1. **实验目的**

学习和使用Verlilog HDL 进行电路的设计方法

掌握灵活的运用Verilog HDL 进行各种描述与建模的技巧和方法

学习寄存器堆的数据传送与读写工作原理，掌握寄存器读写的设计方法

1. **实验内容**

根据实验原理设计32个32位的寄存器堆，有俩个读端口，1个写端口，既能同时读出两个寄存器的值，写入一个寄存器，设计寄存器只需要定义一个reg类型的数组即可完成，利用数组下标操作寄存器即可，同时也有Rest 和CLk 信号控制我们的操作的进行。8位的LED进行显示我们读出的数据的一部分！或用数码管显示读出的数据。

**三、实验平台：硬件平台：Basys 3       软件平台：Vivado**

**四、实验步骤**

**（1）. 创建工程**

**工程名为RF\_ALU， 选择保存文件夹**

**点击next，并选择RTL Project，点击next，直到出现选择与basys 3相应的内核，选**xc7a35tcpg236-1

（2）添加设计源文件

a. ALU运算器模块代码：

`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

**b.寄存器堆模块代码**：

`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

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]<=0;

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;

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

**c. 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,F//ALU运算

);

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] F;//运算结果F

input Write\_Select;//写入数据选择信号

wire [SIZE:1]ALU\_F;//ALU运算结果中间变量

wire [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

begin

assign W\_Data=ALU\_F;

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

(3) 测试模块代码：

添加仿真源文件

`timescale 1ns / 1ps

module Test(

);

reg Clk, Clr, Write\_Reg;

reg Write\_Select;

reg [4:0] R\_Addr\_A ,R\_Addr\_B, W\_Addr;

reg [31:0] Input\_Data;

reg [3:0] OP;

wire [31:0] R\_Data\_A, R\_Data\_B, F;

wire ZF,CF,OF,SF,PF;

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\_fffd;

W\_Addr=5'b00000;

Clk=0;#50;Clk=1;#50; //将32'h7fff\_ffff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; 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'b00010;

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

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),

.F(F)

);

Endmodule

实验报告

仿真波形图：仿真电路图：心得体会：

**拓展**

1. 仿真加入所有运算的仿真（所提供文件中只有减法）。
2. ALU模块加如果输入移位次数:

input [4:0] sa

改动左移操作程序，并加入右移和lui指令。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| R型指令 | | | | | | | |
| 指令 | [31:26] | [25:21] | [20:16] | [15:11] | [10: 6] | [5:0] | 功能 |
| Add | 000000 | rs | rt | rd | 000000 | 100000 | 寄存器加 |
| Sub | 000000 | rs | rt | rd | 000000 | 100010 | 寄存器减 |
| And | 000000 | rs | rt | rd | 000000 | 100100 | 寄存器与 |
| Or | 000000 | rs | rt | rd | 000000 | 100101 | 寄存器或 |
| Xor | 000000 | rs | rt | rd | 000000 | 100110 | 寄存器异或 |
| Sll | 000000 | 00000 | rt | rd | sa | 000000 | 左移 |
| Srl | 000000 | 00000 | rt | rd | sa | 000010 | 逻辑右移 |
| Sra | 000000 | 00000 | rt | rd | sa | 000011 | 算术右移 |
| Lui | 001111 | 00000 | rt | immediate | 设置高位 |  |  |

表：ALU功能真值表

|  |  |
| --- | --- |
| ALU\_Operation | 功能描述 |
| 0100 | Add（加） |
| 0101 | Sub（减） |
| 0000 | And（与） |
| 0001 | Or（或） |
| 0010 | Xor（异或） |
| 0110 | SLT小于设置 |
| 0011 | Lui（设置高位） |
| 0111 | Sll（左移） |
| 1000 | Srl（右移） |
| 1001 | Sra（算术右移） |

