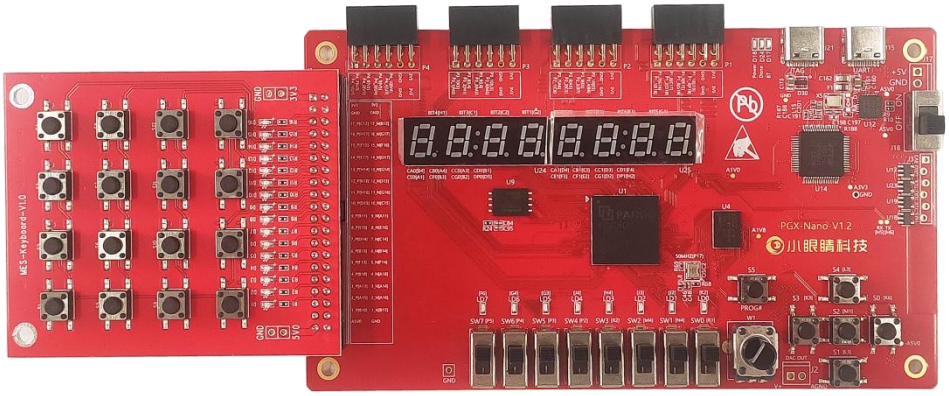
33. 矩阵按键位置显示实验说明

33.1. 实验简介

实验室用“小眼睛科技”开发板、矩阵按键模块完成矩阵按键位置使用数码管显示实验。



33.2. 实验要求

矩阵按键共四行 H0~H3 四列 L0~L3,按键按下第 0 行第 1 列时，数码管显示 01、按键按下第 3 行第 2 列时，数码管显示 32。

33.3. 实验使用模块介绍

实验使用“小眼睛科技”矩阵按键扩展模块，其中矩阵按键为四行四列矩阵按键。

33.4. 实验原理

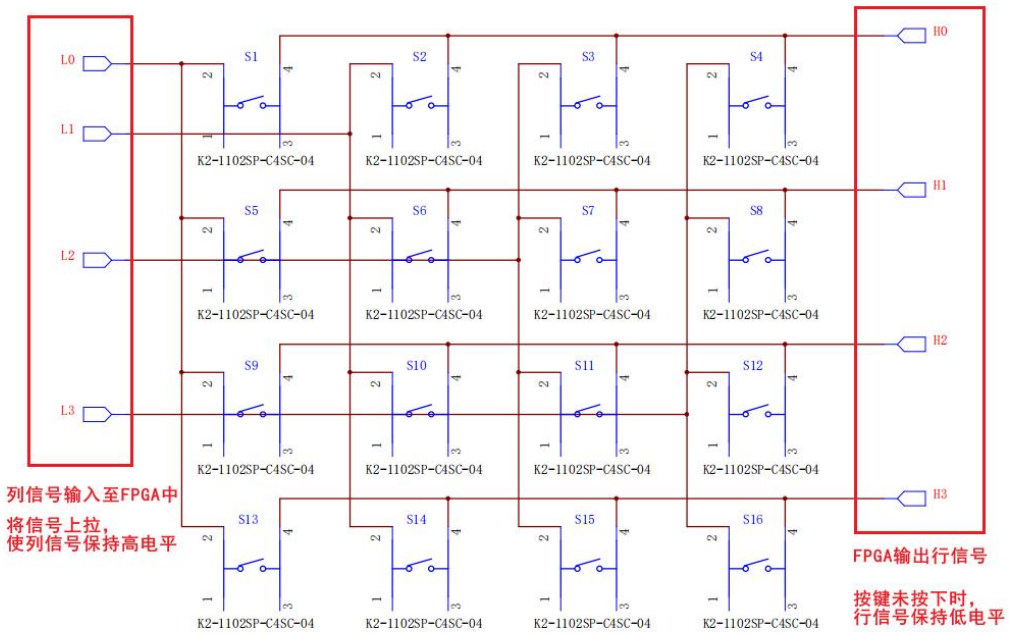
矩阵按键是一种排布类似矩阵的按键组，再使用时比简单的按键要复杂一些，一般矩阵按键的行线或者列线需要上拉接电源，使 IO 默认保持高电平，通过扫描列线或者行线的方式，取判断按键是否按下，以及按键按键的位置。

矩阵扫描过程：

1、将列信号上拉，使列信号 IO 默认高电平状态，将列信号输入至 FPGA 中。具体设置为在使用 UCE 绑定管脚时，在输入矩阵按键列信号位置将 BUS_KEEPER 选项设置为 PULLUP。

	I/O NAME	I/O DIRECTION	LOC	BANK	VCCIO	IOSTANDARD	DRIVE	BUS_KEEPER
17	key_c[3]	INPUT	C15	BANKL4	3.3	LVCMOS33		PULLUP
18	key_c[2]	INPUT	G16	BANKL4	3.3	LVCMOS33		PULLUP
19	key_c[1]	INPUT	F16	BANKL4	3.3	LVCMOS33		PULLUP
20	key_c[0]	INPUT	G14	BANKL4	3.3	LVCMOS33		PULLUP

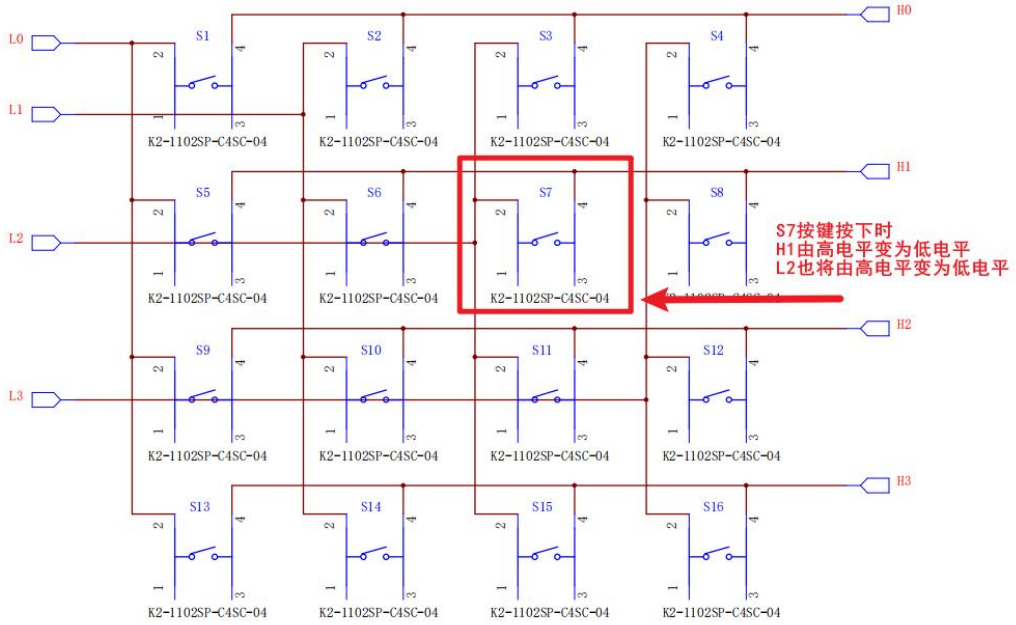
2、FPGA 输出行信号，使行信号默认保持低电平。



3、在按键未按下时，矩阵按键列信号为高电平，当按键按下时，此处按键的列信号与行信号相连，此时矩阵按键列信号将被拉低。

4、检测到矩阵按键的列信号被拉低，通过拉低列信号的位可以确定按键的列数。

5、使用行信号逐行扫描，具体操作为将行信号依次拉高，在行信号拉高期间，FPGA 接收到的列信号被拉高，则可以确定按键的行数。



6、确定按键的位置后，将行信号保持低电平，按键按下期间，列信号将持续拉低。将拉低的列信号进行消抖后可作为按键信号使用。

33.5. 实验源码设计

顶层设计：

1、key_det 模块用于确定按键信号的行列数，并将消抖后的按键信号输出。

2、seq_dis 模块用于显示按键信号的行列数。

```
`timescale 1ns / 1ps
`define UD #1

module top_matrix_key(
    input       clk                 ,
    input       rst_n               ,
    input       key                 ,
    input       wire   [3:0]key_c   ,
    output      wire   [3:0]key_r   ,
    output      wire   [3:0]dig     ,
    output      wire   [7:0]smg     ,
    output      led
   );

wire [3:0]key_location ;

key_det #(
    .btn_delay  (20'hf423f)
)key_det(
    .clk                        (clk ) ,

    .rst_n                      (rst_n ) ,

    .key_c                      (key_c ) ,

    .key_r                      (key_r ) ,

    .key_location               (key_location ) ,

    .key_flag                   (btn_flag  )

);

seq_dis seq_dis(
    .clk                        (clk        ),
```

```
        .key_location                   (key_location),

        .dig                            (dig         ),

        .smg                            (smg         )

    );

    assign led = ~btn_flag ;

    endmodule
```

矩阵按键检测模块：将行信号输出为低电平，当检测到列信号不全为高电平时，说明由按键按下，并可以判断列信号的位置，然后一次拉高行信号，通过判断行信号拉高器件，对应被拉低的列信号是否被拉高，可以判断行信号的位置。

```
    `timescale 1ns / 1ps
    `define UD #1

    module key_det #(
        parameter   btn_delay = 20'hf423f
    )(
        input       wire        clk ,
        input       wire        rst_n ,
        input       wire   [3:0]key_c /*synthesis
PAP_MARK_DEBUG="1"*/,  //The signal is high by default
        output      reg    [3:0]key_r /*synthesis PAP_MARK_DEBUG="1"*/,
        output      wire   [3:0]key_location /*synthesis
PAP_MARK_DEBUG="1"*/,
        output      wire        key_flag /*synthesis PAP_MARK_DEBUG="1"*/
    );

    reg [3:0]key_creg1 ;
    reg [3:0]key_creg2 ;
    reg [1:0]cnt_check ;
    reg [1:0]col_value /*synthesis PAP_MARK_DEBUG="1"*/;
    reg [1:0]row_value /*synthesis PAP_MARK_DEBUG="1"*/;
    reg [19:0]cnt_delay ;
    reg flag ;

    /*--------The input signal eliminates metastability---------*/

    always @(posedge clk ) begin
        if (~rst_n) begin
            key_creg1 <= 4'hf ;
```

```
            key_creg2 <= 4'hf ;
        end
        else begin
            key_creg1 <= key_c ;
            key_creg2 <= key_creg1 ;
        end
    end

    /*----------------------state machine----------------------*/
    reg [7:0]state /*synthesis PAP_MARK_DEBUG="1"*/;

    parameter idle      = 8'b0000_0001 ;
    parameter col_det   = 8'b0000_0010 ;
    parameter row_wait  = 8'b0000_0100 ;
    parameter row_det   = 8'b0000_1000 ;
    parameter key_eli_1 = 8'b0001_0000 ;
    parameter key_end   = 8'b0010_0000 ;
    parameter key_eli_2 = 8'b0100_0000 ;
    parameter state_end = 8'b1000_0000 ;

    always @(posedge clk ) begin
        if (~rst_n)
            state <=  idle;
        else begin
            case (state)
                idle   :begin
                    if (key_creg2 != 4'hf)
                        state <= col_det ;
                    else
                        state <= idle ;
                end
                col_det:begin
                    if ((cnt_check == 2'd3)&&(flag == 1'b1 ))
                        state <= row_wait ;
                    else
                        state <= col_det ;
                end
                row_wait:begin
                        state <= row_det ;
                end
                row_det :begin
                    if ((cnt_check == 2'd3)&&(flag == 1'b1 ))
                        state <= key_eli_1;
                    else
```

```
                        state <= row_det ;
                end
                key_eli_1 :begin
                    if (cnt_delay == btn_delay)
                        state <= key_end ;
                    else
                        state <= key_eli_1 ;
                end
                key_end :begin
                    if ((key_creg2[col_value] ==
1'b0)&&(key_creg1[col_value] == 1'b1))
                        state <= key_eli_2 ;
                    else
                        state <= key_end ;
                end
                key_eli_2 :begin
                    if (cnt_delay == btn_delay)
                        state <= state_end ;
                    else
                        state <= key_eli_2 ;
                end
                state_end :begin
                        state <= idle ;
                end
                default   :state <= idle ;
            endcase
        end
    end

    /*----------Set counter to detect column by column----------*/

    always @(posedge clk ) begin
        if (~rst_n)
            flag <= 1'b0 ;
        else if ((state == col_det)||(state == row_det))
            flag <= ~flag ;
        else
            flag <= 1'b0 ;
    end

    always @(posedge clk ) begin
        if (~rst_n)
            cnt_check <= 2'd0 ;
        else if (((state == col_det)||(state == row_det))&&(flag == 1'b1))
```

```
            cnt_check <= cnt_check + 2'd1 ;
        else if ((state == col_det)||(state == row_det))
            cnt_check <= cnt_check ;
        else
            cnt_check <= 2'd0 ;
    end

    /*---------------Detects key column position--------------*/

    always @(posedge clk ) begin
        if (~rst_n)
            col_value <= 2'd0 ;
        else if ((state == col_det)&&(key_c[cnt_check]== 1'b0)&&(flag == 1'b1 ))
            col_value <= cnt_check ;
        else
            col_value <= col_value ;
    end

    /*---------------Detects key row position--------------------*/

    always @(posedge clk ) begin
        if (~rst_n)
            row_value <= 2'd0 ;
        else if ((state == row_det)&&(key_c == 4'hf)&&(flag == 1'b1 ))
            row_value <= cnt_check ;
        else
            row_value <= row_value ;
    end

    /*---------------ROW signal assignment----------------------*/

    always @(*) begin
            key_r <= 4'b0000 ;
        if (state == row_det) begin
            case (cnt_check)
                2'd0    : key_r <= 4'b0001 ;
                2'd1    : key_r <= 4'b0010 ;
                2'd2    : key_r <= 4'b0100 ;
                2'd3    : key_r <= 4'b1000 ;
                default : key_r <= 4'b0000 ;
            endcase
        end
        else
```

```
            key_r <= 4'b0000 ;
    end

    /*---------------key-vibration eliminate-----------------*/

    always @(posedge clk ) begin
        if (~rst_n)
            cnt_delay <= 20'd0 ;
        else if ((state == key_eli_1)||(state == key_eli_2))
            cnt_delay <= cnt_delay + 20'd1 ;
        else
            cnt_delay <= 20'd0 ;
    end

    /*------------------------------*/

    assign key_flag = (state == key_end)? key_creg2[col_value] : 1'b1 ;

    assign key_location = {row_value , col_value} ;

    endmodule
```

数码管显示模块不再赘述，完整代码请参考源码文件。

33.6. 实验现象

按下矩阵按键按下第 0 行第 1 列时，数码管显示 01、按键按下第 3 行第 2 列时，数码管显示 32。