

Department of  
**Computer Science  
& Engineering**



## 计算机系统结构实验指导书-LAB04

# 1. OVERVIEW

---

## 1.1 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

## 1.2 实验目的

1. 理解 CPU 寄存器与存储器
2. Register 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

## 1.3 实验报告与验收办法

需提交电子版实验报告，本实验检查三个仿真结果并验收登记

## 1.4 注意事项

可建立 E:\Archlabs 文件夹，用于放置实验工程等

## 2. 新建工程

### 2.1 实验描述

#### 2.1.1 新建工程

1. 新建工程 lab04
2. 选择 FPGA 参数:

选择 xc7k325tffg676-2

Parts | Boards

[Reset All Filters](#)

Category: All

Package: ffg676

Temperature: All Remaining

Family: Kintex-7

Speed: -2

Search: Q-

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gt
xc7k160tffg676-2	676	400	101400	202800	325	0	600	8
xc7k325tffg676-2	676	400	203800	407600	445	0	840	8
xc7k410tffg676-2	676	400	254200	508400	795	0	1540	8

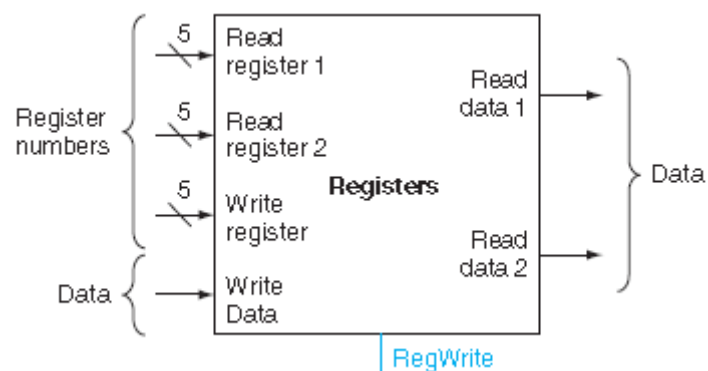
3. 点击 Next
4. 点击 Finish

## 3. 寄存器

### 3.1 实验描述

#### 3.1.1 模块描述

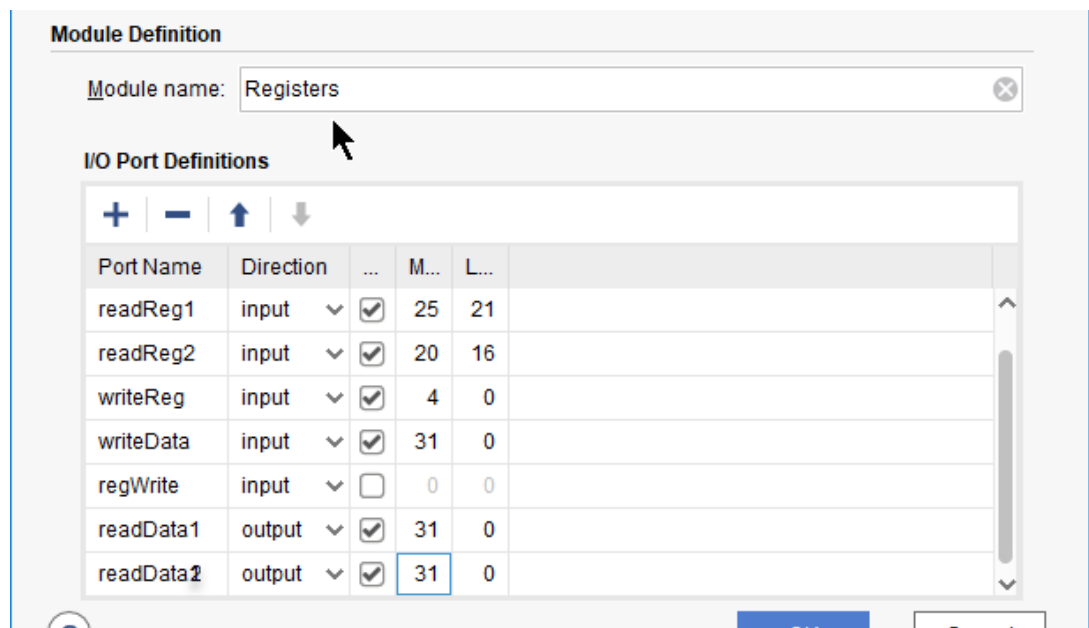
寄存器是指令操作的主要对象，MIPS 中一共有 32 个 32 位的寄存器。



寄存器模块

#### 3.1.2 新建模块源文件

新建文件 Registers。I/O 端口定义，这里加入了时钟信号 Clk



### 3.1.3 编写功能

由于不确定 WriteReg, WriteData, RegWrite 信号的先后次序，我们采用时钟（这里名为 Clk）的下降沿作为写操作的同步信号，防止发生错误。

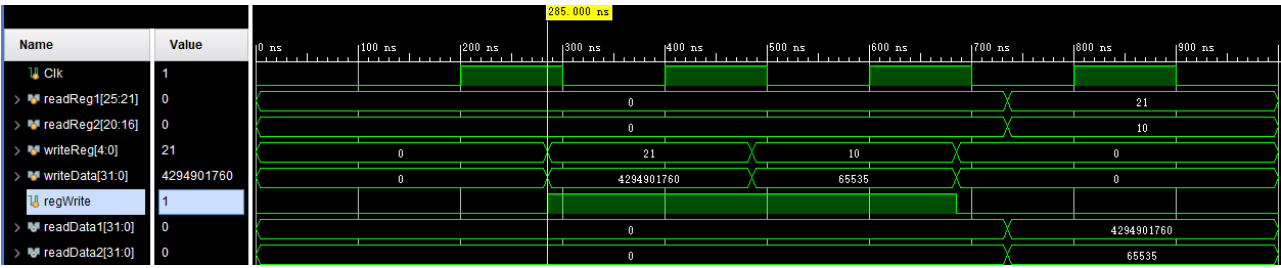
```
31      reg [31:0] regFile[31:0];
32
33
34
35      always @(readReg1 or readReg2 or writeReg)
36      begin
37          // ToDo
38      end
39
40      always @ (negedge Clk)
41      begin
42          // ToDo
43      end
44
```

### 3.1.4 仿真测试

1. 测试文件名 Registers\_tb
2. 添加如下激励信号，进行行为仿真。使用 Clk 作为时钟输入，仿真周期自定，时钟周期可设为 200ns。

```
59      initial begin
60          // Initialize Inputs
61          //.....
62
63          //Current Time: 285ns
64          #285;
65          regWrite = 1'b1;
66          writeReg  = 5'b10101;
67          writeData = 32'b11111111111111110000000000000000;
68
69          //Current Time: 485ns
70          #200;
71          writeReg = 5'b01010;
72          writeData = 32'b00000000000000001111111111111111;
73
74          #200;
75          regWrite = 1'b0;
76          writeReg = 5'b00000;
77          writeData = 32'b00000000000000000000000000000000;
78
79          //Current Time: 735ns
80          #50;
81          readReg1 = 5'b10101;
82          readReg2 = 5'b01010;
83
84      end
```

3. 下面给出一个简单仿真样例：



### 3.2 实验报告

## 4. 内存单元模块 MEMORY

### 4.1 实验描述

#### 4.1.1 模块描述

存储器本模块与 register 类似，由于写数据也要考虑信号同步，因此也需要时钟。内存单元的实现，也可用系统 Block Memory 来生成。参见本手册最后附录的部分图示。



#### 4.1.2 新建模块源文件

**Module Definition**

Module name:

**I/O Port Definitions**

Port Name	Direction	Bus	MSB	LSB
Clk	input	<input type="checkbox"/>	0	0
address	input	<input checked="" type="checkbox"/>	31	0
writeData	input	<input checked="" type="checkbox"/>	31	0
memWrite	input	<input type="checkbox"/>	0	0
memRead	input	<input type="checkbox"/>	0	0
readData	output	<input checked="" type="checkbox"/>	31	0

OK Cancel

### 4.1.3 编写功能

端口声明好了，可如下编写 Verilog 代码

```

13
14     reg [31:0] memFile[0:63];
15
16     always@ ( /* conditions */ )
17     begin
18         // ToDo
19     end
20
21     always@ (/* which edg */)
22     begin
23         // ToDo
24     end

```

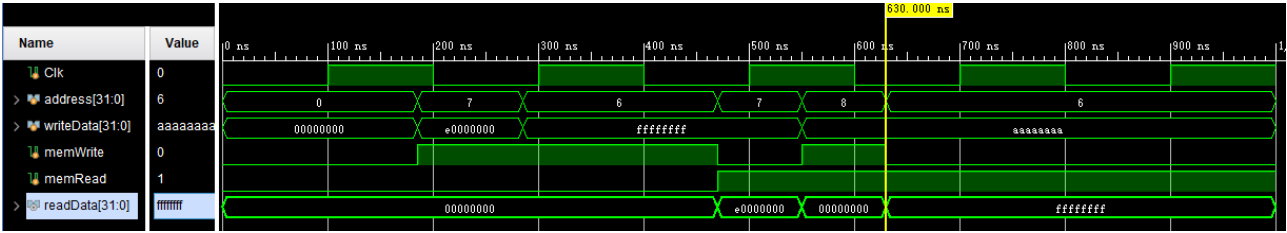
#### 4.1.4 功能仿真

1. 文件名可取 dataMemory\_tb
2. 添加激励信号如下，设定不同的输入，以保证逻辑的正确

[illegible]



3. 下面给出参考样例：



## 4.2 实验报告

## 5. 带符号扩展

### 5.1 实验描述

#### 5.1.1 模块描述

将 16 位有符号数扩展为 32 位有符号数。

补码：

(1) 正数的补码：与原码相同。

+9 的补码是 00001001。

(2) 负数的补码：符号位为 1，其余位为该数绝对值的原码按位取反；然后整个数加 1。  
求-7 的补码。

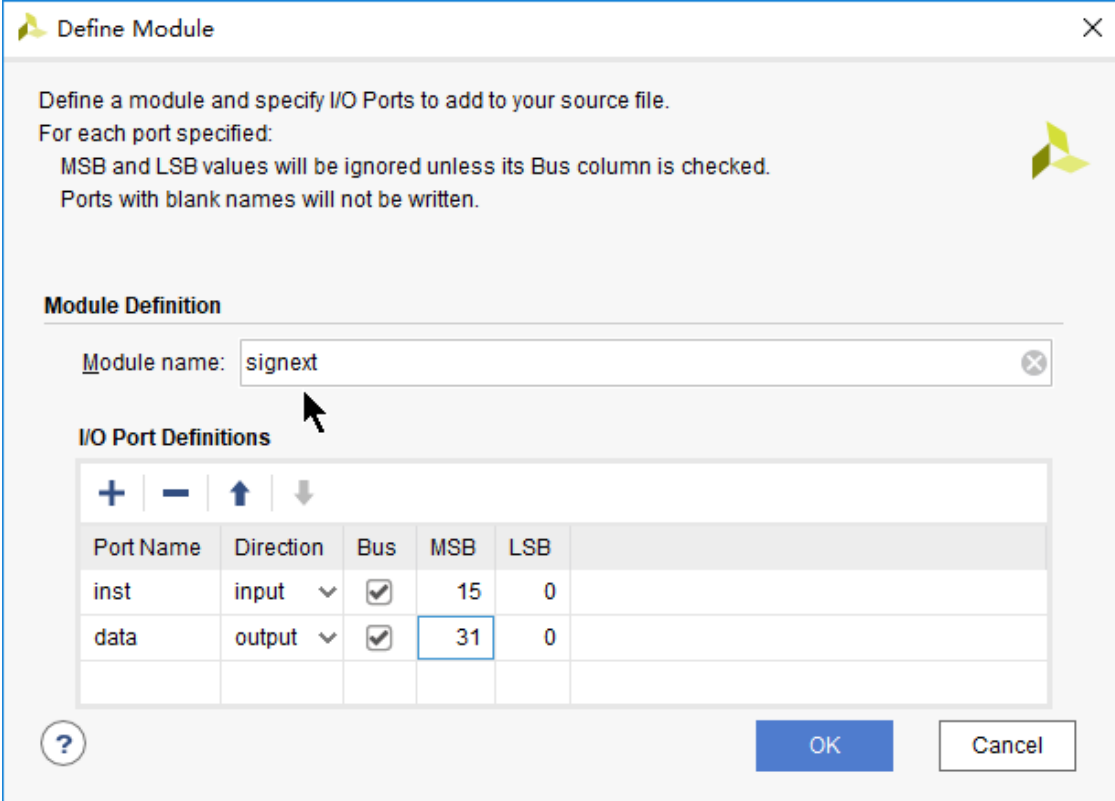
因为给定数是负数，则符号位为“1”。

后七位：+7 的原码（0000111）→按位取反（1111000）→加 1（1111001）

所以-7 的补码是 11111001。

注意：带符号扩展只需要在前面补足符号即可

#### 5.1.2 新建模块源文件



Define a module and specify I/O Ports to add to your source file.  
For each port specified:  
MSB and LSB values will be ignored unless its Bus column is checked.  
Ports with blank names will not be written.

**Module Definition**

Module name:

**I/O Port Definitions**

Port Name	Direction	Bus	MSB	LSB
inst	input	<input checked="" type="checkbox"/>	15	0
data	output	<input checked="" type="checkbox"/>	31	0

Buttons: ? OK Cancel

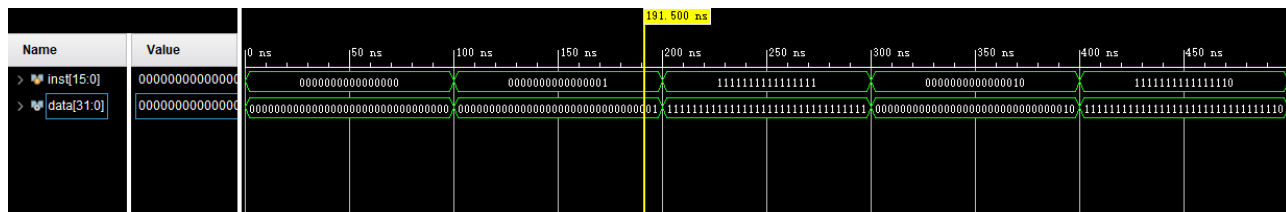
### 5.1.3 实现功能

有多种方法将符号补齐

```
module signext(  
    input [15:0] inst,  
    output [31:0] data  
);  
  
    assign data= //How to;  
  
endmodule
```

### 5.1.4 仿真测试

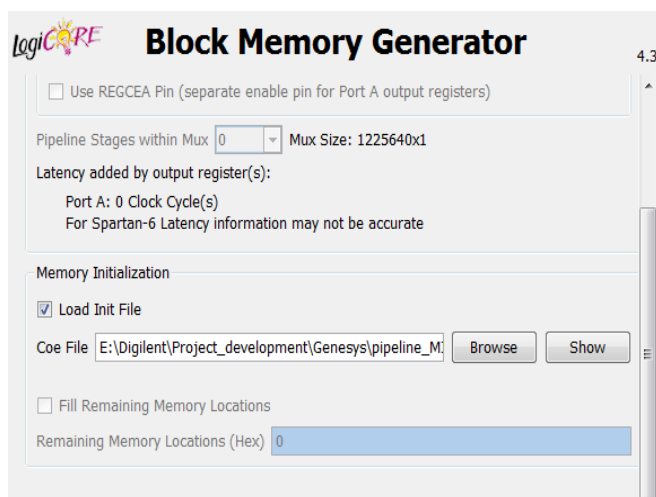
1. 添加激励信号，进行行为仿真
2. 观察波形是否满足设计逻辑
3. 参考波形如下



## 5.2 实验报告

## 附录：

内存既可用类似寄存器的方法来实现，也可用 Block Memory 实现。采用 BRAM 来设计 Data memory 还是较方便的。



```
1 memory_initialization_radix=16;  
2 memory_initialization_vector=  
3 00000001,  
4 00000005,  
5 00000008,  
6 00000000,  
7 00000000,  
8 00000000,  
9 00000000,  
10 00000000,  
11 00000000,  
12 00000000,  
13 00000000,  
14 00000000,  
15 00000000,  
16 00000000,  
17 00000000,  
18 00000000,  
19 00000000,  
20 00000000,
```