中山大学数据科学与计算机学院本科生实验报告

（2019 学年秋季学期）

课程名称：**计算机组成原理实验**　　任课教师：郭雪梅　　助教：丁文、汪庭葳

年级&班级	2019 级 04 班	专业(方向)	计算机科学与技术（超算方向）
学号	19335112	姓名	李钰
电话	19847352856	Email	1643589912@qq.com
开始日期	2020 年 11 月 13 日	完成日期	2020 年 11 月 13 日

一、实验题目

寄存器堆设计 ALU 实验

二、实验目的

1. 学习和使用 Verlilog HDL 进行电路的设计方法；

2. 掌握灵活的运用 Verilog HDL 进行各种描述与建模的技巧和方法；

3. 学习寄存器堆的数据传送与读写工作原理，掌握寄存器堆的设计方法。

三、实验内容

1. 实验步骤

（1）创建工程

（2）添加设计源文件

ALU 运算器模块

寄存器堆模块

顶层综合模块

（3） 添加仿真文件 进行仿真

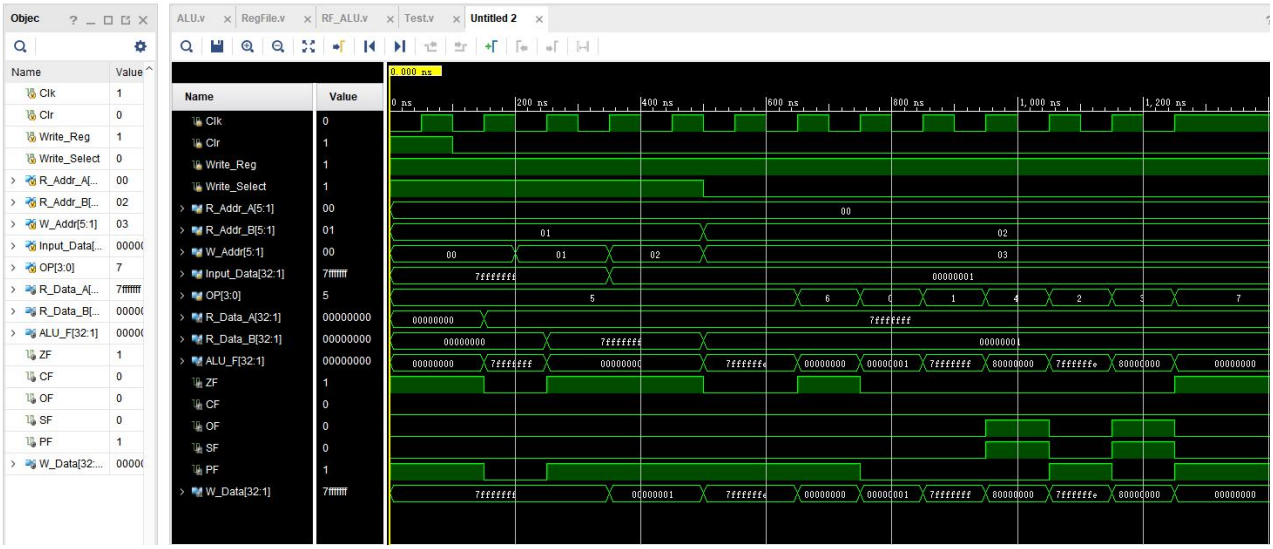
（4） 分析结果

2. 实验原理

ALU根据不同指令做出相应运算并储存在寄存器中

四、实验结果

仿真波形图



下面进行分析：

① OP = 5 -->做减法 ；

A = 00000000 ; B = 00000000

F = 0 ； ZF(0标志位) = 1

A = 7fffffff ; B = 00000000

F = 7fffffff

A = 7fffffff ; B = 7fffffff

F = 00000000 ZF= 1

② OP = 6 -->A < B ； A = 7fffffff ; B = 00000001

F = 00000000(因为A不小于B) ； ZF(0标志位) = 1

③ OP = 0 -->按位与 ； A = 7fffffff ; B = 00000001

F = 00000001 ； ZF(0标志位) = 0

④ OP = 1 -->按位或 ； A = 7fffffff ; B = 00000001

F = 7fffffff ； ZF(0标志位) = 0

⑤ OP = 4 -->做加法 ； A = 7fffffff ; B = 00000001

F = 80000000 ； ZF(0标志位) = 0 OF = 1, SF = 1

⑥ OP = 2 -->按位异或; A = 7fffffff ; B = 00000001

F = 7ffffffe ； ZF(0标志位) = 0

⑦ OP = 3 -->按位或非 ； A = 7fffffff ; B = 00000001

F = 80000000 ； ZF(0标志位) = 0 OF = 1, SF = 1

⑧ OP = 7 -->将B左移A位 ； A = 7fffffff ; B = 00000001

F = 00000000 ； ZF(0标志位) = 1

五、实验感想

通过这次学习，我学习掌握了寄存器堆的数据传送与读写工作原理，掌握寄存器堆的设计方法。更加熟悉了 vivado 软件以及 verilog 编程语言。

附录（流程图，注释过的代码）:

ALU 运算器模块

```
`timescale 1ns / 1ps

module ALU(OP,A,B,F,ZF,CF,OF,SF,PF

    );
    parameter SIZE = 32;//运算位数
        input [3:0] OP;//运算操作
        input [SIZE:1] A;//左运算数
        input [SIZE:1] B;//右运算数
        output [SIZE:1] F;//运算结果
        output          ZF, //0 标志位, 运算结果为 0(全零)则置 1, 否则置 0
                        CF, //进借位标志位, 取最高位进位 C,加法时 C=1 则 CF=1 表示有进位,减法时 C=0
```

```
则 CF=1 表示有借位
                    OF, //溢出标志位，对有符号数运算有意义，溢出则 OF=1，否则为 0
                    SF, //符号标志位，与 F 的最高位相同
                    PF; //奇偶标志位，F 有奇数个 1，则 PF=1，否则为 0
        reg [SIZE:1] F;
        reg C,ZF,CF,OF,SF,PF;//C 为最高位进位
         always@(*)
         begin
              C=0;
              case(OP)
                   4'b0000:begin F=A&B; end       //按位与
                   4'b0001:begin F=A|B; end       //按位或
                   4'b0010:begin F=A^B; end       //按位异或
                   4'b0011:begin F=~(A|B); end //按位或非
                   4'b0100:begin {C,F}=A+B; end //加法
                   4'b0101:begin {C,F}=A-B; end //减法
                   4'b0110:begin F=A<B; end//A<B 则 F=1，否则 F=0
                   4'b0111:begin F=B<<A; end     //将 B 左移 A 位
              endcase
              ZF = F==0;//F 全为 0，则 ZF=1
              CF = C; //进位借位标志
              OF = A[SIZE]^B[SIZE]^F[SIZE]^C;//溢出标志公式
              SF = F[SIZE];//符号标志,取 F 的最高位
              PF = ~^F;//奇偶标志，F 有奇数个 1，则 F=1；偶数个 1，则 F=0
         end

Endmodule
```

寄存器堆模块

```
`timescale 1ns / 1ps//寄存器堆模块
module RegFile(
Clk,Clr,Write_Reg,R_Addr_A,R_Addr_B,W_Addr,W_Data,R_Data_A,R_Data_B
);
     parameter ADDR = 5;//寄存器编码/地址位宽
     parameter NUMB = 1<<ADDR;//寄存器个数
     parameter SIZE = 32;//寄存器数据位宽
     input Clk;//写入时钟信号
     input Clr;//清零信号
     input Write_Reg;//写控制信号
     input [ADDR:1]R_Addr_A;//A 端口读寄存器地址
     input [ADDR:1]R_Addr_B;//B 端口读寄存器地址
     input [ADDR:1]W_Addr;//写寄存器地址
     input [SIZE:1]W_Data;//写入数据
     output [SIZE:1]R_Data_A;//A 端口读出数据
     output [SIZE:1]R_Data_B;//B 端口读出数据
```

```
    reg [SIZE:1]REG_Files[0:NUMB-1];//寄存器堆本体  第一个设置每一个寄存器的宽度,然后寄存器堆的名字以及个数
    integer i;//用于遍历 NUMB 个寄存器
        initial//初始化 NUMB 个寄存器，全为 0
        for(i=0;i<NUMB;i=i+1)
REG_Files[i]<=i;
    always@(posedge Clk or posedge Clr)//时钟信号或清零信号上升沿
    begin
        if(Clr)//清零
                for(i=0;i<NUMB;i=i+1)
                    REG_Files[i]<=0;//看这个赋值
        else//不清零,检测写控制，高电平则写入寄存器
                if(Write_Reg)
                    REG_Files[W_Addr]<=W_Data;
   end          //读操作没有使能或时钟信号控制，使用数据流建模(组合逻辑电路,读不需要时钟控制)
    assign R_Data_A=REG_Files[R_Addr_A];
    assign R_Data_B=REG_Files[R_Addr_B];
endmodule
```

顶层综合模块

```
`timescale 1ns / 1ps
module RF_ALU(
Clk,Clr,Write_Reg,Write_Select,//控制信号
 R_Addr_A,R_Addr_B,W_Addr,//读写地址
 Input_Data,R_Data_A,R_Data_B,//数据 IO
 OP,ZF,CF,OF,SF,PF,ALU_F//ALU 运算
  ,W_Data );
        parameter ADDR = 5;//地址位宽
        parameter SIZE = 32;//数据位宽          //寄存器堆
        input Clk, Clr;//写入时钟信号，清零信号
        input Write_Reg;//写控制信号
        input [ADDR:1]R_Addr_A;//A 读端口寄存器地址
        input [ADDR:1]R_Addr_B;//B 读端口寄存器地址
        input [ADDR:1]W_Addr;//写寄存器地址
        input [SIZE:1]Input_Data;//外部输入数据

        output [SIZE:1]R_Data_A;//A 端口读出数据
        output [SIZE:1]R_Data_B;//B 端口读出数据          //ALU
        input [3:0] OP;//运算符编码
        output ZF,//零标志
               CF,//进借位标志(只对无符号数运算有意义)
                OF,//溢出标志(只对有符号数运算有意义)
               SF,//符号标志(只对有符号数运算有意义)
                PF;//奇偶标志
       output [SIZE:1] ALU_F;//运算结果 F
```

```
    wire [SIZE:1]ALU_F;//ALU 运算结果中间变量
    input wire Write_Select;//写入数据选择信号

    output   reg [SIZE:1]W_Data;//写入数据        //Write_Select 高电平则写外部输入，否则写运算结果
     always@(*)//*表示总是发生
               begin
   case (Write_Select)
     1'b0: W_Data=ALU_F;
     1'b1: W_Data=Input_Data;
     endcase
     end
     RegFile RF_Test(              //输入
  .Clk(Clk),//时钟信号
  .Clr(Clr),//清零信号
  .Write_Reg(Write_Reg),//写入控制
  .R_Addr_A(R_Addr_A),//A 端口读地址
   .R_Addr_B(R_Addr_B),//B 端口读地址
  .W_Addr(W_Addr),//写入地址
   .W_Data(W_Data),//写入数据, 由外部或 ALU 输入                    //输出
  .R_Data_A(R_Data_A),//A 端口读出数据
  .R_Data_B(R_Data_B)//B 端口读出数据
  );                    //实例化 ALU 模块

   ALU ALU_Test(      //输入
  .OP(OP),//运算符
  .A(R_Data_A),//从寄存器读 A 操作数
   .B(R_Data_B),//从寄存器读 B 操作数
   .F(ALU_F),//ALU_F 作为中间变量暂存运算结果, 与 Input_Data 选择输入寄存器                    //输出
   .ZF(ZF),//零标志
   .CF(CF),//进借位标志(只对无符号数运算有意义)
  .OF(OF),//溢出标志(只对有符号数运算有意义)
  .SF(SF),//符号标志(只对有符号数运算有意义)
   .PF(PF)//奇偶标志
  );

 endmodule
```

仿真

```
`timescale 1ns / 1p
module Test(

    );
    reg    Clk, Clr, Write_Reg;
    reg Write_Select;
    reg [5:1] R_Addr_A ,R_Addr_B, W_Addr;
    reg [32:1]    Input_Data;
```

```
    reg [3:0] OP;
    wire [32:1] R_Data_A, R_Data_B, ALU_F;
    wire ZF,CF,OF,SF,PF;
    wire [32:1]    W_Data;
    initial
        begin
            Clr=0;//不清零
            Write_Reg=1;//写进去之后读读看
            R_Addr_A=5'b00000;//A 端口先读寄存器 0
            R_Addr_B=5'b00001;//B 端口先读寄存器 1
            OP=4'b0101;//先做一个减法运算
            Write_Select=1'b1;//先写外部输入  //将 32'h7fff_fffd 写入寄存器 0，作为左操作数
            Input_Data=32'h7fff_ffff;
            W_Addr=5'b00000;
       end
    initial begin
    Clr=1;
    Clk=0;#50;Clk=1;#50; //寄存器清零
    Clr=0;
    Clk=0;#50;Clk=1;#50;                    //将 32'h7fff_fffd 写入寄存器 0，作为右操作数
    Input_Data=32'h7fff_ffff;
    W_Addr=5'b00001;
    Clk=0;#50;Clk=1;#50;Clk=0;#50
    Input_Data=32'h1;//1 寄存器写入 7fffffff
    W_Addr=5'b00010;Clk=1;#50;Clk=0;//2 寄存器写入 1
    #50;Clk=1;
    #50;
    Write_Select=1'b0;//改为写入运算结果
    R_Addr_B=5'b00010;//B 端口改为读寄存器 2
            //将 ALU 的运算结果写入寄存器，并读读看
    W_Addr=5'b00011;
    Clk=0;#50;
    Clk=1;#50;                    //到此为止我们完成了一次减法运算          //（其实不是（ALU 实时
运算），实际上 ALU 已经做了 5 次减法）          //左右操作数跟运算结果依次写入了寄存器 0,1,2
    Clk=0;#50;          OP=4'b0110; Clk=1;#50;
    Clk=0;#50;          OP=4'b0000; Clk=1;#50;
    Clk=0;#50;          OP=4'b0001; Clk=1;#50;
    Clk=0;#50;          OP=4'b0100; Clk=1;#50;
    Clk=0;#50;          OP=4'b0010; Clk=1;#50;
    Clk=0;#50;          OP=4'b0011; Clk=1;#50;
    Clk=0;#50;          OP=4'b0111; Clk=1;#50;
end         //实例化寄存器堆模块

    RF_ALU RF_Test(
            .Clk(Clk),
            .Clr(Clr),
```

```
        .Write_Reg(Write_Reg),
        .Write_Select(Write_Select),
        .R_Addr_A(R_Addr_A),
        .R_Addr_B(R_Addr_B),
        .W_Addr(W_Addr),
        .Input_Data(Input_Data),
        .R_Data_A(R_Data_A),
        .R_Data_B(R_Data_B),
        .OP(OP),
        .ZF(ZF),
        .CF(CF),
        .OF(OF),
        .SF(SF),
        .PF(PF),
        .ALU_F(ALU_F),
        .W_Data(W_Data)

    );

endmodule
```