按键消抖实验

键盘分编码键盘和非编码键盘。键盘上闭合键的识别由专用的硬件编码器实现，并产生键编码号或键值的称为编码键盘，如计算机键盘。而靠软件编程来识别的称为非编码键盘。

在一般嵌入式应用中，用的最多的是非编码键盘，也有用到编码键盘的。非编码键盘有分为：独立键盘和行列式（又称为矩阵式）键盘。

如图 5.3 所示，按键在闭合和断开时，触点会存在抖动现象。在按键按下或者是释放的时候都会出现一个不稳定的抖动时间，如果不处理好这个抖动时间，我们就无法处理好按键编码，所以我们的设计中必须有效消除按键抖动。

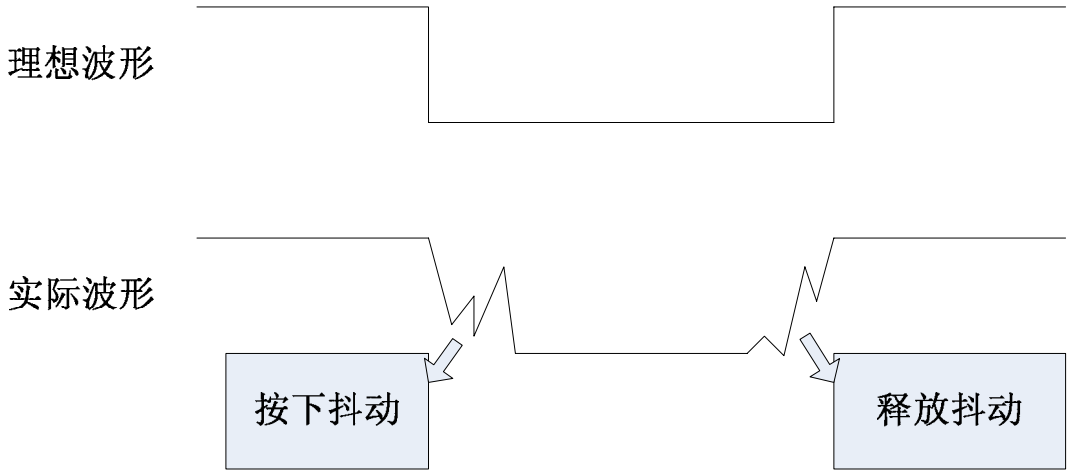

图 5.3 按键波形

如图 5.4 所示，4 个独立按键一端接地，另一端在上拉的同时连接到 CPLD 的 I/O 口。当 I/O 口（SW0/SW1/SW2/SW3）的电平为高时，说明按键没有被按下，当 I/O 口的电平为低时，说明按键被按下了。

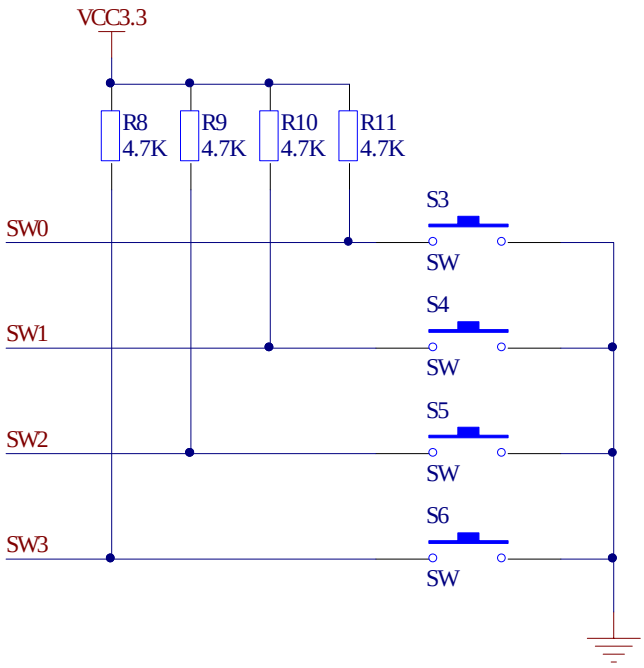

图 5.4 独立按键接口

如图 5.5 所示，4 个发光二极管通过串入一个 510 欧姆的分压电阻后与 CPLD 的 I/O 口连接。发光二极管的负极接地。因此，CPLD 的 I/O 口输出高电平时，二极管导通，则点亮；CPLD 的 I/O 口输出低电平时，二极管截止，则不亮。

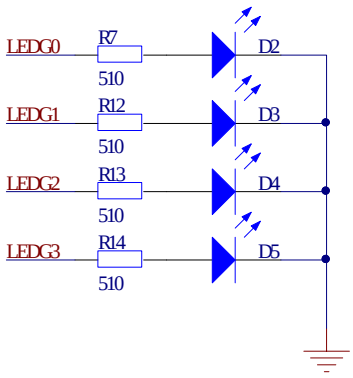

图 5.5 发光二极管接口

该实验需要实现一个简单的三个按键分别控制三个发光二极管亮或暗的控制。例如，按键 1 控制发光二极管 1。上电初始发光二极管 1 不亮，当检测到按键 1 被按下后，发光二极管 1 则点亮，按键 1 再次被按下时，发光二极管 1 则不亮，如此反复。该实验需要把握好按键消抖检测的设计技巧。其接口定义如表 5.4 所示。

表 5.4 按键消抖实验接口定义

信号名称	方向	描述

clk	input	时钟信号，50MHz
rst_n	input	复位信号，低电平有效
sw1_n	input	按键 1，控制发光二极管 1
sw2_n	input	按键 2，控制发光二极管 2
sw3_n	input	按键 3，控制发光二极管 3
led_d1	output	发光二极管 1
led_d2	output	发光二极管 2
led_d3	output	发光二极管 3

```
module sw_debounce(
            clk,rst_n,
            sw1_n,sw2_n,sw3_n,
            led_d1,led_d2,led_d3
            );

input   clk;    //主时钟信号，50MHz
input   rst_n;  //复位信号，低有效
input   sw1_n,sw2_n,sw3_n;  //三个独立按键，低表示按下
output  led_d1,led_d2,led_d3;   //发光二极管，分别由按键控制

//-----------------------------------------------------------------------
reg[2:0] key_rst;

always @(posedge clk  or negedge rst_n)
    if (!rst_n) key_rst <= 3'b111;
    else key_rst <= {sw3_n,sw2_n,sw1_n};

reg[2:0] key_rst_r;        //每个时钟周期的上升沿将 low_sw 信号锁存到 low_sw_r 中

always @ ( posedge clk  or negedge rst_n )
    if (!rst_n) key_rst_r <= 3'b111;
    else key_rst_r <= key_rst;

//当寄存器 key_rst 由 1 变为 0 时，led_an 的值变为高，维持一个时钟周期
wire[2:0] key_an = key_rst_r & ( ~key_rst);

//-----------------------------------------------------------------------
reg[19:0]  cnt; //计数寄存器
```

```
always @ (posedge clk  or negedge rst_n)
    if (!rst_n) cnt <= 20'd0;   //异步复位
    else if(key_an) cnt <=20'd0;
    else cnt <= cnt + 1'b1;

reg[2:0] low_sw;

always @(posedge clk  or negedge rst_n)
    if (!rst_n) low_sw <= 3'b111;
    else if (cnt == 20'hfffff)
                  //满 20ms，将按键值锁存到寄存器 low_sw 中   cnt == 20'hfffff
      low_sw <= {sw3_n,sw2_n,sw1_n};

//-----------------------------------------------------------------------
reg  [2:0] low_sw_r;       //每个时钟周期的上升沿将 low_sw 信号锁存到 low_sw_r 中

always @ ( posedge clk  or negedge rst_n )
    if (!rst_n) low_sw_r <= 3'b111;
    else low_sw_r <= low_sw;

//当寄存器 low_sw 由 1 变为 0 时，led_ctrl 的值变为高，维持一个时钟周期
wire[2:0] led_ctrl = low_sw_r[2:0] & ( ~low_sw[2:0]);

reg d1;
reg d2;
reg d3;

always @ (posedge clk or negedge rst_n)
    if (!rst_n) begin
        d1 <= 1'b0;
        d2 <= 1'b0;
        d3 <= 1'b0;
      end
    else begin      //某个按键值变化时，LED 将做亮灭翻转
        if ( led_ctrl[0] ) d1 <= ~d1;
        if ( led_ctrl[1] ) d2 <= ~d2;
        if ( led_ctrl[2] ) d3 <= ~d3;
      end
```

```
assign led_d3 = d1 ? 1'b1 : 1'b0;        //LED 翻转输出
assign led_d2 = d2 ? 1'b1 : 1'b0;
assign led_d1 = d3 ? 1'b1 : 1'b0;

endmodule
```

也许初学者看到这段代码似乎有点吃力，好多的 always 好多的 wire 啊。的确是这样，一个好的 verilog 代码，它不应该把一大堆寄存器的赋值都放到一个 always 语句里，那样虽然看上去代码很简洁，但是维护起来却很痛苦。所以设计中往往应该一个（或者少数几个）寄存器的赋值对应单独的一个 always 来执行。其次是 wire 连线很多，要是仔细研究代码，大家就不难发现所有的寄存器的连线关系设计者都考虑到了，设计者就是必须对自己的代码要实现的每一个细节都做到心中有数。细化到每一个寄存器、每一条连线。

上面说的是代码风格，下面就看看按键消抖的设计思想。前两个 always 语句其实是对输入的键盘进行两次锁存，key_rst 和 key_rst_r 是前后两次采样时刻获得的键值。key_an 是通过一定的逻辑关系，当键值从 0 到 1 变化时刻到得一个时钟宽度的高脉冲。这里的 key_an 是为了后面的 20ms 消抖复位使用的。

第三个 alwsys 做了一个（大约）20ms 的计数，这个计数器会在上面得到的 key_an 高脉冲时复位，也就是说，一旦发现按键出现从 0 到 1 的释放抖动时，就对计数寄存器清零。然后隔 20ms 就会再读取键值，把这个键值放到寄存器 low_sw 中，接下来的一个 always 语句就是把 low_sw 的值锁存到 low_sw_r 里，这样一来，low_sw 和 low_sw_r 就是前后两个时钟周期里的键值了，为什么要这样呢？看下一个语句吧：

```
wire [2:0] led_ctrl = low_sw_r[2:0] & ( ~low_sw[2:0]);
```

仔细分析，你会发现当没有键按下时，low_sw=low_sw_r=3’b111，此时的 led_ctrl=3’b000；只有当 low_sw_r 和 low_sw 的某一位分别为 1 和 0 时，才可能使 led_ctrl 的值改变（也就是把 led_ctrl 的某一位拉高一个时钟周期）。这就意味着当键值由 0 跳变到 1 时（按键释放时）才可能把 led_ctrl 拉高。回顾前面的 20ms 赋键值，也就是说每 20ms 内如果出现按键被按下，那么有一个时钟周期里 led_ctrl 是会被拉高的，而再看后面的代码，led_ctrl 的置高就使得相应的 LED 灯的亮灭做一次翻转，这就达到了目的。

欢迎加入 EDN 网站 FPGA/CPLD 助学小组 http://group.ednchina.com/1375/

购买 BJ-EPM240 CPLD 学习板 http://group.ednchina.com/1375/23842.aspx

购买 SF-EP1C FPGA 开发板 http://group.ednchina.com/1375/27650.aspx

北航出版社将于 2010 年 3 月份前后出版《深入浅出玩转 FPGA》一书，欢迎各位网友到时购买，作为本视频和学习板/开发板的参考教材

特权

2009.11