lab05

金泽文 pb15111604

实验目的：

* 熟练掌握时序逻辑电路的设计方法
* 了解寄存器文件的实现原理

实验内容：

* 设计一4\*4bit的寄存器文件
  + 具备一组读端口及一组写端口
  + 通过读端口可从0~3号的任意地址读取数据
  + 通过写端口可向0~3号的任意地址写入数据
  + 读写端口为“全双工”的工作方式
  + 0~3号寄存器的复位值依次为“1、2、4、8”
  + sw4~sw7为写数据端口;sw2~sw3为写地址；sw0~sw1为读地址；led0~led3用来显示读数据；写使能用按键实现；读使能可选

实验结果：

除了基本功能，还实现了消除抖动功能，以及数码管显示数据功能。

实验分析：

首先，本次实验吸取了上次的教训，认真阅读了verilog语法教程。

其次，实现了一些额外的模块：

消除抖动模块，同上次。

数码管显示模块，数码管sel<3>显示读地址 sel<2>读数据，十六进制 sel<1>写地址 sel<0>写数据，十六进制。

意见建议：

暂无

思考题：

由于此寄存器在电路实现实质上是触发器，而输出端同样是个触发器。当上升沿到来时，两边同时非阻塞赋值（锁存器被同时触发），所以可以正常写入，读出的是上次写入的数据。

附录：

代码：

最高层模块：

lab5.v

`timescale 1ns/1ps

module lab5(

input clk,

input [1:0] btn,

input [7:0] sw,

output reg [7:0] led,

output [7:0] seg\_out,

output [3:0] sel

);

wire [15:0] data;

wire [1:0] btn\_kj;

reg [3:0] regs [3:0];

wire [1:0] read\_address = sw[1:0];

wire [1:0] write\_address = sw[3:2];

wire [3:0] write\_regs = sw[7:4];

reg [3:0] regs\_seg;

always@(posedge clk)begin

case (read\_address)

0: regs\_seg[3:0] <= regs[0][3:0];

1: regs\_seg[3:0] <= regs[1][3:0];

2: regs\_seg[3:0] <= regs[2][3:0];

3: regs\_seg[3:0] <= regs[3][3:0];

endcase

end

assign data[3:0] = write\_regs[3:0];

assign data[7:4] = write\_address[1:0];

assign data[11:8]= regs\_seg[3:0];

assign data[15:12] = read\_address[1:0];

wire rst = btn\_kj[0];

wire we = btn\_kj[1];//write\_enable

key\_jitter kj1(clk, btn[0], btn\_kj[0]);

key\_jitter kj2(clk, btn[1], btn\_kj[1]);

seg\_display sd(clk, data, seg\_out, sel);

initial begin

regs[3] = 8;

regs[2] = 4;

regs[1] = 2;

regs[0] = 1;

end

always@(posedge clk or posedge rst)begin

if(rst)begin

led[7:0] <= 8'b11111111;//once rst all leds will be on

end

else if(~rst)begin

case(read\_address)

0:led[3:0] <= regs[0];

1:led[3:0] <= regs[1];

2:led[3:0] <= regs[2];

3:led[3:0] <= regs[3];

endcase

led[7:4] <= write\_regs;

end

end

always @(posedge clk or posedge rst) begin

if (rst) begin

regs[3] <= 8;

regs[2] <= 4;

regs[1] <= 2;

regs[0] <= 1;

end

else if (we) begin

case(write\_address)

0: regs[0] <= write\_regs;

1: regs[1] <= write\_regs;

2: regs[2] <= write\_regs;

3: regs[3] <= write\_regs;

endcase

end

end

endmodule

消除抖动模块 key\_jitter

module key\_jitter(

input clk, // 100MHz clock in

input key\_in,

output key\_out

);

reg [19:0] count\_high;

reg [19:0] count\_low;

reg key\_reg;

assign key\_out = key\_reg;

always@(posedge clk)

if( key\_in == 1'b0 )

count\_low <= count\_low + 1;

else

count\_low <= 20'd 0\_000\_000;

always@(posedge clk)

if(key\_in == 1'b1)

count\_high <= count\_high +1;

else

count\_high <= 20'd 0\_000\_000;

// deJitter key out

always@(posedge clk)

begin

if (count\_high == 20'd 1\_000\_000)

key\_reg <= 1'b1;

else

if(count\_low == 20'd 1\_000\_000)

key\_reg <= 1'b0;

else

key\_reg <= key\_reg;

end

endmodule

数码管显示模块

module seg\_display(

input clk,

input [15:0] dataflow,

output [7:0] seg\_out,

output [3:0] sel);

reg [7:0] seg\_out\_reg;

reg [20:0] counter;

reg [3:0] sel\_reg;

assign sel = sel\_reg;

assign seg\_out = seg\_out\_reg;

function [7:0] display;

input[3:0] a;

case (a)

4'h0 : display=8'b11000000;

4'h1 : display=8'b11111001;

4'h2 : display=8'b10100100;

4'h3 : display=8'b10110000;

4'h4 : display=8'b10011001;

4'h5 : display=8'b10010010;

4'h6 : display=8'b10000010;

4'h7 : display=8'b11111000;

4'h8 : display=8'b10000000;

4'h9 : display=8'b10010000;

4'ha : display=8'b10001000;

4'hb : display=8'b10000011;

4'hc : display=8'b11000110;

4'hd : display=8'b10100001;

4'he : display=8'b10000110;

4'hf : display=8'b10001110;

default : display=8'b11111111;

endcase

endfunction

always@(posedge clk)begin

if(counter > 1\_000) begin

counter <= 0;

case(sel\_reg)

4'b1110: begin seg\_out\_reg <= display(dataflow[7:4]); sel\_reg <= 4'b1101; end

4'b1101: begin seg\_out\_reg <= display(dataflow[11:8]); sel\_reg <= 4'b1011; end

4'b1011: begin seg\_out\_reg <= display(dataflow[15:12]); sel\_reg <= 4'b0111; end

4'b0111: begin seg\_out\_reg <= display(dataflow[3:0]); sel\_reg <= 4'b1110; end

default: sel\_reg <= 4'b1110;

endcase

end

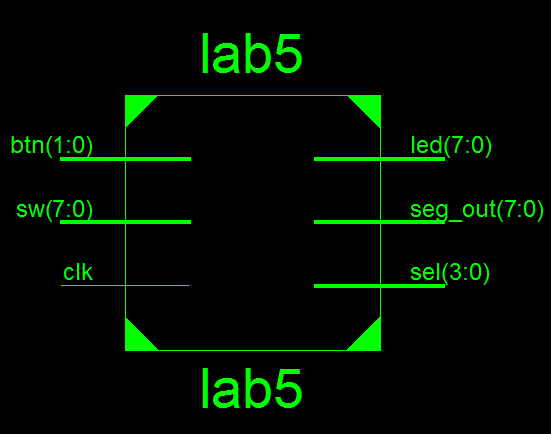
else

counter <= counter + 1;

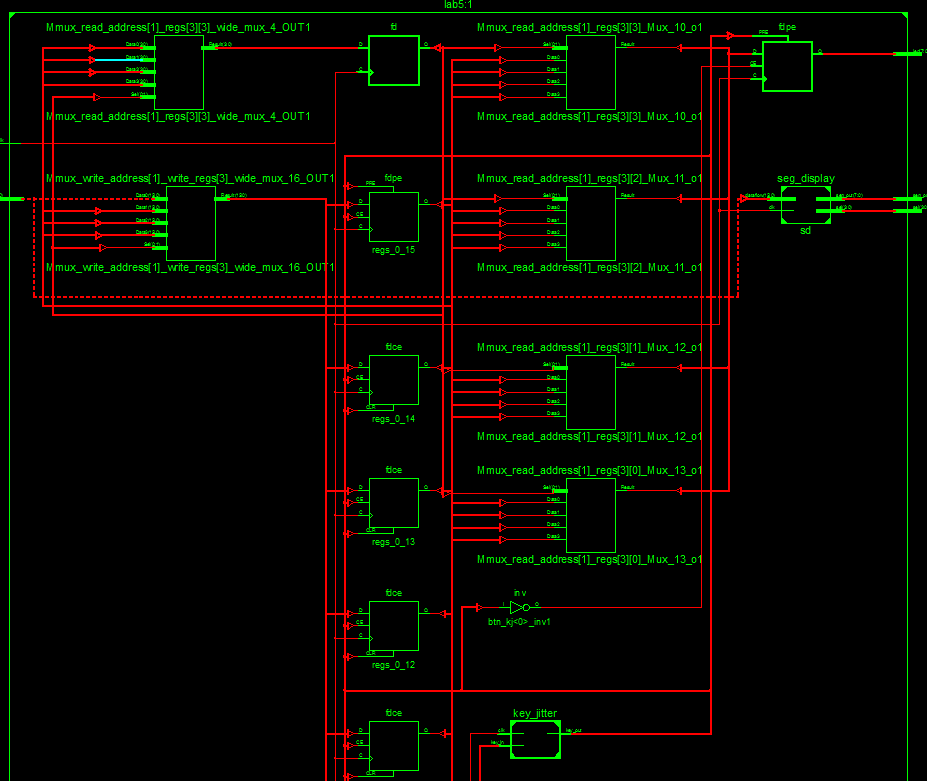
end

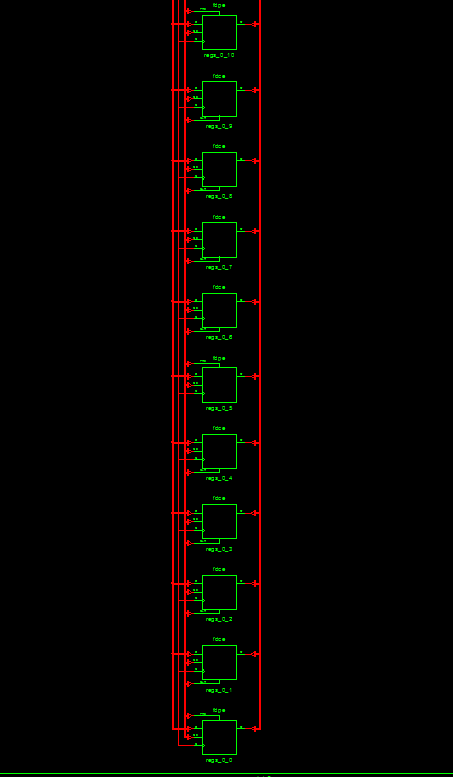
endmodule

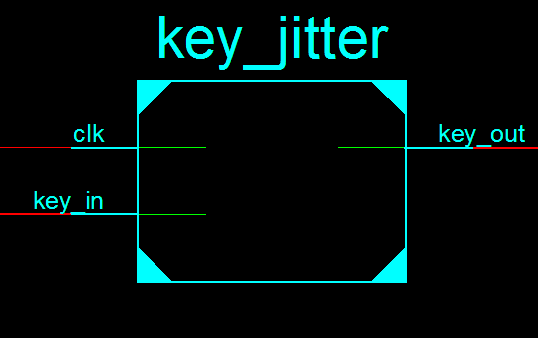
原理图：

顶层模块：

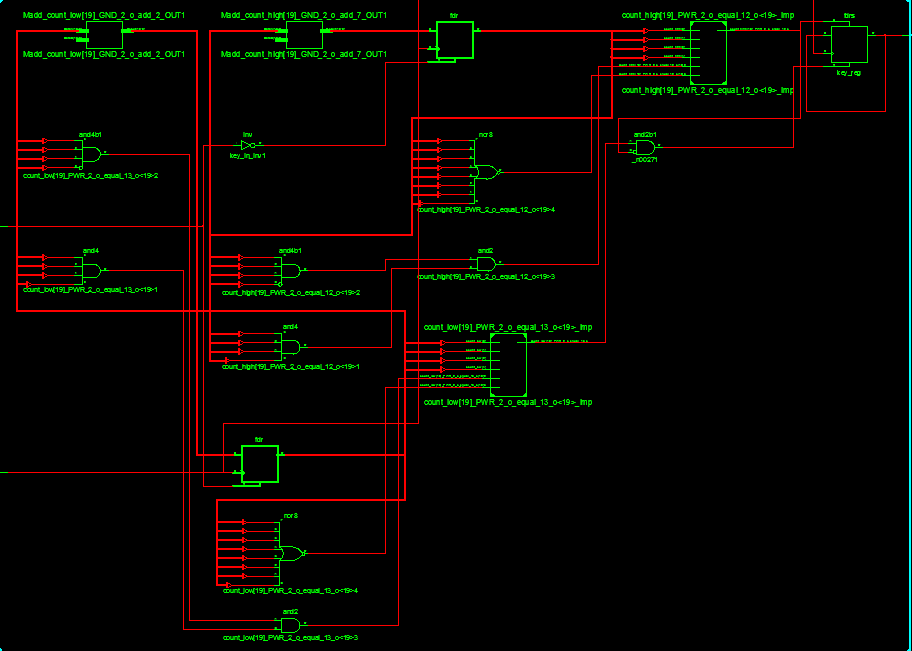
lab5







消除抖动模块 key\_jitter



数码管模块

seg\_display

