上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

计算机系统结构实验报告

实验名称：简单的类 MIPS 单周期处理器部件实现-寄存器与存储器

姓　　名：洪瑄锐

学　　号：517030910227

班　　级：F1703302

手　　机：15248246044

邮　　箱：1204378645@qq.com

目录

1. 实验目的

1) 理解 CPU 寄存器与存储器；

2) egister 的实现；

3) Data Memory 的实现；

4) 有符号扩展的实现；

5) 使用行为仿真。

2. 实验原理与代码实现

2.1 寄存器 Register 的实现

2.1.1 主要思想

MIPS 中一共有 32 个 32 位的寄存器，因此需要设置 reg [31:0] regSets [31:0]，类似二维数组。

该模块共有 7 个输入，分别为 readReg1（所读寄存器 1 编号），readReg2（所读寄存器 2 编号），writeReg（所写寄存器编号），writeData（将要向寄存器内写入的数值），regWrite（是否要向寄存器中写入）。

共有 2 个输出，readData1（读寄存器 1 的数值），readData2（读寄存器 2 的数值）。

读寄存器时只要给出 readReg1 和 readReg2 即可，写寄存器时需要 clk 下跳沿、允许写入信号 regWrite，所写寄存器编号 writeReg，所写值 writeData 同时存在。

2.1.2 Register 模块代码

```
module Registers(
    input clk,
    input [25:21] readReg1,
    input [20:16] readReg2,
    input [4:0] writeReg,
    input [31:0] writeData,
    input regWrite,
    output reg [31:0] readData1,
    output reg [31:0] readData2
    );

    reg [31:0] regSets[31:0];

    always @ (readReg1 or readReg2 or writeReg)
        begin
            readData1 = regSets[readReg1];
            readData2 = regSets[readReg2];
        end

     always @ (negedge clk)
        begin
            if(regWrite==1)
                regSets[writeReg] = writeData;
        end
endmodule
```

2.1.3 仿真测试

通过给 regWrite、WriteData、WriteReg、readReg1、readReg2 赋值观察 readData1 和 readData2 是否符合正确输出。

部分仿真代码如下所示：

```
initial begin
    Clk = 1'b0;
    ReadReg1 = 0;
    ReadReg2 = 0;
    WriteReg = 0;
    WriteData = 32'b00000000000000000000000000000000;
    RegWrite = 1'b0;

    #200;
    RegWrite = 1'b1;
    WriteReg = 5'b00001;
    WriteData = 32'b11111111111111110000000000000000;

    #200;
    RegWrite = 1'b0;
    WriteReg = 5'b00010;
    WriteData = 32'b00000000000000001111111111111111;

    #200
     ReadReg1 = 5'b00010;
     ReadReg2 = 5'b00001;

    #200;
    RegWrite = 1'b1;
    WriteReg = 5'b00000;
    WriteData = 32'b11111111111111111111111111111111;

    #200;
    ReadReg1 = 5'b00001;
    ReadReg2 = 5'b00000;
end
endmodule
```

仿真结果：

向 0 号寄存器写入，但 regWrite 为 0 即不允许写入，向 1 号寄存器写入，并且 regWrite 为 1 即允许写入，readData1 读入 0 号寄存器，readData2 读入 1 号寄存器，可以看出上图 readData1 仍为未知数 x，readData1 读入 ffff0000。再向 0 号寄存器写入 ffffffff 且 writeReg 为 1，使 readData1 读入 1 号寄存器，readData2 读入 0 号寄存器，则如上图所示 readData1 为 ffff0000，readData2 为 ffffffff。仿真结果正确。

2.2 内存单元模块 Memory 的实现

2.2.1 主要思想

内存单元模块与 register 类似，由于写数据也要考虑信号同步，因此也需要时钟。

共 4 个输入，address（写入或读出的内存地址），writeData（将要写入内存的数据），

memWrite（是否允许写数据信号），memRead（是否允许读数据信号）。

共 1 个输出，readData（所读数据）。

2.2.2 Memory 模块代码

```
module dataMemory(
    input clk,
    input [31:0] address,
    input [31:0] writeData,
    input memWrite,
    input memRead,
    output reg [31:0] readData
    );

    reg [31:0] memSets[63:0];

    always @(address or memRead or clk)
        begin
            if(memRead)
                readData = memSets[address];
        end

     always @(negedge clk)
        begin
            if(memWrite)
                memSets[address] = writeData;
        end

endmodule
```

2.2.3 仿真测试

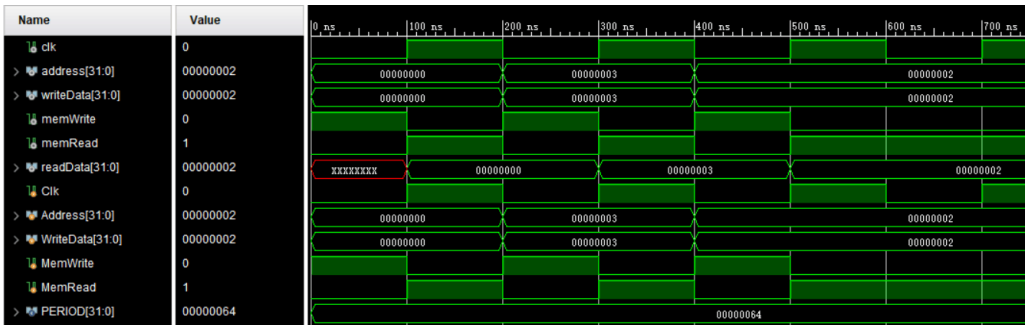
通过给 memread，memwrite，address，writeData 赋值检查 readData。本次仿真采用先写再读的方式。

部分代码：

```
initial begin
    Clk = 1'b0;
    Address = 32'b00000000000000000000000000000000;
    WriteData = 32'b00000000000000000000000000000000;
    MemWrite = 1'b1;
    MemRead = 1'b0;

     #100
     MemWrite = 0;//读，不写
     MemRead = 1'b1;

    #100;
    MemWrite = 1'b1;//写，不读
    MemRead = 1'b0;
    Address = 32'b00000000000000000000000000000011;
    WriteData = 32'b00000000000000000000000000000011;

     #100
     MemWrite = 0;//读，不写
     MemRead = 1'b1;

    #100
    MemWrite = 1'b1;//写。不读
    MemRead = 1'b0;
    Address = 32'b00000000000000000000000000000010;
    WriteData = 32'b00000000000000000000000000000010;


     #100
     MemWrite = 0;//读，不写
     MemRead = 1'b1;


end
endmodule
```

仿真结果：

第 100nsreadData 开始读值，后每 200ns 变化一次。分析可知仿真结果正确。



2.3 带符号扩展的实现

2.3.1 主要思想

将 16 位有符号数扩展为 32 位有符号数，采用 ifelse 语句，如果 16 位有符号数最高位为 0，则在其前补 16 个 0，否则补 16 个 1。

2.3.2 带符号扩展模块代码

```
module signext(
    input [15:0] init,
    output reg [31:0] data
    );

    always @(init)
    begin
    if(init[15]== 1'b0)
        begin
            data = 32'b00000000000000000000000000000000;
            data [15:0] = init;
         end
     else
        begin
            data = 32'b11111111111111111111111111111111;
            data [15:0] = init;
        end
     end

endmodule
```

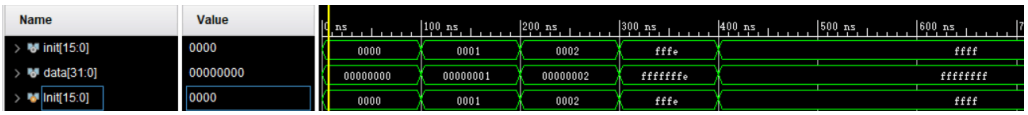
2.3.3 仿真测试

通过对 init 赋最高位不同的值得到正确的扩展数 data。

部分代码：

```
initial
begin
    Init = 0;

    #100 Init = 1;

    #100 Init = 2;

    #100 Init = 16'b1111111111111110;

    #100 Init = 16'b1111111111111111;
end
endmodule
```

仿真结果：



分析知仿真正确。

3. 感想

在本次实验中我实现了简单 cpu 中的寄存器模块，内存模块和带符号扩展，深入理解了寄存器和内存模块的相关信号是如何起作用，知道了其内部工作原理。另外本次实验使我对 verilog 的基础语法使用的更加熟练。

感谢老师和同学们在本次实验中对我的教导和帮助。

4. 参考文献

《cs145 实验指导书 lab04》