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：