实验报告二

PB14000556 陈晓彤

实验题目：寄存器文件设计 及 模块调用完成斐波那契数列计算

实验要求：

设计一32\*32bit的寄存器文件，即32个32位的寄存器文件（寄存器组）

具备两组读端口及一组写端口

通过读端口可从0~31号的任意地址读取数据

通过写端口可向0~31号的任意地址写入数据

寄存器的复位值自行制定

调用实验一ALU，完成以下功能

寄存器文件组r0,r1初始化为1，1，其他所有寄存器初始化为0

在clk控制下，依次完成以下计算，注意每个clk至多允许完成一次计算

r0+r1->r2

r1+r2->r3

r2+r3->r4

……

结果在仿真中显示

设计思路：

采用二维数组实现寄存器文件；

使用控制模块调用加法器和寄存器，模块调用时，采用固定机制，变换地址的方法实现循环，即：加法器与寄存器的入口不断变换，以达到不停取数，读数，计算的目的；

源代码:

module lab02\_ctrl(

input clk,

input rst\_n,

output [31:0] a,

output [31:0] b,

output [31:0] sum

);

reg [4:0] ad0;

reg [4:0] ad1;

reg [4:0] ad2;

lab02\_alu alu1(

.alu\_a (a),

.alu\_b (b),

.alu\_op (5'b1),

.alu\_out (sum)

);

lab02\_reg reg1(

.clk (clk),

.rst\_n (rst\_n),

.r1\_addr (ad0),

.r2\_addr (ad1),

.r3\_addr (ad2),

.r3\_din (sum),

.r3\_wr (1'b1),

.r1\_dout (a),

.r2\_dout (b)

);

always@(posedge clk or negedge rst\_n)

begin

if(~rst\_n)

begin

ad0<=5'h0;

ad1<=5'h1;

ad2<=5'h2;

end

else

begin

ad0<=ad0+5'b1;

ad1<=ad1+5'b1;

ad2<=ad2+5'b1;

end

end

endmodule

module lab02\_alu(

input signed [31:0] alu\_a,

input signed [31:0] alu\_b,

input [4:0] alu\_op,

output reg sign,

output reg [31:0] alu\_out

);

parameter A\_NOP = 5'h00; //空运算

parameter A\_ADD = 5'h01; //符号加

parameter A\_SUB = 5'h02; //符号减

parameter A\_AND = 5'h03; //与

parameter A\_OR = 5'h04; //或

parameter A\_XOR = 5'h05; //异或

parameter A\_NOR = 5'h06; //或非

always@(\*)

begin

case(alu\_op)

A\_ADD:{sign,alu\_out}=alu\_a+alu\_b;

A\_SUB:{sign,alu\_out}=alu\_a-alu\_b;

A\_AND:alu\_out=alu\_a&alu\_b;

A\_OR:alu\_out=alu\_a|alu\_b;

A\_XOR:alu\_out=alu\_a^alu\_b;

A\_NOR:alu\_out=~(alu\_a|alu\_b);

A\_NOP:alu\_out=alu\_out;

default:alu\_out=alu\_out;

endcase

end

endmodule

module lab02\_reg(

input clk,

input rst\_n,

input [4:0] r1\_addr,

input [4:0] r2\_addr,

input [4:0] r3\_addr,

input [31:0] r3\_din,

input r3\_wr,

output reg [31:0] r1\_dout,

output reg [31:0] r2\_dout

);

reg [31:0] regfile [31:0];

integer r1\_add,r2\_add,r3\_add;

integer i;

always@(\*)

begin

r1\_add=16\*r1\_addr[4]+8\*r1\_addr[3]+4\*r1\_addr[2]+2\*r1\_addr[1]+r1\_addr[0];

r2\_add=16\*r2\_addr[4]+8\*r2\_addr[3]+4\*r2\_addr[2]+2\*r2\_addr[1]+r2\_addr[0];

r3\_add=16\*r3\_addr[4]+8\*r3\_addr[3]+4\*r3\_addr[2]+2\*r3\_addr[1]+r3\_addr[0];

end

always@(posedge clk or negedge rst\_n)

begin

if(~rst\_n)

begin

regfile[0]<=32'b1;

regfile[1]<=32'b1;

for(i=2;i<32;i=i+1)

regfile[i]<=32'b0;

end

else

if(r3\_wr)

regfile[r3\_add]<=r3\_din;

end

always@(negedge clk)

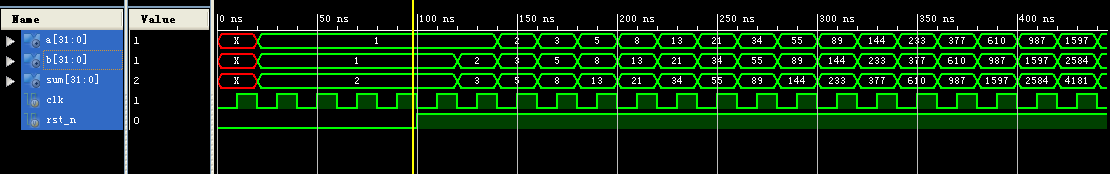
begin

r1\_dout<=regfile [r1\_add];

r2\_dout<=regfile [r2\_add];

end

endmodule

仿真图：