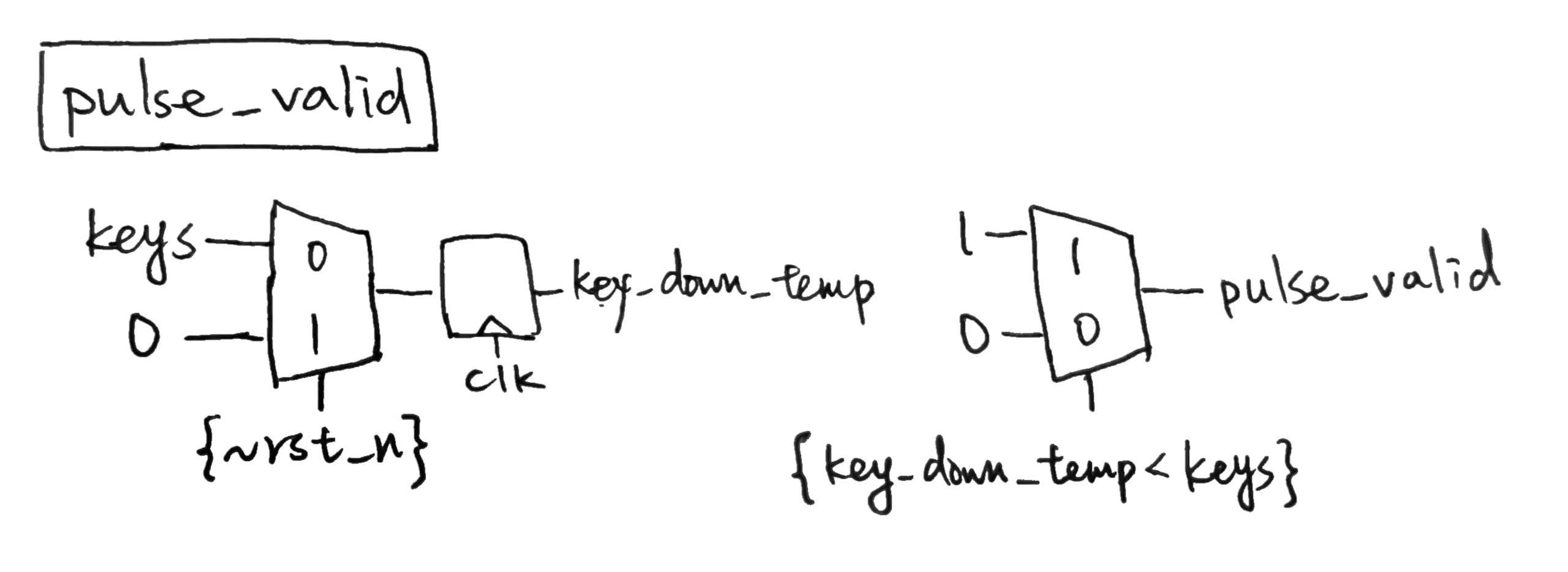
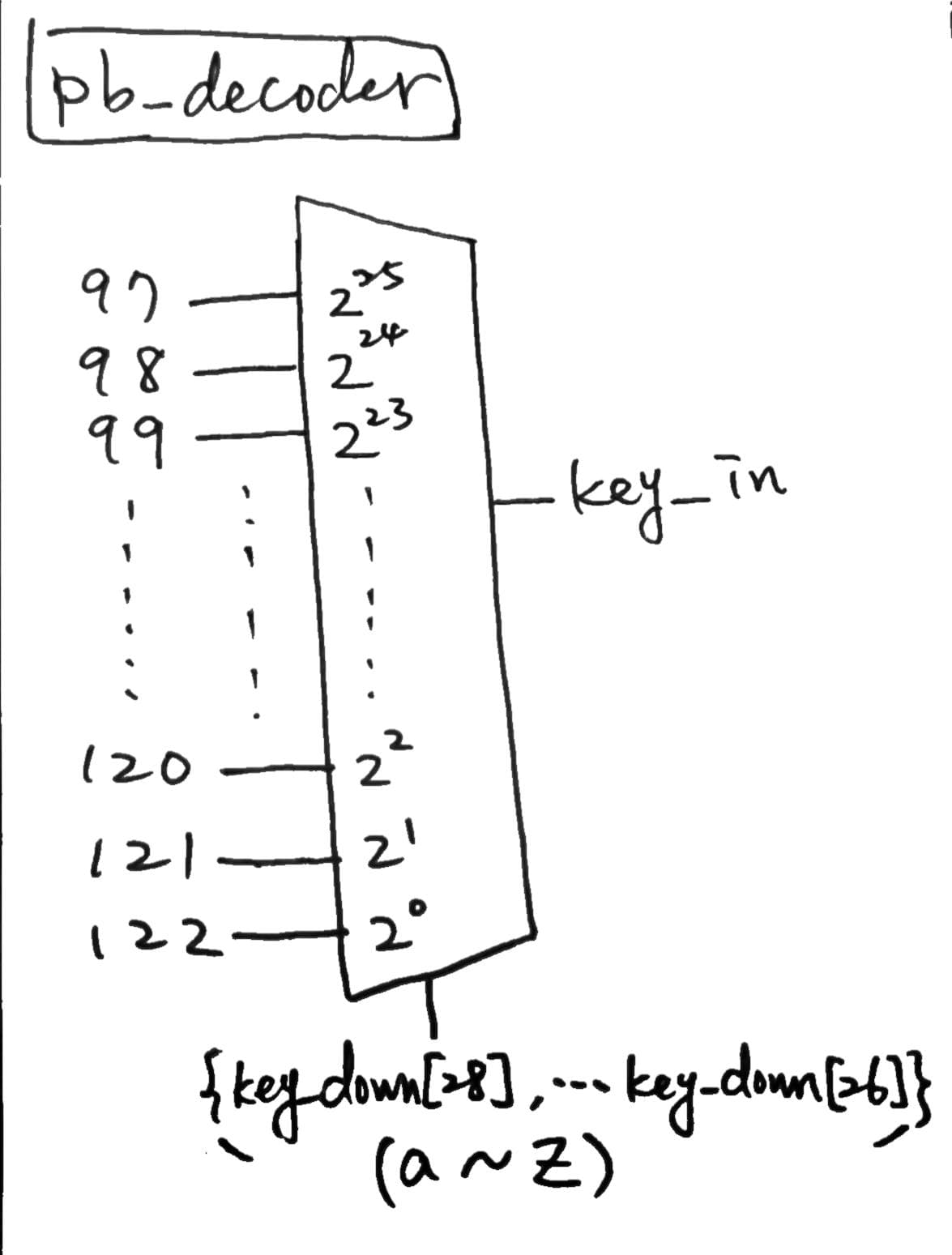
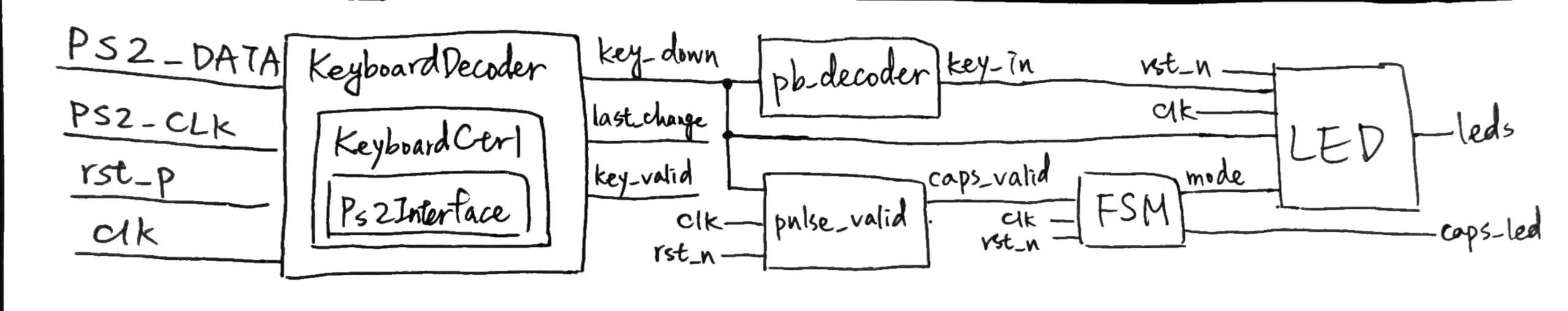
**Design Specification**

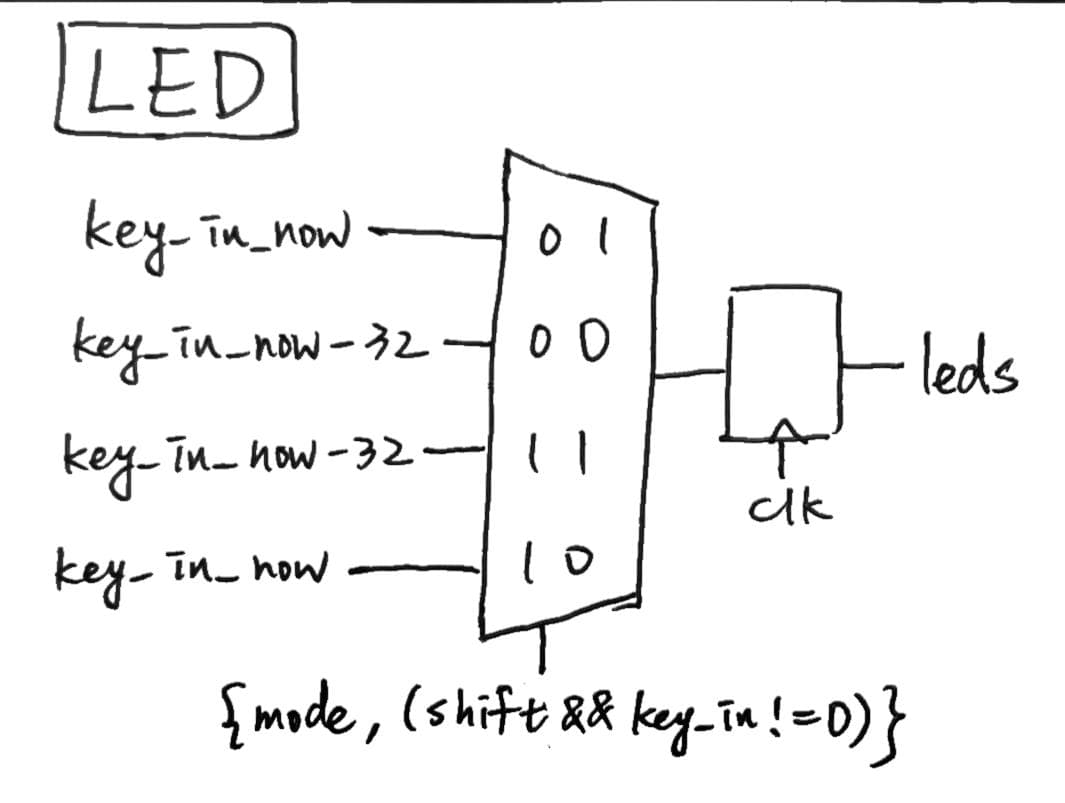
inout : PS2\_DATA, PS2\_CLK

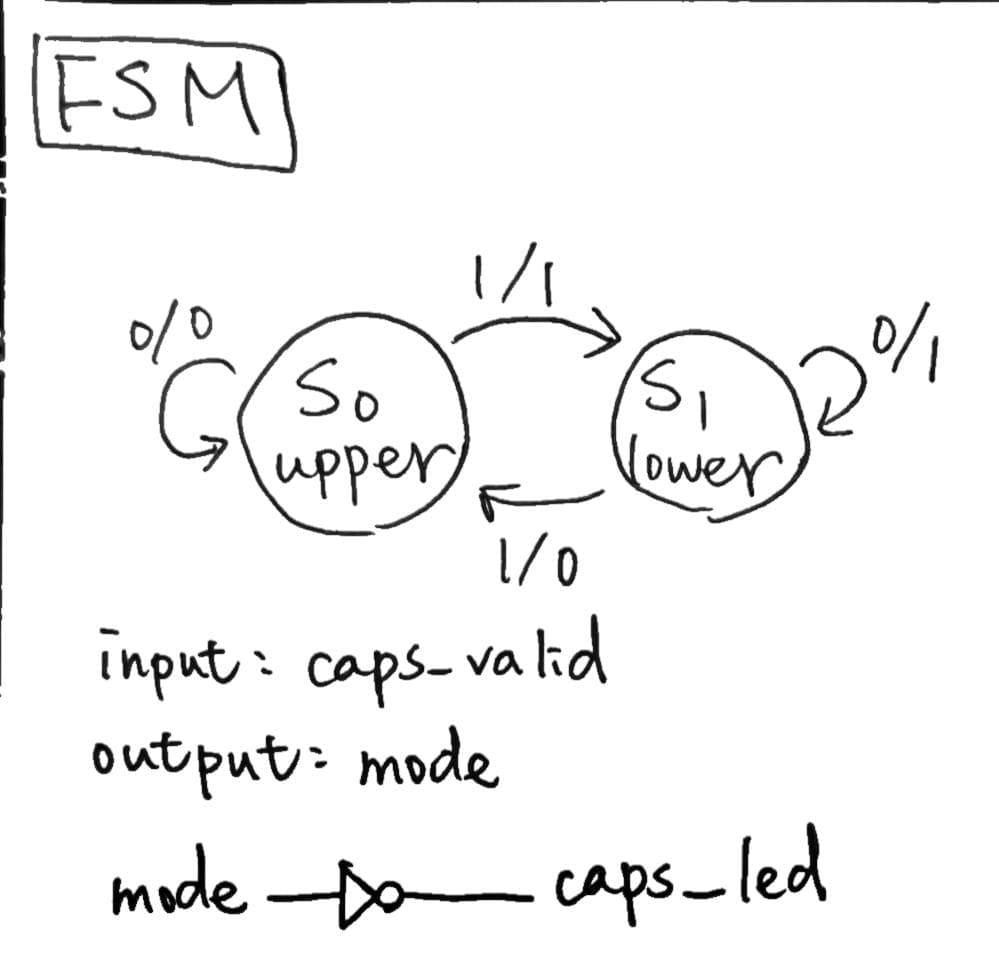
input : clk (100Mhz), rst\_n (reset)

output : caps\_led (大小寫LED指示燈), [6:0]leds (LED output)

logic diagram







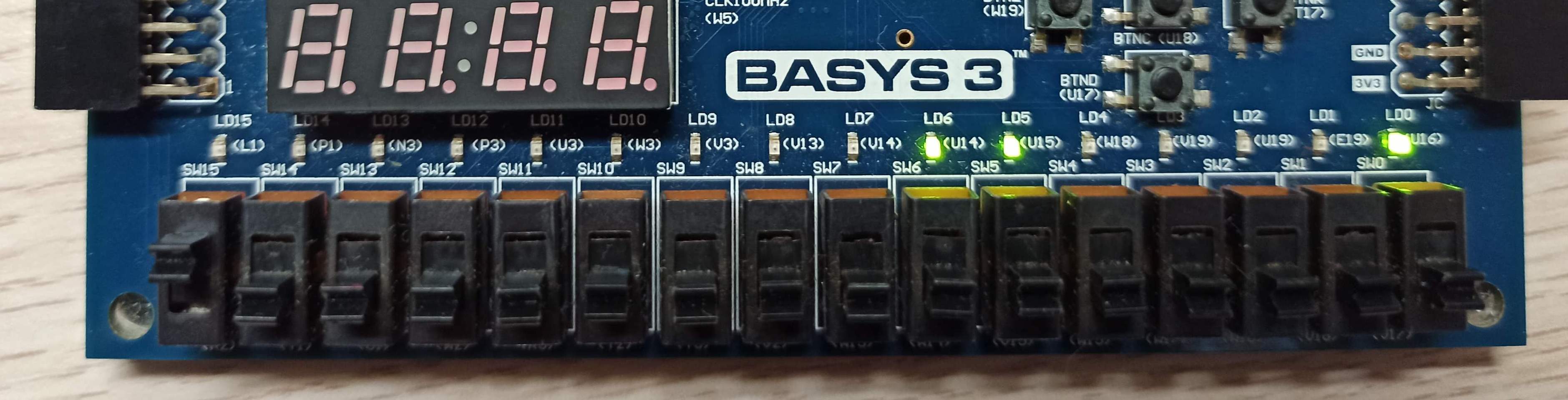
I/O pin assignment:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PS2\_DATA | PS2\_CLK | caps\_led | rst\_n | clk |
| B17 | C17 | L1 | R2 | W5 |

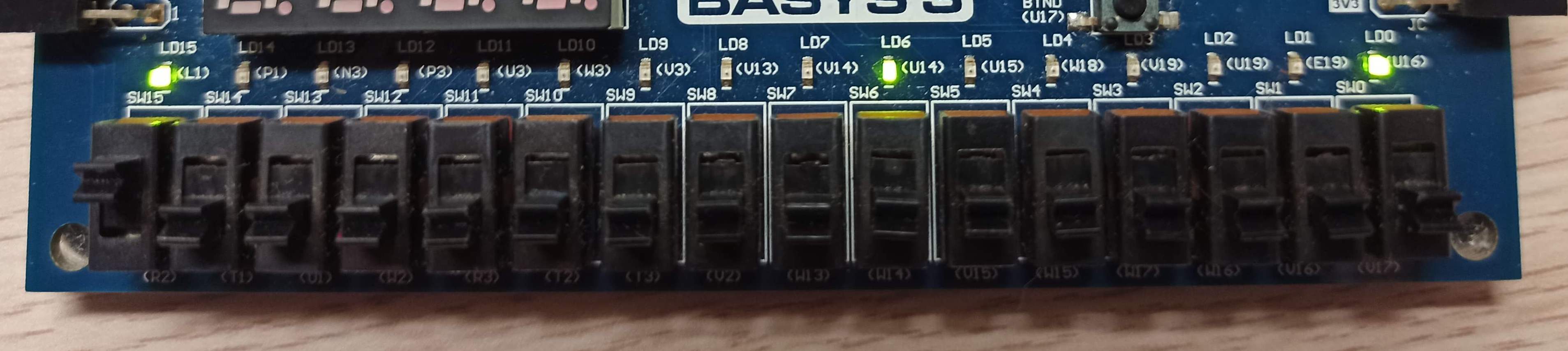
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| leds[6] | leds[5] | leds[4] | leds[3] | leds[2] | leds[1] | leds[0] |
| U14 | U15 | W18 | V19 | U19 | E19 | U16 |

**Design Implementation**

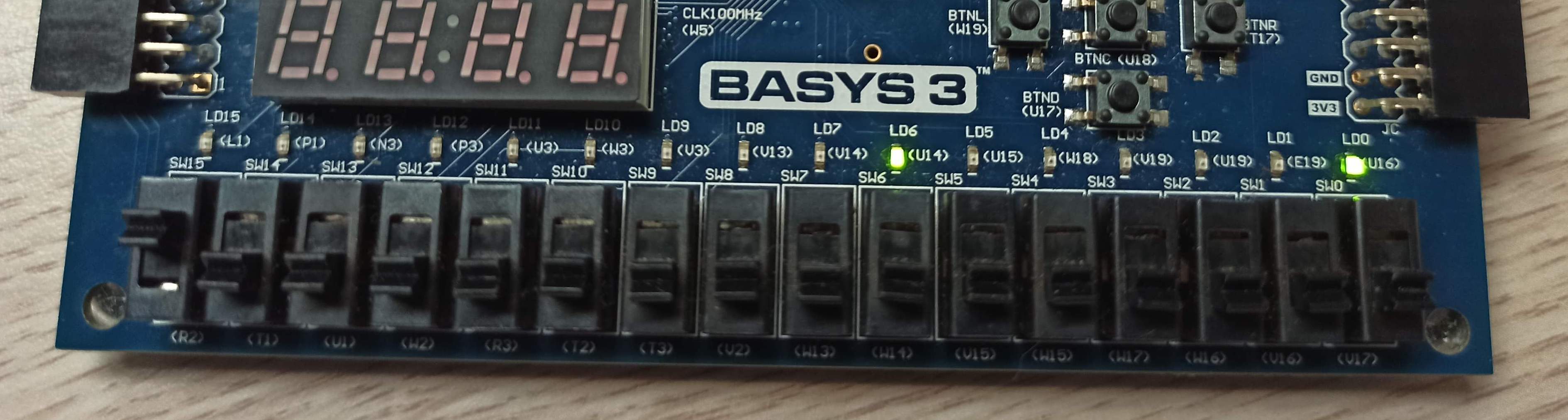
**a(97 = 1100001)**



**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。