**按键消抖**

开发板上的按钮和拨动开关都是机械弹性开关，当机械触点断开、闭合时，由于机械触点的弹性作用，一个按键开关在闭合时不会马上稳定地接通，在断开时也不会一下子断开。因而在闭合及断开的瞬间均伴随有一连串的抖动，为了不产生这种现象而作的措施就是按键消抖。DE10-Standard开发板上的KEY0-KEY3是经过按键消抖的，按下时为低电平，一般情况下可直接使用。

在时序逻辑电路中，我们通常需要手动产生一个上升沿或者一个下降沿，如将拨动开关拨上去（由0到1），这样就产生了一个上升沿；吧拨动开关从上面拨下来（由1到0），这样就产生了一个下降沿。如果开关有机械抖动的话，那么将开关从下拨到上或从上拨到下面，就会产生若干个上升沿或下降沿，如图1，这样电路就会产生意想不到的情况。因此，如果将拨动开关当作时钟使用，必须对其进行消抖。

抖动时间的长短由按键的机械特性决定，一般为20ms左右。

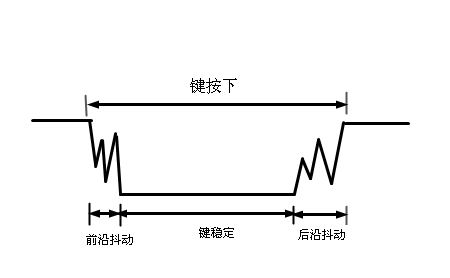


图1 按键抖动图

消抖是为了避免在按键按下或是抬起时电平剧烈抖动带来的影响。按键的消抖，可用硬件或软件两种方法。

**硬件消抖**

在键数较少时可用硬件方法消除键抖动。图2所示的RS触发器为常用的硬件去抖电路。

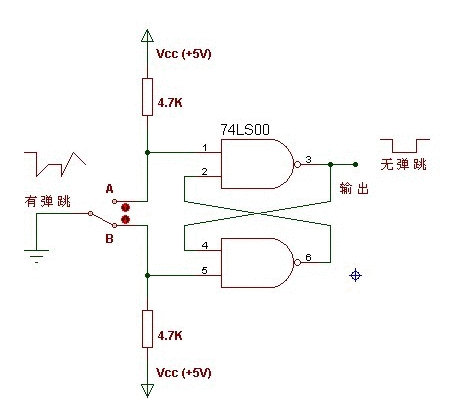


图2 利用RS触发器消除按键抖动

图中两个“与非”门构成一个RS触发器。当按键未按下时，输出为0;当键按下时，输出为1。此时即使用按键的机械性能，使按键因弹性抖动而产生瞬时断开（抖动跳开B），只要按键不返回原始状态A，双稳态电路的状态不改变，输出保持为0，不会产生抖动的波形。也就是说，即使B点的电压波形是抖动的，但经双稳态电路之后，其输出为正规的矩形波。

**软件消抖**

如果按键较多，常用软件方法去抖，即检测出键闭合后执行一个延时程序，20ms左右的延时，让前沿抖动消失后再一次检测键的状态，如果仍保持闭合状态电平，则确认为真正有键按下。当检测到按键释放后，也要给20ms左右的延时，待后沿抖动消失后才能转入该键的处理程序。

一般来说，软件消抖的方法是不断检测按键值，直到按键值稳定。实现方法：假设未按键时输入1，按键后输入为0，抖动时不定。可以做以下检测：检测到按键输入为0之后，延时20ms，再次检测，如果按键还为0，那么就认为有按键输入。延时的20ms恰好避开了抖动期。

下面代码是Verilog HDL语言实现按键消抖的参考代码，是对4输入的信号，即一次对四个按键或者按钮进行了消抖，平时设计中，一般只需要一个时钟信号就可以了，对于只对一个按键进行消抖，可在此基础上修改。

module debounce4(

input wire [3:0] inp, // 4位宽的输入开关或按键

input wire clk, // 开发平台的内部时钟，100MHZ，周期10ns

input wire clr, // 复位开关，

output [3:0] outp ); // 消抖后的输出

reg [3:0] delay1; // 第一次采样

reg [3:0] delay2; // 第二次采样

reg [19:0] count; // 计数器计数约220≈106次

reg clk\_20ms; // 时钟，周期约20ms

always @ (posedge clk  or negedge clr)

    if (!clr)  begin        //异步复位

        count <= 20'd0;

clk\_20ms 〈= 1'b0;

end

/\* 计数器count在每个时钟clk的上升沿（约10ns）加1计数一次，共计220次，约10ms后清0，重头计数。\*/

    else if (count == 20'hfffff) begin

clk\_20ms <= ~clk\_20ms; // clk\_20ms每10ms取反一次

count <= 20'd0;

end

else count <= count + 1'b1;

always @ (posedge clk\_20ms or negedge clr)

 if (!clr)  begin

        delay1 <= 4'b0000;

       delay2 <= 4'b0000;

end

else

   begin

    delay1 <= inp;

delay2 <= delay1;

   end

assign outp = delay1 & delay2 ;

endmodule

### 

重点看看消抖程序代码中的采样段代码：这里巧妙地用了非阻塞赋值语句完成两次采样。关于阻塞赋值和非阻塞赋值请查看“实验6 锁存器和触发器”。

always @ (posedge clk\_20ms)

   begin

    delay1 <= inp;

delay2 <= delay1;

   end

在第一个时钟（clk\_20ms）上升沿，inp采样当前输入值，程序运行出always块时，delay1 等于当前 inp值（第一次采样值），delay2维持原delay1的值，到下一个时钟上升沿到来时，delay1 等于现在的 inp（第二次采样值），delay2维持上一次delay1（第一次采样值）的值。