计算机系统结构实验报告

Lab04

简单的类 MIPS 单周期处理器部件实现——寄存器、存储器与有符号扩展

姓　　名：　徐薪

学　　号：　519021910726

日　　期：　2021 年 6 月 09 日

摘 要

本实验要求在 Vivado 集成设计环境下实现简单的类 MIPS 单周期处理器部件寄存器（Register）、存储器（Data Memory）以及有符号扩展（Signext）。他们分别用于暂时存放 ALU 的计算结果和一些数据，储存大量的内存数据以及对立即数进行带符号扩展。本实验通过软件仿真的方式进行结果验证。

目录

1. 实验概述

1.1 实验内容

本实验要求在 Vivado 集成设计环境下实现简单的类 MIPS 单周期处理器部件寄存器（Register）、存储器（Data Memory）以及有符号扩展(Signext)。他们分别用于暂时存放 ALU 的计算结果和一些数据，储存大量的内存数据以及对立即数进行带符号扩展。本实验通过软件仿真的方式进行结果验证。

1.2 实验目的

（1）理解 CPU 的寄存器、存储器、有符号扩展；

（2）寄存器 Register 的实现；

（3）存储器 Data Memory 的实现；

（4）有符号扩展 Signext 的实现；

（5）使用功能仿真。

2. 原理分析

2.1 寄存器 Register 的设计

寄存器（Register）的功能是暂时存储一些数据以及中间运算结果，其主要接口如图一所示。寄存器单元内部一共有 32 个寄存器，所以需要五位的二进制数来确定选中的目标寄存器。由于 MIPS 处理器的存储数据都是 32 位的，所以寄存器输入输出的 data bus 都是 32 位的。除此之外，寄存器还有两个控制信号：时钟 CLK 和写操作 RegWrite。其中，CLK 信号用于控制寄存器的读写，我们要求寄存器

在 CLK 发生变化时都能读，在时钟下跳沿写，这样设计能很好地解决多周期 MIPS 处理器不同阶段的寄存器读写冲突问题；RegWrite 信号控制是否进行写入寄存器的操作。

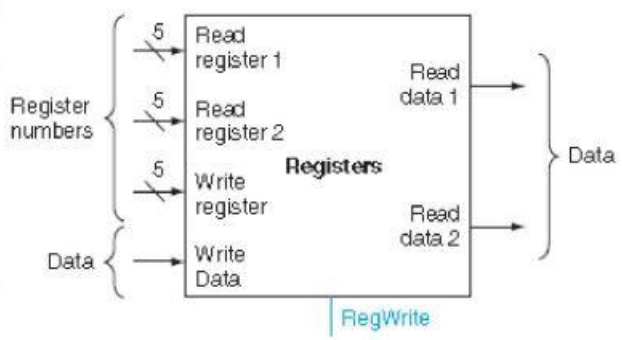

图 1. 寄存器（Register）的图示

2.2 存储器模块 Data Memory

存储器（Data Memory）的功能是存储大量内存数据并支持数据的读写，其示意图如图 2 所示。MIPS 处理器能够支持 32 位的数据地址单元，所以 address 接口是 32 位的 input wire。由于 MIPS 支持存储的内存数据的大小是 32 bit，所以 Write data 输入接口和 Read data 输出接口都是 32 位的。除此之外，存储器还有三个控制信号：CLK、MemWrite 以及 MemRead。其中，时钟 CLK 上升沿支持存储器读，下跳沿支持存储器写，这样通过硬件的方式，能够解决 MIPS 多周期处理器在不同阶段同时用到存储器的冒险冲突问题。MemWrite 信号控制数据是否写入存储器，MemRead 信号控制是否从存储器中读取数据。

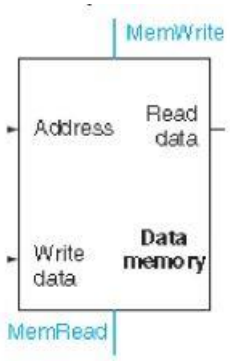

图 2. 存储器（Data Memory）的图示

2.3 有符号扩展单元 Signext

有符号扩展单元（Signext）的功能是将指令中的 16 位有符号立即数扩展成为 32 位有符号立即数。由于是带符号扩展，所以只需要将 16 位有符号立即数的符号位扩展到 32 位有符号立即数的高 16 位即可。这个功能可以用数字逻辑运算来进行实现，在最终的代码实现过程中，我将它设计为了一个 module。

3. 功能实现

3.1 寄存器模块 Register

3.1.1 Register 模块实现代码

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

reg [31:0] regFile[31:0];
reg [31:0] ReadData1;
reg [31:0] ReadData2;

always @ (Clk)
    begin
        ReadData1 = regFile[readReg1];
        ReadData2 = regFile[readReg2];
    end
```

```
    always @ (posedge Clk or Reset)
        if(Reset)
            begin
                for(integer i = 0; i <= 31; i = i + 1)
                    regFile[i] = 0;
            end
        else
        begin
            if(regWrite)
                regFile[writeReg] = writeData;
        end

    assign readData1 = ReadData1;
    assign readData2 = ReadData2;

endmodule
```

3.2 存储器模块 Data Memory

3.2.1 Data Memory 模块实现代码

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

    reg [31:0] memFile[0:63];
    reg [31:0] ReadData;

    always@(posedge Clk)
        begin
            if(memRead == 1)
                ReadData = memFile[address];
        end

    always@(negedge Clk)
        begin
            if(memWrite == 1)
                memFile[address] = writeData;
        end

    assign readData = ReadData;
endmodule
```

3.3 有符号立即数单元 Signext

3.3.1 Signext 模块实现代码

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

    reg [31:0] Data;

    always@(inst)
        begin
            if(inst < 16'b1000000000000000)
                Data = 32'b00000000000000000000000000000000 | inst;
            else
                Data = 32'b11111111111111110000000000000000 | inst;
        end

    assign data = Data;

endmodule
```

4. 仿真测试

4.1 寄存器模块 Register

4.1.1 Register 模块仿真结果

在 Register 的测试文件中依次执行了寄存器写和寄存器读两种操作，测试结果如下图所示：

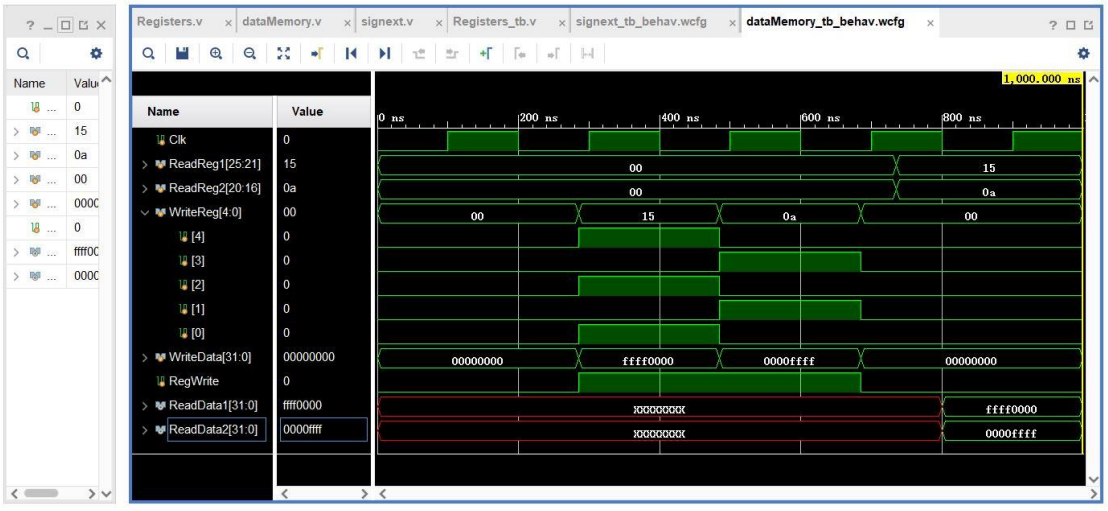

图 3. Register 模块仿真结果

注：这里的红线表示没有执行寄存器读操作，寄存器的 Read Data 可以为任意值

检验测试结果，正确。

4.2 存储器模块 Data Memory

4.2.1 Data Memory 模块仿真结果

在 Data Memory 的测试文件中依次执行了存储器写和存储器读两种操作，测试结果如下图所示：

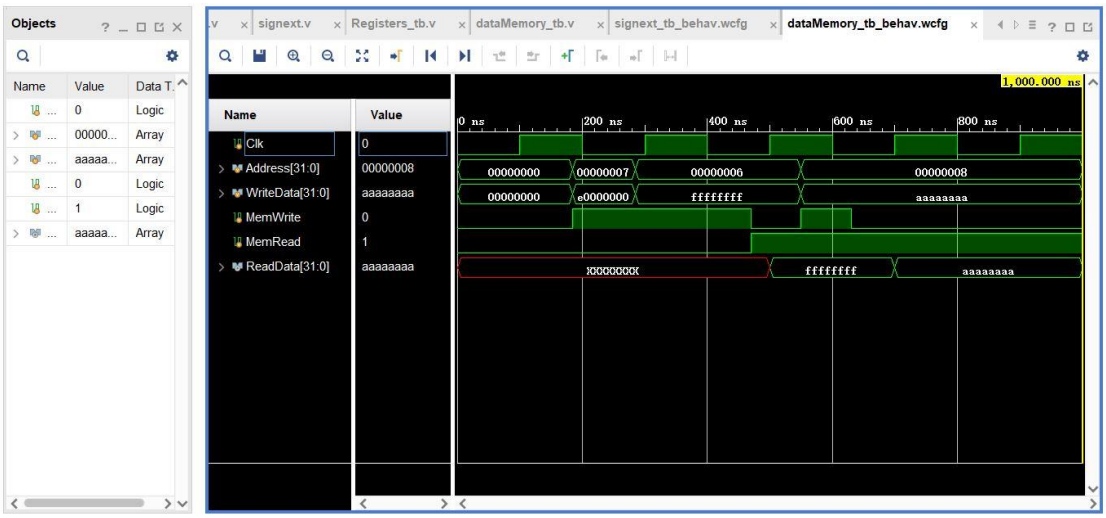

图 4. Data Memory 模块仿真结果

注：这里的红线表示没有执行存储器读操作，存储器的 Read Data 可以为任意值

检验测试结果，正确。

4.3 有符号扩展单元 Signext

4.3.1 Signext 模块仿真结果

在 Signext 的测试文件中依次测试了正数和负数的有符号扩展指令，测试结果如下：

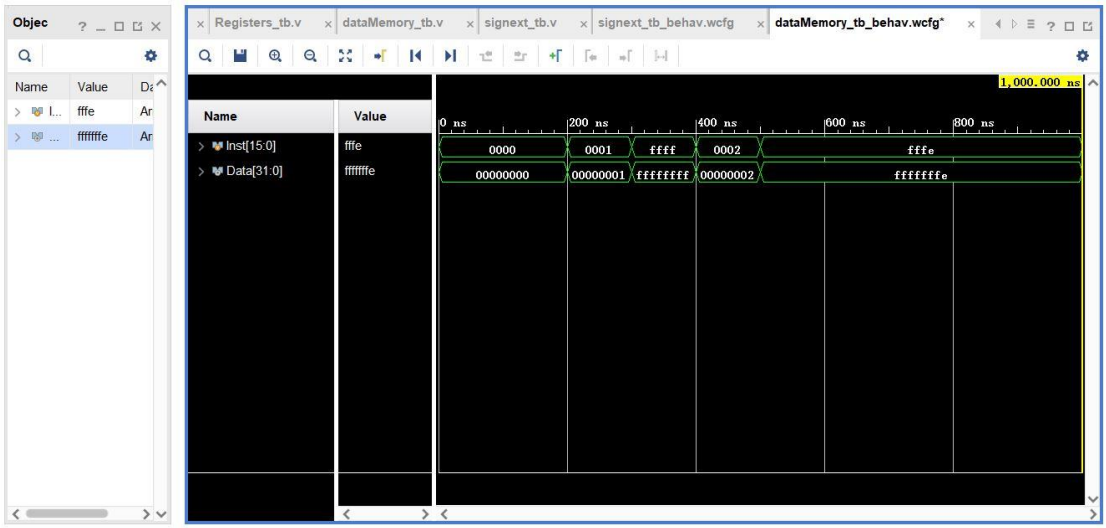

图 5. Signext 模块仿真结果

检验测试结果，正确。

5. 实验总结

5.1 实验评价

综合以上模拟仿真结果，可以验证 Register、Data Memory 以及 Signext 模块均已成功实现：

Register 能够暂时存储一些数据以及中间运算结果，并且能进行寄存器数据的读写；

Data Memory 能够存储大量的内存数据，并且能够进行内存数据的读写；

Signext 能够将 16 位有符号立即数带符号扩展为 32 位有符号立即数。

5.2 实验心得

总体来说，这次实验比较简单，只要弄清楚了各个模块的输入输

出接口的意义，实验就成功了一大半。

这个实验里我最大的收获是深刻理解了 MIPS 多周期处理器的相关性与冒险问题里面的结构性冒险。结构性冒险的原因在于资源相关性，也就是所需的硬件部件正在为之前的指令工作，从而造成硬件冲突。由于在多周期 MIPS 处理器中，只有一个寄存器单元和一个存储器单元，当不同的周期阶段需要同时用到这两个元件时，就会造成硬件资源的冲突,从而引发结构性冒险。于是我在 Register 模块和 Data Memory 模块对读写进行了分离操作。因为在一个周期 CLK 里面，有两次 CLK 跳变：上升沿跳变和下降沿跳变，因此，如果让读操作和写操作分别在两次跳变时刻进行,就能将一个周期里面同时对 Register 或 Data Memory 的读写操作分离开，从而能使读写操作能在同一个周期同一个元件内部执行。所以在我的实现代码里面 ，对于寄存器单元：上升沿和下跳沿都能进行寄存器读操作，寄存器写操作只能在时钟下跳沿执行；对于存储器：存储器读操作只能在时钟上升沿执行，存储器写操作只能在时钟下跳沿执行。

这次的实验让我加深了对计算机系统结构课上所学的相关性与冒险问题的理解，让人受益匪浅。