邏輯設計實驗 Lab8 結報

105060012 張育菘

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.

Design Specification

Input：clk,rst;

Output：ssd_ctl[3:0],seg[7:0];

Inout：PS2_DATA,PS2_CLK;

*V17 控制 rst，當 rst==0 時會產生 rst 的訊號

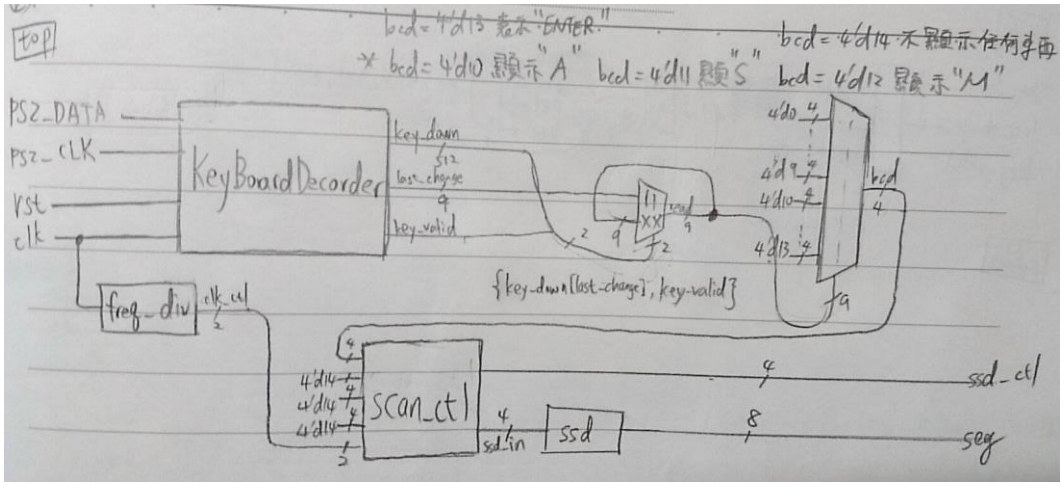
Design Implementation

Logic function :

1. top：

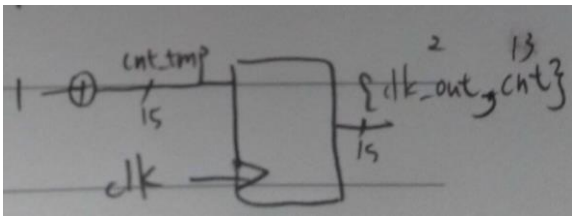
此 top module，專門拿來呼叫其他小 moudule 的。

我設計 key_down[last_change] && key_valid 去讀值，因為 key_valid 是一當按下或放下才會產生的訊號，因此拿他與 key_down[last_change]做交集可判斷 last_change 是否有被按下去，最後至 mux 轉換成 BCD(Binary-Coded Decimal)的形式。



2. freq_div：

   此為除頻的 module，輸出的 clk_ctl 當作 scan_ctl 的 clk，其頻率極高。

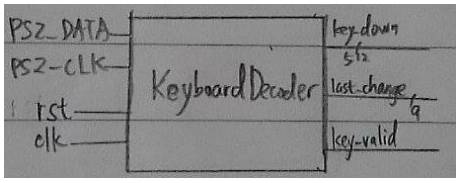


3. KeyboardDecoder：

   這個 module 主要的目的是讀取鍵盤的值。

   - key_down 是表示那一個鍵有沒有被按下去；last_change 是表示按鍵按下去的值；key_valid 在按鍵被按下去或放開的當下方會產生一個突波(理論上)。

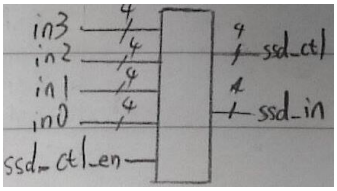
   * PS2_DATA,PS2_CLK 都是 inout port

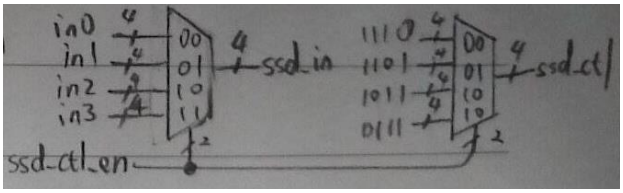


4. scan_ctl：

   此 module 是讓七段顯示器同時顯示不同位數的 module。

   - ssd_ctl_en(from freq_div 的 clk_ctl[2:0])的用意為顯示兩個不同的數，因為七段顯示器一次只能顯示一種數次，因此需要一個頻率介於內建時間到 1hz 的 ssd_ctl_en 產生同時顯示兩個數的錯覺。





5. ssd：

   當 bcd 為 4'd0: segs = 8'b00000011; //七段顯示器顯示”0”
   當 bcd 為 4'd1: segs = 8'b10011111; //七段顯示器顯示”1”
   當 bcd 為 4'd2: segs = 8'b00100101; //七段顯示器顯示”2”
   當 bcd 為 4'd3: segs = 8'b00001101; //七段顯示器顯示”3”
   當 bcd 為 4'd4: segs = 8'b10011001; //七段顯示器顯示”4”
   當 bcd 為 4'd5: segs = 8'b01001001; //七段顯示器顯示”5”
   當 bcd 為 4'd6: segs = 8'b01000001; //七段顯示器顯示”6”
   當 bcd 為 4'd7: segs = 8'b00011111; //七段顯示器顯示”7”
   當 bcd 為 4'd8: segs = 8'b00000001; //七段顯示器顯示”8”
   當 bcd 為 4'd9: segs = 8'b00001001; //七段顯示器顯示”9”
   當 bcd 為 4'd10:segs=8'b00010001; //七段顯示器顯示”A” //按下’A’鍵

當 bcd 為 4'd11:segs=8'b01001001; //七段顯示器顯示”S” //按下’S’鍵
當 bcd 為 4'd12:segs=8'b01100001; //七段顯示器顯示”E” //按下’M’鍵
當 bcd 為 4'd13:segs=8'b11111111; //七段顯示器顯示” ” //按下’ENTER’鍵//清空
default: segs = 8'11111111; //七段顯示器顯示” “

Result

I. 圖一~圖五分別是輸入”5”、”9”、”A”、”S”、”M”的結果。

圖一

圖二

圖三

圖四

圖五

Discussion

1. 我設計 key_down[last_change] && key_valid 去讀值，因為 key_valid 是一當按下或放下才會產生的訊號，因此拿他與 key_down[last_change]做交集可判斷 last_change 是否有被按下去，最後至 mux 轉換成 BCD(Binary-Coded Decimal)的形式。
2. 我在處理按”ENTER”鍵會清空七段顯示器的方式是使他完全不要顯示任何東西的方式呈現。
3. 由於”M”無法在七段顯示上顯示，因此我改用”倒 M”的方式呈現。

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)..

digit1	digit2	clk	rst
R2	T1	W5	V17

*digit1、digit2 分別是第一個數和第二個數能否給值的 enable 訊號

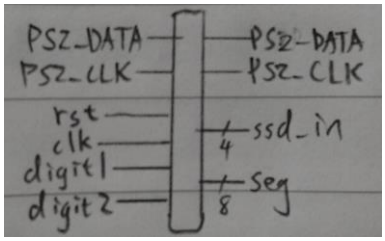
Design Specification

Input：rst,clk;

Output：ssd_ctl[3:0],seg[7:0];

Inout：PS2_DATA,PS2_CLK;

block diagram :

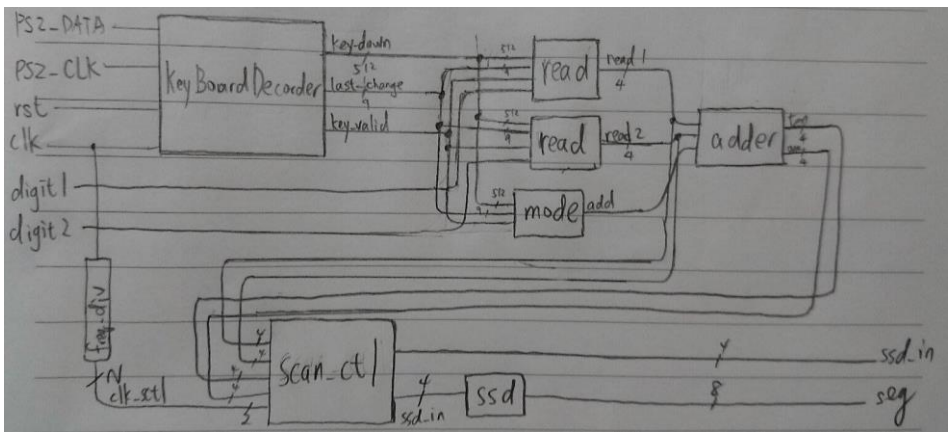


Design Implementation

Logic function :

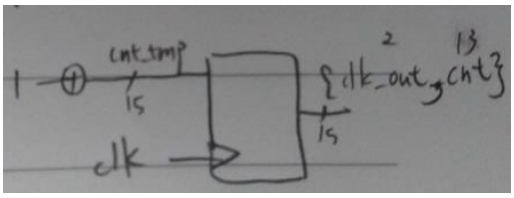
1. top：

此 top module，專門拿來呼叫其他小 moudule 的。



2. freq_div：

   此為除頻的 module，輸出的 clk_ctl 當作 scan_ctl 的 clk，其頻率極高。

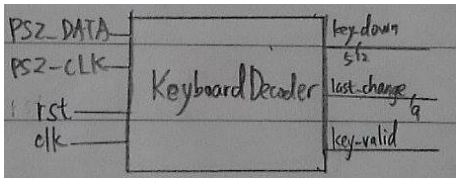


3. KeyboardDecoder：

   這個 module 主要的目的是讀取鍵盤的值。

   key_down 是表示那一個鍵有沒有被按下去；last_change 是表示按鍵按下去的值；key_valid 在按鍵被按下去或放開的當下方會產生一個突波(理論上)。

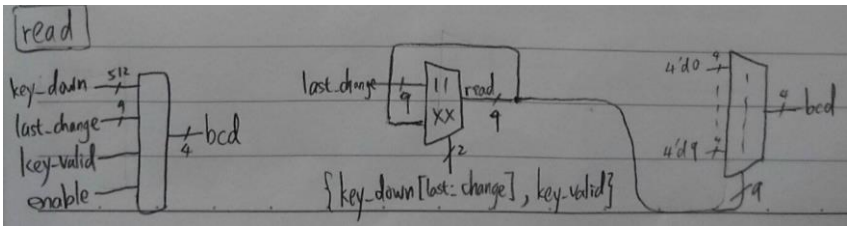
   * PS2_DATA,PS2_CLK 都是 inout port



4. read：

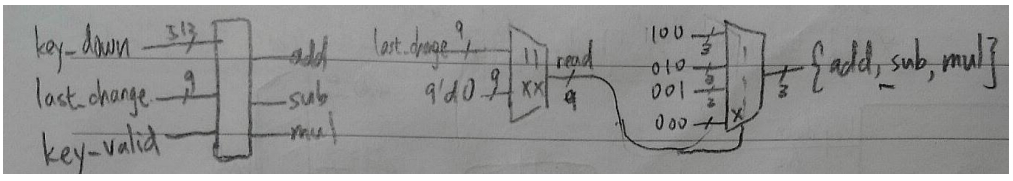
   此 module 是用來讀取從 key_board 輸入值，並轉成 BCD 的形式。

   - 取值的方法與第一題同，都是用 key_down[last_change] && key_valid 的方式判斷 last_change 為何值的被按下去。



5. mode：

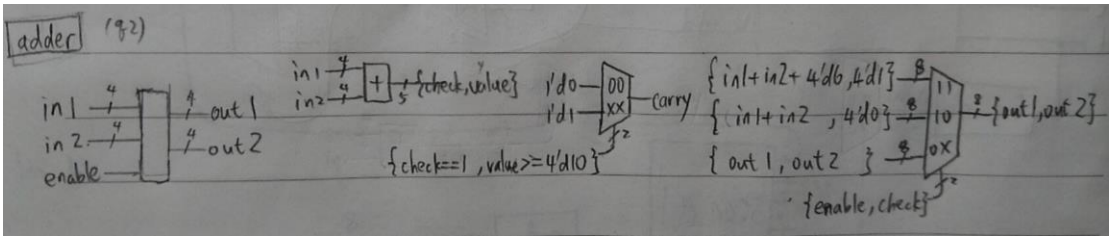
   此 module 是用來判斷要做”加”or”減”or”乘”，產生的 add 訊號會接出去作為 adder 的 enable，判斷能否開始進行加法。



6. adder：

   此 module 用來進行加法功能，有接一個來自 mode 產生的 enable 訊號，作為判斷能否進行加法的依據。

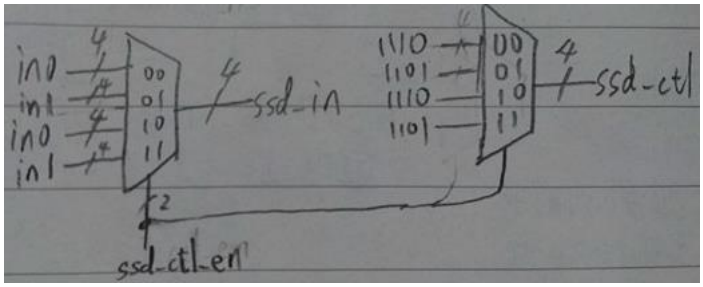
   - 設計{check,value} = in1 + in2，並用產生的 check 和 enable 的值去判斷是否有進位情形的產生。
   - 當{in1+in2}>=4'd16 時會有 overflow 的情形產生，我是利用"規律性"發現當有 overflow 情形產生時，需加 4’d6 並可以解決此問題。



7. scan_ctl：

   此 module 是讓七段顯示器同時顯示不同位數的 module。

   ssd_ctl_en(from freq_1hz 的 clk_ctl[2:0])的用意為顯示兩個不同的數，因為七段顯示器一次只能顯示一種數次，因此需要一個頻率介於內建時間到 1hz 的 ssd_ctl_en 產生同時顯示兩個數的錯覺。



8. ssd：

   當 bcd 為 4'd0: segs = 8'b00000011; //七段顯示器顯示”0”

   當 bcd 為 4'd1: segs = 8'b10011111; //七段顯示器顯示”1”

   當 bcd 為 4'd2: segs = 8'b00100101; //七段顯示器顯示”2”

   當 bcd 為 4'd3: segs = 8'b00001101; //七段顯示器顯示”3”

   當 bcd 為 4'd4: segs = 8'b10011001; //七段顯示器顯示”4”

   當 bcd 為 4'd5: segs = 8'b01001001; //七段顯示器顯示”5”

   當 bcd 為 4'd6: segs = 8'b01000001; //七段顯示器顯示”6”

   當 bcd 為 4'd7: segs = 8'b00011111; //七段顯示器顯示”7”

   當 bcd 為 4'd8: segs = 8'b00000001; //七段顯示器顯示”8”

   當 bcd 為 4'd9: segs = 8'b00001001; //七段顯示器顯示”9”

   當 bcd 為 4'd10:segs=8'b00010001; //七段顯示器顯示”A” //按下’A’鍵

   當 bcd 為 4'd11:segs=8'b01001001; //七段顯示器顯示”S” //按下’S’鍵

   當 bcd 為 4'd12:segs=8'b01100001; //七段顯示器顯示”E” //按下’M’鍵

   當 bcd 為 4'd13:segs=8'b11111111; //七段顯示器顯示” ” //按下’ENTER’鍵//清空

   default: segs = 8'11111111; //七段顯示器顯示” “

Result

(digit1、digit2 分別是第一個數和第二個數能否給值的 enable 訊號

R2：digit1 ； T1：digit2)

1. 圖一(digit1==1；digit2==0)，表示能對第一個數輸值
2. 圖二(digit1==0；digit2==1)，表示能對第二個數輸值
3. 圖三，當按下”+”時，會進行加法
4. 圖四，”8+9 = 17”

圖一

圖二

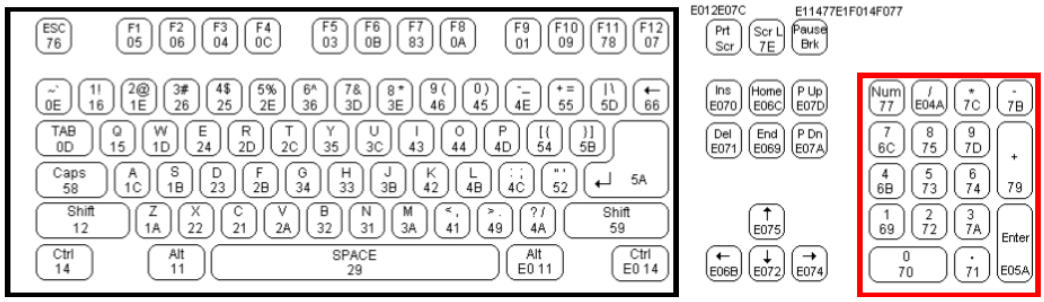
圖三

圖二

Discussion

1. adder
   當{in1+in2}>=4'd16 時會有 overflow 的情形產生，我是利用"規律性"發現當有 overflow 情形產生時，需加 4’d6 並可以解決此問題。
2. 設計 key_down[last_change] && key_valid 去讀值，因為 key_valid 是一當按下或放下才會產生的訊號，因此拿他與 key_down[last_change]做交集可判斷 last_change 是否有被按下去，最後至 mux 轉換成 BCD(Binary-Coded Decimal)的形式。

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.**



state[1]	state[0]	clk	rst
P1	N3	W5	V17

Design Specification

Input：clk,rst;

Output：ssd_ctl[3:0],seg[7:0],state[1:0];

Inout：PS2_DATA,PS2_CLK;

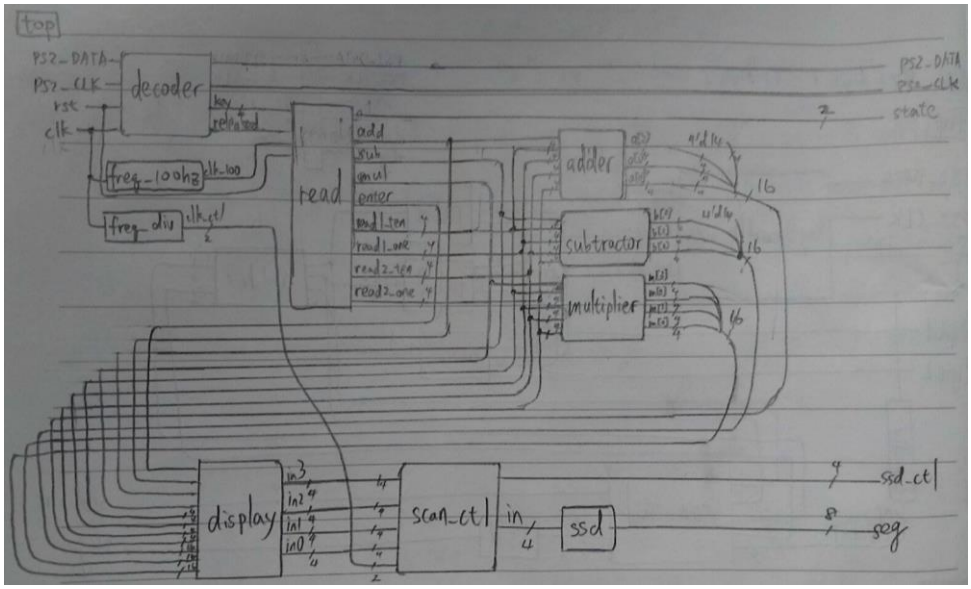
*V17 控制 rst，當 rst==0 時會產生 rst 的訊號

Design Implementation

Logic function :

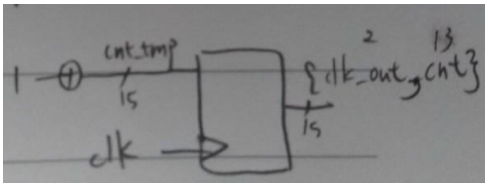
1. top：

   此 top module，專門拿來呼叫其他小 moudule 的。



2. freq_div：

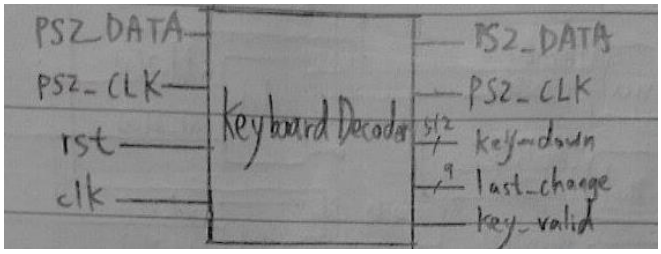
   此為除頻的 module，輸出的 clk_ctl 當作 scan_ctl 的 clk，其頻率極高。



3. KeyboardDecoder：

   這個 module 主要的目的是讀取鍵盤的值。

   - key_down 是表示那一個鍵有沒有被按下去；last_change 是表示按鍵按下去的值；key_valid 在按鍵被按下去或放開的當下方會產生一個突波(理論上)。

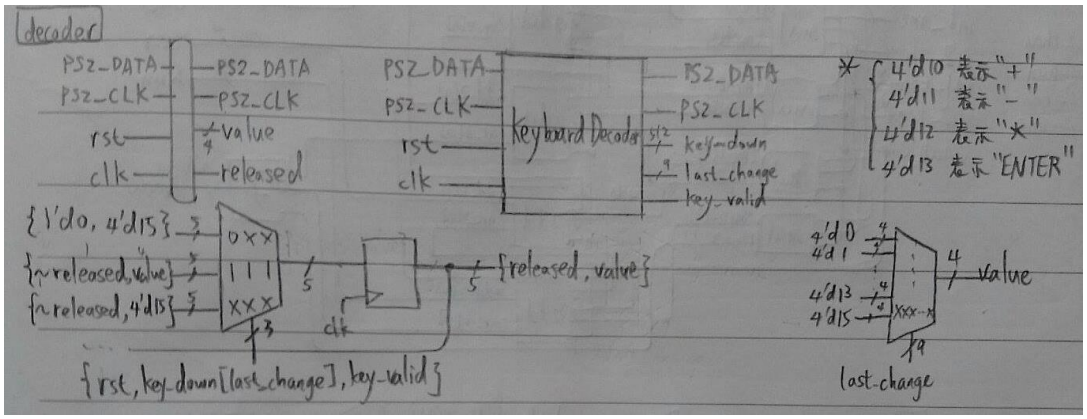


4. decoder：

   此 module 是用來讀取從 key_board 輸入值，並轉成 BCD 的形式。

   - 取值的方法與第一題同，都是用 key_down[last_change] && key_valid 的方式判斷 last_change 為何值的被按下去。

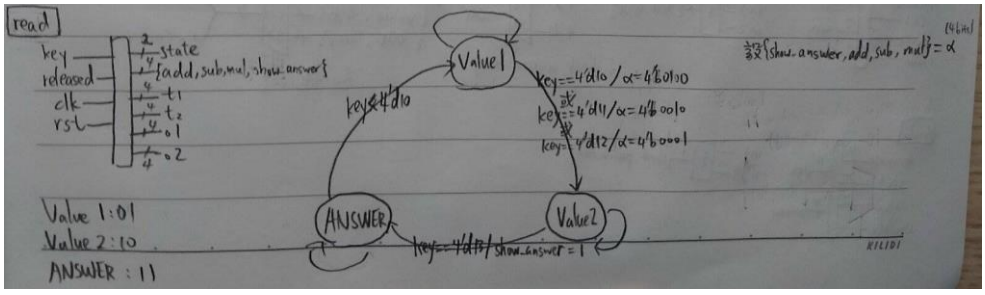
- 把 KeyboradDecoder 包在這個 module 的目的在於，使 KeyboradDecoder 影響範圍只限在這個 module 內，不會因產生不規則的訊號而破壞結果。
- 此輸出的 released 是在 read module 內讀值用。
- 4'd10 表示"+"；4'd11 表示"-"；4'd12 表示"*"；4'd10 表示"ENTER"

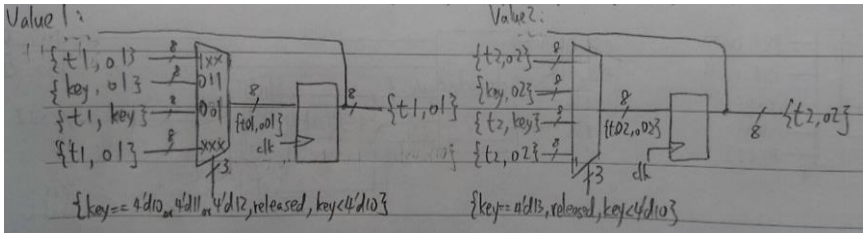


5. read：

   此 module 是用來讀取兩個數的值。

   - 特別在此設計切換每一個 state 的條件都不同，目的在於能避免彼此間互相影響，以及避免 key_valid 產生多的突波使 state 跳得太快。
   - 從`Value1 →`Value2 是當讀取到"+","-"or"*"的鍵盤訊號才會切換
     從`Value2→`ANSWER 是當讀取到"ENTER"的鍵盤訊號才會切換
     從`ANSWER →`Value1 是當讀取到”數字”的鍵盤訊號才會切換
   - state==`Value1 表示讀取第一個數；state==`Value2 表示讀取第二個數
     state==`ANSWER 表示顯示運算結果。

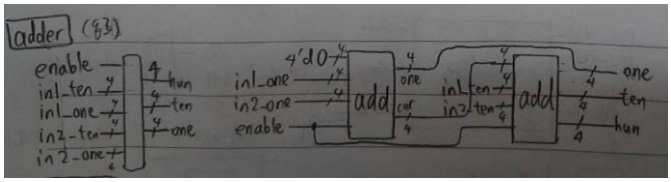




6. adder：

   此 module 是用來做二位數的加法，內含 add(做一位數的加法)。

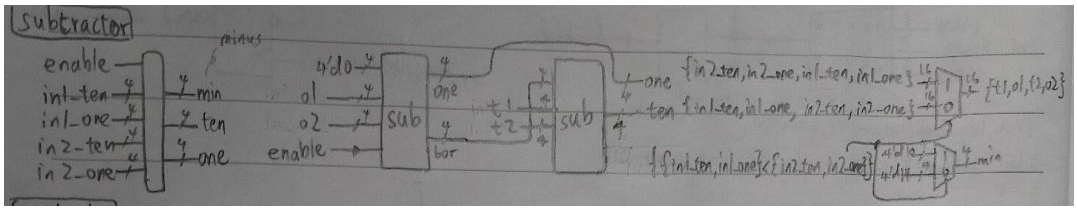
   - enable 是從 read module 接出來的 add 訊號。



7. subtractor：

   此 module 是用來做二位數的減法，內含 sub(做一位數的減法)。

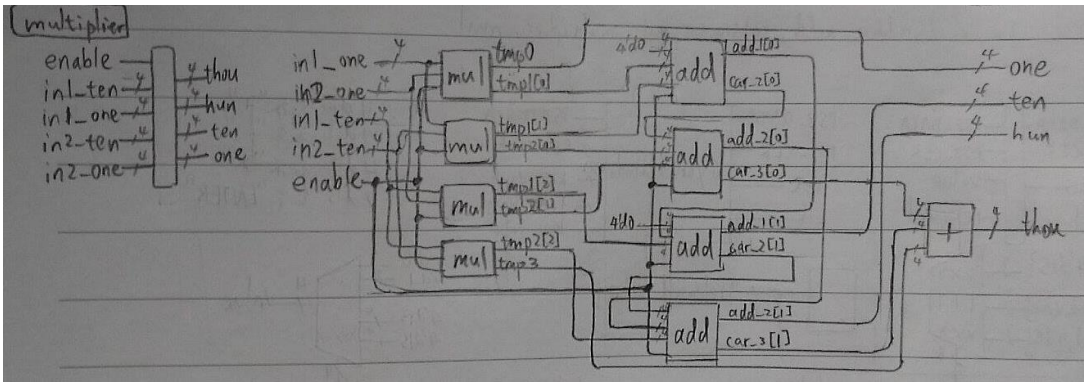
   - min 是當"(較小的數) - (較大的數)"產生的"負號"，在七段顯示器上顯示。
   - enable 是從 read module 接出來 sub 訊號。



8. multiplier：

   此 module 是用來做二位數的乘法，內含 add(做一位數的加法)以及 mul(做一位數的乘法)。

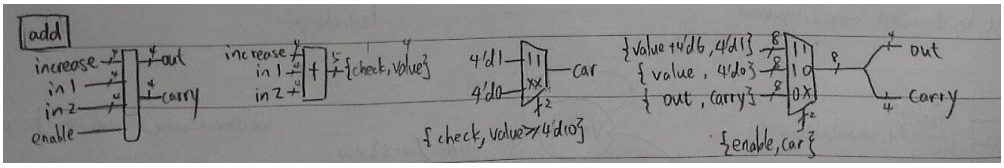
   - 由於二位數的乘法是 4 個一位數乘法產生的值，依照位數一一做相加，因此才會包 add 以及 mul module。
   - enable 是從 read module 接出來的 mul 訊號。



9. add：

   此 module 是用來做一位數的加法。

   - 使用 enable 作為 mux 的判斷值，可以使在 enable==1 的當下才會運算，其餘時間則維持當下的值。
   - 當 increase + in1 + in2 超過 4'd15 時會有 overflow 的產生，因此多接一個 check 訊號出來判斷是否有 overflow 的產生，若有 overflow 的產生，便使運算出來的值再加上 4'd6(經驗使然)使結果產生應有的值。

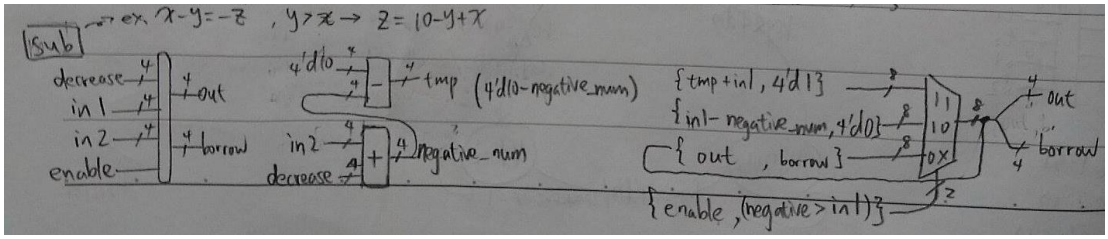


10. sub：

    此 module 是用來做一位數的減法。

    - 使用 enable 作為 mux 的判斷值，可以使在 enable==1 的當下才會運算，其餘時間則維持當下的值。

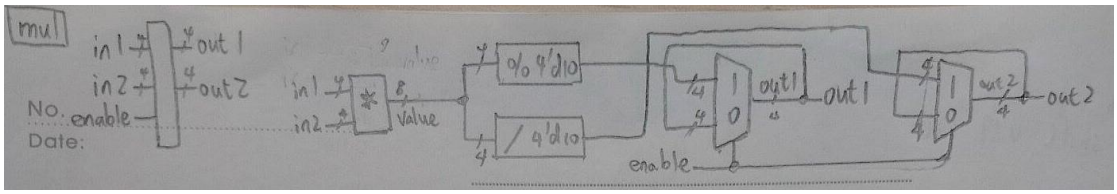
- 想法來源：ex. X-Y=-Z, Y>X → Z = 10 – Y + X



11. mul：

此 module 是用來做一位數的乘法。

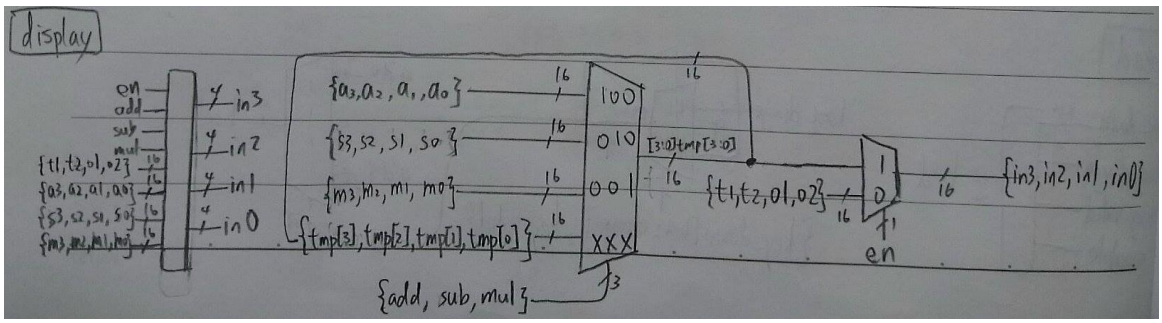
- 使用 enable 作為 mux 的判斷值，可以使在 enable==1 的當下才會運算，其餘時間則維持當下的值。



12. display：

此 module 是用來決定 scan_ctl 的 input :in3, in2,in1,in0。

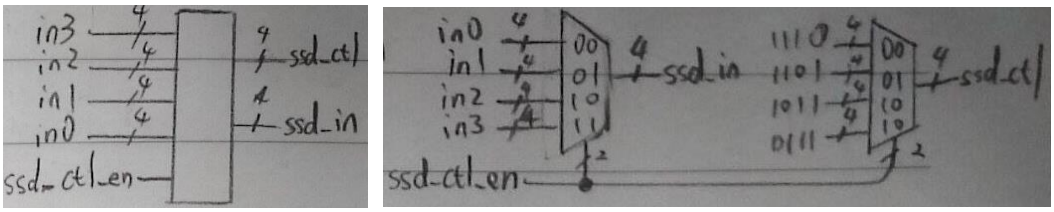
- en 是從 read module 接出來的 enter 訊號，目的是判斷要顯示的數是計算前抑或是計算後的值。
- 這裡接{add,sub,mul}作為 mux 訊號用意在判斷要顯示經個何種運算的結果。



13. scan_ctl：

此 module 是讓七段顯示器同時顯示不同位數的 module。

- ssd_ctl_en(from freq_div 的 clk_ctl[2:0])的用意為顯示兩個不同的數，因為七段顯示器一次只能顯示一種數次，因此需要一個頻率介於內建時間到 1hz 的 ssd_ctl_en 產生同時顯示兩個數的錯覺。



14. ssd：

當 bcd 為 4'd0: segs = 8'b00000011; //七段顯示器顯示”0”
當 bcd 為 4'd1: segs = 8'b10011111; //七段顯示器顯示”1”
當 bcd 為 4'd2: segs = 8'b00100101; //七段顯示器顯示”2”
當 bcd 為 4'd3: segs = 8'b00001101; //七段顯示器顯示”3”
當 bcd 為 4'd4: segs = 8'b10011001; //七段顯示器顯示”4”
當 bcd 為 4'd5: segs = 8'b01001001; //七段顯示器顯示”5”
當 bcd 為 4'd6: segs = 8'b01000001; //七段顯示器顯示”6”
當 bcd 為 4'd7: segs = 8'b00011111; //七段顯示器顯示”7”
當 bcd 為 4'd8: segs = 8'b00000001; //七段顯示器顯示”8”
當 bcd 為 4'd9: segs = 8'b00001001; //七段顯示器顯示”9”
當 bcd 為 4'd10:segs=8'b00010001; //七段顯示器顯示”A” //按下’A’鍵
當 bcd 為 4'd11:segs=8'b01001001; //七段顯示器顯示”S” //按下’S’鍵
當 bcd 為 4'd12:segs=8'b01100001; //七段顯示器顯示”E” //按下’M’鍵
當 bcd 為 4'd13:segs=8'b11111111; //七段顯示器顯示” ” //按下’ENTER’鍵//清空
default: segs = 8'11111111; //七段顯示器顯示” “

Result (state==`Value1(2’b01)表示讀取第一個數；state==`Value2(2’b10)表示讀取第二個數 state==`ANSWER(2’b11)表示顯示運算結果。)

1. 圖一~圖四是在進行"19 + 59 = 78"的過程(加法)。
   I. 圖一當 state==2'b01，可以對第一個數輸值。
   II. 圖二當按下"+"鍵 state 會從 2'b01 轉換至 2'b10。
   III. 圖三當 state==2'b10，可以對第二個數輸值。
   IV. 圖四當按下"ENTER"鍵會顯示出運算結果。

圖一 圖二

圖三　　圖四

2. 圖五~圖八是在進行"87 - 98 = -21"的過程(減法)。
   I. 圖五當 state==2'b01，可以對第一個數輸值。
   II. 圖六當按下"-"鍵 state 會從 2'b01 轉換至 2'b10。
   III. 圖七當 state==2'b10，可以對第二個數輸值。
   IV. 圖八當按下"ENTER"鍵會顯示出運算結果。

圖五　　圖六

圖七

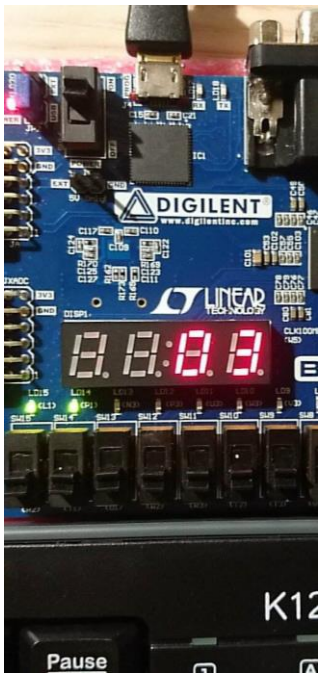
圖八

3. 圖九~圖十一是在進行"52 - 49 = 03"的過程(減法)。
   I. 圖九當按下"-"鍵 state 會從 2'b01 轉換至 2'b10。
   II. 圖十當按下" ENTER "鍵 state 會從 2'b10 轉換至 2'b11。
   III. 圖十一當按下"ENTER"鍵會顯示出運算結果。

圖九

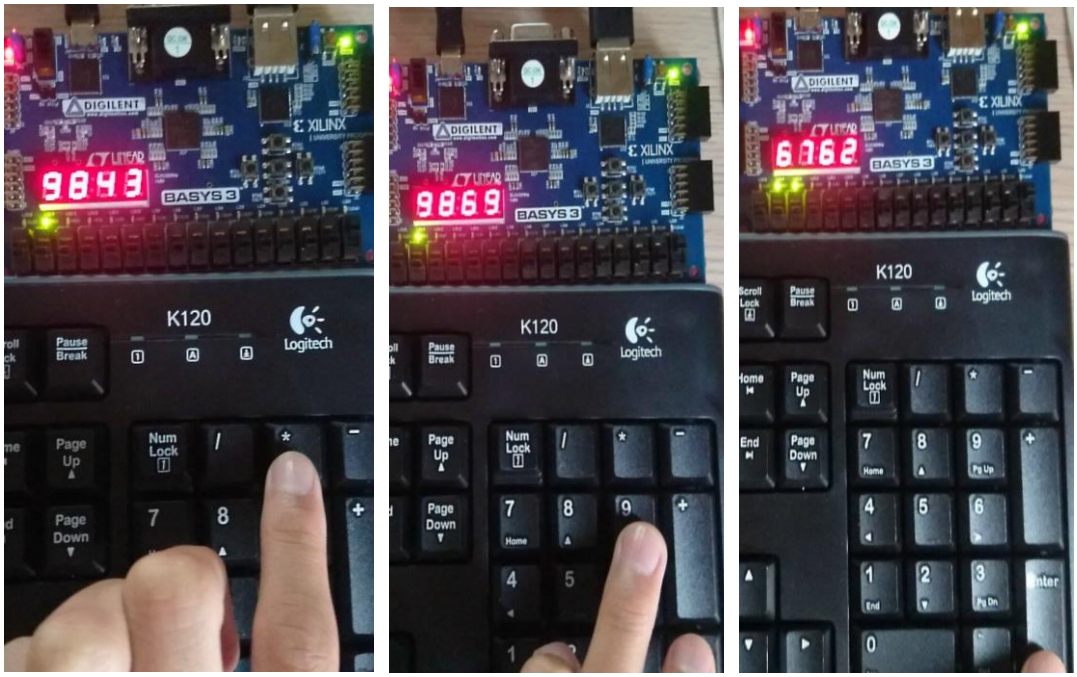
圖十

圖十一



4. 圖十二~圖十四是在進行"98 * 69 = 6762"的過程(乘法)。
    I. 圖十二當按下"*"鍵 state 會從 2'b01 轉換至 2'b10。
    II. 圖十三當按下" ENTER "鍵 state 會從 2'b10 轉換至 2'b11。
    III. 圖十四當按下"ENTER"鍵會顯示出運算結果。

圖十二 圖十三 圖十四



Discussion

1. decoder
   把 KeyboradDecoder 包在這個 module 的目的在於，使 KeyboradDecoder 影響範圍只限在這個 module 內，不會因產生不規則的訊號而破壞結果。
2. read
   特別在此設計切換每一個 state 的條件都不同，目的在於能避免彼此間互相影響，以及避免 key_valid 產生多的突波使 state 跳得太快。
3. add
   當 increase + in1 + in2 超過 4'd15 時會有 overflow 的產生，因此多接一個 check 訊號出來判斷是否有 overflow 的產生，若有 overflow 的產生，便使運算出來的值再加上 4'd6(經驗使然)使結果產生應有的值。
4. sub
   想法來源：ex. X-Y=-Z, Y>X → Z = 10 – Y + X。

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.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Design Specification

Input：clk,rst;

Output：ssd_ctl[3:0],seg[7:0], caps,shift, ASCII[7:0];

Inout：PS2_DATA,PS2_CLK; //V17 控制 rst，當 rst==0 時會產生 rst 的訊號

Design Implementation

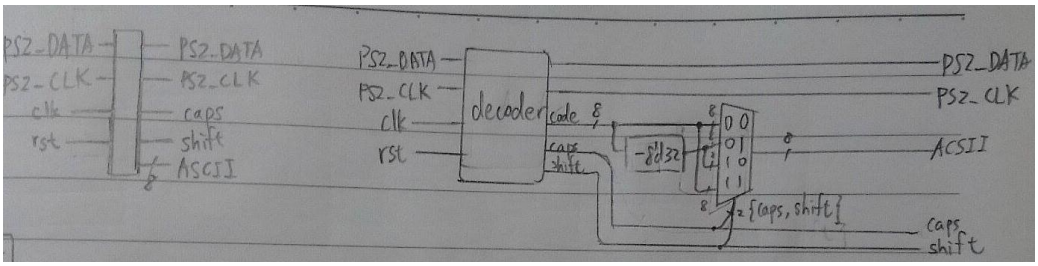
Logic function :

1. top：

此 module，專門拿來呼叫其他小 moudule 的。

此 module 還以 decoder 接出來的 caps 和 shift 訊號做為要顯示大寫或小寫英文字母的 ASCII code。

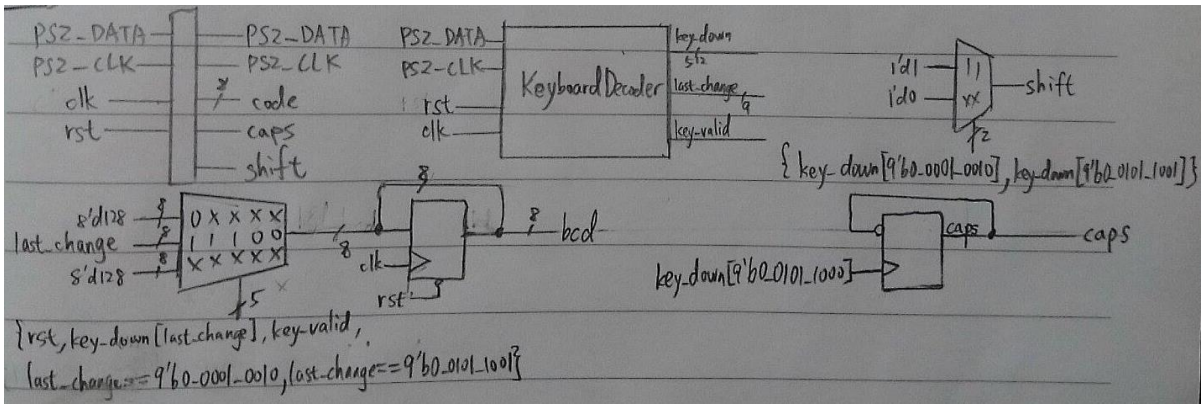
- caps==shift 時，顯示小寫英文字母的 ASCII code。
- caps!=shift 時，顯示大寫英文字母的 ASCII code。



2. decoder：

   此 module 是用來讀取從 key_board 輸入值，並轉成 BCD 的形式。

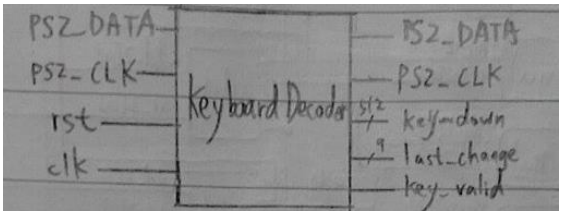
   - 取值的方法與第一題同，都是用 key_down[last_change] && key_valid 的方式判斷 last_change 為何值的被按下去。
   - 這設計以 key_down[9'b0_0101_1000]作為給 caps 值的 DFF 的 clk。這樣可以使當按下"caps"鍵時就會進行存值。
   - 當按下"caps"鍵時，caps<=~caps 即可改變現在的狀態。
   - shift 是做一個當按下"shift"鍵時 shift==1，沒按時 shift==0。



3. KeyboardDecoder：

   這個 module 主要的目的是讀取鍵盤的值。

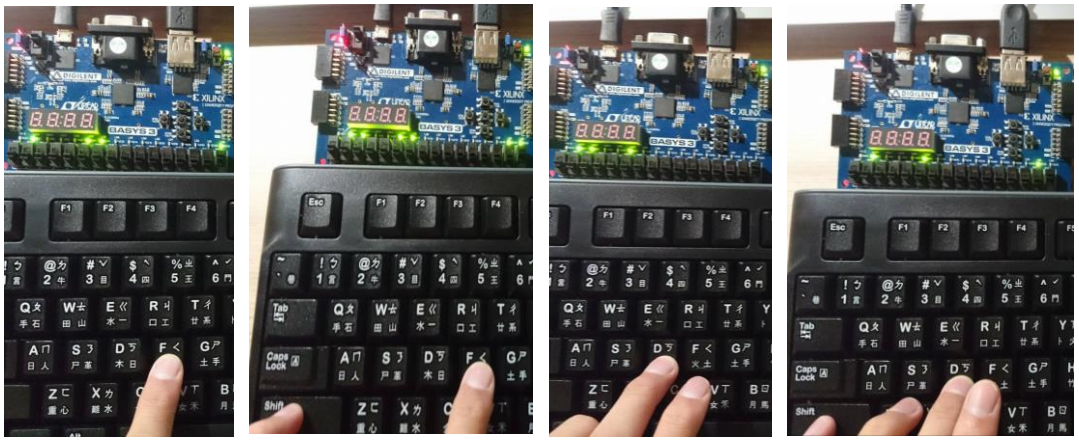
   - key_down 是表示那一個鍵有沒有被按下去；last_change 是表示按鍵按下去的值；key_valid 在按鍵被按下去或放開的當下方會產生一個突波(理論上)。



Result (caps==1 時顯示大寫；caps==0 時顯示小寫)(按下”shift”鍵時 shift==1)

1. 圖一：caps==1,shift==0 時，當按下”f”鍵→顯示 01000110(F)。
2. 圖二：caps==1,shift==1 時，當按下”f”鍵→顯示 01100110(f)。
3. 圖三：caps==0,shift==0 時，當按下”d”鍵→顯示 01100100(D)。
4. 圖四：caps==0,shift==1 時，當按下”d”鍵→顯示 01000100(d)。

圖一 圖二 圖三 圖四



Discussion

1. decoder

   這設計以 key_down[9’b0_0101_1000]作為給 caps 值的 DFF 的 clk。這樣可以使當按下”caps”鍵時就會進行存值。

2. 由於大寫與小寫字母的 ASCII code 都剛好相差 8’d32(小寫>大寫)，因此在做給 ASCII code 值時可以運用這個特性，不用把每個字母大小寫的 ASCII code 都解碼。

Conclusion：

這次的 lab 做的真的頗挫折的，尤其在做讀 key_board 值的部分不得已看了好幾天的日出，因為好像當按下按鍵的剎那間，key_valid 訊號貌似會產生多個小突波，導致輸值的時候都無法如心所願，不過很感謝助教細心的指導我，真的很感恩助教教我”盡量使切換每個 state 的條件都不同”這個方法，巧妙的避開 key_valid 和 key_down 可能產生的雜訊，真的很感恩。