计算机系统结构实验报告　实验 4

颜培深 518030910094

2020 年 5 月 19 日

摘要

本实验实现了简易 MIPS 处理器中的几个重要部件: 寄存器 (Register)、存储器 (Data Memory) 和有符号拓展 (Signal Extend)。他们分别用于暂时存放参与运算的数据和运算结果（具有接收数据、存放数据和输出数据的功能），用于大量存储数据以及对立即数进行有符号拓展，实验通过软件仿真的方式进行结果验证。

目录

1 实验目的

1. 理解 CPU 的寄存器、存储器、有符号拓展
2. 寄存器 (Register) 的实现
3. 存储器 (Data Memory) 的实现
4. 有符号拓展 (Signal Extend) 的实现
5. 使用行为仿真验证实现的正确性

2 原理分析

2.1 寄存器 (Register) 的设计

寄存器的基本功能是暂时存放参与运算的数据和运算结果，接收数据、存放数据和输出数据。不同的控制信号对应不同的器件操作方式，本实验中用到的输入输出端口及其功能如表1所示。

端口名称	具体功能
readReg1	5 位操作数，用于选择读取寄存器 1
readReg2	5 位操作数，用于选择读取寄存器 2
writeReg	5 位操作数，用于选择写入寄存器
writeData	32 位操作数，在特定条件下写入指定寄存器
regWrite	寄存器写使能信号，高电平有效
readData1	从寄存器 1 中读取的 32 位操作数
readData1	从寄存器 2 中读取的 32 位操作数

表 1: 寄存器输入输出端口及其功能

三个用于选择寄存器的操作数都是 5 位二进制数，是由于寄存器文件一般设置为含有 32 个寄存器，同时为了每个时钟周期上升沿从寄存器文件中读取到的数据为最新的，我们在寄存器文件的写操作时设置为在时钟下降沿进行更改。

2.2 存储器是 (Data Memory) 的设计

存储器是时序逻辑电路的一种。按存储器的使用类型可分为只，这里我们讨论的是可读可写存储器，读存储器 (ROM) 和随机存取存储器 (RAM)，这里我们讨论的是可读可写存储器，本实验中用到的输入输出端口及其功能如表2所示。

端口名称	具体功能
address	32 位操作数，用于选择读取的内存数据
writeData	32 操作数，在特定条件下写入指定存储器
memWrite	存储器写使能信号，高电平有效
memRead	存储器读使能信号，高电平有效
readData	从存储器指定地址中读取的 32 位操作数

表 2: 存储器输入输出端口及其功能

；存储器模块与寄存器文件模块类似，都具有读写数据的功能，在仿真层面差距不大，同时也存在数据读写信号同步的问题，所以存储器中写操作也是在时钟下降沿进行，保证了时钟周期上升沿从 readData 端口读出的数据为最新的。

2.3 有符号拓展 (Signal Extend) 的设计

有符号拓展模块的主要功能就是将指令中 16 位有符号立即数拓展为 32 位有符号数，即仅需将此 16 位有符号数的最高位在 32 位有符号数中高 16 位补足即可，这里我们将输入定为 $[15:0]inst$, 输出定为 $[31:0]data$，从而输出公式为：

$$data = (inst\&(32'h8000)) * 32'h1fffe + inst;$$

3 功能实现

3.1 寄存器 (Register) 的实现

寄存器在不同的控制信号进行选择是否进行读写操作，所以我们将读和写操作分开，读寄存器操作一直保持进行，而写寄存器操作仅在时钟下降沿进行，实现部分代码如下：

```
always @(readReg1 or readReg2) begin
    readData1 = regFile[readReg1];
    readData2 = regFile[readReg2];
end
always @(negedge Clk) begin
    if(regWrite)
        regFile[writeReg] = writeData;
end
```

之所以能让寄存器一直保持读操作，是因为我们在寄存器文件中不关心该读取是否有效，在 MIPS 处理器中由寄存器读取数据的接收方来决定是否使用并传递这些数据。

3.2 存储器 (Data Memory) 的实现

与寄存器操作类似，存储器在不同的控制信号进行选择是否进行读写操作，所以我们也将读和写操作分开，在读使能信号保持高电平的状态下读寄存器操作一直保持进行，而在写使能信号高电平状态下写寄存器操作仅在时钟下降沿进行，实现部分代码如下：

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

与寄存器文件不同的是，这里我们对写操作进行了限制，使其仅在读使能信号为高电平的状态下才输出读取的数据。

3.3 有符号拓展 (Signal Extend) 的实现

有符号拓展将符号位进行拓展，代码部分较为简单描述部分就不再赘述，实现部分代码如下：

```
always @(inst) begin
    data = inst;
    data = (data&(32'h8000))*32'h1fffe + data;
end
```

4 结果验证

对于本实验中实现的简易 MIPS 处理器中寄存器存储器以及有符号拓展模块采用软件仿真的方法进行测试，用 Verilog 语言编写测试激励文件，进行仿真，观察各信号波形变化，比对是否符合预期。

4.1 寄存器 (Register) 的测试

对寄存器文件进行两次写操作和一次读取两个寄存器的读操作，用于分别测试寄存器文件读写的仿真性能，仿真结果如图1所示。

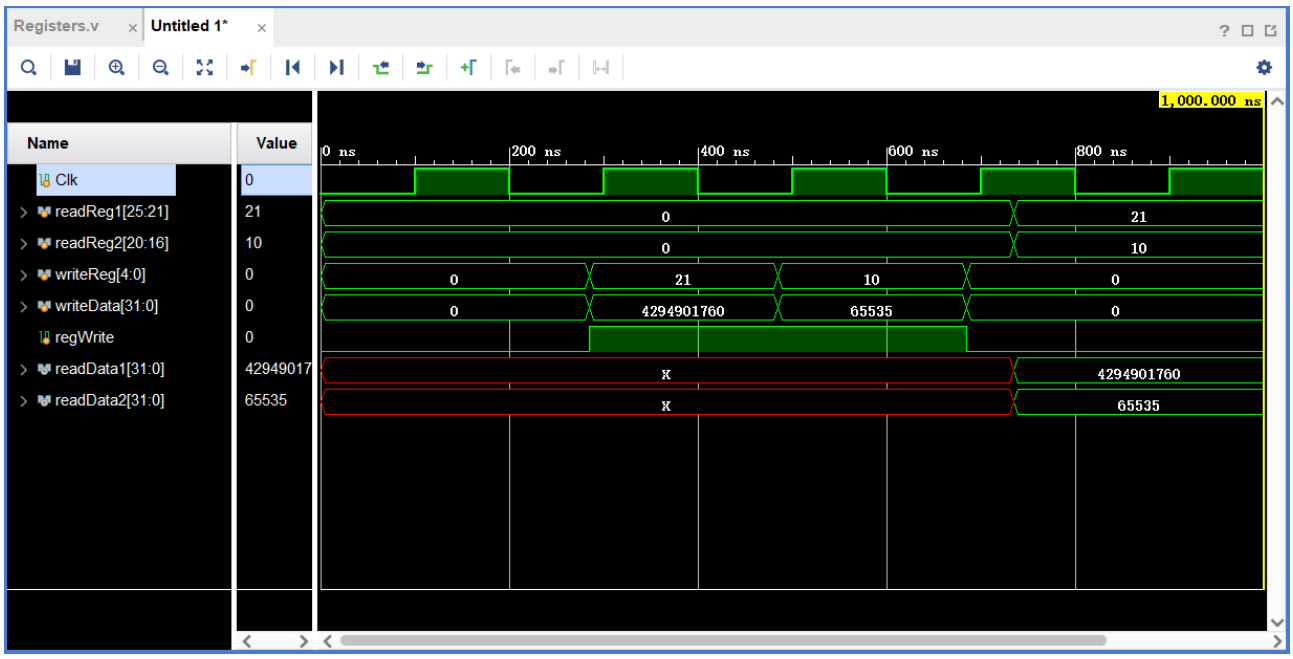

图 1: 寄存器仿真波形

通过图像可以看出两次写入的数据在读出的过程中显示正常，说明仿真成功。

4.2 存储器 (Data Memory) 的测试

对存储器进行多次读写操作，用于分别测试存储器读写的仿真性能，仿真结果如图2所示。

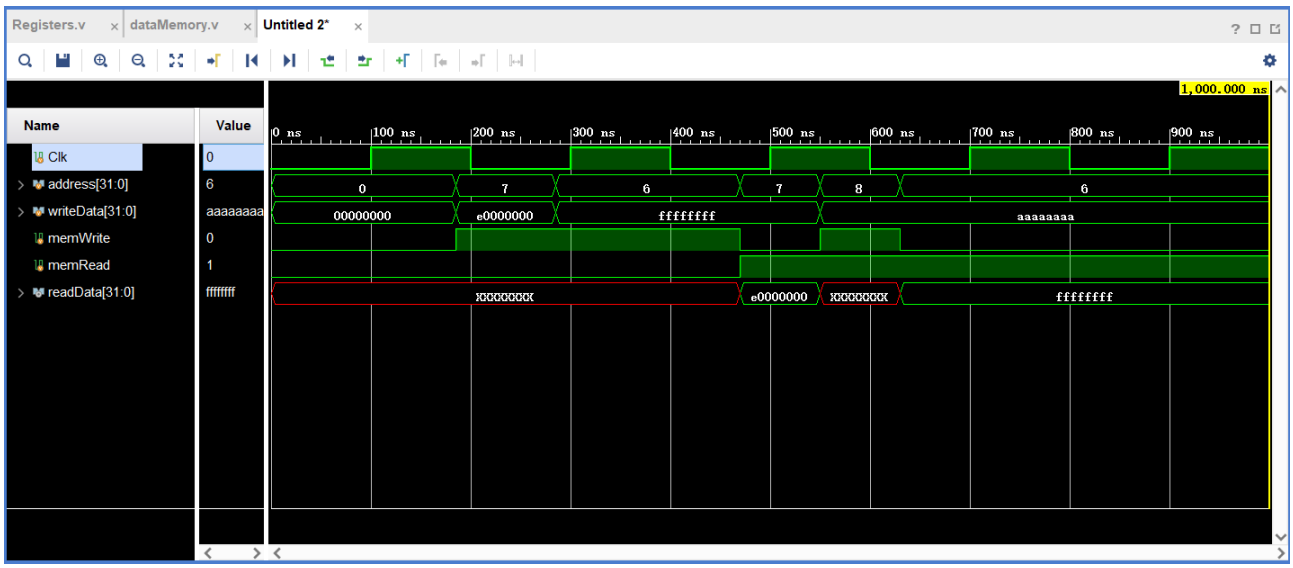

图 2: 存储器仿真波形

存储器多次读写波形正常，仿真过程成功。

4.3 有符号拓展 (Signal Extend) 的测试

每隔 $100ns$ 对有符号拓展模块的输入数据进行一次赋值，每次输入数据如表**??**所示，仿真得到的波形如图3所示。

测试序号	输入数据
1	$16'b0000000000000000$
2	$16'b0000000000000001$
3	$16'b1111111111111111$
4	$16'b0000000000000010$
5	$16'b1111111111111110$

表 3: 有符号拓展模块测试内容

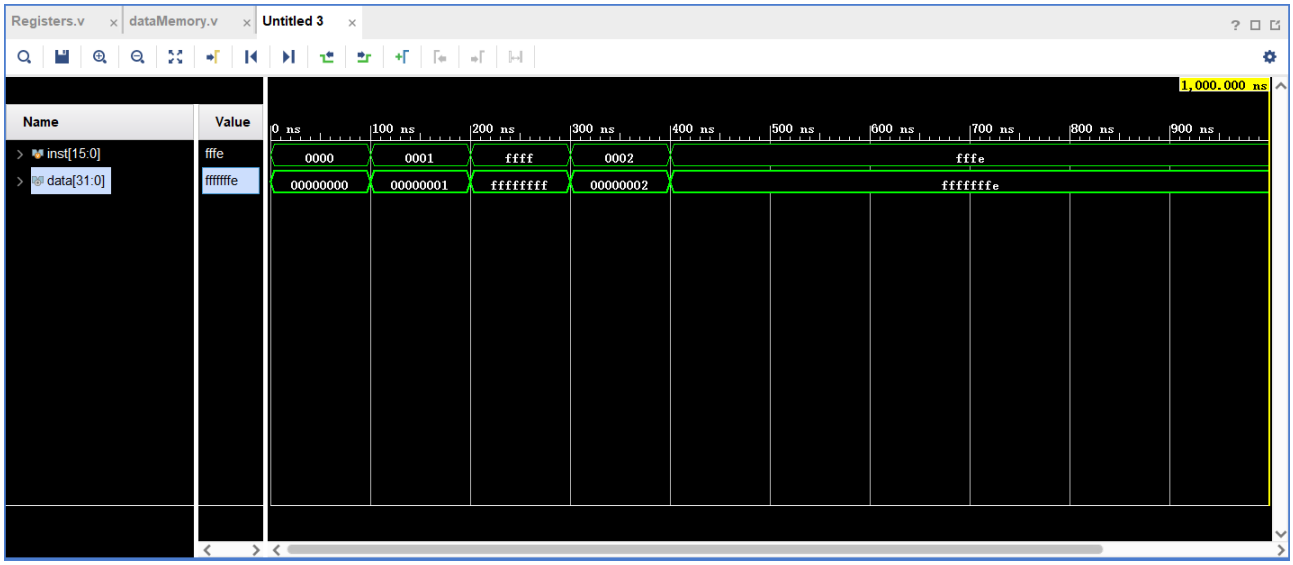

图 3: 有符号拓展模块仿真波形

可以看到有符号拓展模块正确带符号拓展了每次的输入数据，证明本次实验验证成功。

5 总结与反思

本实验实现了简易 MIPS 处理器中寄存器、存储器以及有符号拓展的简单仿真，为接下来的单周期 MIPS 处理器以及多周期流水线 MIPS 处理器的实现奠定了基础，利用时钟周期下降沿进行写入操作是基于时钟上升沿读出数据的假定进行设置的，这样能够实现读写数据先后次序的分辨，更有利于对后面实现完整处理器中数据流处理。另外在寄存器以及存储器文件的初始化过程中，在 Verilog 指导书中查到了直接将整个数据文件读入的操作，好像在后面两次实验中能用到，在这里就不实现了。这三个模块在实现上都没有什么难度，所以在代码规范上多加注意，为后面的复杂代码关系做铺垫。