矩阵键盘检测例程

矩阵键盘是常用输入设备，在按键数量较多时，为了节省 I/O 口等单片机资源，一般采取扫描的方式来识别到底是哪一个键被按下。即通过确定被按下的键处在哪一行哪一列来确定该键的位置，获取键值以启动相应的功能程序。

本实验内容将介绍矩阵键盘的检测及按键的防抖的程序设计，通过点亮/点灭开发板上相应的 LED 灯来指示矩阵键盘的动作。

矩阵键盘 K1 按一下----------LED1 反转；

矩阵键盘 K2 按一下----------LED2 反转；

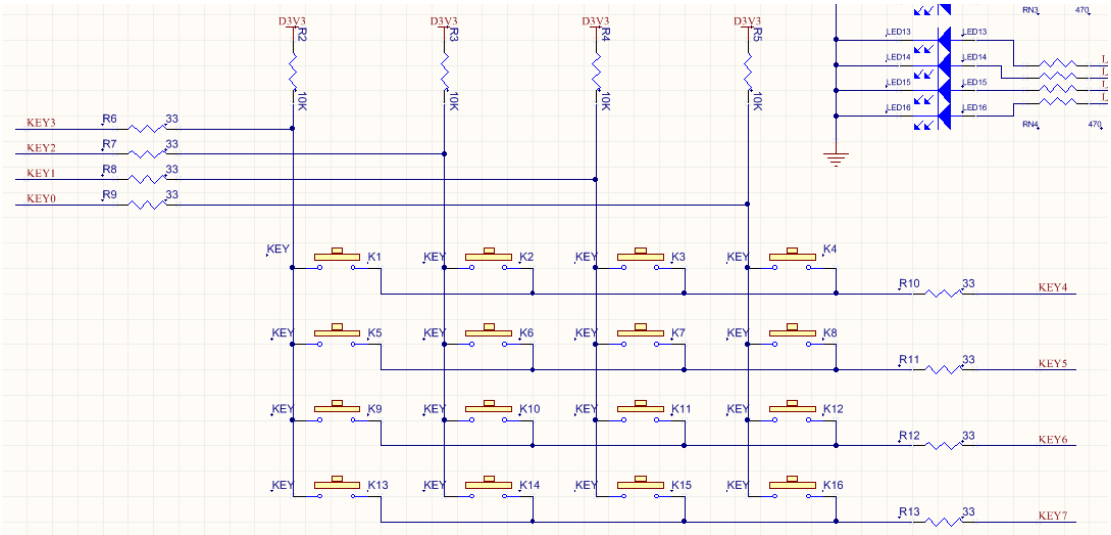
矩阵键盘 K3 按一下----------LED3 反转；

矩阵键盘 K4 按一下----------LED4 反转；

……

1 硬件介绍

下图为 4*4 矩阵键盘的原理图（具体原理图参考矩阵键盘 AN4040 资料）。在矩阵键盘模块中，矩阵键盘的四列和四行依次接到 FPGA 的 IO；同时，将列线上拉，通过 10K 电阻接电源。



信号 KEY4 对应矩阵键盘的第一行，KEY5 对应矩阵键盘的第二行，KEY6 对应矩阵键盘的第三行，KEY7 对应矩阵键盘的第四行。

信号 KEY3 对应矩阵键盘的第 1 列，KEY2 对应矩阵键盘的第 2 列，KEY1

对应矩阵键盘的第 3 列，KEY0 对应矩阵键盘的第 4 列。

2 程序设计

查找哪个按键被按下的方法为：一个一个地查找。

先第一行输出 0，检查列线是否非为高；

再第二行输出 0，检查列线是否非为高；

再第三行输出 0，检查列线是否非为高；

再第三行输出 0，检查列线是否非为高；

如果某行输出 0 时，查到列线非全高，则该行有按键按下；

根据第几行线输出 0 与第几列线读入为 0，即可判断在具体什么位置的按键按下。这是一种比较经典的矩阵键盘识别方法，实现起来较为简单，程序短小精炼。

下面是具体程序：

```
//=============================================================
// Module name: key4x4_test.v
// 描述: 检测开发板上的四乘四的矩阵键盘 K1~K164, 当检测到按键按下时,相应的 LED 灯翻转
//=============================================================
`timescale 1ns / 1ps
module key4x4_test  (
                                clk,                // 开发板上输入时钟: 50Mhz
                                rst_n,              // 开发板上复位按键
                                key_in_y,           // 输入矩阵键盘的列信号(KEY0~KEY3)
                                key_out_x,          // 输出矩阵键盘的行信号(KEY4~KEY7)
                                led_out                 // 输出 LED 灯,用于矩阵键盘板上 16 个
LED(LED1~LED16)
                            );

//=============================================================
// PORT declarations
//=============================================================

input           clk;
input           rst_n;
input   [3:0] key_in_y;
output reg [3:0] key_out_x;
output [15:0] led_out;

//寄存器定义
```

```
reg [19:0] count;

//================================================
// 输出矩阵键盘的行信号，20ms 扫描矩阵键盘一次,采样频率小于按键毛刺频率，相当于滤除掉了高频毛刺信号。
//================================================
always @(posedge clk or negedge rst_n)        //检测时钟的上升沿和复位的下降沿
begin
    if(!rst_n) begin                          //复位信号低有效
        count <= 20'd0;             //计数器清 0
        key_out_x <= 4'b1111;
    end
    else begin
                if(count == 20'd0)                  //0ms 扫描第一行矩阵键盘
                begin
                    key_out_x <= 4'b1110;     //开始扫描第一行矩阵键盘,第一行输出 0
                    count <= count + 20'b1; //计数器加 1
                end
            else if(count == 20'd249_999) //5ms 扫描第二行矩阵键盘,5ms 计数(50M/200-1=249_999)
                begin
                    key_out_x <= 4'b1101;     //开始扫描第二行矩阵键盘,第二行输出 0
                    count <= count + 20'b1; //计数器加 1
                end
                  else   if(count   ==20'd499_999)       //10ms  扫 描 第 三 行 矩 阵 键 盘 ,10ms  计 数
(50M/100-1=499_999)
                begin
                    key_out_x <= 4'b1011;     //扫描第三行矩阵键盘,第三行输出 0
                    count <= count + 20'b1; //计数器加 1
                end
                  else   if(count   ==20'd749_999)       //15ms  扫 描 第 四 行 矩 阵 键 盘 ,15ms  计 数
(50M/67.7-1=749_999)
                begin
                    key_out_x <= 4'b0111;     //扫描第四行矩阵键盘,第四行输出 0
                    count <= count + 20'b1; //计数器加 1
                end
            else if(count ==20'd999_999)    //20ms 计数(50M/50-1=999_999)
               begin
                        count <= 0;                     //计数器为 0
                end
               else
                         count <= count + 20'b1;      //计数器加 1

       end
end
//====================================================
```

```
// 采样列的按键信号
//=====================================================
reg [3:0] key_h1_scan;      //第一行按键扫描值 KEY
reg [3:0] key_h1_scan_r;    //第一行按键扫描值寄存器 KEY
reg [3:0] key_h2_scan;      //第二行按键扫描值 KEY
reg [3:0] key_h2_scan_r;    //第二行按键扫描值寄存器 KEY
reg [3:0] key_h3_scan;      //第三行按键扫描值 KEY
reg [3:0] key_h3_scan_r;    //第三行按键扫描值寄存器 KEY
reg [3:0] key_h4_scan;      //第四行按键扫描值 KEY
reg [3:0] key_h4_scan_r;    //第四行按键扫描值寄存器 KEY
always @(posedge clk)
      begin
           if(!rst_n) begin                    //复位信号低有效
                key_h1_scan <= 4'b1111;
                key_h2_scan <= 4'b1111;
                key_h3_scan <= 4'b1111;
                key_h4_scan <= 4'b1111;
           end
           else begin
             if(count == 20'd124_999)             //2.5ms 扫描第一行矩阵键盘值
                   key_h1_scan<=key_in_y;           //扫描第一行的矩阵键盘值
             else if(count == 20'd374_999)       //7.5ms 扫描第二行矩阵键盘值
                   key_h2_scan<=key_in_y;           //扫描第二行的矩阵键盘值
             else if(count == 20'd624_999)       //12.5ms 扫描第三行矩阵键盘值
                   key_h3_scan<=key_in_y;           //扫描第三行的矩阵键盘值
             else if(count == 20'd874_999)       //17.5ms 扫描第四行矩阵键盘值
                   key_h4_scan<=key_in_y;           //扫描第四行的矩阵键盘值
           end
end

//=====================================================
// 按键信号锁存一个时钟节拍
//=====================================================
always @(posedge clk)
    begin
            key_h1_scan_r <= key_h1_scan;
            key_h2_scan_r <= key_h2_scan;
            key_h3_scan_r <= key_h3_scan;
            key_h4_scan_r <= key_h4_scan;
      end

wire [3:0] flag_h1_key = key_h1_scan_r[3:0] & (~key_h1_scan[3:0]);  //当检测到按键有下降沿变化时，代表该按键被按下，按键有效
wire [3:0] flag_h2_key = key_h2_scan_r[3:0] & (~key_h2_scan[3:0]);  //当检测到按键有下降沿变化时，代表该按键被按下，按键有效
```

```
wire [3:0] flag_h3_key = key_h3_scan_r[3:0] & (~key_h3_scan[3:0]);  //当检测到按键有下降沿变化时，代表该按键被按下，按键有效
wire [3:0] flag_h4_key = key_h4_scan_r[3:0] & (~key_h4_scan[3:0]);  //当检测到按键有下降沿变化时，代表该按键被按下，按键有效

//=====================================================
// LED 灯控制,按键按下时,相关的 LED 输出翻转
//=====================================================
reg [15:0] temp_led;
always @ (posedge clk or negedge rst_n)         //检测时钟的上升沿和复位的下降沿
begin
    if (!rst_n)                         //复位信号低有效
        temp_led <= 16'd0;        //LED 灯控制信号输出为低, LED 灯全灭
    else
        begin
            if ( flag_h1_key[0] ) temp_led[0] <= ~temp_led[0];    //按键第一行的 KEY1 值变化时，LED1 将做亮灭翻转
            if ( flag_h1_key[1] ) temp_led[1] <= ~temp_led[1];    //按键第一行的 KEY2 值变化时，LED2 将做亮灭翻转
            if ( flag_h1_key[2] ) temp_led[2] <= ~temp_led[2];    //按键第一行的 KEY3 值变化时，LED3 将做亮灭翻转
            if ( flag_h1_key[3] ) temp_led[3] <= ~temp_led[3];    //按键第一行的 KEY4 值变化时，LED4 将做亮灭翻转
                if ( flag_h2_key[0] ) temp_led[4] <= ~temp_led[4];    //按键第二行的 KEY5 值变化时，LED5 做亮灭翻转
            if ( flag_h2_key[1] ) temp_led[5] <= ~temp_led[5];    //按键第二行的 KEY6 值变化时，LED6 将做亮灭翻转
            if ( flag_h2_key[2] ) temp_led[6] <= ~temp_led[6];    //按键第二行的 KEY7 值变化时，LED7 将做亮灭翻转
            if ( flag_h2_key[3] ) temp_led[7] <= ~temp_led[7];    //按键第二行的 KEY8 值变化时，LED8 将做亮灭翻转
                if ( flag_h3_key[0] ) temp_led[8] <= ~temp_led[8];    //按键第三行的 KEY9 值变化时，LED9 将做亮灭翻转
            if ( flag_h3_key[1] ) temp_led[9] <= ~temp_led[9];    //按键第三行的 KEY10 值变化时，LED10 将做亮灭翻转
            if ( flag_h3_key[2] ) temp_led[10] <= ~temp_led[10];    //按键第三行的 KEY11 值变化时，LED11 将做亮灭翻转
            if ( flag_h3_key[3] ) temp_led[11] <= ~temp_led[11];    //按键第三行的 KEY12 值变化时，LED12 将做亮灭翻转
                if ( flag_h4_key[0] ) temp_led[12] <= ~temp_led[12];    //按键第四行的 KEY13 值变化时，LED13 将做亮灭翻转
            if ( flag_h4_key[1] ) temp_led[13] <= ~temp_led[13];    //按键第四行的 KEY14 值变化时，LED14 将做亮灭翻转
            if ( flag_h4_key[2] ) temp_led[14] <= ~temp_led[14];    //按键第四行的 KEY15 值变化时，LED15 将做亮灭翻转
```

```
                if ( flag_h4_key[3] ) temp_led[15] <= ~temp_led[15];    //按键第四行的 KEY16 值变化
时，LED16 将做亮灭翻转
            end
end

 assign led_out = temp_led;

endmodule
```

程序里都有中文说明，这里还是再简单说明一下，每 20ms 一个周期程序会扫描矩阵键盘 4 行上按键的状态（每行时间为 5ms），如果这次检测为低电平而且上次检测为高电平的话，说明有按键按下，相对应的 LED 灯反转。

3 管脚约束

配置 FPGA 对应的管脚约束，具体管脚约束需要参考矩阵模块的原理图和具体 FPGA 开发板的原理图。以下为 AX301 开发板的管脚约束。

clk	Input	PIN_E1	1	B1_N0	3.3-V LV...default)	8mA
key_in_y[3]	Input	PIN_B5	8	B8_N0	3.3-V LV...default)	8mA
key_in_y[2]	Input	PIN_A4	8	B8_N0	3.3-V LV...default)	8mA
key_in_y[1]	Input	PIN_B4	8	B8_N0	3.3-V LV...default)	8mA
key_in_y[0]	Input	PIN_A3	8	B8_N0	3.3-V LV...default)	8mA
key_out_x[3]	Output	PIN_B3	8	B8_N0	3.3-V LV...default)	8mA
key_out_x[2]	Output	PIN_A2	8	B8_N0	3.3-V LV...default)	8mA
key_out_x[1]	Output	PIN_B1	1	B1_N0	3.3-V LV...default)	8mA
key_out_x[0]	Output	PIN_C2	1	B1_N0	3.3-V LV...default)	8mA
led_out[15]	Output	PIN_B9	7	B7_N0	3.3-V LV...default)	8mA
led_out[14]	Output	PIN_A8	8	B8_N0	3.3-V LV...default)	8mA
led_out[13]	Output	PIN_B8	8	B8_N0	3.3-V LV...default)	8mA
led_out[12]	Output	PIN_A7	8	B8_N0	3.3-V LV...default)	8mA
led_out[11]	Output	PIN_B7	8	B8_N0	3.3-V LV...default)	8mA
led_out[10]	Output	PIN_A6	8	B8_N0	3.3-V LV...default)	8mA
led_out[9]	Output	PIN_B6	8	B8_N0	3.3-V LV...default)	8mA
led_out[8]	Output	PIN_A5	8	B8_N0	3.3-V LV...default)	8mA
led_out[7]	Output	PIN_B13	7	B7_N0	3.3-V LV...default)	8mA
led_out[6]	Output	PIN_A12	7	B7_N0	3.3-V LV...default)	8mA
led_out[5]	Output	PIN_B12	7	B7_N0	3.3-V LV...default)	8mA
led_out[4]	Output	PIN_A11	7	B7_N0	3.3-V LV...default)	8mA
led_out[3]	Output	PIN_B11	7	B7_N0	3.3-V LV...default)	8mA
led_out[2]	Output	PIN_A10	7	B7_N0	3.3-V LV...default)	8mA
led_out[1]	Output	PIN_B10	7	B7_N0	3.3-V LV...default)	8mA
led_out[0]	Output	PIN_A9	7	B7_N0	3.3-V LV...default)	8mA
rst_n	Input	PIN_N13	5	B5_N0	3.3-V LV...default)	8mA

4 按键实验

编译工程并生成 FPGA 配置文件，然后把矩阵按键模块插到相应的扩展口上，比如 AX301 开发板是插到扩展口 J2 上。再把 FPGA 配置文件下载到 FPGA 中，接下去我们就可以测试按键的效果了。

按一下矩阵键盘上的 K1，矩阵键盘 LED1 变亮，再按一下 K1，LED1 熄灭。

实验结果是否跟我们的设计的思路一样。是否感觉就像家里的灯一样由墙上的开关控制？同样您可以再试试矩阵键盘的其它按键的效果。