计算机组成原理实验报告
LAB02

题目：寄存器文件

姓名：林祥

学号：PB16020923

实验目的

1、复习 Verilog 语法

2、知道如何用 Verilog 实现寄存器文件

实验内容

1、设计一 64*32bit 的寄存器文件，即 64 个 32 位的寄存器文件（寄存器组）

- 具备一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
- 寄存器的复位值可自行指定

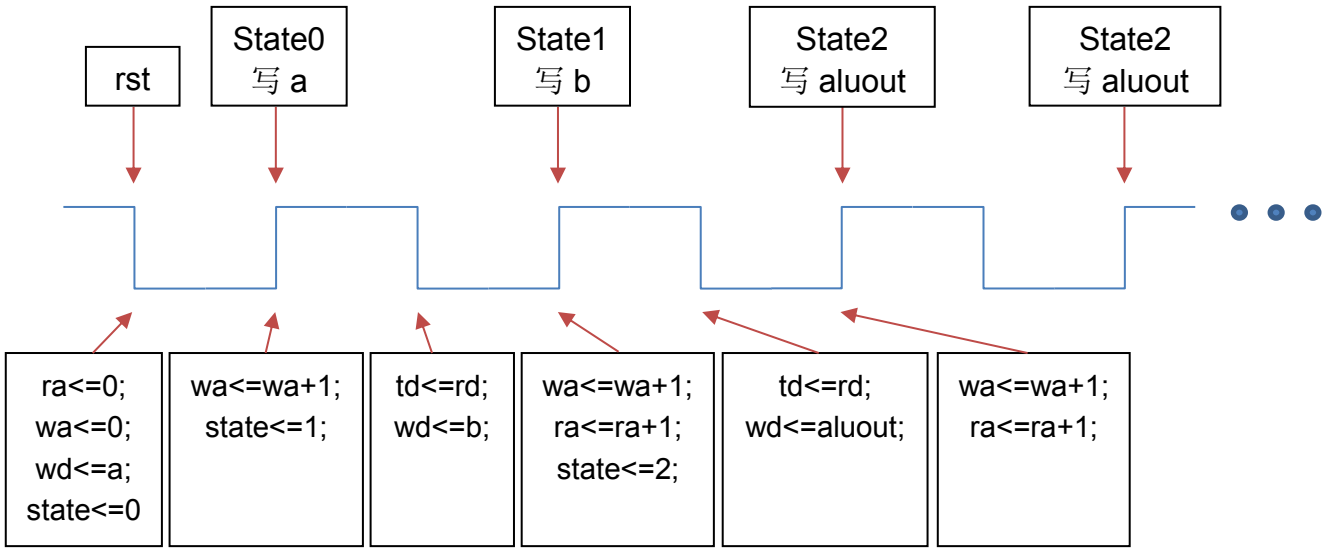
实验分析

1、模块化设计，alu 一个模块，regfile 一个模块，计算斐波那契数列一个 top 模块，初始两个数 a,b 从 top 的参数输入。

2、alu 模块使用 case 语句判断 7 种操作类型。

3、regfile 模块用组合逻辑读，时序逻辑写。

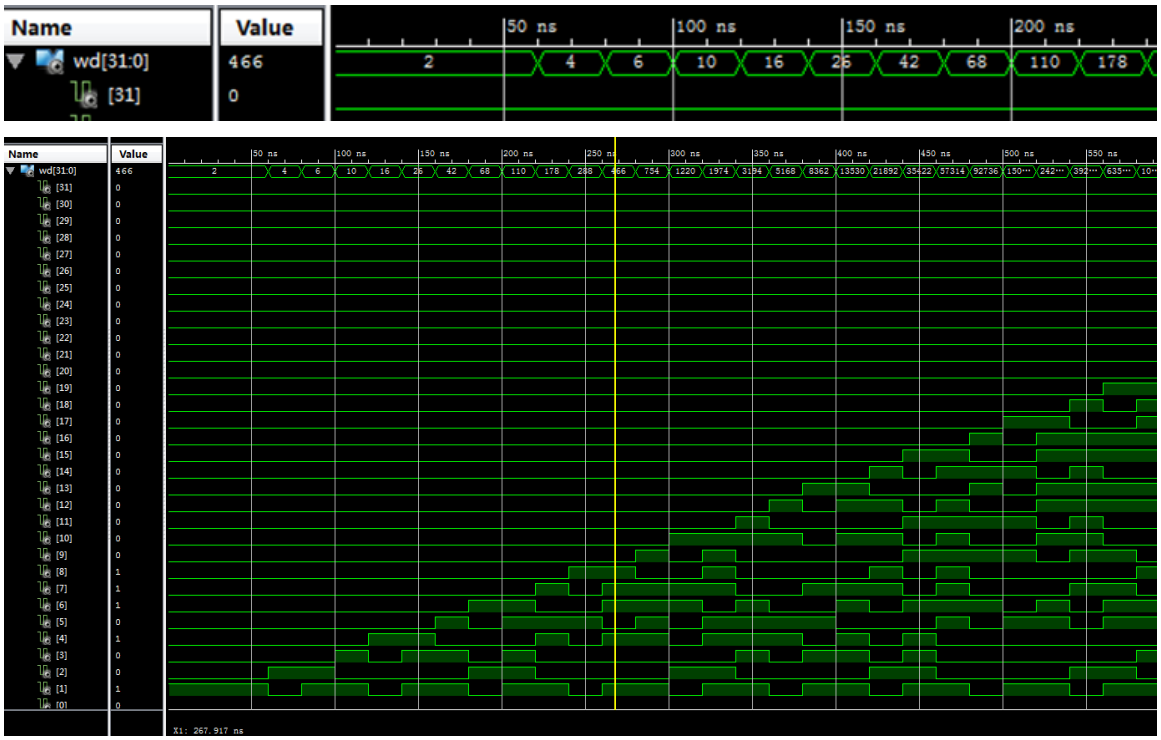
4、思路



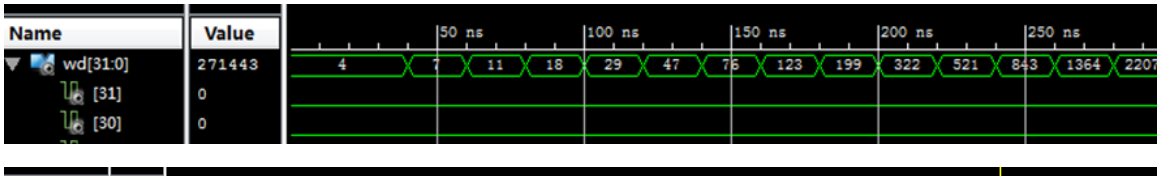
实验结果

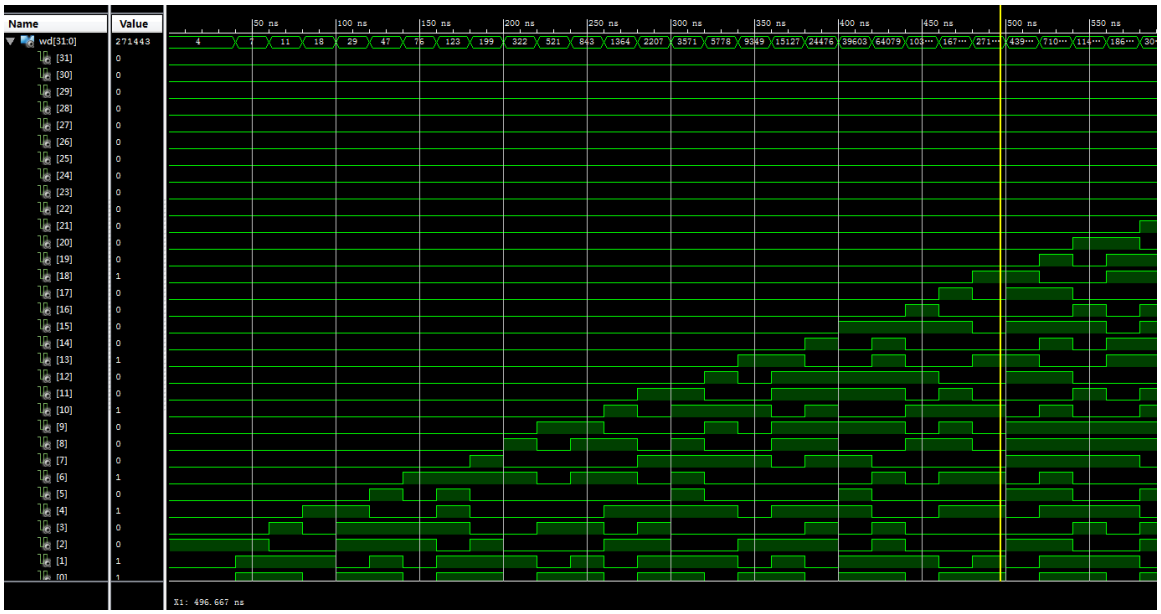
仿真结果（设 a，b 为斐波那契数列初始值，下面测试两组）

（1）Test 中输入 a=2，b=2 的结果



（2）Test 中输入 a=4，b=7 的结果





附录:

一、 模块源代码

top.v

```
module top(
    input clk,rst_n,
    input [31:0] a, //斐波那契数列第一个数
    input [31:0] b, //斐波那契数列第二个数
    output reg [31:0] wd //写入 regfile 的数据，同时作为数列输出
);

    reg[5:0] ra=6'd0;
    reg[5:0] wa=6'd0;

    wire [31:0] aluout;
    reg [1:0] state=2'b00;
    reg we=1'b1;

    reg [31:0] td;
    wire [31:0] rd;

    alu alu1(td,rd,5'h01,aluout);
    regfile regfile1(clk,rst_n,ra,wa,wd,we,rd);

    always@(negedge clk) //下降沿改变数据
        if(state==2'b00) //写 a 状态
            begin
                wd<=a;
            end
        else if(state==2'b01)//写 b 状态
            begin
                wd<=b;
                td<=rd;
            end
        else if(state==2'b10) //写 aluout 状态
            begin
                wd<=aluout;
                td<=rd;
            end

    always@(posedge clk or negedge rst_n) //这里上升沿改变地址（同时 regfile 会将 wd 写入，wd 已在上一个下降沿更新）
    begin
```

```
        if(~rst_n)//初始化
            begin
                wa<=6'd0;
                ra<=6'd0;
                state<=2'b00;
            end
        else
            begin
                if(state==2'b00)//写 a 状态，读地址不+1，使读地址落后写地址一周期
                    begin
                        state<=2'b01;
                    end
                else if(state==2'b01)//写 b 状态
                    begin
                        ra<=ra+6'd1;

                        state<=2'b10;
                    end
                else if(state==2'b10) //写 aluout 状态
                    begin
                        ra<=ra+6'd1;
                    end
                wa<=wa+6'd1;
            end
    end
endmodule
```

alu.v

```
parameter A_NOP =5'h00; //nop
parameter A_ADD =5'h01; //sign_add
parameter A_SUB =5'h02; //sign_sub
parameter A_AND =5'h03; //and
parameter A_OR  =5'h04; //or
parameter A_XOR =5'h05; //xor
parameter A_NOR =5'h06; //nor

module alu(
    input [31:0] alu_a,
    input [31:0] alu_b,
    input [4:0] alu_op,
    output reg [31:0] alu_out
    );
    always@(*)
```

```
        case (alu_op)
            A_NOP: alu_out = 0;
            A_ADD: alu_out = alu_a + alu_b;
            A_SUB: 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);
            default: alu_out = 0;
        endcase
endmodule
```

regfile.v

```
module regfile(
    input   clk,
    input rst_n,
    input [5:0] rAddr1,//读地址
    input [5:0] wAddr,//写地址
    input [31:0] wDin,//写数据
    input wEna,//写使能
    output [31:0] rDout1//读数据 1
);
    reg [31:0] data [0:63];
    integer i;
    assign rDout1=data[rAddr1];//读

    always@(posedge clk or rst_n)//写和复位
        if(~rst_n)
        begin
            for(i=0; i<64; i=i+1) data[i]<=0;
        end
        else
        begin
            if(wEna)
                data[wAddr]<=wDin;
        end
endmodule
```

test.v

```
module test(
    );
    reg clk,rst_n;
    reg [31:0] a;
    reg [31:0] b;

    wire [31:0] wd;


    top test(
        .clk(clk),
        .rst_n(rst_n),
        .a(a),
        .b(b),
        .wd(wd),
    );

    always #10 clk=~clk;
    initial begin
        clk=0;
        a=2;
        b=2;
        rst_n=0;
        #20;
        rst_n=1;
    end
endmodule
```