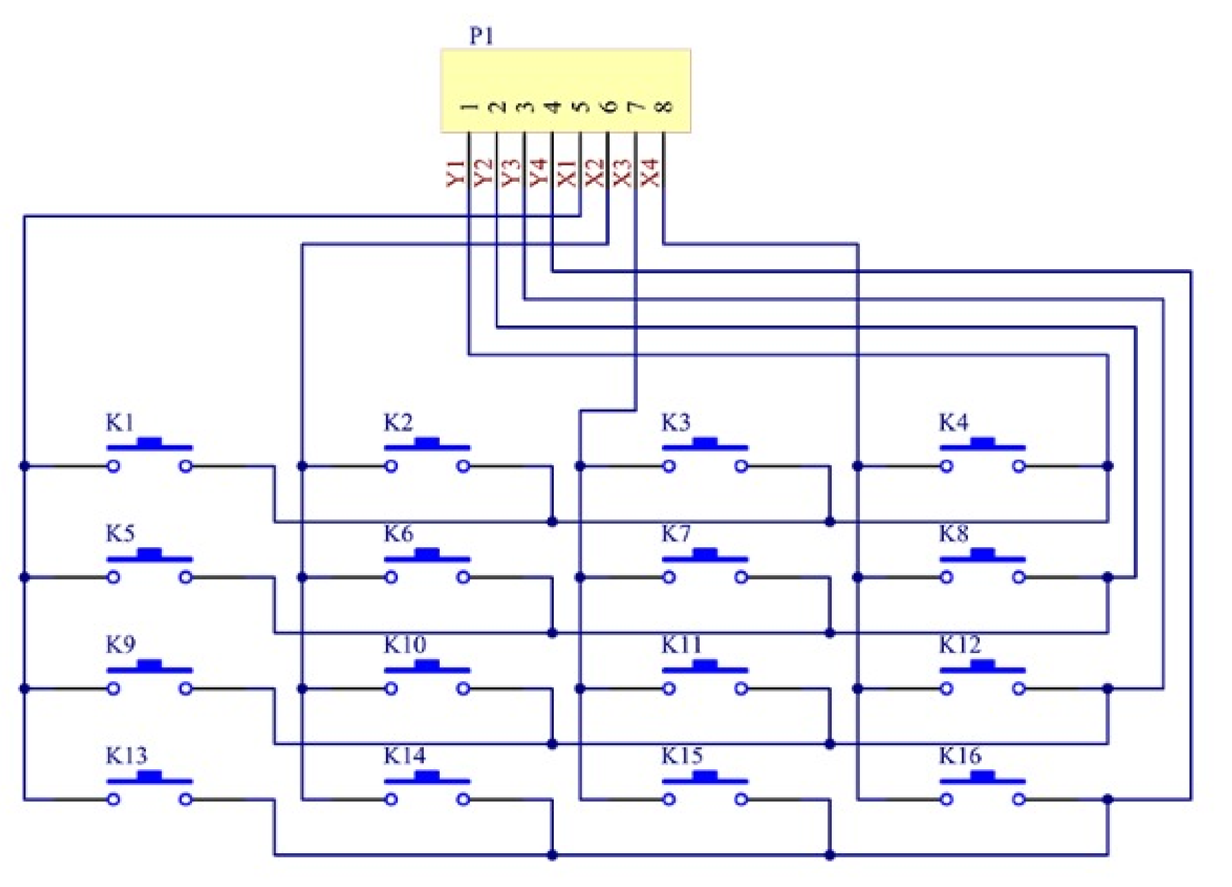
矩阵键盘

只有选择做附加题2的同学才需用到矩阵键盘。

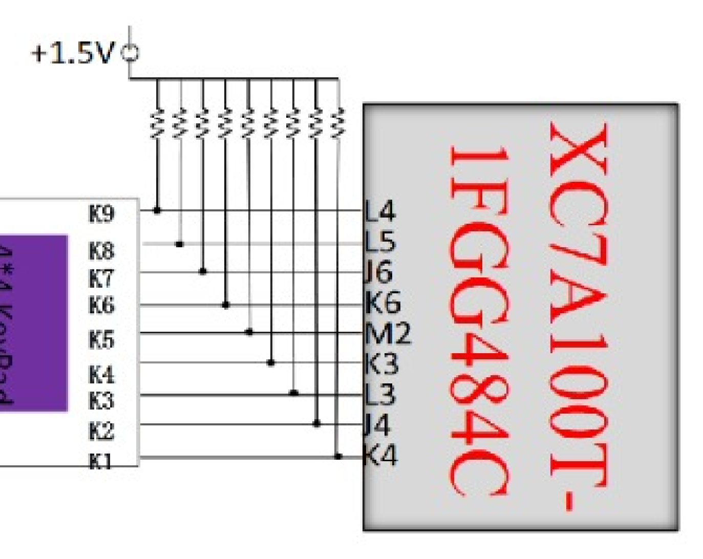
1、矩阵键盘介绍

Minisys FPGA开发板所用矩阵键盘是标准4*4矩阵键盘，与电话按键布局一致，右边加了一列A/B/C/D方便十六进制处理，外观如下。

矩阵键盘通过行列交叉编码连接，通过分时扫描的方法识别按键，节约I/O资源，但编程相对复杂。矩阵键盘通常对外8个引脚，4个行信号，4个列信号。FPGA开发板矩阵键盘内部连接原理图：

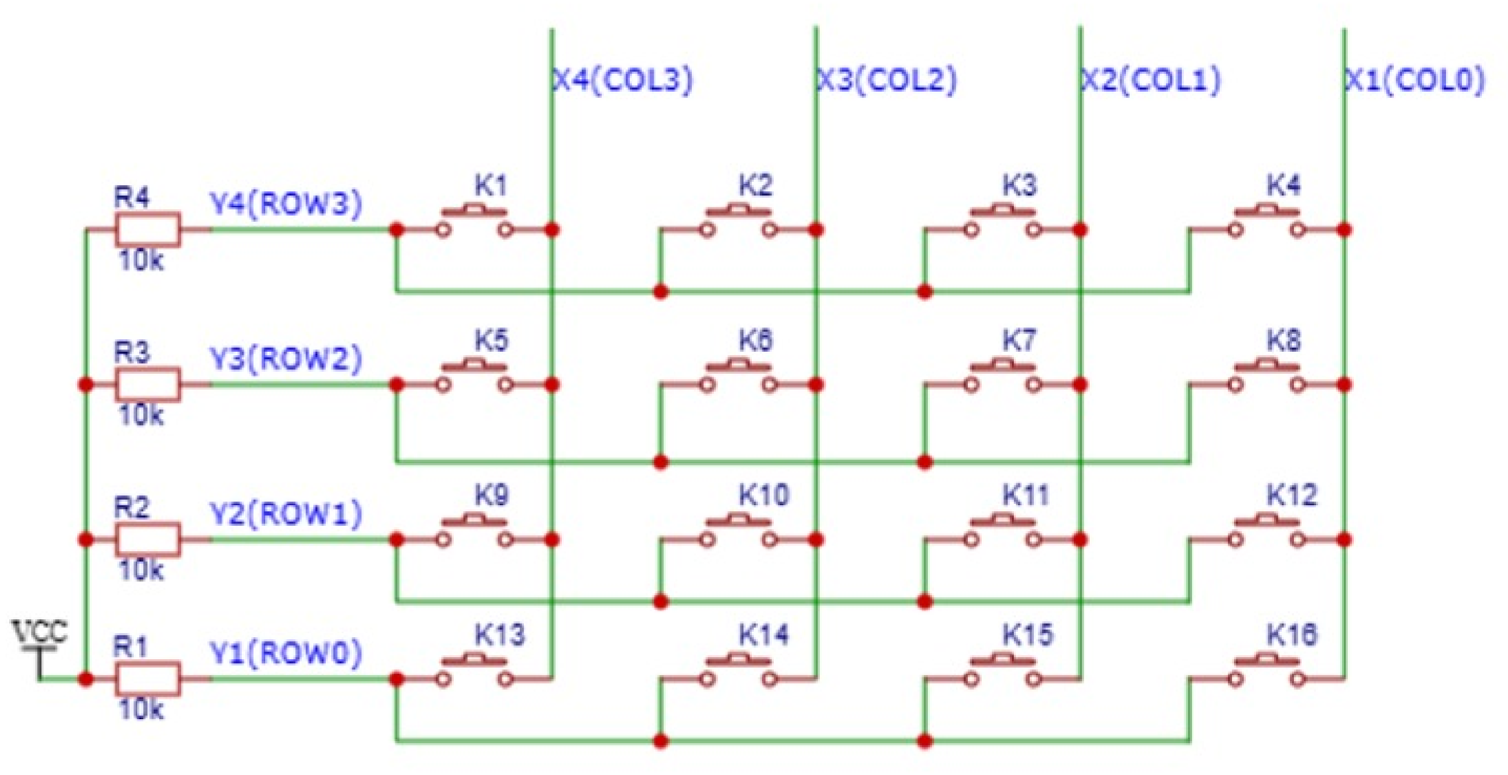


FPGA开发板矩阵键盘行列信号与FPGA芯片引脚的连接图，其中行信号Y4-Y1分别接FPGA引脚K4/J4/L3/K3，列信号X4-X1分别接L5/J6/K6/M2。



2、矩阵键盘按键检测：列扫描法

矩阵键盘的识别相对比较复杂，需要逐行或者逐列扫描。下面讲解列扫描法，列扫描时列信号由FPGA输出，行信号被FPGA读入，逐列扫描是否有按键按下。为方便理解列扫描，将上图接了上拉电阻的矩阵连线展开，得到如下的原理图。



其中X1(COL0)指的是COL0列，对应到上图中的X1信号，Y1(ROW0)指的是ROW0行，对应上图中的Y1信号。COL[3:0]从FPGA输出，ROW[3:0]被FPGA读取。实现时只需考虑同一时刻只有一个按键按下，每行之间的按键互不影响，故只分析一行的工作原理，其他行同理。先分析ROW0行即上拉电阻旁的行信号Y1，有以下两种状态：

- 无按键按下，该位置悬空，通过上拉电阻接到电源与电源同电位，电平为高电平。
- 该行有一个按键按下，假设K13、K14、K15未按下，K16按下，ROW0与COL0被相连，两者电平相同，COL0由FPGA控制，FPGA输出高电平则ROW0读到高电平，若输出低电平则读到低电平。

正是利用以上特性，FPGA控制电路依次拉低列信号，读取行信号的值，译码判断按下的按键，故称为列扫描。列扫描具体过程：

- 首先，扫描COL0列的K4、K8、K12、K16四个按键，FPGA拉低COL0，其他列为高电平，即列信号COL[3:0]=4'b1110，读取行信号进行译码。若读取到4b'1111则没有按键按下，若K16按下，读取到ROW[3:0]应是4b'1110，若K12按下，ROW[3:0]=4b'1101，依次类推。
- 接着扫描COL1列的K3、K7、K11、K15四个按键，FPGA拉低COL1，列信号COL[3:0]=4'b1101，读取行信号进行译码。若读取到4b'1111则没有按键按下，若K15按下，读取到ROW[3:0]应是4b'1110，若K11按下，ROW[3:0]=4b'1101，依次类推。
- COL2和COL3的扫描方式一致，不再赘述。

反过来行信号由FPGA输出，列信号由FPGA读入则是行扫描。开发板对列信号也接了上拉电阻，同时支持行列扫描。列信号的变化快慢即扫描频率，通过计数器得到扫描信号的控制信号，通常为100HZ。扫描得到的行列信号是按键坐标，进行译码才能得到按键的键值，结合按键布局得到以下的编码表。

```
case ({col, row})  // 编码表
   8'b1110_1110 : key_num <= 4'hd;   // 按键D
   8'b1110_1101 : key_num <= 4'hc;   // 按键C
   8'b1110_1011 : key_num <= 4'hb;   // 按键B
   8'b1110_0111 : key_num <= 4'ha;   // 按键A

   8'b1101_1110 : key_num <= 4'hf;   // 按键#
   8'b1101_1101 : key_num <= 4'h9;   // 按键9
   8'b1101_1011 : key_num <= 4'h6;   // 按键6
   8'b1101_0111 : key_num <= 4'h3;   // 按键3

   8'b1011_1110 : key_num <= 4'h0;   // 按键0
   8'b1011_1101 : key_num <= 4'h8;   // 按键8
   8'b1011_1011 : key_num <= 4'h5;   // 按键5
   8'b1011_0111 : key_num <= 4'h2;   // 按键2

   8'b0111_1110 : key_num <= 4'he;   // 按键*
   8'b0111_1101 : key_num <= 4'h7;   // 按键7
   8'b0111_1011 : key_num <= 4'h4;   // 按键4
   8'b0111_0111 : key_num <= 4'h1;   // 按键1
endcase
```

完整的矩阵按键扫描的verilog参考实现如下：

```
module keyboard #(
    parameter CNT_THRESHOLD=1000000-1        // 计数器值，控制扫描频率
)(
    input  wire       clk,
    input  wire       rst,
    input  wire [3:0] row,          //行信号，作为输入被FPGA读取
    output reg  [3:0] col,          //列信号，作为输出被FPGA驱动
    output reg        key_valid,    //是否有按键按下，扫描到按键按下时输出1
    output reg  [3:0] key_num       //键值对应的十六进制，只维持一个周期
);

wire cnt_end;

counter #(CNT_THRESHOLD, 24) u_counter(
    .clk(clk),
    .rst(rst),
    .cnt_inc(1),
    .cnt_end(cnt_end)
);

reg  [3:0] row_r_0;
reg  [3:0] row_r_1;
wire [3:0] row_debounce;
always @(posedge clk, posedge rst) begin
    if (rst) begin
        row_r_0 <= 0;
        row_r_1 <= 0;
    end
    else begin
        row_r_0 <= row;
        row_r_1 <= row_r_0;
    end
end

// 列信号生成
always @(posedge clk, posedge rst) begin
    if (rst)                  col <= 4'b1111;
    else if (col ==  4'b1111) col <= 4'b1110;
    else if (cnt_end)         col <= {col[2:0], col[3]};
end

reg [1:0] row_f_cnt;       // 一轮扫描中row为f的次数，若为3说明无按键按下
always @(posedge clk, posedge rst) begin
    if      (rst)     row_f_cnt <= 0;
    else if (cnt_end && col == 4'b1110) row_f_cnt <= 0;
    else if (cnt_end && row == 4'b1111) row_f_cnt <= row_f_cnt + 1;
end

reg [3:0] row_last;
always @(posedge clk, posedge rst) begin
    if (rst) row_last <= 0;
    else if(cnt_end ) begin
        if(row != 4'b1111)       row_last <= row;   // 长按只识别一次，一轮扫描中长按中
        else if (row_f_cnt == 3) row_last <= row;   // 连续短按的处理，若没按键按下，更
    end
end


always @(posedge clk or posedge rst) begin
    if (rst) begin
        key_num   <= 4'd0;
        key_valid <= 0;
    end
    else if (cnt_end) begin
        if (row != 4'b1111 && row!= row_last) begin
            key_valid <= 1;
            case ({col, row})  // 编码表
                8'b1110_1110 : key_num <= 4'hd;   // 按键D
                8'b1110_1101 : key_num <= 4'hc;   // 按键C
                8'b1110_1011 : key_num <= 4'hb;   // 按键B
                8'b1110_0111 : key_num <= 4'ha;   // 按键A

                8'b1101_1110 : key_num <= 4'hf;   // 按键#
                8'b1101_1101 : key_num <= 4'h9;   // 按键9
                8'b1101_1011 : key_num <= 4'h6;   // 按键6
                8'b1101_0111 : key_num <= 4'h3;   // 按键3

                8'b1011_1110 : key_num <= 4'h0;   // 按键0
                8'b1011_1101 : key_num <= 4'h8;   // 按键8
                8'b1011_1011 : key_num <= 4'h5;   // 按键5
                8'b1011_0111 : key_num <= 4'h2;   // 按键2

                8'b0111_1110 : key_num <= 4'he;   // 按键*
                8'b0111_1101 : key_num <= 4'h7;   // 按键7
                8'b0111_1011 : key_num <= 4'h4;   // 按键4
                8'b0111_0111 : key_num <= 4'h1;   // 按键1
                default:       key_num <= 4'h0;   // 多键按下
            endcase
         end
         else begin
            key_valid <= 0;
            key_num   <=0;
         end
     end
     else begin
        key_valid <= 0;
        key_num   <=0;
     end
end

endmodule

module counter #(
    parameter END=15,
    parameter WIDTH=4
)(
    input clk,
    input rst,
    input cnt_inc,
    output cnt_end,
    output reg[WIDTH-1:0] cnt
);

assign cnt_end = (cnt == END);

always @(posedge clk, posedge rst) begin
    if      (rst)   cnt <= 0;
    else if (cnt_end) cnt <= 0;
    else if (cnt_inc) cnt <= cnt + 1;
end
endmodule
```

参考的约束文件如下：

```
set_property -dict {PACKAGE_PIN Y18 IOSTANDARD LVCMOS33} [get_ports clk]
set_property -dict {PACKAGE_PIN R1  IOSTANDARD LVCMOS33} [get_ports rst]

set_property -dict {PACKAGE_PIN K4 IOSTANDARD LVCMOS33} [get_ports {row[0]}]
set_property -dict {PACKAGE_PIN J4 IOSTANDARD LVCMOS33} [get_ports {row[1]}]
set_property -dict {PACKAGE_PIN L3 IOSTANDARD LVCMOS33} [get_ports {row[2]}]
set_property -dict {PACKAGE_PIN K3 IOSTANDARD LVCMOS33} [get_ports {row[3]}]
set_property -dict {PACKAGE_PIN M2 IOSTANDARD LVCMOS33} [get_ports {col[0]}]
set_property -dict {PACKAGE_PIN K6 IOSTANDARD LVCMOS33} [get_ports {col[1]}]
set_property -dict {PACKAGE_PIN J6 IOSTANDARD LVCMOS33} [get_ports {col[2]}]
set_property -dict {PACKAGE_PIN L5 IOSTANDARD LVCMOS33} [get_ports {col[3]}]


set_property -dict {PACKAGE_PIN K17 IOSTANDARD LVCMOS33} [get_ports {key_valid}]
set_property -dict {PACKAGE_PIN A21 IOSTANDARD LVCMOS33} [get_ports {key_num[0]}]
set_property -dict {PACKAGE_PIN E22 IOSTANDARD LVCMOS33} [get_ports {key_num[1]}]
set_property -dict {PACKAGE_PIN D22 IOSTANDARD LVCMOS33} [get_ports {key_num[2]}]
set_property -dict {PACKAGE_PIN E21 IOSTANDARD LVCMOS33} [get_ports {key_num[3]}]
```