因为不想写五个模块，弄五个消抖，尝试多个按键一起消抖。起初由于对消抖代码理解太浅，甚至尝试把“复位”键也放进消抖里，结果发现按下复位之后，输出一直是“复位”，直到按下另外一个按键后原先的“复位“才被取消掉。后发现代码中的btn0，1，2寄存器会一直出0，直到另一个按键按下之后才能停止“复位”。将复位的优先级置于消抖模块之上，即消抖模块无需对“复位“进行消抖，在此思路的指导下成功完成多个按键消抖的模块。

**设计代码：**

module key\_debounce(

input clk\_btn,

input rst,

input stop\_key,

input up,

input down,

input left,

input right,

output wire[4:0] btn\_out

);

reg btn0;

reg btn1;

reg btn2;

reg flag;

reg[4:0] temp\_out;

always@ (posedge clk\_btn or negedge rst)

begin

if (!rst) begin

btn0<=1'b0;

btn1<=1'b0;

btn2<=1'b0;

temp\_out = 5'b00000;

end

else begin

btn0 <= stop\_key || up || right || down || left;

btn1 <= btn0;

btn2 <= btn1;

end

end

//同一时间只能??测一个按键，多个按键??起按下的话优先级：暂??>up>down>left>rignt;

always @(\*) begin

if (btn0&btn1&btn2 == 1)

begin

if (stop\_key) begin

temp\_out = 5'b10000;

end

else if(up) begin

temp\_out = 5'b01000;

end

else if(down) begin

temp\_out = 5'b00100;

end

else if(left) begin

temp\_out = 5'b000010;

end

else if(right) begin

temp\_out = 5'b00001;

end

else begin

temp\_out = 5'b00000; //消除延迟。

end

end

else begin

temp\_out = 5'b00000;

end

end

assign btn\_out = temp\_out;

endmodule

**仿真代码：**

`timescale 1ns / 1ps

module debounce\_sim();

reg[5:0] btns;

reg[5:0] temp;

reg clk;

wire[4:0] out\_btn;

key\_debounce uut(clk,btns[5],btns[4],btns[3],btns[2],btns[1],btns[0],out\_btn);

initial begin

btns = 6'b100000;

temp = 6'b000001;

clk=0;

#160

btns[5] = 0;

#20

btns[5] = 1;

end

always #10 clk = !clk;

always

begin

#100

temp = {temp[4:0],temp[5]}; //verilog中好像没有循环左/右移啊。。。。

btns = {btns[5],temp[5:1]};

#100

btns = 6'b100000;

end

endmodule

