实验一报告

07152001 班 1120201198 史桎彬

实验内容

设计并实现一个 ALU。

实验环境

ASUS VivoBook + Windows10 + Vivado2019.2，语言为 Verilog HDL。

实验要求

1. 支持至少 8 种运算
2. 输出 5 个标志符号
3. 支持左右移位操作
4. 可支持至少两种舍入操作

实验过程

1. 顶层设计

- 输入

变量名	具体含义	位宽
reset	初始化置零位	1 bit
cut	舍入方式位	1 bit
in0	第一个操作数	32 bit
in1	第二个操作数	32 bit
op	操作符	11 bit

- 输出

变量名	具体含义	位宽
out	运算结果	32 bit
overflow	溢出标志位	1 bit
carryout	进借位标志位	1 bit
zero	零标志位	1 bit
parity	奇偶校验位	1 bit
signal	符号标志位	1 bit

2. 运算操作

运算	OP 编码	具体含义
add	00000100000	有符号数加法
sub	00000100001	有符号数减法
and	00000100010	按位与
or	00000100011	按位或
xor	00000100100	异或
nor	00000100101	或非
equ	00000100110	有符号数相等运算

com	00000100111	有符号数比较运算

3. 移位操作

运算	OP 编码	具体含义
sllv	00000000000	算数左移
srlv	00000000010	算数右移
srav	00000000011	逻辑右移

4. 舍入操作

cut 值	具体含义
0	恒舍
1	恒置 1

注：设计过程中，将不同类型的舍入操作包含在算术右移操作中。

5. 设计代码

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: Beijing Institute of Technology
// Engineer: Yabin Shi
// Create Date: 2022/12/24 17:39:50
//////////////////////////////////////////////////////////////////////////////////
module mine(
 reset, in0, in1, op, cut, out, overflow, zero, carryout, parity, signal
    );
    input reset;                //用于初始化置零
    input[31:0] in0,in1;        //操作数
    input[10:0] op;             //操作运算符
    input cut;
    output[31:0] out;           //运算结果
    output overflow,zero,carryout,parity,signal;  //溢出判断位、零值判断位、进借位判断位、奇偶校验位、符号位
    reg[31:0] out;              //标明为寄存器类型变量
    reg overflow,zero,carryout,parity,signal;//标明为寄存器类型变量
always@(*)                      //使用 always 语句进行运算
begin
     if(reset)              //判断 reset 值，为 1 进行初始化，为 0 进行 ALU 运算
```

```
        begin
         out=0;
         overflow=0;
         zero=0;
         carryout=0;
         parity=0;
         signal=0;
        end
    else
        alutask( in0, in1, op, cut, out, overflow, zero, carryout, parity, signal);
//把具体运算功能模块封装成一个任务
end
task alutask;                    //运算任务定义
   input[31:0] in0,in1;
   input[10:0] op;
   input cut;
   output[31:0] out;
   output overflow,zero,carryout,parity,signal;
   reg[31:0] out;
   reg tmp,pmt,overflow,zero,carryout,parity,signal;

begin
            overflow=0;             //每次进行运算前，标志位置 0
            carryout=0;
            zero=0;
            parity=0;
            signal=0;
            case( op )
            11'b00000100000://有符号数加法
                            begin
                            {tmp,out}=in0+in1;
                            end
            11'b00000100001://有符号数减法
                            begin
                            {tmp,out}=in0-in1;
                            end
            11'b00000100010: out=in0&in1;//按位与
            11'b00000100011: out=in0|in1;//按位或
            11'b00000100100: out=in0^in1;//异或
            11'b00000100101: out=~(in0|in1);//或非
            11'b00000100110: out=( $signed(in0)==$signed(in1) )? 1:0;//有符号数
相等运算
            11'b00000100111: out=( $signed(in0)>$signed(in1) )? 1:0;//有符号数比
较运算
```

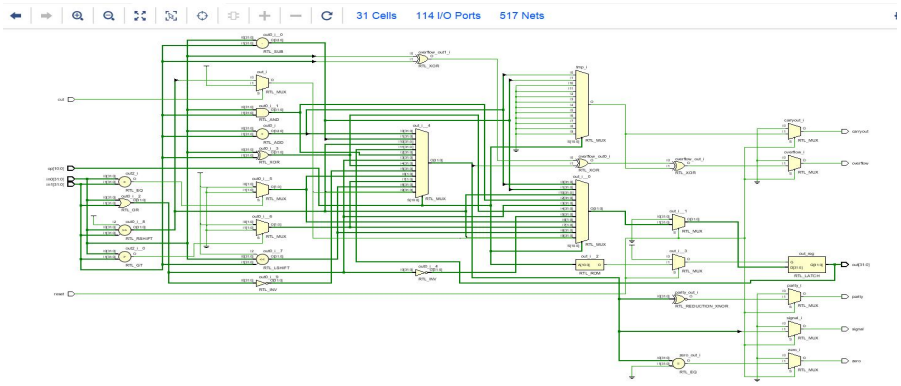
```
            11'b00000000000: out=in0<<in1;
            11'b00000000010:
                            begin
                            out=in0>>in1;
                            case( cut )
                            1'b0://恒舍
                            out[0]=out[0];
                            1'b1://恒置 1
                            out[0]=1;
                            endcase
                            end
            11'b00000000011: out=in0>>>in1;

            endcase
            zero=out==0;          //zero 通过直接判断 out 是否为 0
            carryout=tmp;
            overflow=in0[31]^in1[31]^out[31]^tmp;
            signal=out[31];
            parity=~^out;
end
endtask
endmodule
```

6. 仿真文件

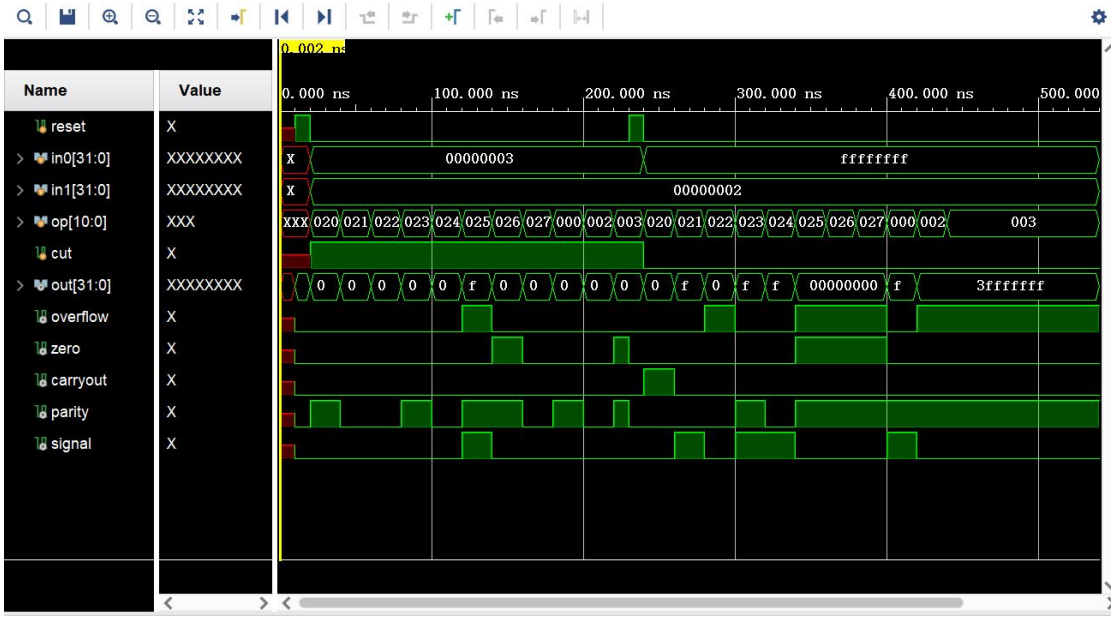
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: Beijing Institute of Technology
// Engineer: Yabin Shi
// Create Date: 2022/12/24 17:39:50
//////////////////////////////////////////////////////////////////////////////////
    module mine1;
    reg reset;
    reg [31:0] in0,in1;
    reg [10:0] op;
    reg cut;
    wire [31:0] out;
    wire overflow,zero,carryout,parity,signal;

    mine unit(            //模块实例化
        .reset(reset),
        .in0(in0),
        .in1(in1),
        .op(op),
```

```
    .cut(cut),
    .out(out),
    .overflow(overflow),
    .zero(zero),
    .carryout(carryout),
    .parity(parity),
    .signal(signal)
);
initial
begin
    #10 reset=1;
    #10 reset=0;in0=32'd3;in1=32'd2;cut=1'b1;
    for(op=11'b00000100000;op<11'b00000100111;op=op+1)
    #20;
    #20 op=11'b00000000000;
    #20 op=11'b00000000010;
    #20 op=11'b00000000011;

    #10 reset=1;
    #10 reset=0;in0=-32'd1;in1=32'd2;cut=1'b0;
    for(op=11'b00000100000;op<11'b00000100111;op=op+1)
    #20;
    #20 op=11'b00000000000;
    #20 op=11'b00000000010;
    #20 op=11'b00000000011;

    #100 $finish;
end
initial
$monitor ($time,,,"reset=%b in0=%b in1=%b op=%b cut=%b out=%b overflow=%b
zero=%b carryout=%b parity=%b signal=%b",
reset,in0,in1,op,cut,out,overflow,zero,carryout,parity,signal);
endmodule
```

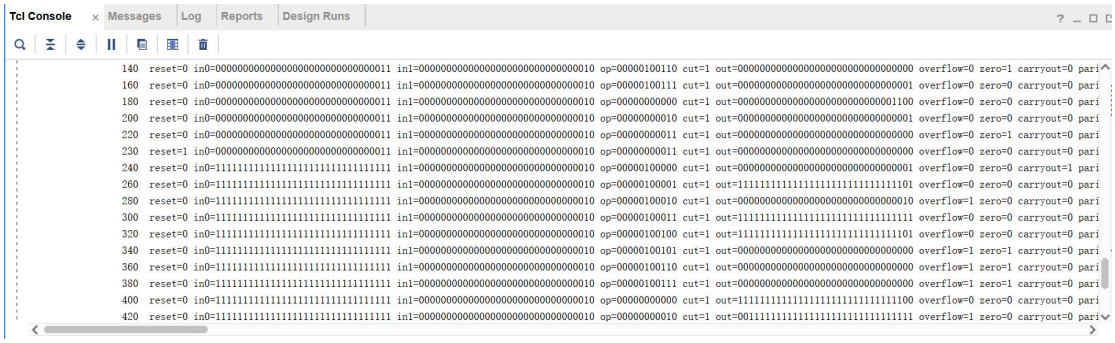
7. 电路图



8. 仿真波形图



9. Monitor 监视器结果



实验心得

在本次实验过程中，我将书本中学到的计算机组成原理和体系结构内容进行了实践，基本掌握了 Vivado 的使用及设计代码、仿真代码的书写，实现了知识的沉淀、巩固，本次实验让我收获颇丰。

实验期间，我遇到了许多难题，花费了大量时间在学习软件使用及编程规范上，所幸最终我成功完成了实验。其中，安徽大学刘峰老师在 Bilibili 平台发布的“Verilog 的仿真代码和约束文件的编写”视频对我帮助尤大，这也是网络上为数不多的专门介绍仿真代码编写的课程，在此特加推荐与感谢。