數位系統技術

Calculator

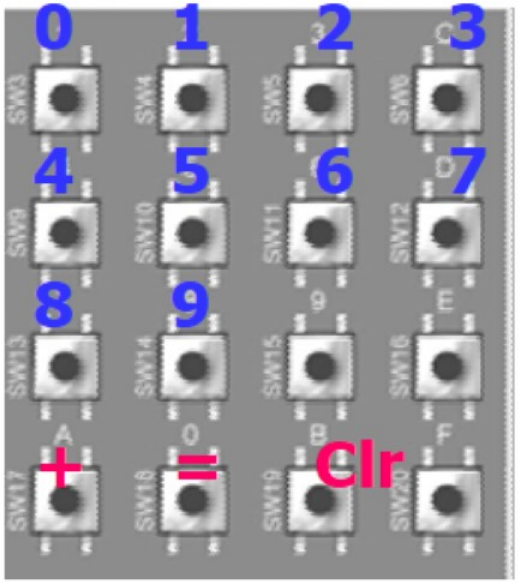
Ren-Der Chen (陳仁德)
Department of Computer Science and
Information Engineering
National Changhua University of Education
E-mail: rdchen@cc.ncue.edu.tw
Spring, 2025

說明

- 本實驗前半段為使用ModelSim (or Quartus II)模擬電路行為，同學完成key_buffer.v之設計，並視需要觀察模擬結果。
- 後半段為使用Quartus II驗證電路於實驗板之行為，同學修改key_buffer.v之輸入/輸出信號，設計keyboard2_top.v，並完成電路燒錄，觀察驗證結果。

實驗內容 (1/3)

- 修改keyboard單元，設計一簡易之計算機電路，由keyboard輸入兩個數字後，執行加法運算 (X + Y = Z)，輸入之數字最多為6位數。
- Ex. 386 + 92 = 478，Clr 為清除鍵



Keyboard

實驗內容 (2/3)

- 設計一個資料儲存電路key_buf_code_1[23:0]以存放 X (最多6位數字)，每個數字4位元。
- 按下keyboard的其中一個0~9鍵時，數字會存入key_buf_code_1[3:0]位置，原先在key_buf_code_1[19:0]的數字則會左移。
- 另外再設計一個資料儲存電路key_buf_code_2[23:0]以存放 Y。

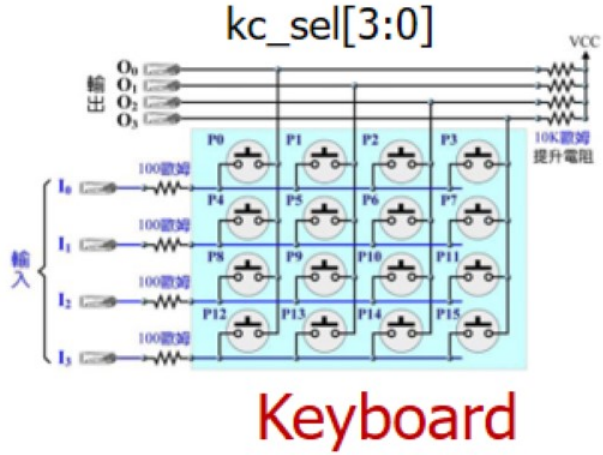
實驗內容 (3/3)

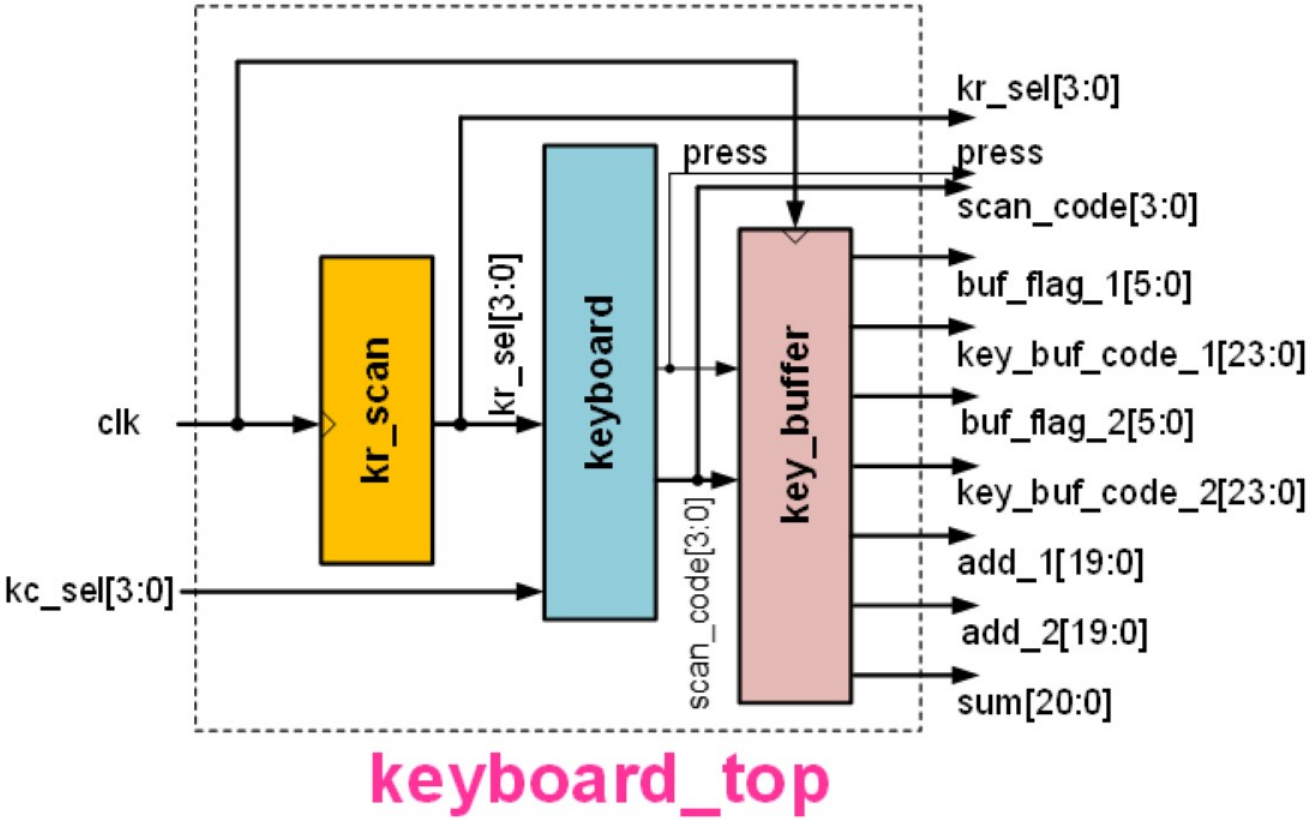
- 一開始，當keyboard有按鍵按下時(press=1)，所按之鍵的二進位值會顯示於輸出信號scan_code[3:0]。
- 當0-9鍵按下時，按鍵的值會存放於key_buf_code_1，接著當“+”鍵按下後，後續的0-9鍵則會存放於key_buf_code_2。
- 最後再按下”=“鍵時，會將key_buf_code_1 (BCD編碼)之值轉碼於add_1[19:0]，將key_buf_code_2 (BCD編碼)之值轉碼於add_2[19:0]，之後將add_1與add_2相加之結果顯示於sum[20:0]。
- 當按下”Clr”鍵時，所有紀錄之資料均清除為0。

Example

- 按下**3**，key_buf_code_1 = _____3
- 按下**8**，key_buf_code_1 = ____38
- 按下**6**，key_buf_code_1 = ___386
- 按下**+**
- 按下**9**，key_buf_code_2 = _____9
- 按下**2**，key_buf_code_2 = ____92
- 按下**=**，sum = 478
- 按下**Clr**，所有紀錄之資料均清除為0

keyboard_top 電路方塊圖

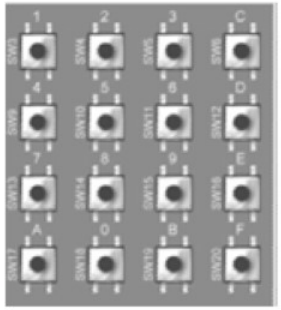




Design 1: kr_scan

- 設計keyboard之row掃描電路kr_scan.v，由上至下掃描keyboard。

1110
1101
1011
0111



```
module kr_scan(clk, rst, kr_out);
    input clk, rst;
    output reg [3:0] kr_out;

    always@(posedge clk or posedge rst) begin
        if (rst)
            kr_out = 4'b1110; // top row
        else // left circular shift
            kr_out = {kr_out[2:0], kr_out[3]};
            // 1110 -> 1101 -> 1011 -> 0111
            // select row from up to down
    end
endmodule
```

Design 2: keyboard (1/6)

- 設計keyboard偵測電路keyboard.v，判斷keyboard是否有按鍵按下，並產生按鍵之二進位編碼。

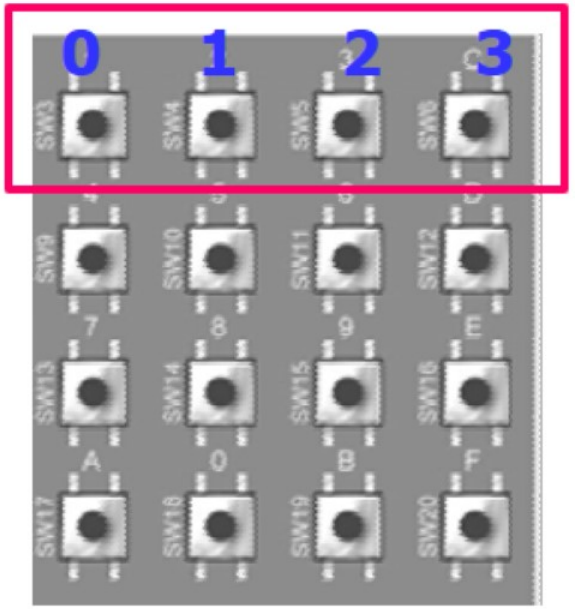
```
module keyboard(kr, kc, press, scan_code);
    input [3:0] kr;
    input [3:0] kc;
    output reg press;
    output reg [3:0] scan_code;

    always@(kr or kc) begin
        case(kr)
```

1. kr[3:0]為keyboard的row掃描輸入
2. kc[3:0]為keyboard的column偵測輸出
3. 若keyboard偵測到有任何鍵按下去，press之值為1
4. scan_code[3:0]為所按之鍵的二進位編碼

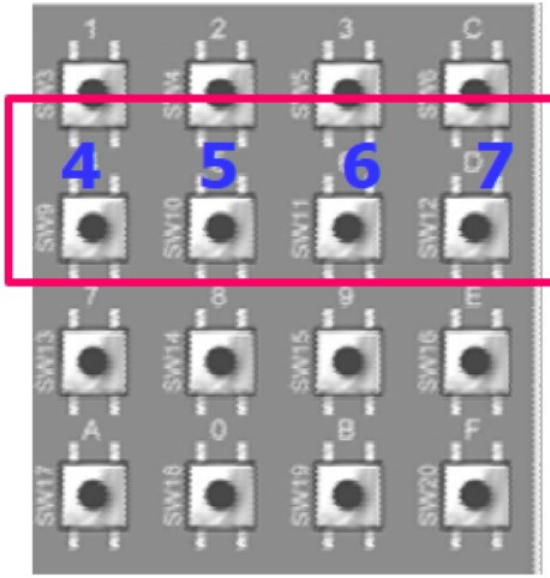
Design 2: keyboard (2/6)

```
4'b1110: // 1st row
    case(kc)
        4'b1110: begin  press =1'b1;
                        scan_code = 4'h0;
                 end // 0
        4'b1101: begin  press =1'b1;
                        scan_code = 4'h1;
                 end // 1
        4'b1011: begin  press =1'b1;
                        scan_code = 4'h2;
                 end // 2
        4'b0111: begin  press =1'b1;
                        scan_code = 4'h3;
                 end // 3
        default: begin  press =1'b0;
                        scan_code = 4'h0;
                 end // default
    endcase
```



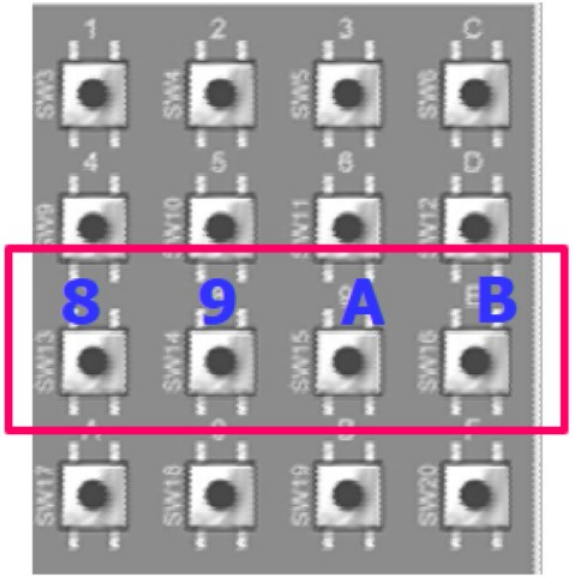
Design 2: keyboard (3/6)

```
4'b1101: // 2nd row
    case(kc)
        4'b1110: begin  press =1'b1;
                        scan_code = 4'h4;
                 end // 4
        4'b1101: begin  press =1'b1;
                        scan_code = 4'h5;
                 end // 5
        4'b1011: begin  press =1'b1;
                        scan_code = 4'h6;
                 end // 6
        4'b0111: begin  press =1'b1;
                        scan_code = 4'h7;
                 end // 7
        default: begin  press =1'b0;
                        scan_code = 4'h0;
                 end // default
    endcase
```



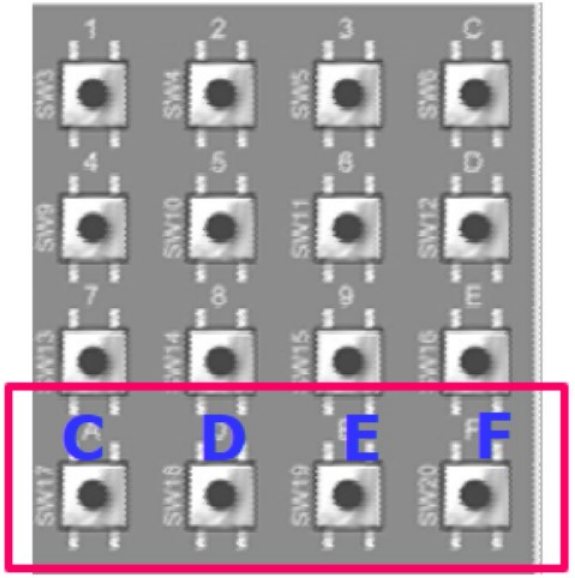
Design 2: keyboard (4/6)

```
4'b1011: // 3rd row
    case(kc)
        4'b1110: begin  press =1'b1;
                        scan_code = 4'h8;
                 end // 8
        4'b1101: begin  press =1'b1;
                        scan_code = 4'h9;
                 end // 9
        4'b1011: begin  press =1'b1;
                        scan_code = 4'hA;
                 end // A
        4'b0111: begin  press =1'b1;
                        scan_code = 4'hB;
                 end // B
        default: begin  press =1'b0;
                        scan_code = 4'h0;
                 end // default
    endcase
```



Design 2: keyboard (5/6)

```
4'b0111: // 4th row
    case(kc)
        4'b1110: begin  press =1'b1;
                        scan_code = 4'hC;
                 end // C
        4'b1101: begin  press =1'b1;
                        scan_code = 4'hD;
                 end // D
        4'b1011: begin  press =1'b1;
                        scan_code = 4'hE;
                 end // E
        4'b0111: begin  press =1'b1;
                        scan_code = 4'hF;
                 end // F
        default: begin  press =1'b0;
                        scan_code = 4'h0;
                 end // default
    endcase
```



Design 2: keyboard (6/6)

```
            default: begin   press = 1'b0;
                             scan_code = 4'h0;
                     end
        endcase
    end
endmodule
```

Design 3: key_buffer (1/7)

```
module key_buffer(clk, rst, press, scan_code,
buf_flag_1, key_buf_code_1, buf_flag_2, key_buf_code_2, add_1, add_2, sum);

    input   clk, rst, press;
    input   [3:0] scan_code;

    output  reg [5:0] buf_flag_1; // 6 numbers
    output  reg [23:0] key_buf_code_1;

    output  reg [5:0] buf_flag_2; // 6 numbers
    output  reg [23:0] key_buf_code_2;

    output  reg [19:0] add_1, add_2;
    output  reg [20:0] sum;
```

1. press為1時表示keyboard有按鍵被按下。
2. scan_code[3:0]為所按下之鍵的二進位編碼。
3. buf_flag_1[5:0] & buf_flag_2[5:0]為記錄buffer電路中每個位置是否有數字存入。
4. key_buf_code_1[23:0] & key_buf_code_2[23:0]為存放6個數字之24位元buffer電路，每個數字4個位元。
5. add_1[19:0] & add_2[19:0] 為欲相加之兩個數字，sum[20:0] 為相加後之結果。

Design 3: key_buffer (2/7)

```
reg cal_state;
// cal_state=0, add_1
// cal_state=1, add_2
// sum = add_1 + add_2

reg [3:0] decimal_num;
integer i, j;
```

1. cal_state記錄目前所輸入之值為add_1 (cal_state=0)或是add2 (cal_state=1)。
2. decimal_num紀錄key_buf_code_1[23:0] & key_buf_code_2[23:0]中所存放的某一個4-bit數字之值。

Design 3: key_buffer (3/7)

```
always@(posedge clk or posedge rst) begin
    if (rst) begin
        buf_flag_1 = 6'b000000;
        key_buf_code_1 = 24'h000000;

        buf_flag_2 = 6'b000000;
        key_buf_code_2 = 24'h000000;

        add_1 = 20'd0;
        add_2 = 20'd0;
        sum = 21'd0;

        cal_state = 1'b0; // add_1
    end
```

初始時，buffer電路是空的

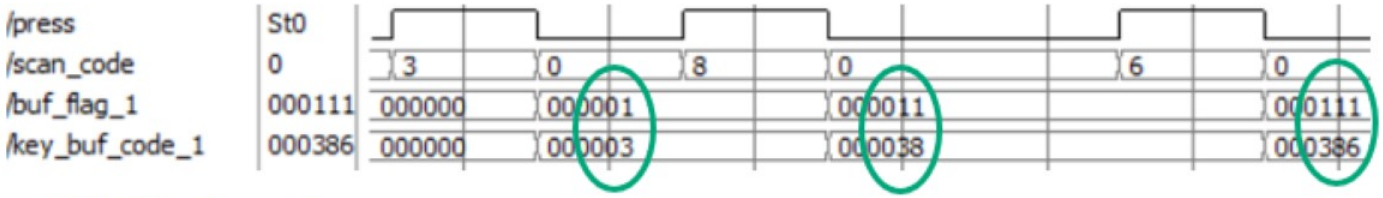
初始時，buffer電路是空的

Design 3: key_buffer (4/7)

若有0-9鍵按下，數字將存入buffer，buf_flag_1[0]設為1，原buf_flag_1[4:0]之值左移

所按之鍵的二進位編碼scan_code[3:0]存入key_buf_code_1[3:0]，原本的[23:20]左移出buffer，[19:0]左移1個數字(4個位元)

16進位




```
else
    if (press == 1'b1) begin
        if (scan_code >= 4'h0 && scan_code <= 4'h9) begin // key 0-9
            if (cal_state == 1'b0) begin // add_1
                buf_flag_1 = {buf_flag_1[4:0], 1'b1};
                key_buf_code_1[23:0] =
                    {key_buf_code_1[19:0], scan_code};
            end
            end
        end
```

由cal_state決定資料是要存入key_buf_code_1或是 key_buf_code_2

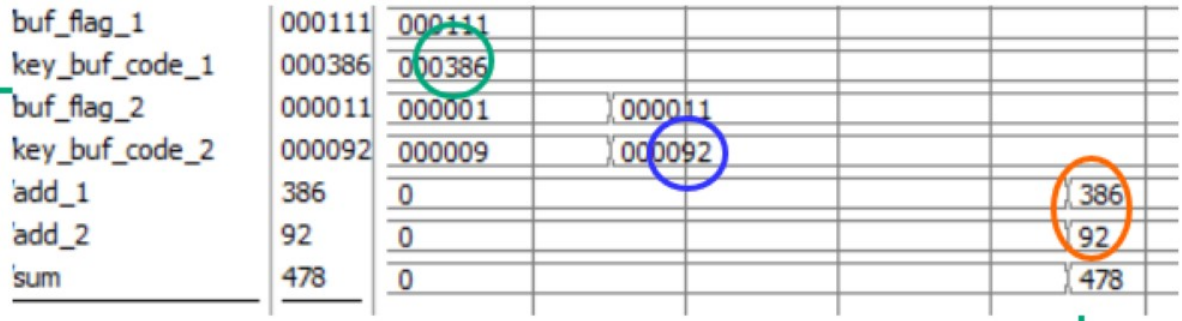
Design 3: key_buffer (5/7)

```
else if (scan_code == 4'hc) begin // key C, "+"
    cal_state = [      ] // to add_2
end
```

若有" +" 鍵按下，將add_1 (cal_state=0) 切換至add_2 (cal_state=1)

Design 3: key_buffer (6/7)

```
else if (scan_code == 4'hd) begin // key D, "=", sum = add_1 + add_2
    產生add_1
    for (i=0; i < 6; i=i+1) begin
        decimal_num = 4'd0;
        if (buf_flag_1[i] == 1'b1) begin
            for (j=0; j < 4; j=j+1)    擷取key_buf_code_1中4-bit之資料
                decimal_num[j] = key_buf_code_1[i*4+j];
            add_1 = add_1 + decimal_num * (10**i);
        end
    end
```




```
    產生add_2

    end

    sum =
end
```

若有”=”鍵按下，先將被加數BCD編碼轉換為add_1之二進位編碼，再將加數BCD編碼轉換為add_2之二進位編碼，最後將add_1 與 add_2相加，存至sum

Design 3: key_buffer (7/7)

若有" Clr" 鍵按下，清除所有資料，類似reset的動作

```
                else if (scan_code == 4'he) begin // key E, clear

                    buf_flag_1 = 6'b000000;
                    key_buf_code_1 = 24'h000000;

                    buf_flag_2 = 6'b000000;
                    key_buf_code_2 = 24'h000000;

                    add_1 = 20'd0;
                    add_2 = 20'd0;
                    sum = 21'd0;

                    cal_state = 1'b0; // to add_1
                end
            end
        end
    end
endmodule
```

Design 4: keyboard_top (1/2)

```
module keyboard_top (clk, rst, kc_sel, kr_sel,
press, scan_code,
buf_flag_1, key_buf_code_1,
buf_flag_2, key_buf_code_2,
add_1, add_2, sum);

    input clk, rst;
    input [3:0] kc_sel;
    output [3:0] kr_sel;
    output press;
    output [3:0] scan_code;

    output [5:0] buf_flag_1;
    output [23:0] key_buf_code_1;
    output [5:0] buf_flag_2;
    output [23:0] key_buf_code_2;

    output  [19:0] add_1, add_2;
    output  [20:0] sum;
```

此為產生波形，模擬電路行為之用，有需要再設計。

Design 4: keyboard_top (2/2)

```
    kr_scan      m1(clk, rst, kr_sel);
    keyboard     m2(kr_sel, kc_sel, press, scan_code);
    key_buffer   m3(clk, rst, press, scan_code,
                        buf_flag_1, key_buf_code_1,
                        buf_flag_2, key_buf_code_2,
                        add_1, add_2, sum);
endmodule
```

Design 5: keyboard_top_tb (1/4)

```
`timescale 1ns/100ps

module keyboard_top_tb;
    parameter CLK_CYCLE = 10; // 10 ns

    reg clk, rst;
    reg [3:0] kc_sel;
    wire [3:0] kr_sel;

    wire press;
    wire [3:0] scan_code;

    wire [5:0] buf_flag_1;
    wire [23:0] key_buf_code_1;
    wire [5:0] buf_flag_2;
    wire [23:0] key_buf_code_2;

    wire [19:0] add_1, add_2;
    wire [20:0] sum;
```

此為產生波形，模擬電路行為之用，有需要再設計。

Design 5: keyboard_top_tb (2/4)

待測電路

```
keyboard_top m1(clk, rst, kc_sel, kr_sel,
                press, scan_code,
                buf_flag_1, key_buf_code_1,
                buf_flag_2, key_buf_code_2,
                add_1, add_2, sum);

// clock generation
always begin
    clk = 0; #(CLK_CYCLE/2);
    clk = 1; #(CLK_CYCLE/2);
end

//reset generation
initial begin
    rst = 1'b1;
    kc_sel = 4'b0000;
    #10 rst = 1'b0;
end
```

Design 5: keyboard_top_tb (3/4)

若改用Quartus II模擬
手動產生以下輸入信號值

```
initial begin
    #45 kc_sel = 4'b0111;  輸入按鍵3
    #10 kc_sel = 4'b0000;
    #10 kc_sel = 4'b1110;  輸入按鍵8
    #10 kc_sel = 4'b0000;
    #20 kc_sel = 4'b1011;  輸入按鍵6
    #10 kc_sel = 4'b0000;
    #50 kc_sel = 4'b1110;  輸入按鍵+
    #10 kc_sel = 4'b0000;
    #20 kc_sel = 4'b1101;  輸入按鍵9
    #10 kc_sel = 4'b0000;
    #10 kc_sel = 4'b1011;  輸入按鍵2
    #10 kc_sel = 4'b0000;
    #20 kc_sel = 4'b1101;  輸入按鍵=
    #10 kc_sel = 4'b0000;
    #30 kc_sel = 4'b1011;  輸入按鍵Clr
    #10 kc_sel = 4'b0000;
    #60 $finish;
end
```

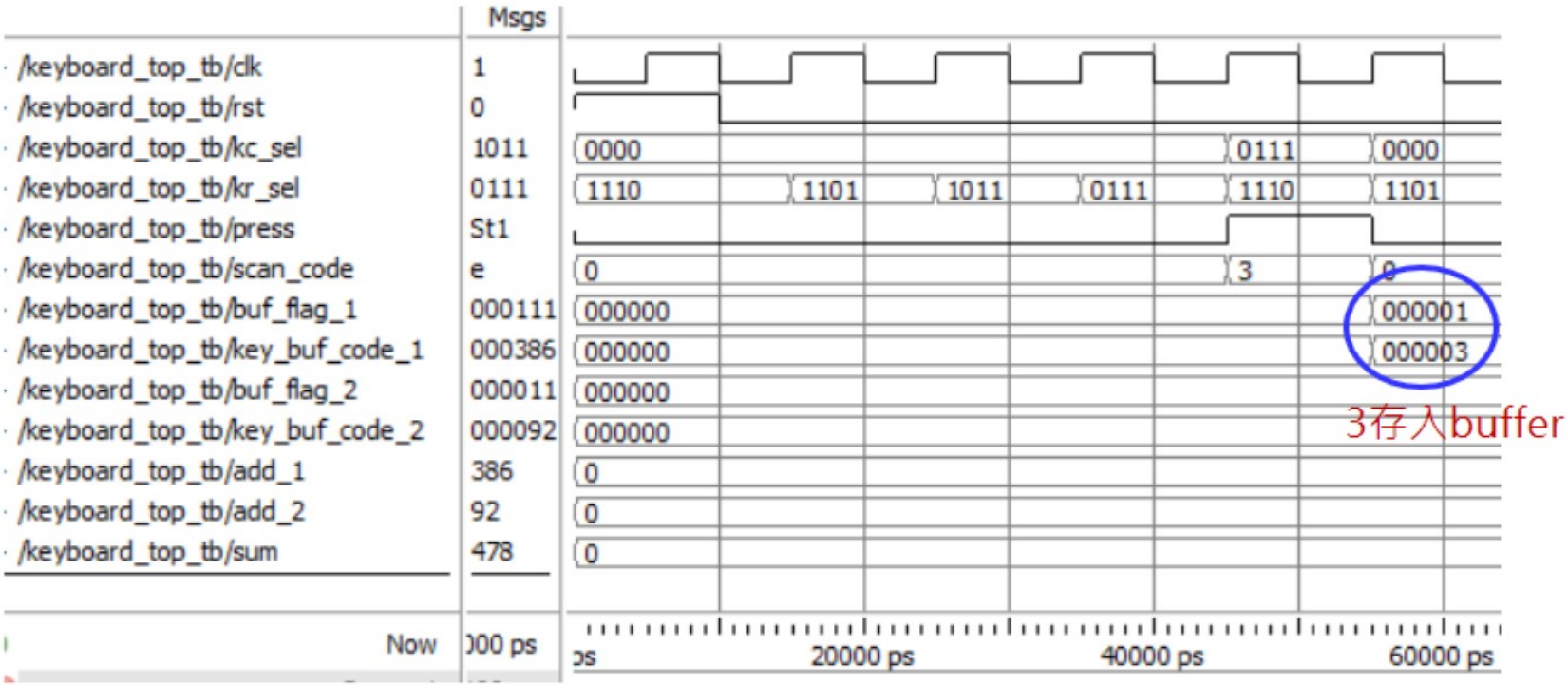
Design 5: keyboard_top_tb (4/4)

文字訊息輸出

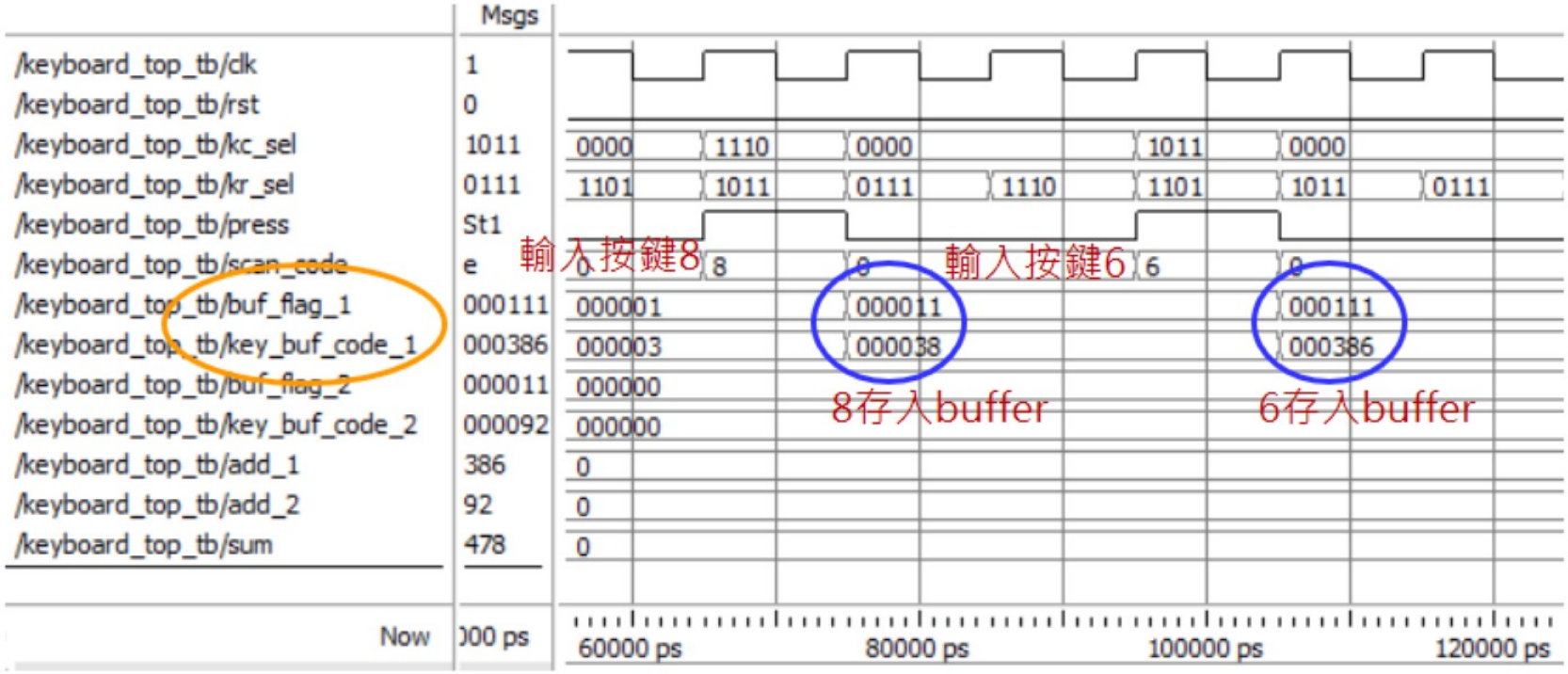
```
    always @(posedge clk) begin
            $display("%d ns: press:%d, scan_code:%h, buf_flag_1:%b,
            key_buf_code_1:%h, buf_flag_2:%b, key_buf_code_2:%h,
            add_1:%6d, add_2:%6d, sum:%7d", $stime, press, scan_code,
            buf_flag_1, key_buf_code_1,
            buf_flag_2, key_buf_code_2,
            add_1, add_2, sum);
    end
endmodule
```

模擬結果之文字訊息輸出

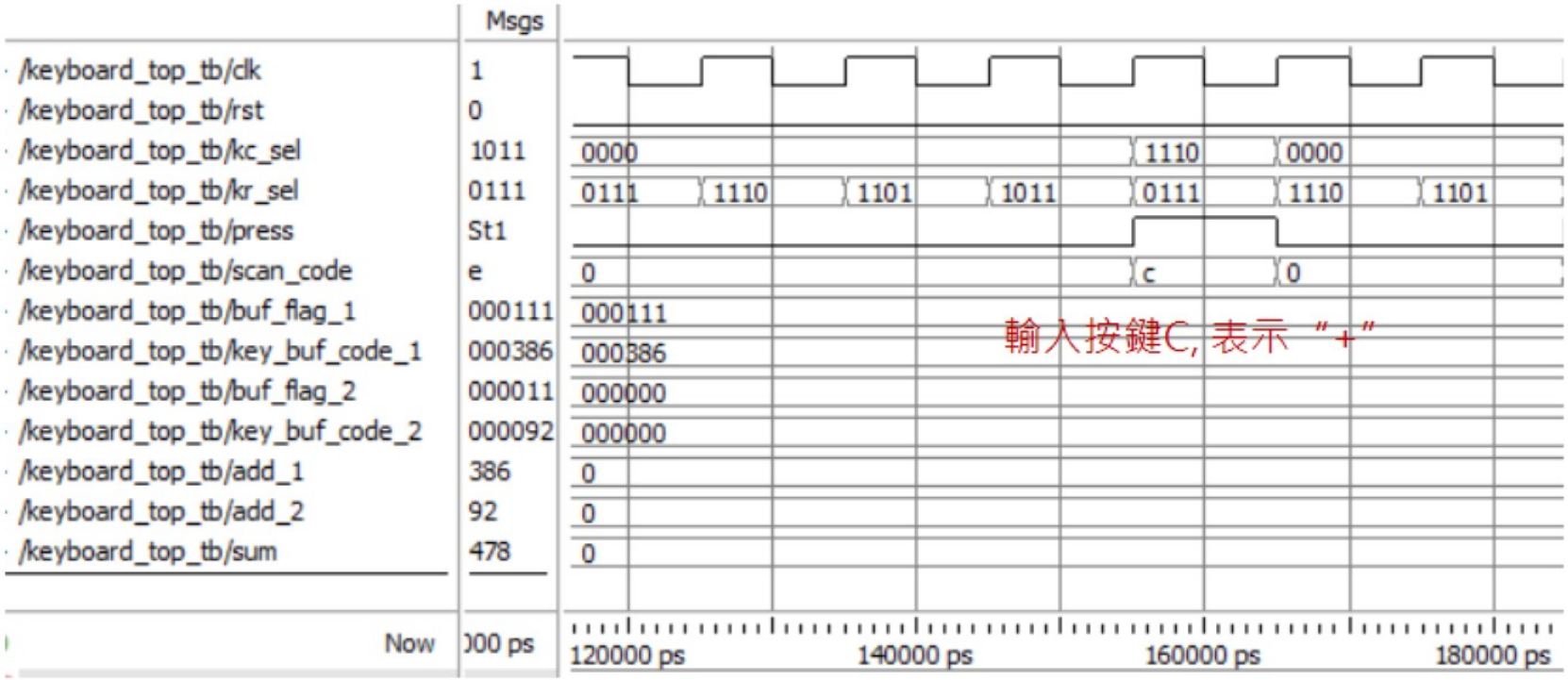
模擬波形圖 (1/5)


Msgs
/keyboard_top_tb/clk
/keyboard_top_tb/rst
/keyboard_top_tb/kc_sel
/keyboard_top_tb/kr_sel
/keyboard_top_tb/press
/keyboard_top_tb/scan_code
/keyboard_top_tb/buf_flag_1
/keyboard_top_tb/key_buf_code_1
/keyboard_top_tb/buf_flag_2
/keyboard_top_tb/key_buf_code_2
/keyboard_top_tb/add_1
/keyboard_top_tb/add_2
/keyboard_top_tb/sum
1
0
1011
0111
St1
e
000111
000386
000011
000092
386
92
478
0000
0111
0000
1110
1101
1011
0111
1110
1101
0
3
000000
000001
000000
000003
000000
000000
0
0
0
Now
00 ps
ps
20000 ps
40000 ps
60000 ps
3存入buffer

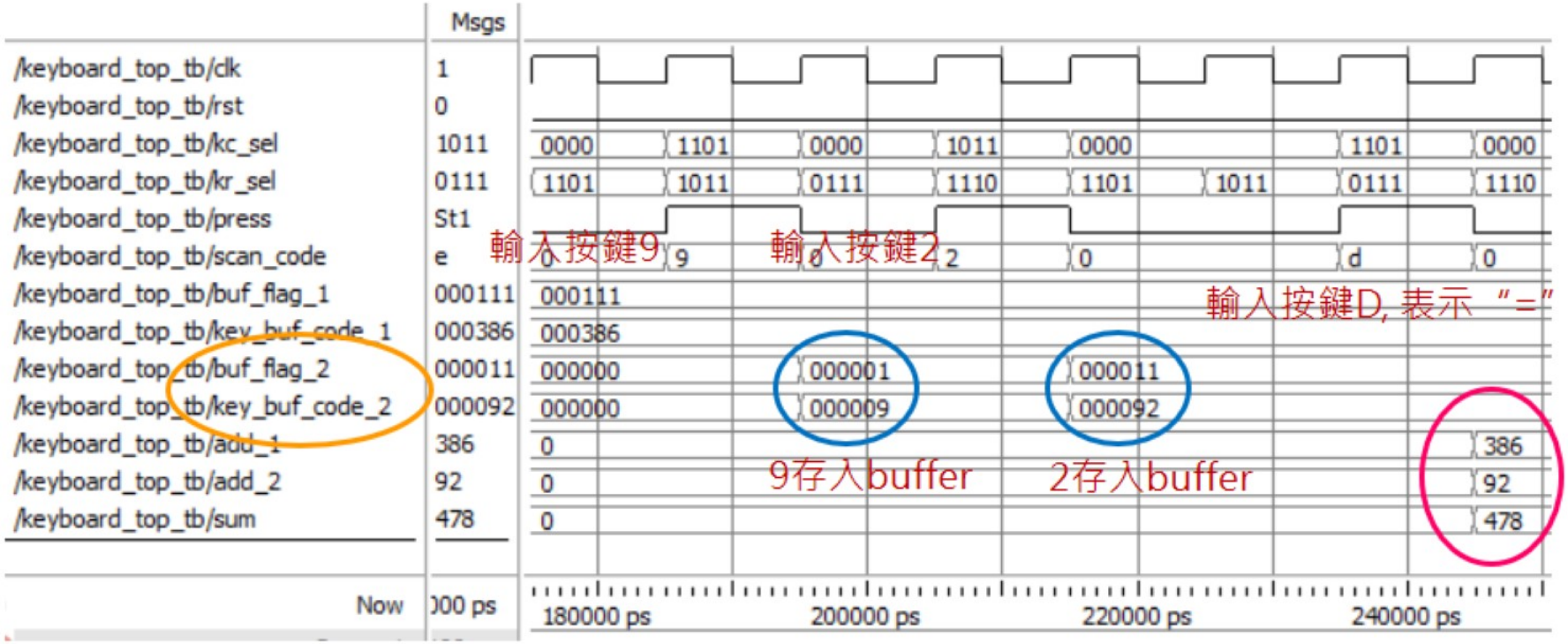
模擬波形圖 (2/5)


Msgs
/keyboard_top_tb/clk
/keyboard_top_tb/rst
/keyboard_top_tb/kc_sel
/keyboard_top_tb/kr_sel
/keyboard_top_tb/press
/keyboard_top_tb/scan_code
/keyboard_top_tb/buf_flag_1
/keyboard_top_tb/key_buf_code_1
/keyboard_top_tb/buf_flag_2
/keyboard_top_tb/key_buf_code_2
/keyboard_top_tb/add_1
/keyboard_top_tb/add_2
/keyboard_top_tb/sum
Now
1
0
1011
0111
St1
e
000111
000386
000011
000092
386
92
478
0000
1110
0000
1011
0000
1101
1011
0111
1110
1101
1011
0111
輸入按鍵8
輸入按鍵6
0
8
8
6
6
000001
000011
000111
000003
000038
000386
000000
000000
8存入buffer
6存入buffer
0
0
0
60000 ps
80000 ps
100000 ps
120000 ps

模擬波形圖 (3/5)

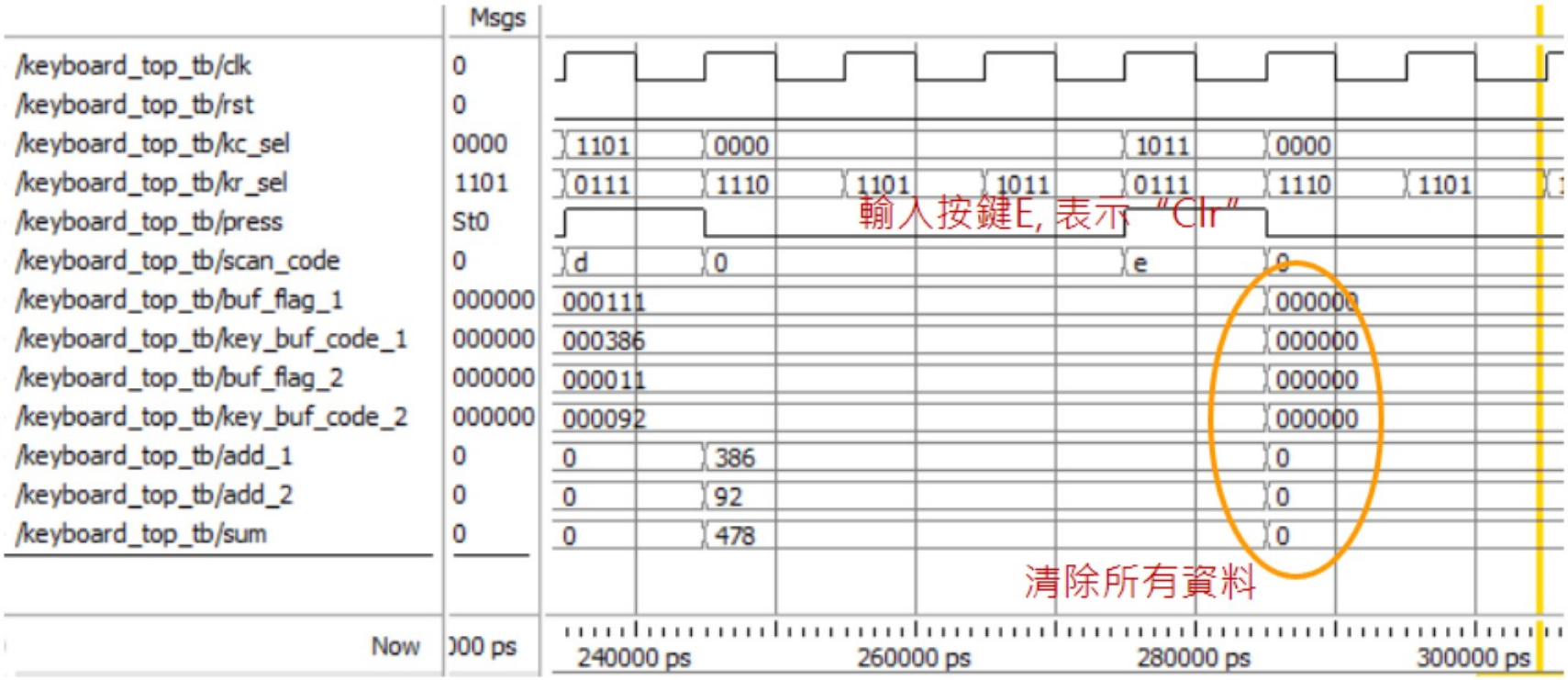

Msgs
/keyboard_top_tb/clk
/keyboard_top_tb/rst
/keyboard_top_tb/kc_sel
/keyboard_top_tb/kr_sel
/keyboard_top_tb/press
/keyboard_top_tb/scan_code
/keyboard_top_tb/buf_flag_1
/keyboard_top_tb/key_buf_code_1
/keyboard_top_tb/buf_flag_2
/keyboard_top_tb/key_buf_code_2
/keyboard_top_tb/add_1
/keyboard_top_tb/add_2
/keyboard_top_tb/sum
1
0
1011
0111
St1
e
000111
000386
000011
000092
386
92
478
Now
)00 ps
0000
1110
0000
0111
1110
1101
1011
0111
1110
1101
0
c
0
000111
000386
000000
000000
0
0
0
輸入按鍵C, 表示 "+"
120000 ps
140000 ps
160000 ps
180000 ps

模擬波形圖 (4/5)



運算結果
386+92=478

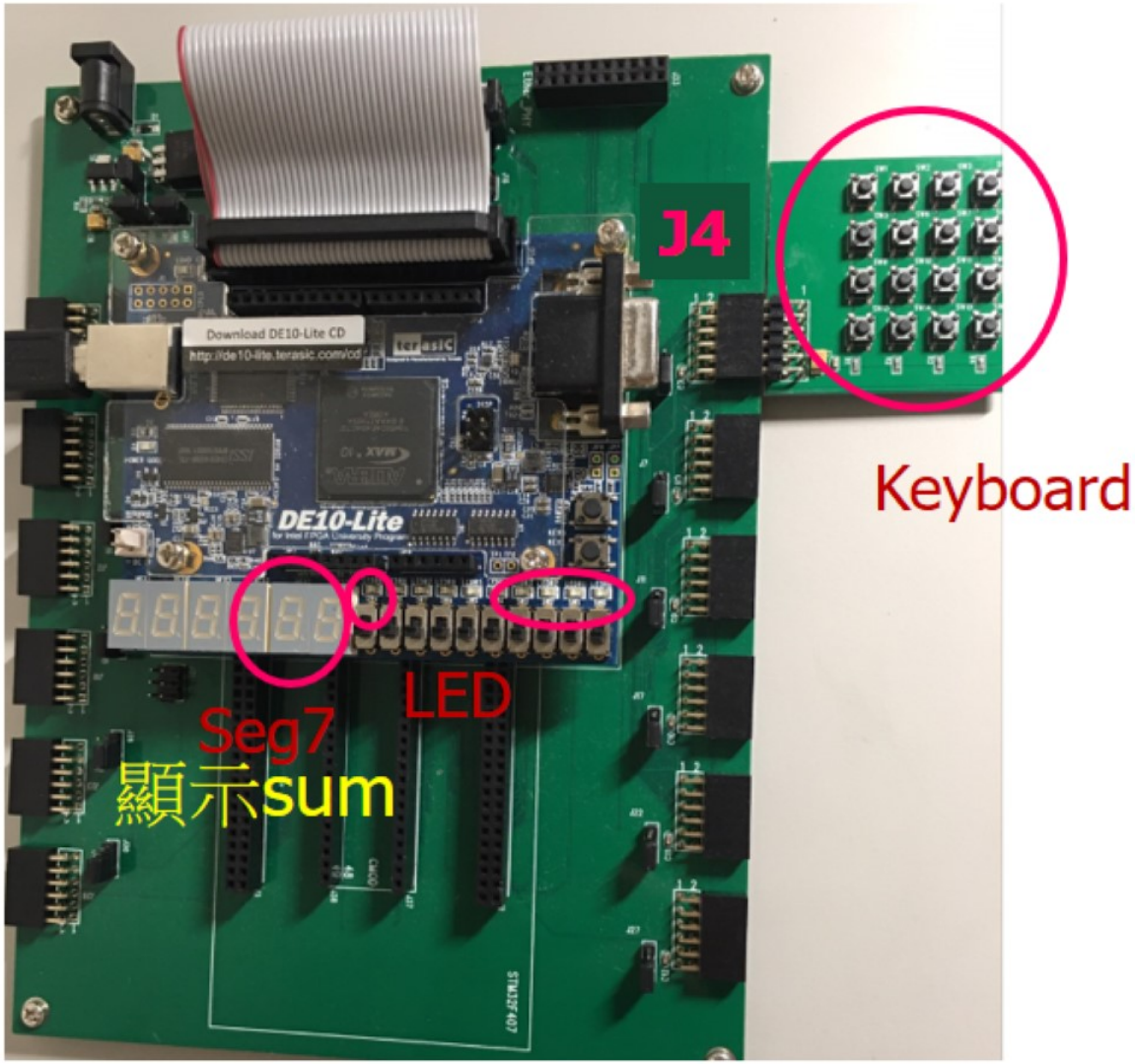
模擬波形圖 (5/5)


Msgs
/keyboard_top_tb/clk
/keyboard_top_tb/rst
/keyboard_top_tb/kc_sel
/keyboard_top_tb/kr_sel
/keyboard_top_tb/press
/keyboard_top_tb/scan_code
/keyboard_top_tb/buf_flag_1
/keyboard_top_tb/key_buf_code_1
/keyboard_top_tb/buf_flag_2
/keyboard_top_tb/key_buf_code_2
/keyboard_top_tb/add_1
/keyboard_top_tb/add_2
/keyboard_top_tb/sum
Now
0
0
0000
1101
St0
0
000000
000000
000000
000000
0
0
0
)00 ps
1101
0000
1011
0000
0111
1110
1101
1011
0111
1110
1101
d
0
e
0
000111
000386
000011
000092
0
386
0
92
0
478
000000
000000
000000
000000
0
0
0
輸入按鍵E, 表示 "Clr"
清除所有資料
240000 ps
260000 ps
280000 ps
300000 ps

File List for Simulation

- Verilog files
  - kr_scan.v (產生列掃描的信號)
  - keyboard.v (產生按鍵編碼)
  - key_buffer.v (將按鍵資料送入buffer，並執行加法運算)
  - keyboard_top.v (最上層top電路)

keyboard2_top 電路圖



File List for Quartus

- Verilog files
  - freq_div.v (二的次方之除頻電路)
  - num_to_seg7_0_F.v (16進位數字轉Seg7編碼)
  - kr_scan.v (產生列掃描的信號)
  - keyboard.v (產生按鍵編碼)
  - bin2bcd.v (將8-bit二進位編碼轉為12-bit BCD編碼)
  - key_buffer.v (將按鍵資料送入buffer，並執行加法運算)
  - keyboard2_top.v (最上層top電路)

Design 6: 修改 key_buffer.v

修改 keyboard_top 所使用之 key_buffer, 減少輸出信號

```
module key_buffer(clk, rst, press, scan_code,
buf_flag_1, key_buf_code_1, buf_flag_2, key_buf_code_2, add_1, add_2, sum);

    input   clk, rst, press;
    input   [3:0] scan_code;

    output  reg [5:0] buf_flag_1; // 6 numbers
    output  reg [23:0] key_buf_code_1;

    output  reg [5:0] buf_flag_2; // 6 numbers
    output  reg [23:0] key_buf_code_2;

    output  reg [19:0] add_1, add_2;
    output  reg [20:0] sum;
```

此為產生波形，模擬電路行為之用。

此為燒錄至實驗板，驗證電路行為之用。

```
module key_buffer(clk, rst, press, scan_code, sum);

    input   clk, rst, press;
    input   [3:0] scan_code;

    reg [5:0] buf_flag_1; // 6 numbers
    reg [23:0] key_buf_code_1;

    reg [5:0] buf_flag_2; // 6 numbers
    reg [23:0] key_buf_code_2;

    reg [19:0] add_1, add_2;
    output  reg [20:0] sum;
```

Design 7: keyboard2_top (1/3)

```
module keyboard2_top (clk, rst, J4_col, J4_row, press, scan_code,
seg7_0, seg7_0_dpt, seg7_1, seg7_1_dpt, seg7_2, seg7_2_dpt);

    input clk, rst;
    input [3:0] J4_col;
    output [3:0] J4_row;
    output press;
    output [3:0] scan_code;
    output [6:0] seg7_0, seg7_1, seg7_2;
    output seg7_0_dpt, seg7_1_dpt, seg7_2_dpt;

    wire [3:0] kr_sel, kc_sel;
```

Design 7: keyboard2_top (2/3)

```
assign J4_row[0] = kr_sel[0];
assign J4_row[1] = kr_sel[1];
assign J4_row[2] = kr_sel[2];
assign J4_row[3] = kr_sel[3];
assign kc_sel[0] = J4_col[0];
assign kc_sel[1] = J4_col[1];
assign kc_sel[2] = J4_col[2];
assign kc_sel[3] = J4_col[3];

wire clk_key;
wire [20:0] sum;
wire [11:0] key_buf_code;
```

運算結果, 輸出8-bit (0-255)

顯示12-bit(3-digit)於seg7

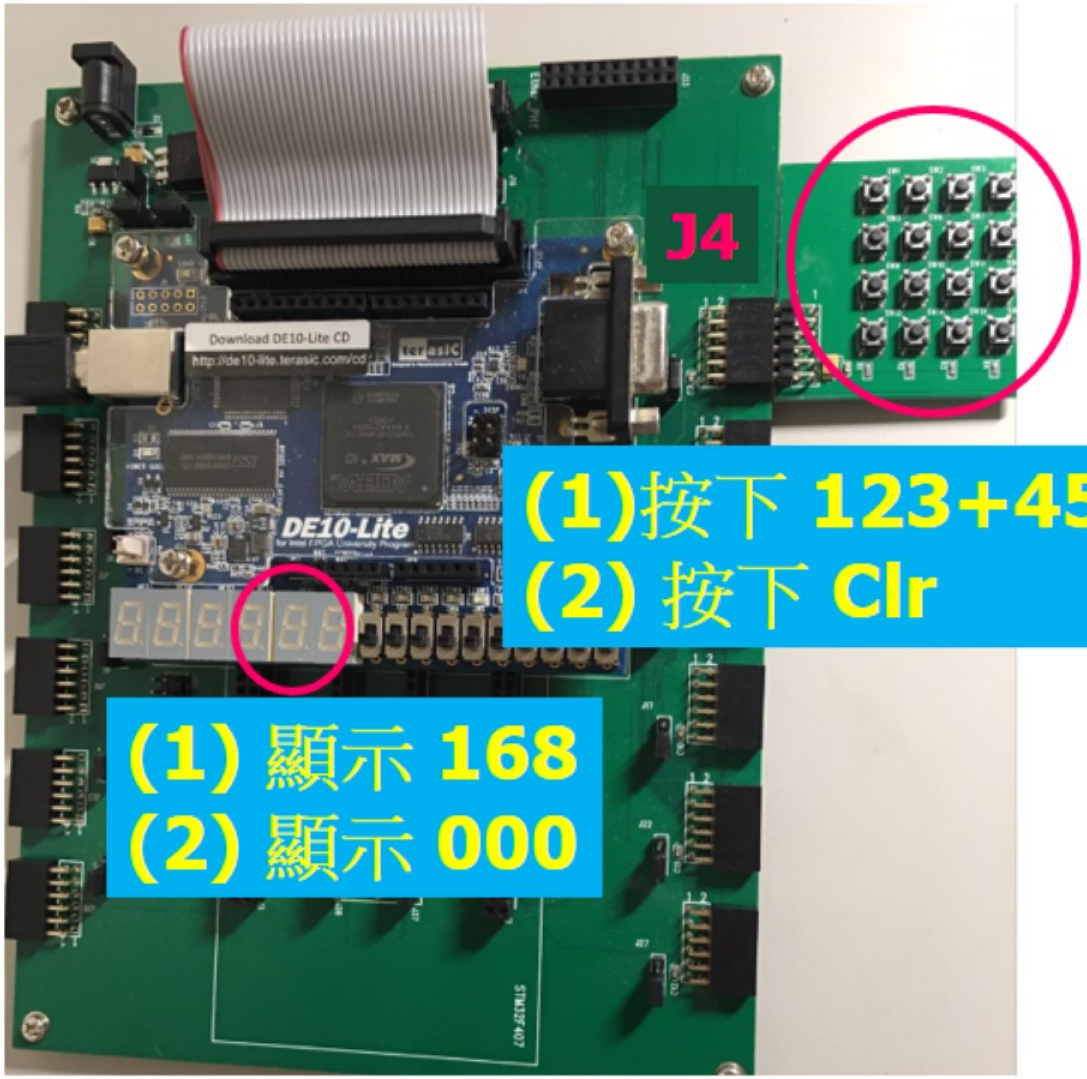
Design 7: keyboard2_top (3/3)

```
freq_div      #(21)  m0(clk, ~rst, clk_key);
kr_scan              m1(clk_key, ~rst, kr_sel);
keyboard           m2(kr_sel, kc_sel, press, scan_code);

// ***                    注意 ~rst
key_buffer  m3(                                          );
bin2bcd            m7(sum[7:0], key_buf_code[11:0]);
// ***
num_to_seg7_0_F       m4(1,
num_to_seg7_0_F       m5(1,
num_to_seg7_0_F       m6(1,
endmodule
```

8-bit 運算結果 sum[7:0], 轉為12-bit BCD編碼 key_buf_code[11:0]

keyboard2_top 電路圖



(1)按下 123+45=
(2) 按下 Clr

(1) 顯示 168
(2) 顯示 000

實驗結果驗收

- 請老師或助教驗收 keyboard2_top 電路於實驗板顯示之結果
- **計算 123+45=168**