计算机系统结构实验 4
简单的类 MIPS 单周期处理器功能部件的设计与实现（二）

胡晨志 521021910107

摘要

在 lab04 中，我继续学习 Verilog 语言并实现了寄存器、数据存储器和有符号扩展单元等功能，为后续搭建单周期处理器和流水线作准备。

关键字：Vivado，Verilog

目录

1 实验目的

- 理解寄存器、数据存储器、有符号扩张单元的 IO 定义
- Register 的设计实现
- Data Memory 的设计实现
- 有符号扩展部件的实现
- 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 Vivado 工程的基本组成

Vivado 工程的基本组成如下：

- design source .v 文件：`dataMemory.v`，`Registers.v`，`signext.v`
- simulation source .v 文件：`Datamemory_tb.v`，`Registers_tb.v`，`signext_tb.v`

2.2 实现原理

按照实验指导书上各个部件的端口创建模块。统一通过时钟下降沿来同步写数据操作。注意 Register 写数据后要改变输出端口的值。

3 功能实现

基于上述即可完成主控制模块的功能。三个模块的代码分别如 </>CODE 1，</>CODE 2，</>CODE 3 所示。

</> CODE 1: dataMemory.v

```
`timescale 1ns / 1ps

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;
    integer i;

    initial begin
        for (i = 0; i < 64; i = i + 1)
            memFile[i] = 0;
    end

    always@ (memRead or memWrite or address)
        begin
            if (memWrite ==1)
                readdata = 0;
            else if (memRead == 1)
                readdata = memFile[address];
            else readdata = 0;
        end
    always@ (negedge Clk)
        begin
            if (memWrite == 1)
                memFile[address] = writeData;
        end
    assign readData = readdata;
endmodule
```

</> CODE 2: Registers.v

```
`timescale 1ns / 1ps
module Registers(
    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;
    integer i;

    initial begin
        for (i = 0; i < 32; i = i + 1)
            regFile[i] = 0;
    end

    always @(readReg1 or readReg2 or writeReg)
        begin
            readdata1 = regFile[readReg1];
            readdata2 = regFile[readReg2];
        end

    always @(negedge Clk)
        begin
            if (regWrite)
                begin
                    regFile[writeReg] = writeData;
                    if (writeReg == readReg1)
                        readdata1 = regFile[readReg1];
                    if (writeReg == readReg2)
                        readdata2 = regFile[readReg2];
                end
        end
    assign readData1 = readdata1;
    assign readData2 = readdata2;
endmodule
```

</> CODE 3: signext.v

```
`timescale 1ns / 1ps

module signext(
    input [15:0] inst,
    output [31:0] data
    );
    assign data = {{16{inst[15]}}, inst[15:0]};
endmodule
```

实现上述后，生成 `Ctr_tb.v` 的激励文件用以仿真测试.

4 结果验证

4.1 测试用激励文件

按照实验指导书的要求编写 `dataMemory_tb.v`, `Registers_tb.v`, `signext.v`文件, 代码如 </>CODE 4， </>CODE 5， </>CODE 6， 所示。

</> CODE 4: dataMemory_tb.v

```
`timescale 1ns / 1ps
```

```
module dataMemory_tb(

    );
    reg Clk;
    reg [31:0] address;
    reg [31:0] writeData;
    reg memWrite;
    reg memRead;
    wire [31:0] readData;

    dataMemory u0 (
        .Clk(Clk),
        .address(address),
        .writeData(writeData),
        .memWrite(memWrite),
        .memRead(memRead),
        .readData(readData)
    );

    parameter PERIOD = 100;

    always #(PERIOD) Clk = !Clk;

    initial begin
        Clk = 0;
        address = 0;
        writeData = 0;
        memWrite = 0;
        memRead = 0;

        #185;
        memWrite = 1'b1;
        address = 32'b00000000000000000000000000000111;
        writeData = 32'b11100000000000000000000000000000;

        #100;
        memWrite = 1'b1;
        writeData = 32'hffffffff;
        address = 32'b00000000000000000000000000000110;

        #185;
        memRead = 1'b1;
        address = 7;
        memWrite = 1'b0;

        #80;
        memWrite = 1;
        address = 8;
        writeData = 32'haaaaaaaa;

        #80;
        memWrite = 0;
        address = 6;
        memRead = 1;
    end
endmodule
```

</> CODE 5: Registers_tb.v

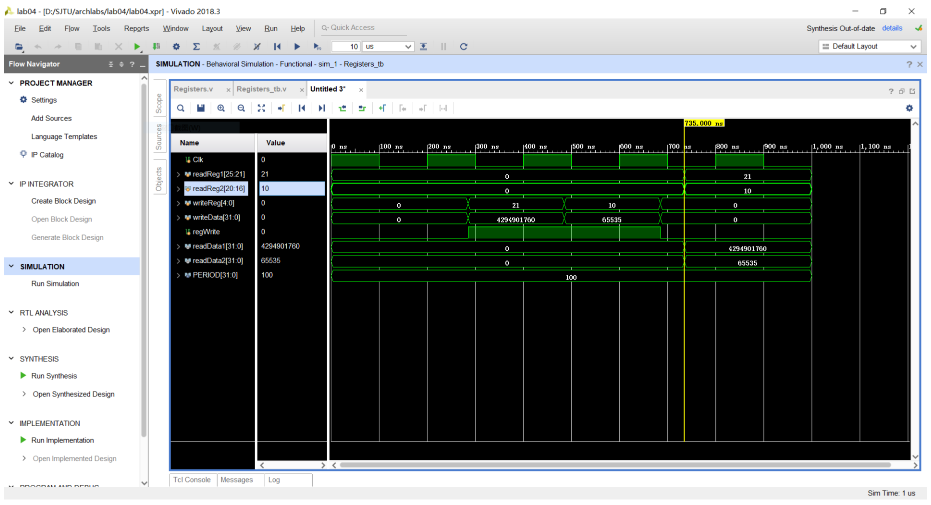
```
`timescale 1ns / 1ps

module Registers_tb(

    );
    reg Clk;
    reg [25:21] readReg1;
    reg [20:16] readReg2;
    reg [4:0] writeReg;
    reg [31:0] writeData;
    reg regWrite;
    wire [31:0] readData1;
    wire [31:0] readData2;

    Registers u0 (
        .Clk(Clk),
        .readReg1(readReg1),
        .readReg2(readReg2),
        .writeReg(writeReg),
        .writeData(writeData),
        .regWrite(regWrite),
        .readData1(readData1),
        .readData2(readData2)
    );

    parameter PERIOD = 100;

    always #(PERIOD) Clk = !Clk;

    initial begin
        // Initialize Inputs
        Clk = 1;
        readReg1 = 0;
        readReg2 = 0;
        writeReg = 0;
        writeData = 0;
        regWrite = 0;

        //Current Time: 285ns
        #285;
        regWrite = 1'b1;
        writeReg = 5'b10101;
        writeData = 32'b11111111111111110000000000000000;

        //Current Time: 485ns
        #200;
        writeReg = 5'b01010;
        writeData = 32'b00000000000000001111111111111111;

        #200;
        regWrite = 1'b0;
        writeReg = 5'b00000;
        writeData = 32'b00000000000000000000000000000000;

        //Current Time: 735ns
        #50;
        readReg1 = 5'b10101;
        readReg2 = 5'b01010;
    end
endmodule
```

</> CODE 6: signext_tb.v

```
`timescale 1ns / 1ps

module signext_tb(

    );
    reg [15:0] inst;
    wire [31:0] data;

    signext u0 (
        .inst(inst),
        .data(data)
    );

    initial begin
        inst = 16'b0000000000000000;
        #100;
        inst = 16'b0000000000000001;

        #100;
        inst = 16'b1111111111111111;

        #100;
        inst = 16'b0000000000000010;

        #100;
        inst = 16'b1111111111111110;
    end
endmodule
```

4.2 仿真测试

数据存储器、寄存器和有符号扩展单元仿真结果分别如图1，2，3所示。


lab04 - [D:/SJTU/archlabs/lab04/lab04.xpr] - Vivado 2018.3
SIMULATION - Behavioral Simulation - Functional - sim_1 - Registers_tb
Registers.v
Registers_tb.v
Untitled 3*
735.000 ns
Sim Time: 1 us

Name	Value
Clk	0
readReg1[25:21]	21
readReg2[20:16]	10
writeReg[4:0]	0
writeData[31:0]	0
regWrite	0
readData1[31:0]	4294901760
readData2[31:0]	65535
PERIOD[31:0]	100

图 1: 数据存储器实验结果

5 反思与总结

本次实验的主要难点在于写数据操作的同步问题，具体实现时通过时钟下降沿进行同步。此处统一使用下降沿也是为了方便后续实验。

在编写有符号扩展扩展单元时刚开始我的写法十分笨拙，先判断高位是一还是零，然后用 if 语句分别进行扩展。在老师的指导下我修改成了现在这个比较简单的版本。

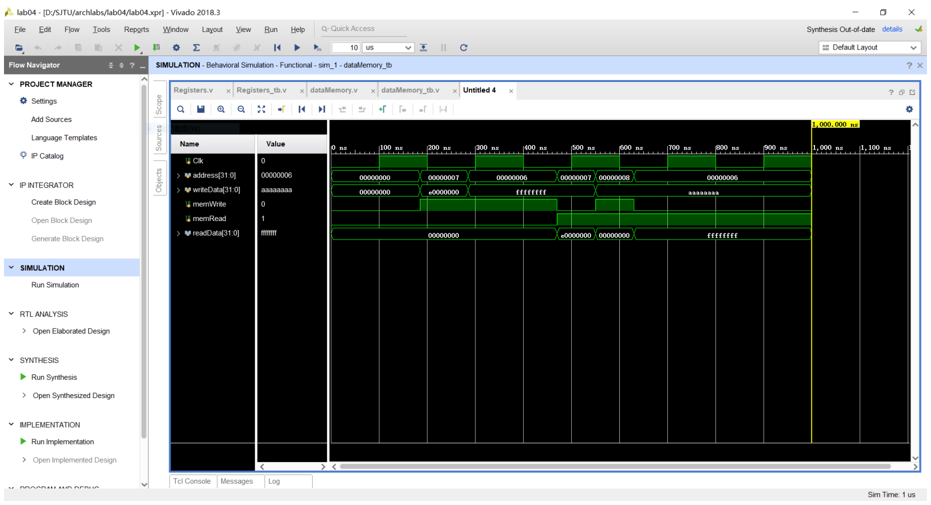


图 2: 寄存器实验结果

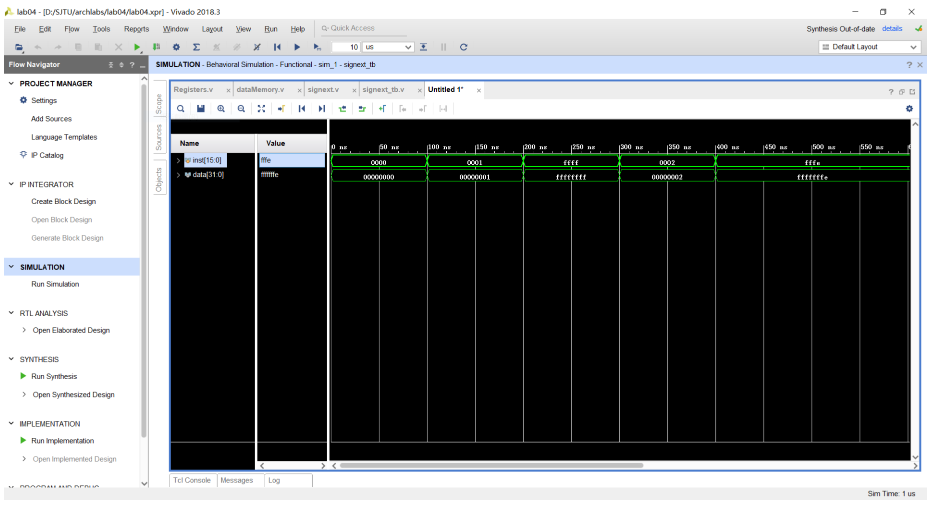


图 3: 有符号扩展单元实验结果