计算机系统结构实验报告 Lab04

简单的类 MIPS 单周期处理器功能部件的设计与实现（二）：寄存器、存储器与有符号扩展单元

姚宣骋 520021910431

2022 年 3 月 23 日

摘要

本实验实现了简单的类 MIPS 处理器的几个重要部件：寄存器（Register）、存储器（Data Memory）以及有符号扩展单元（Sign Extension）。寄存器的作用是暂时存放少量数据，支持读取、存放于写入功能；存储器的作用是存放较大量的数据，支持读取、存放于写入功能；有符号扩展单元的作用是对 16 位立即数进行有符号扩展至 32 位数。本实验通过软件仿真的形式进行实验结果的验证。

目录

1 实验目的 **2**

2 原理分析 **3**

2.1 寄存器（Register）原理分析 3

2.2 存储器（Data Memory）原理分析 4

2.3 有符号扩展单元（Sign Extension）原理分析 4

3 功能实现 **5**
3.1 寄存器（Register）功能实现 5
3.2 存储器（Data Memory）功能实现 6
3.3 有符号扩展单元（Sign Extension）功能实现 7

4 结果验证 **8**
4.1 寄存器（Register）结果验证 8
4.2 存储器（Data Memory）结果验证 8
4.3 有符号扩展单元（Sign Extension）结果验证 9

5 总结反思 **10**

6 致谢 **10**

A 设计文件完整代码实现 **10**
A.1 寄存器（Register）的代码实现 10
A.2 存储器（Data Memory）的代码实现 10
A.3 有符号扩展单元（Sign Extension）的代码实现 11

B 激励文件完整代码实现 **11**
B.1 寄存器（Register）激励文件的代码实现 11
B.2 存储器（Data Memory）激励文件的代码实现 11
B.3 有符号扩展单元（Sign Extension）激励文件的代码实现 . . . 11

1 实验目的

本次实验有 5 个目的：

- 理解寄存器、数据存储器、有符号扩展单元的 IO 定义；
- Registers 的设计实现；

- Data Memory 的设计实现；
- 有符号扩展部件的实现；
- 对功能模块进行仿真；

2 原理分析

2.1 寄存器（Register）原理分析

寄存器（Register）的功能是暂时存储少量的数据，需要同时支持双通道数据的读取、写入功能。该模块内部共有 32 个寄存器，故两个读取地址 Readregister1、Readregister2 和一个写入地址 Writeregister 都是 5 位 2 进制数。此实验中，寄存器内存储的数据是 32 位的二进制数，故写入数据接口 WriteData 和两个读取数据接口 ReadData1、ReadData2 都是 32 位二进制数。寄存器的主要接口如图 1 所示。由于不确定 WriteReg,WriteData,RegWrite

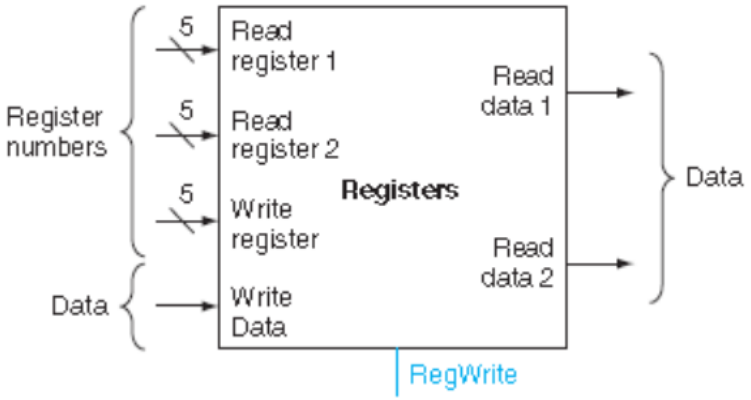

图 1: 寄存器（Register）的主要接口

信号的先后次序，我们采用时钟（样例里命名为 Clk）的下降沿作为写操作的同步信号，防止发生错误。

2.2 存储器（Data Memory）原理分析

存储器（Data Memory）的功能是存储大量的数据，支持数据读取、数据写入的功能。该模块比寄存器大，共能存储 64 个 32 位二进制数。因此地址（Address）是 64 位数。而写入数据接口 Writedata 和读取数据接口 Readdata 都是 32 位二进制数。同时为了防止访问错误，需要对地址访问进行判断，防止给定地址超出存储器范围（0 至 63）。存储器的主要借口如图 2 所示。同寄存器，我们也采用时钟（样例里命名为 Clk）的下降沿作为

图 2: 存储器（Data Memory）的主要接口

写操作的同步信号，防止发生错误。

2.3 有符号扩展单元（Sign Extension）原理分析

有符号扩展单元（Sign Extension）的功能是将指令中的 16 位有符号的立即数拓展为 32 位有符号立即数。因此只需要根据立即数扩展规则，将这个 16 位有符号数的符号位填充在 32 位有符号立即数的高 16 位，再将低 16 位复制到 32 位有符号立即数的低 16 位即可。其主要接口如图 3 所示。

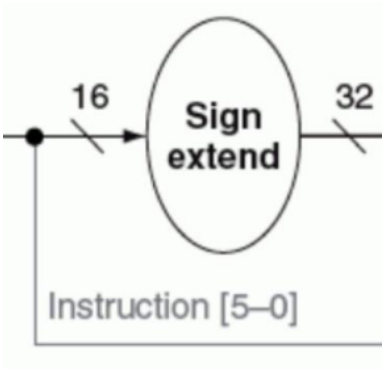

图 3: 有符号扩展单元（Sign Extension）的主要接口

3 功能实现

3.1 寄存器（Register）功能实现

寄存器(Register)可以一直进行读取操作,但是需要用 RegWrite 信号控制其是否允许写入。同原理 2.1 节中所言,由于不确定 WriteReg,WriteData,RegWrite 信号的先后次序，我们采用时钟（样例里命名为 Clk）的下降沿作为写操作的同步信号，防止发生错误。此外，在模块中对 32 个寄存器进行初始化置零处理。部分实现代码如下，详见附录 A.1。

```
1       reg [31:0] regFile [31:0];
2       reg [31:0] ReadData1;
3       reg [31:0] ReadData2;
4       integer i;
5       initial begin
6       for(i=0;i<=31;i=i+1)
7           regFile[i]=0;
8       end
9
10      always @ (readReg1 or readReg2)
```

```
11          begin
12              ReadData1 = regFile[readReg1];
13              ReadData2 = regFile[readReg2];
14          end
15
16      always @ (negedge Clk)
17          begin
18              if(regWrite)
19                  regFile[writeReg]=writeData;
20          end
```

3.2 存储器（Data Memory）功能实现

存储器（Data Memory）由 MemRead 和 MemWrite 分别用来控制读取和写入操作。同原理 2.2 节中所言，我们采用时钟（样例里命名为 Clk）的下降沿作为写操作的同步信号，防止发生错误。同样，对模块中的 64 个存储进行初始化置零操作，并且对地址范围加以检查限制，防止访问错误。部分实现代码如下，详见附录 A.2。

```
1       reg [31:0] ReadData;
2       reg [31:0] memFile[0:63];
3       integer i;
4
5       initial begin
6       for(i=0;i<64;i=i+1)
7           memFile[i]=0;
8       ReadData=0;
9       end
10
```

```
11      always @ (memRead or address)
12      begin
13          // check if the address is valid
14          if (memRead)
15          begin
16              if(address <= 63)
17                  ReadData = memFile[address];
18              else
19                  ReadData = 0;
20          end
21      end
22
23      always @ (negedge Clk)
24      begin
25          if (memWrite && address <= 63)
26              memFile[address] = writeData;
27      end
```

3.3 有符号扩展单元（Sign Extension）功能实现

有符号扩展单元（Sign Extension）将 16 位有符号立即数扩展位 32 位有符号立即数。如原理 2.3 节中扩展规则所述，我们利用 Verilog 提供的拼接操作完成扩展。部分代码如下，详见附录 A.3。

```
1  assign data = { {16 {inst[15]}}, inst[15 : 0] };
```

4　结果验证

4.1　寄存器（Register）结果验证

设置时钟周期（Clk）为 100 ns，我们采取软件仿真对寄存器（Register）模块进行测试，尝试读写数据等不同情况下的操作。激励代码实现详见附录 B.1，测试结果如图 4 所示。从图中可以看出，寄存器（Register）的功

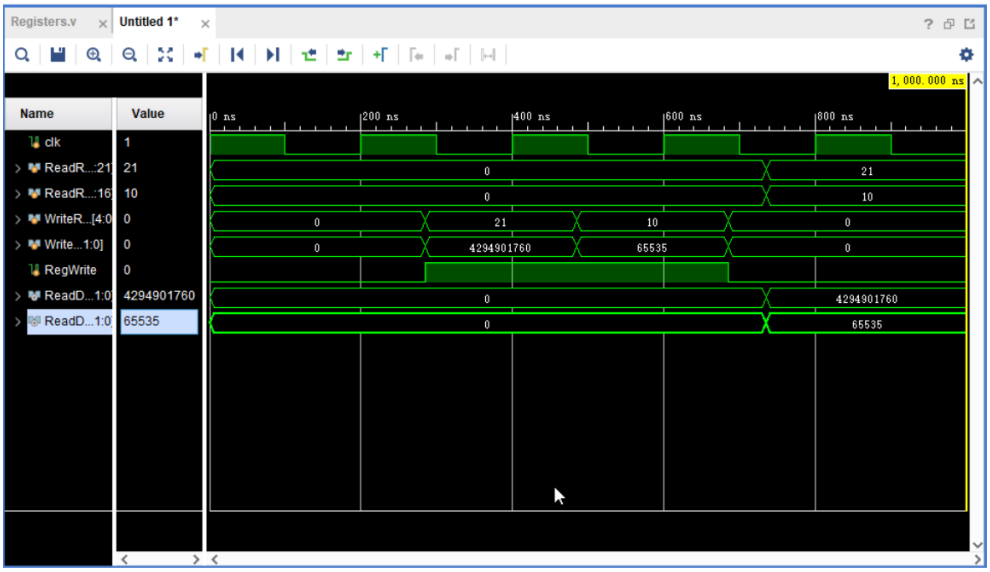

图 4: 寄存器（Register）的测试结果

能实现正确。

4.2　存储器（Data Memory）结果验证

设置时钟周期（Clk）为 100 ns，我们采取软件仿真对存储器（Data Memory）模块进行测试，尝试读写数据等不同情况下的操作。激励代码实现详见附录 B.2，测试结果如图 5 所示。从图中可以看出，存储器（Data Memory）的功能实现正确。

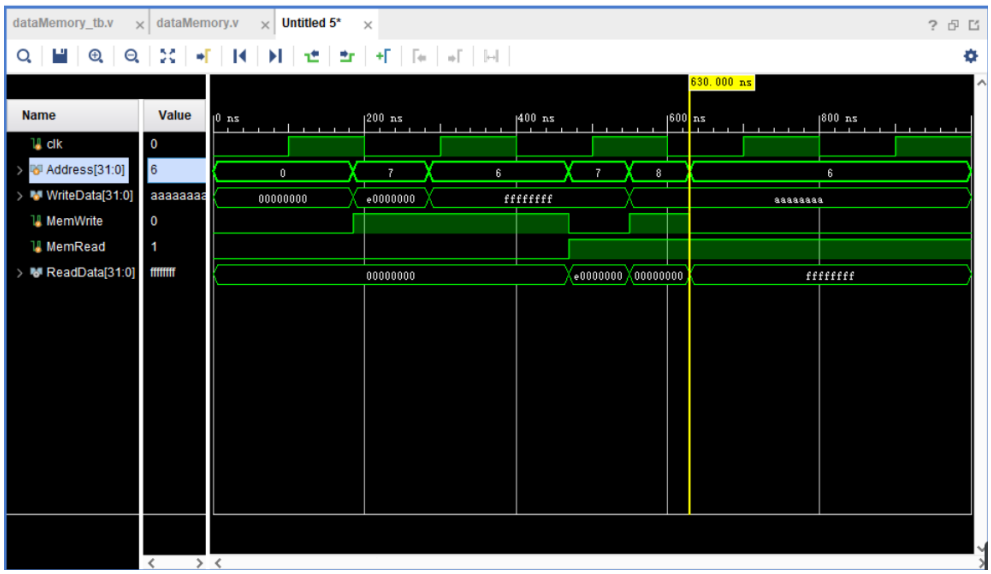

图 5: 存储器（Data Memory）的测试结果

4.3 有符号扩展单元（Sign Extension）结果验证

我们采取软件仿真对有符号扩展单元（Sign Extension）模块进行测试，对 10 个不同的 16 位立即数进行扩展。激励代码实现详见附录 B.3，测试结果如图 6 所示。从图中可以看出，有符号扩展单元（Sign Extension）的

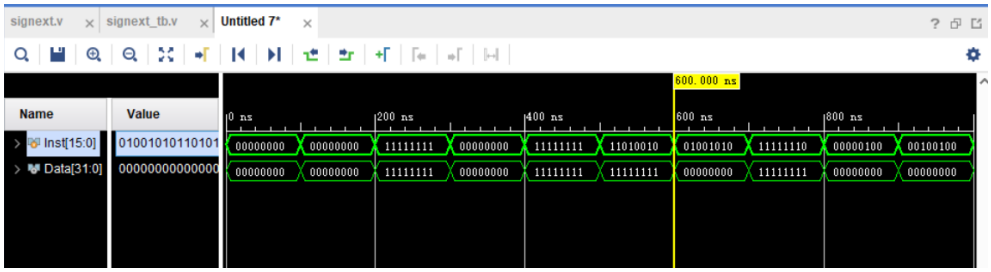

图 6: 存储器（Data Memory）的测试结果

功能实现正确。

5 总结反思

本实验设计并实现了类 MIPS 处理器的三个重要组成部件：寄存器（Register）、存储器（Data Memory）与有符号扩展单元（Sign Extension），并且通过软件仿真模拟的方式验证其正确性，为后续更加复杂的模块设计做铺垫。在本次实验中，我更加深入地理解了 reg、wire 等数据类型的特点与使用方式，在尝试各种方式后，理解了对于寄存器变量初始化的方式与效果。

此外，在本次实验中，我们从软件的角度做到了延迟存储器、寄存器的写操作到时间下降沿的实现。达成同步信号，保证读取数据正确性，确保程序运行正确的效果，让我更加深刻地意识到此思想的重要作用。

6 致谢

感谢本次实验中指导老师在课程微信群里为同学们答疑解惑；

感谢上海交通大学网络信息中心提供的远程桌面资源；

感谢计算机科学与工程系相关老师对于课程指导书的编写以及对于课程的设计，让我们可以更快更好地学习相关知识，掌握相关技能；

感谢电子信息与电气工程学院提供的优秀的课程资源。

A 设计文件完整代码实现

A.1 寄存器（Register）的代码实现

参见代码文件 Register.v

A.2 存储器（Data Memory）的代码实现

参见代码文件 dataMemory.v

A.3 有符号扩展单元（Sign Extension）的代码实现

参见代码文件 signext.v

B 激励文件完整代码实现

B.1 寄存器（Register）激励文件的代码实现

参见代码文件 Register_tb.v。

B.2 存储器（Data Memory）激励文件的代码实现

参见代码文件 dataMemory_tb.v。

B.3 有符号扩展单元（Sign Extension）激励文件的代码实现

参见代码文件 signext_tb.v。