邏輯設計實驗 Lab06 結報

104060012 邱怡庭

1 Implement the keypad press catch function with the FPGA demo board.

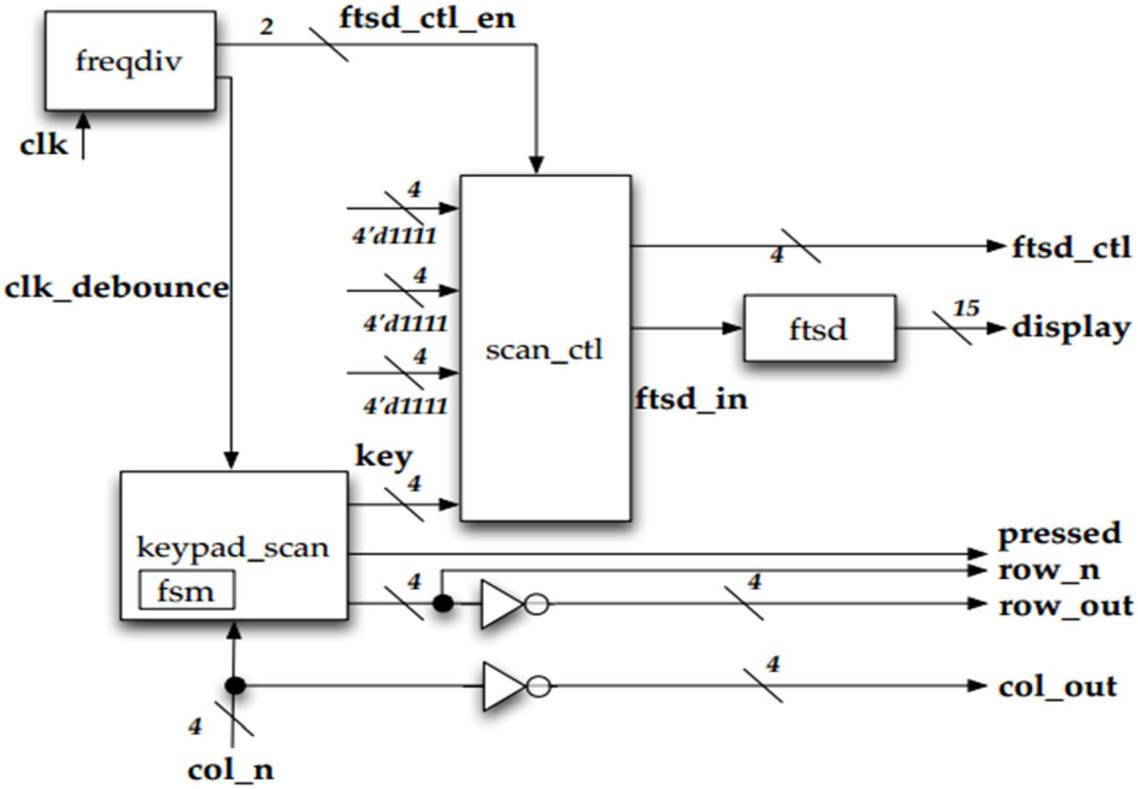
Design Specification

output : [3:0] row_n, pressed, ftsd_ctl, [14:0]display

input : clk, rst_n, [3:0] col_n

wire : clk_debounce, [1:0] ftsd_ctl_en, [3:0] key, [3:0] ftsd_in, [3:0] ftsd_ctl, [3:0] in0, [3:0]in1, [3:0]in2, [3:0]in3

block diagram:



Design Implementation

logic function / logic diagram:

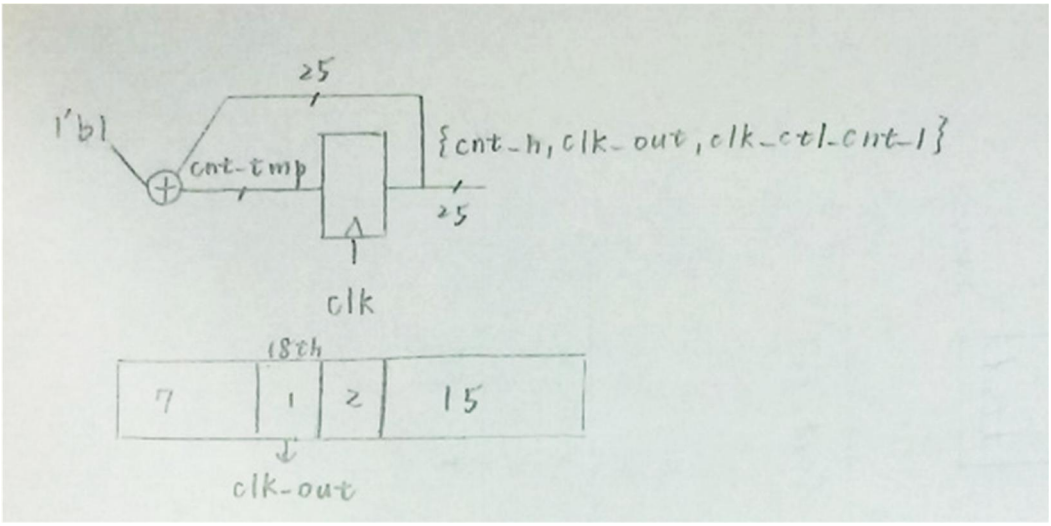
keypad_scan

{row_n,col_n}, the mapping:

// 1st row	// 3rd row
0111_0111 => F	1101_0111 => A
0111_1011 => E	1101_1011 => 2
0111_1101 => D	1101_1101 => 5
0111_1110 => C	1101_1110 => 8
// 2nd row	// 4th row
1011_0111 => B	1110_0111 => 0
1011_1011 => 3	1110_1011 => 1
1011_1101 => 6	1110_1101 => 4
1011_1110 =>9	1110_1110 =>7

Row scan (row_n: 0111->1011->1101->1110, loop every 4 clocks)

freq_div:



scan_ctl

if ftsd_ctl_en=00

->控制第一個 14 段顯示器

if ftsd_ctl_en=01

->控制第二個 14 段顯示器

if ftsd_ctl_en=10

->控制第三個 14 段顯示器

if ftsd_ctl_en=11

->控制第四個 14 段顯示器

ftsd

當 bcd=4'd0: display = 15'b0000_0011_1111_111; //0

當 bcd=4'd1: display = 15'b1111_1111_1011_011; //1

```
當 bcd=4'd2: display = 15'b0010_0100_1111_111; //2
當 bcd=4'd3: display = 15'b0000_1100_1111_111; //3
當 bcd=4'd4: display = 15'b1001_1000_1111_111; //4
當 bcd=4'd5: display = 15'b0100_1000_1111_111; //5
當 bcd=4'd6: display = 15'b0100_0000_1111_111; //6
當 bcd=4'd7: display = 15'b0001_1111_1111_111; //7
當 bcd=4'd8: display = 15'b0000_0000_1111_111; //8
當 bcd=4'd9: display = 15'b0000_1000_1111_111; //9
當 bcd=4'd10: display = 15'b0001_0000_1111_111; //A
當 bcd=4'd11: display = 15'b1100_0000_1111_111; //B
當 bcd=4'd12: display = 15'b0110_0011_1111_111; //C
當 bcd=4'd13: display = 15'b1000_0100_1111_111; //D
當 bcd=4'd14: display = 15'b0110_0000_1111_111; //E
當 bcd=4'd15: display = 15'b0111_0000_1111_111; //F
default: display = 15'b1111_1111_1111_111; //DEF
```

I/O pin assignment:

```
NET "clk" LOC = R10;
NET "rst_n" LOC = T1;
NET "ftsd_ctl[0]" LOC = V8;
NET "ftsd_ctl[1]" LOC = U8;
NET "ftsd_ctl[2]" LOC = V6;
NET "ftsd_ctl[3]" LOC = T6;
NET "display[14]" LOC = P6;
NET "display[13]" LOC = N4;
NET "display[12]" LOC = V5;
NET "display[11]" LOC = T5;
NET "display[10]" LOC = U7;
NET "display[9]" LOC = R3;
NET "display[8]" LOC = N5;
NET "display[7]" LOC = R5;
NET "display[6]" LOC = T3;
NET "display[5]" LOC = T4;
NET "display[4]" LOC = V4;
NET "display[3]" LOC = V7;
NET "display[2]" LOC = R7;
NET "display[1]" LOC = T7;
NET "display[0]" LOC = U5;
```

```
NET "col_n[3]" LOC = H1;
NET "col_n[2]" LOC = H2;
NET "col_n[1]" LOC = J1;
NET "col_n[0]" LOC = J3;
NET "row_n[3]" LOC = L3;
NET "row_n[2]" LOC = L4;
NET "row_n[1]" LOC = K1;
NET "row_n[0]" LOC = K2;
```

Discussion:

row 會隨著時間不停變換，不同案件去控制不同的 col 使 col 和 row 配對產生不同的結果。

2 Implement a single digit decimal adder using the keypad as the input, push button as the plus function, 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).

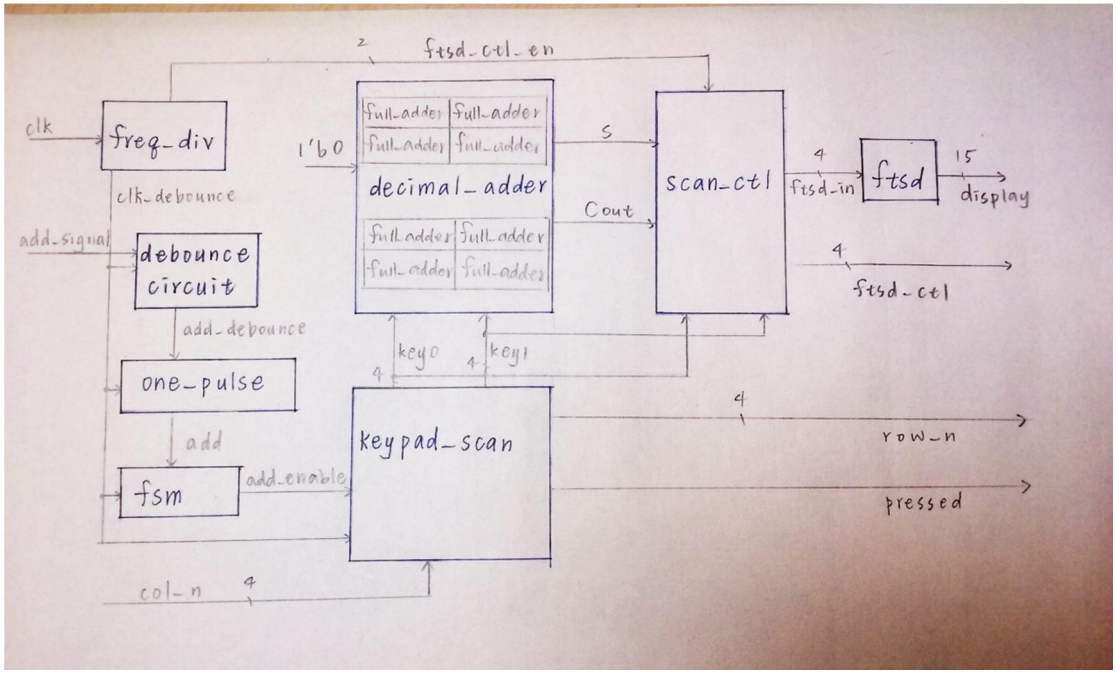
Design Specification

output : [3:0] row_n, [14:0] display, [3:0] ftsd_ctl, pressed

input : clk, rst_n, [3:0] col_n, add_signal

wire : clk_debounce, [3:0] key0, [3:0]key1, [3:0] s, Cout, [1:0] ftsd_ctl_en, [3:0] ftsd_in, add_debounce, add, add_enable

block diagram:



Design Implementation

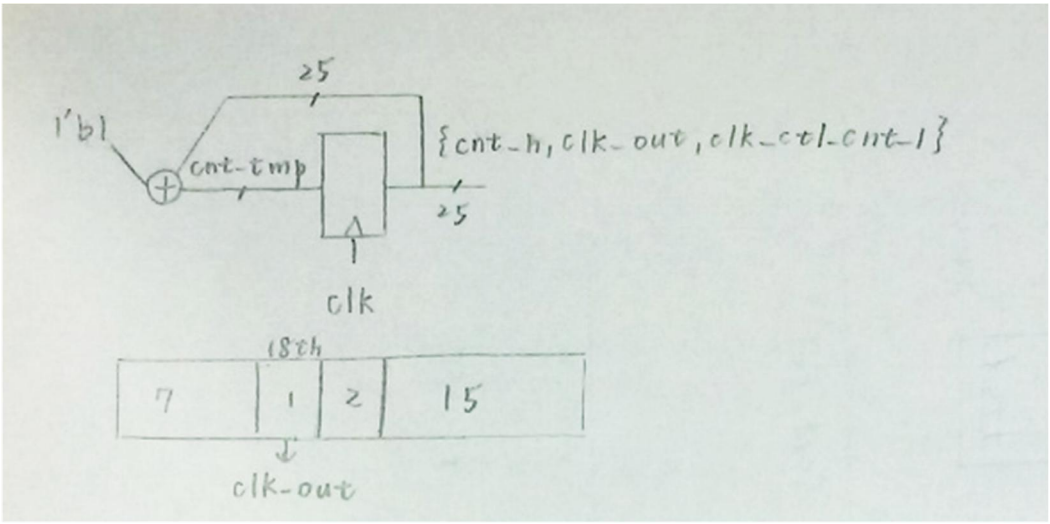
logic function / logic diagram:

keypad_scan

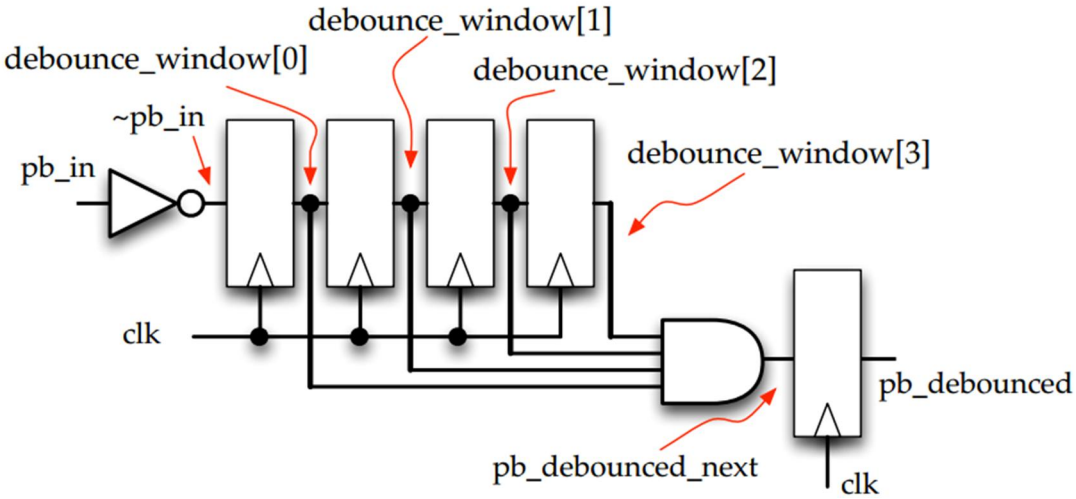
{row_n,col_n}, the mapping:

// 1st row	// 3rd row
0111_0111 => F	1101_0111 => A
0111_1011 => E	1101_1011 => 2
0111_1101 => D	1101_1101 => 5
0111_1110 => C	1101_1110 => 8
// 2nd row	// 4th row
1011_0111 => B	1110_0111 => 0
1011_1011 => 3	1110_1011 => 1
1011_1101 => 6	1110_1101 => 4
1011_1110 =>9	1110_1110 =>7

freq_div:

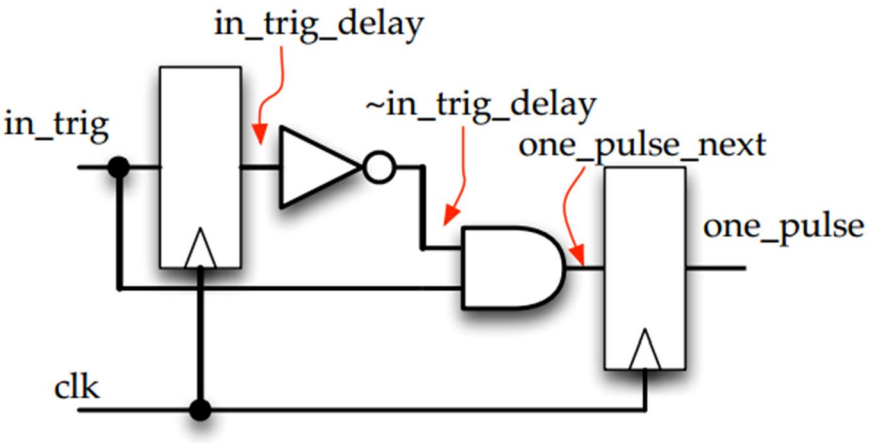


debounce_circuit



When all 4 bits of the registers are high the output of the debounce circuit changes to high

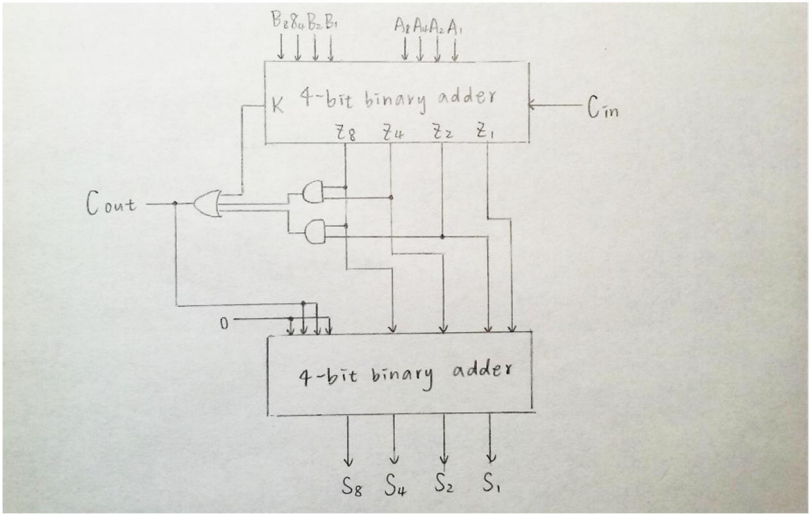
one_pulse



scan_ctl

if ftsd_ctl_en=00
->控制第一個 14 段顯示器
if ftsd_ctl_en=01
->控制第二個 14 段顯示器
if ftsd_ctl_en=10
->控制第三個 14 段顯示器
if ftsd_ctl_en=11
->控制第四個 14 段顯示器

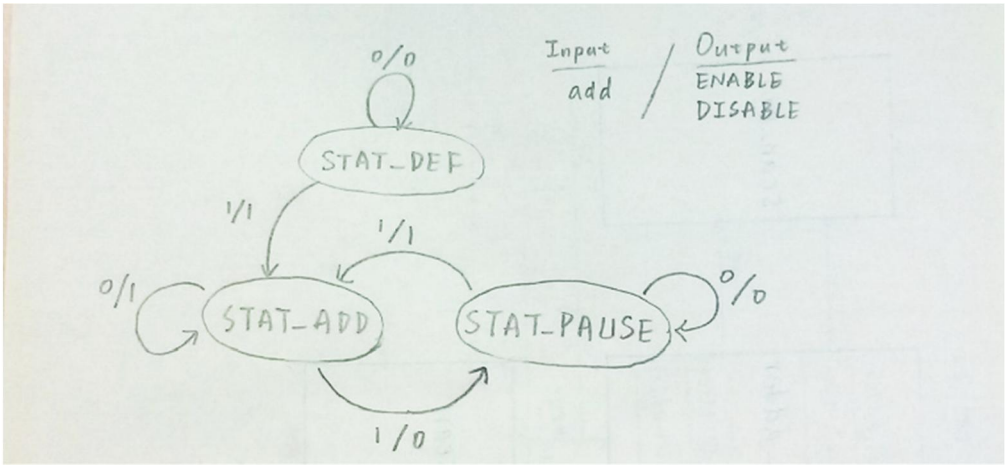
decimal_adder



ftsd

當 bcd=4'd0: display = 15'b0000_0011_1111_111; //0
當 bcd=4'd1: display = 15'b1111_1111_1011_011; //1
當 bcd=4'd2: display = 15'b0010_0100_1111_111; //2
當 bcd=4'd3: display = 15'b0000_1100_1111_111; //3
當 bcd=4'd4: display = 15'b1001_1000_1111_111; //4
當 bcd=4'd5: display = 15'b0100_1000_1111_111; //5
當 bcd=4'd6: display = 15'b0100_0000_1111_111; //6
當 bcd=4'd7: display = 15'b0001_1111_1111_111; //7
當 bcd=4'd8: display = 15'b0000_0000_1111_111; //8
當 bcd=4'd9: display = 15'b0000_1000_1111_111; //9
default: display = 15'b1111_1111_1111_111; //DEF

fsm



I/O pin assignment:

```
NET "col_n[3]" LOC = H1;
NET "col_n[2]" LOC = H2;
NET "col_n[1]" LOC = J1;
NET "col_n[0]" LOC = J3;
NET "row_n[3]" LOC = L3;
NET "row_n[2]" LOC = L4;
NET "row_n[1]" LOC = K1;
NET "row_n[0]" LOC = K2;
NET "display[0]" LOC = U5;
NET "display[1]" LOC = T7;
NET "display[2]" LOC = R7;
NET "display[3]" LOC = V7;
```

```
NET "display[4]" LOC = V4;
NET "display[5]" LOC = T4;
NET "display[6]" LOC = T3;
NET "display[7]" LOC = R5;
NET "display[8]" LOC = N5;
NET "display[9]" LOC = R3;
NET "display[10]" LOC = U7;
NET "display[11]" LOC = T5;
NET "display[12]" LOC = V5;
NET "display[13]" LOC = N4;
NET "display[14]" LOC = P6;
NET "ftsd_ctl[3]" LOC = T6;
NET "ftsd_ctl[2]" LOC = V6;
NET "ftsd_ctl[1]" LOC = U8;
NET "ftsd_ctl[0]" LOC = V8;
NET "add_signal" LOC =U1;
NET "clk" LOC = R10;
NET "rst_n" LOC = T1;
```

Discussion:

利用 fsm 設置在不同 state 的不同功能，並結合 decimal adder 形成簡單的加法計算機。

3 Implement a two-digit decimal adder/subtractor.

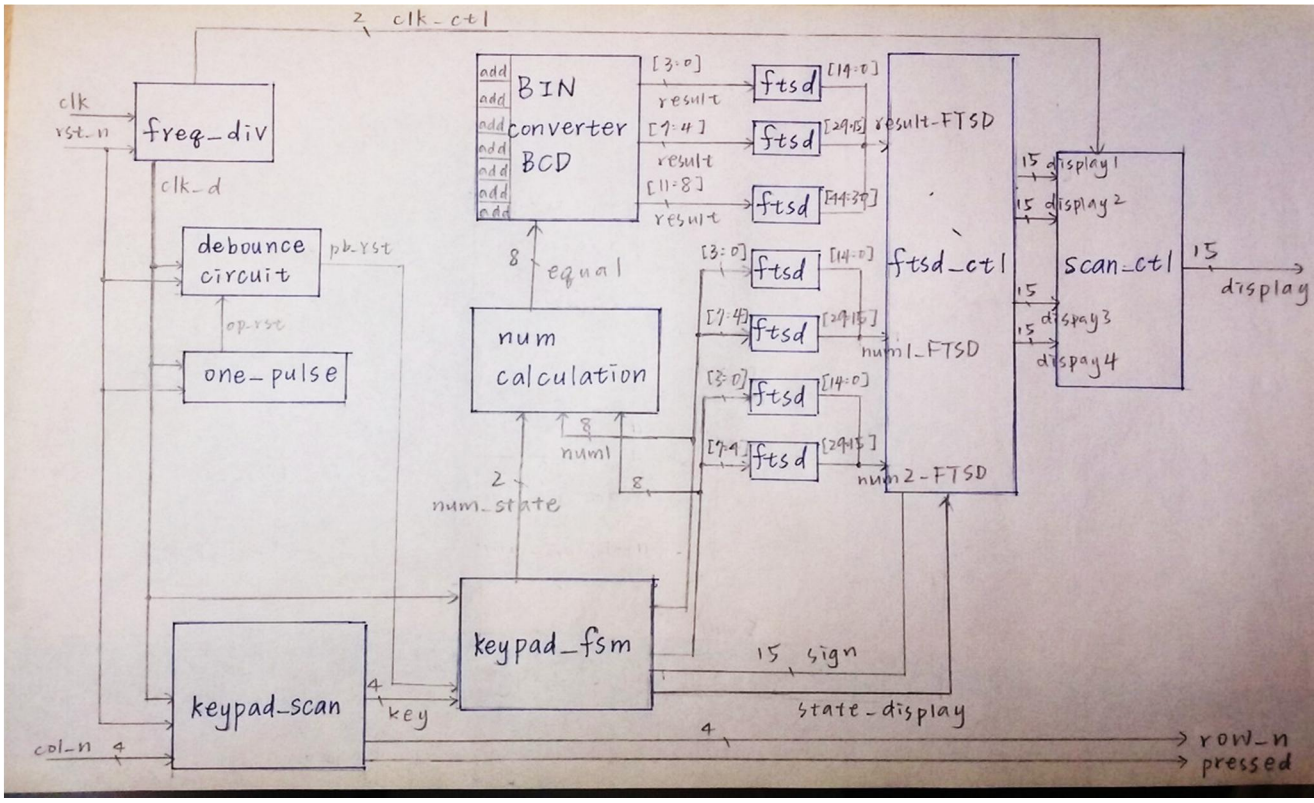
Design Specification

output : [3:0] row_n, [18:0] display

input : clk, rst_n, [3:0] col_n

wire : [1:0] clk_ctl, clk_d, [3:0] key, pressed, [1:0] cal_state, [14:0] sign , [1:0] state_num_input, [7:0] num1, [7:0]num2, [7:0] equal, [11:0] result, [29:0] num1_FTSD, [29:0]num2_FTSD, [44:0] result_FTSD, [14:0] sign_FTSD, state_display, [14:0] display1, [14:0]display2, [14:0]display3, [14:0]display4, pb_rst, op_rst

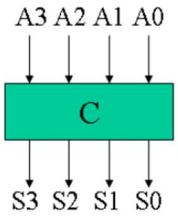
block diagram:



Design Implementation

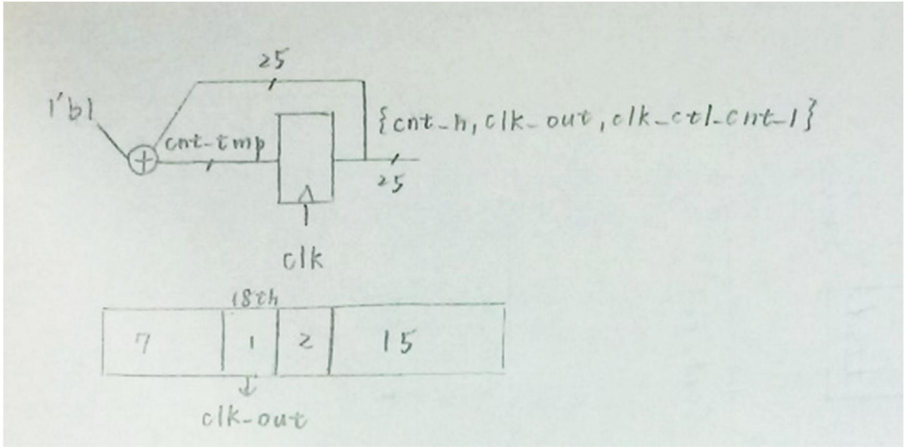
logic function / logic diagram:

add

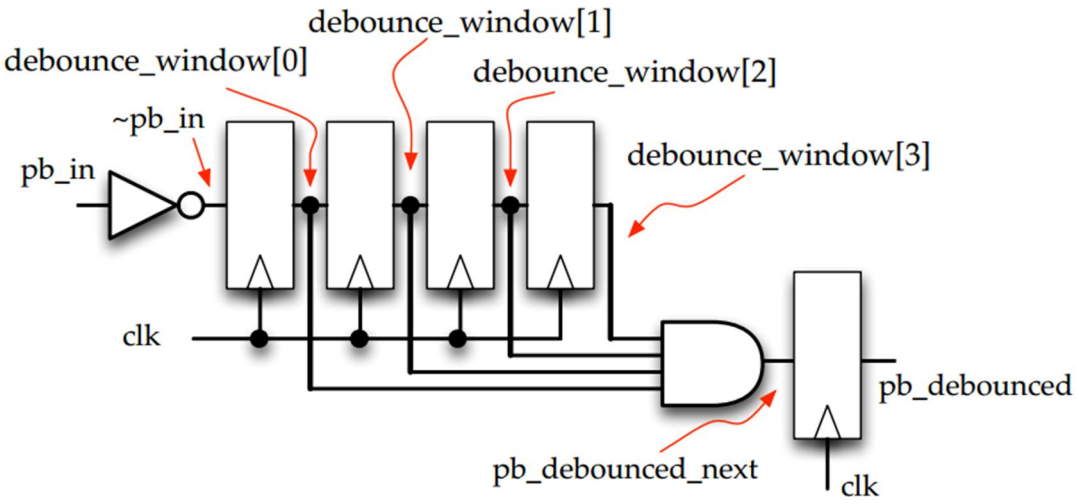


A3	A2	A1	A0	S3	S2	S1	S0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

freq_div:

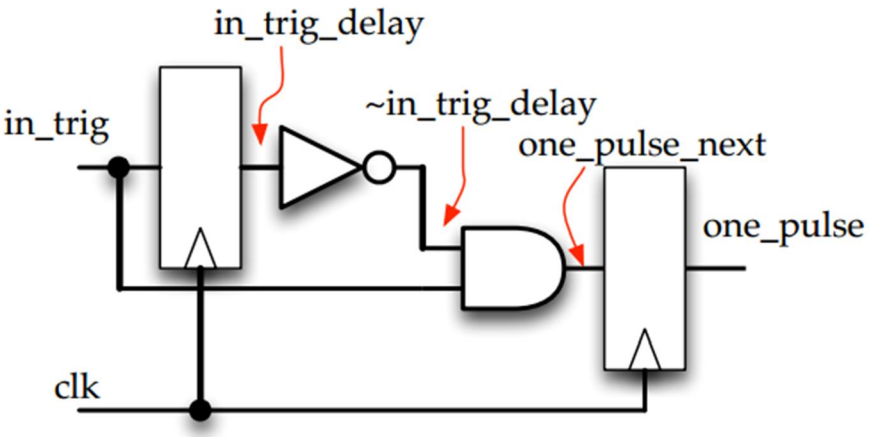


debounce_circuit



When all 4 bits of the registers are high the output of the debounce circuit changes to high

one_pulse



When one presses the push button for a short moment, the time that the switch is closed (ms range) is usually much longer than one clock period (µs or ns range). The one-pulse circuit generates only a one-clockperiod-long pulse every time the push button is hit, independent of the time one keeps the button pressed

scan_ctl

if ftsd_ctl_en=00

->控制第一個 14 段顯示器

if ftsd_ctl_en=01

->控制第二個 14 段顯示器

if ftsd_ctl_en=10

->控制第三個 14 段顯示器

if ftsd_ctl_en=11

->控制第四個 14 段顯示器

ftsd

當 num=4'd0: display = 15'b0000_0011_1111_111; //0

當 num=4'd1: display = 15'b1111_1111_1011_011; //1

當 num=4'd2: display = 15'b0010_0100_1111_111; //2

當 num=4'd3: display = 15'b0000_1100_1111_111; //3

當 num=4'd4: display = 15'b1001_1000_1111_111; //4

當 num=4'd5: display = 15'b0100_1000_1111_111; //5

當 num=4'd6: display = 15'b0100_0000_1111_111; //6

當 num=4'd7: display = 15'b0001_1111_1111_111; //7

當 num=4'd8: display = 15'b0000_0000_1111_111; //8

當 num=4'd9: display = 15'b0000_1000_1111_111; //9

default: display = 15'b1111_1111_1111_111; //DEF

num_calculator

當 cal_state == `ADD

->equal = bin1+bin2;

->sign = 15'b1111_1111_1111_111;(nothing)

當 cal_state == `SUB && bin1 < bin2

-> equal = ~(bin1-bin2-1'b1);

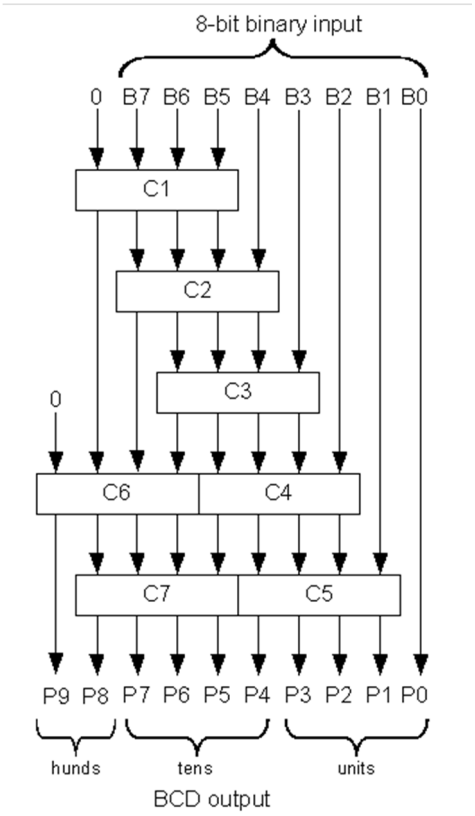
-> sign = 15'b1111_1100_1111_111;(負號)

當 cal_state == `SUB && bin1 >bin2

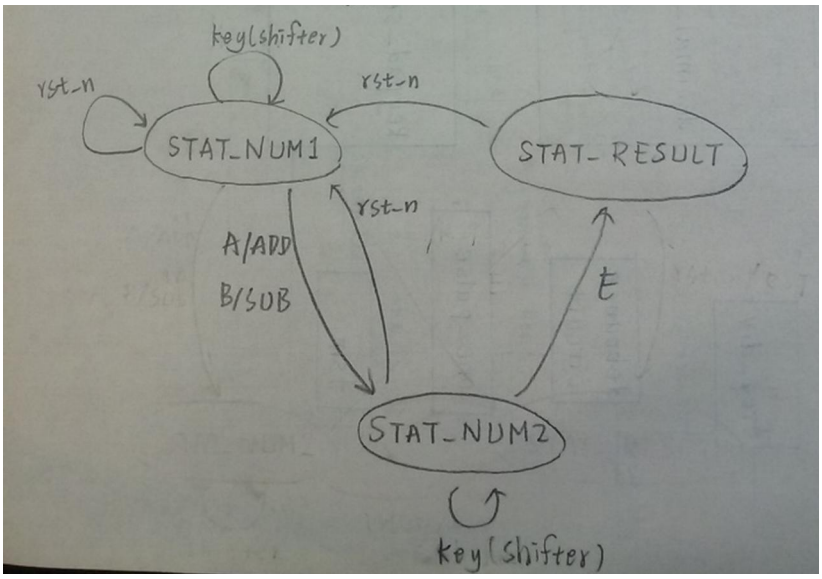
->equal = bin1-bin2;

->sign = 15'b1111_1111_1111_111;(nothing)

BIN_converter_BCD



keypad_fsm



Ftsd_ctl

If state=0 {display1,display2,display3,display4} = {num1_FTSD,num2_FTSD};(顯示所按的數字)

state=1{display1,display2,display3,display4} = {sign_FTSD,result_FTSD};(顯示計算結果)

keypad_scan

{row_n,col_n}, the mapping:

```
// 1st row                  // 3rd row
0111_0111 => F              1101_0111 => A
0111_1011 => E              1101_1011 => 2
0111_1101 => D              1101_1101 => 5
0111_1110 => C              1101_1110 => 8

// 2nd row                  // 4th row
1011_0111 => B              1110_0111 => 0
1011_1011 => 3              1110_1011 => 1
1011_1101 => 6              1110_1101 => 4
1011_1110 =>9               1110_1110 =>7
```

Row scan (row_n: 0111->1011->1101->1110, loop every 4 clocks)

I/O pin assignment:

```
NET "clk" LOC = R10;
NET "rst_n" LOC = N3;

NET "display[18]" LOC = T6;
NET "display[17]" LOC = V6;
NET "display[16]" LOC = U8;
NET "display[15]" LOC = V8;
NET "display[14]" LOC = P6;
NET "display[13]" LOC = N4;
NET "display[12]" LOC = V5;
NET "display[11]" LOC = T5;
NET "display[10]" LOC = U7;
NET "display[9]" LOC = R3;
NET "display[8]" LOC = N5;
NET "display[7]" LOC = R5;
NET "display[6]" LOC = T3;
NET "display[5]" LOC = T4;
```

```
NET "display[4]" LOC = V4;
NET "display[3]" LOC = V7;
NET "display[2]" LOC = R7;
NET "display[1]" LOC = T7;
NET "display[0]" LOC = U5;

NET "col_n[3]" LOC = H1;
NET "col_n[2]" LOC = H2;
NET "col_n[1]" LOC = J1;
NET "col_n[0]" LOC = J3;
NET "row_n[3]" LOC = L3;
NET "row_n[2]" LOC = L4;
NET "row_n[1]" LOC = K1;
NET "row_n[0]" LOC = K2;
```

Discussion:

利用 shifter 將原本存放在個位數的數字轉移到十位數，形成兩位數字，再使用 fsm 設置不同狀態下的不同功能，此外，因為含有減法功能，所以必須考慮兩正數相減亦有可能為負數，因此須分別討論需要加負號的狀態。

Conclusion:

在 lab06 中，我學會了控制按鈕及設計 fsm 去改變狀態始之呈現不同功能，並製造簡易計算機，雖然越來越多的 state 使我有些混亂，但其實只要按照固定的規則，要做多少模式切換應該都不會太困難。