**目录**

[实验二 寄存器实验 2](#_Toc549)

[1、实验目的、重点难点 2](#_Toc26726)

[2、 MIPS相关知识 2](#_Toc29572)

[2.1 数据格式 2](#_Toc23031)

[2.2 寄存器 2](#_Toc2795)

[3、 实验任务实现 3](#_Toc6163)

[3.1 实现程序计数器（PC） 3](#_Toc27070)

[3.2 MIPS寄存器堆 5](#_Toc15814)

[4、 vivado以及verilog知识补充 11](#_Toc2371)

[4.1将defines.v设置为global include 11](#_Toc10236)

[4.2 调整仿真时间 13](#_Toc112)

[4.3 verilog建模规则 13](#_Toc28839)

[4.4 integer类型 13](#_Toc11515)

# 实验二 寄存器实验

## 1、实验目的、重点难点

实验目的：

1.熟悉并掌握MIPS计算机中寄存器堆的原理和设计方法。

2.初步了解MIPS指令结构和源操作数/目的操作数的概念。

3.熟悉并运用verilog语言进行电路设计。

4.为后续设计cpu的实验打下基础。

实验重点：

熟悉使用verilog语言进行设计，仿真的过程。

实验难点：

观察仿真波形，确认电路设计是否正确。

## MIPS相关知识

### 2.1 数据格式

处理器可处理的数据格式定义如下：

 比特（bit, b）

 字节（Byte, 8bits, B）

 半字（Halfword, 16bits, H）

 字（Word, 32bits, W）

### 2.2 寄存器

处理器包含的软件可见的寄存器种类如下：

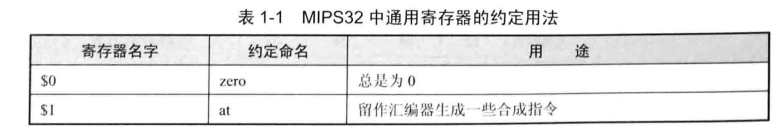
 32 个 32 位通用寄存器，r0~r31。其中有两个被赋予了特殊含义：r0，0 号通用寄存器，值永远为 0；r31， 31 号通用寄存器，被 JAL，BLTZAL 和 BGEZAL 指令隐式的用作目标寄存器，存放返回地址。

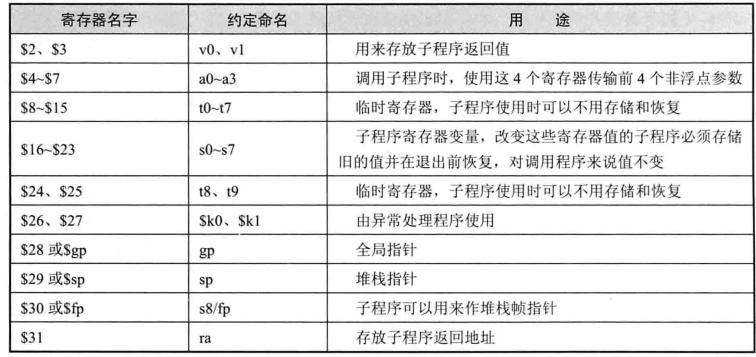
 程序计数器（PC）。这个寄存器软件无法直接访问。

 HI/LO 寄存器。HI 寄存器存放乘法指令结果的高半部分或是除法指令结果的余数，LO 寄存器存放乘法指

令结果的低半部分或是除法指令结果的商。

 控制寄存器（CP0）





## 实验任务实现

### 3.1 实现程序计数器（PC）

程序计数器,有复位信号rst时，输出指令存储器使能信号为ChipDisable（0）,表示指令存储器禁用，此时PC的值保持为0。其余时刻指令存储器使能信号为ChipEnale(1),此时PC的值会在每时钟周期加4。

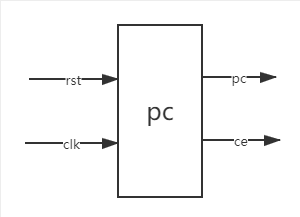
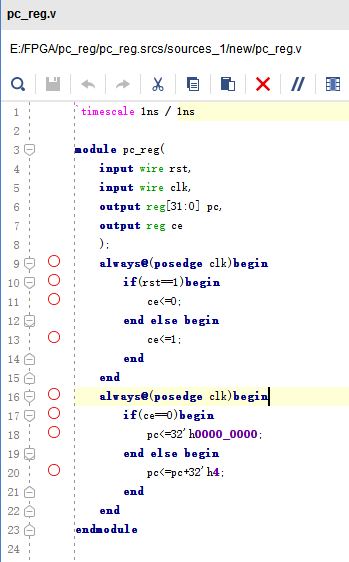


图3.1 程序计数器PC

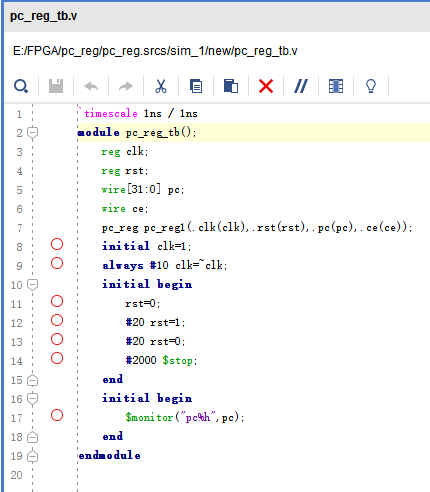
如图3.1所示，是程序计数器pc的设计框图。按照框图进行编程。接口定义如表3.1所示。实验设计代码如代码3.1所示,测试代码如代码3.2所示。

表3.1 pc模块的接口描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 接口名 | 宽度（bit） | 输入/输出 | 作用 |
| 1 | rst | 1 | 输入 | 复位信号 |
| 2 | clk | 1 | 输入 | 时钟信号 |
| 3 | pc | 32 | 输出 | 要读取的指令地址 |
| 4 | ce | 1 | 输出 | 指令存储器使能信号 |



代码3.1 pc\_reg设计代码



代码3.2 pc\_reg测试代码

### 3.2 MIPS寄存器堆

寄存器堆regfile模块实现了32个32位通用寄存器，可以同时进行两个寄存器的读操作和一个寄存器的写操作，其中写操作是同步写。读操作可以在任意时刻进行读。设计框图如图3.2所示。实验设计代码如代码3.3-1和3.3-2所示。测试代码如3.4-1和3.4-2和3.4-3所示。defines.v如代码3.5所示。

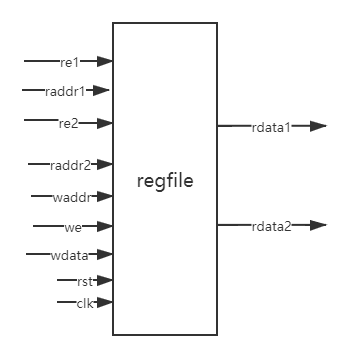
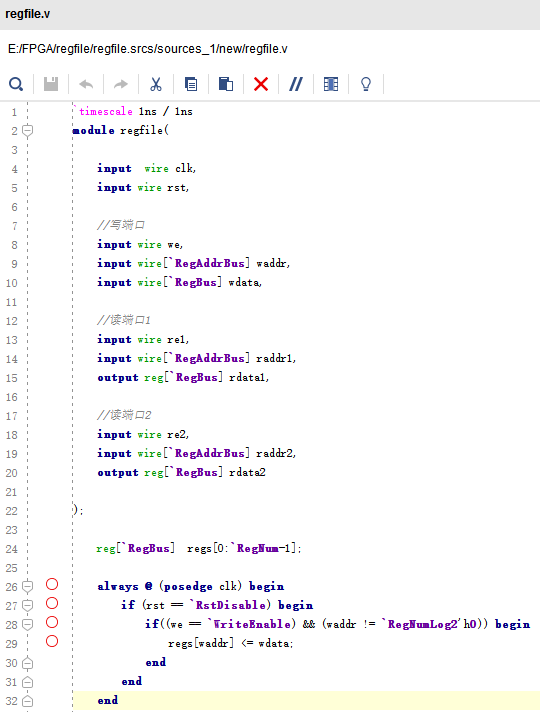
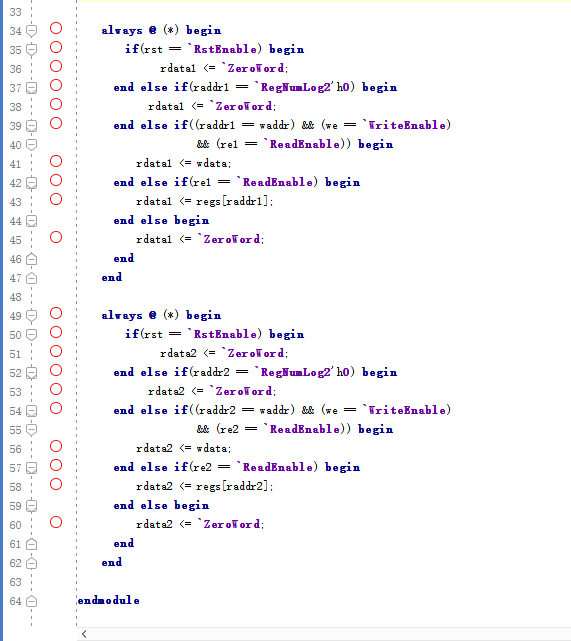


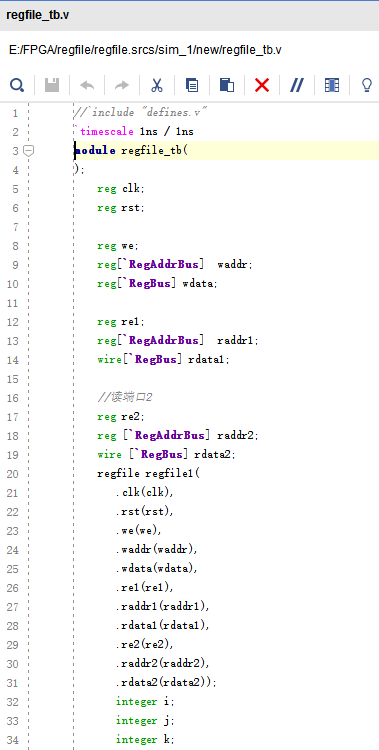
图3.2 regfile设计框图



代码3.3-1 regfile设计代码1



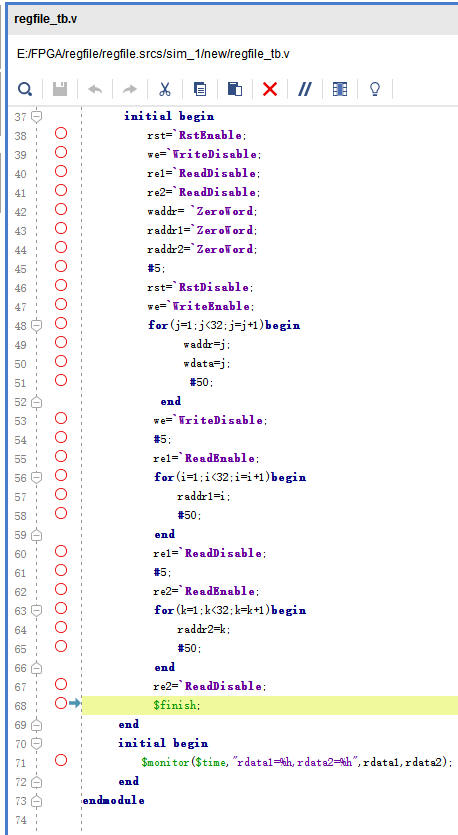
代码3.3-2 regfile设计代码2



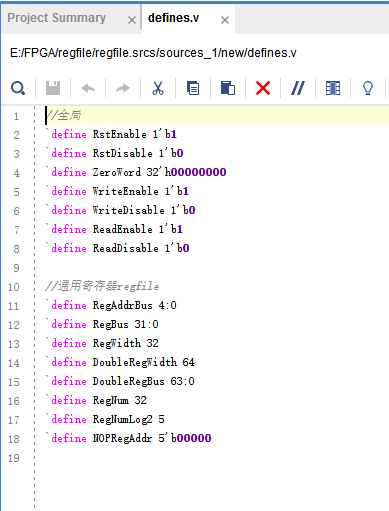
代码3.4-1 regfile测试代码1



代码3.4-2 测试代码2



代码3.4-3 测试代码3



代码3.5 defines.v代码

## vivado以及verilog知识补充

### 4.1将defines.v设置为global include

如图4.1，将defines.v设置成global include，这样的话就可以在别的文件中引用defines.v文件中的内容。而不需要使用`include defines.v。

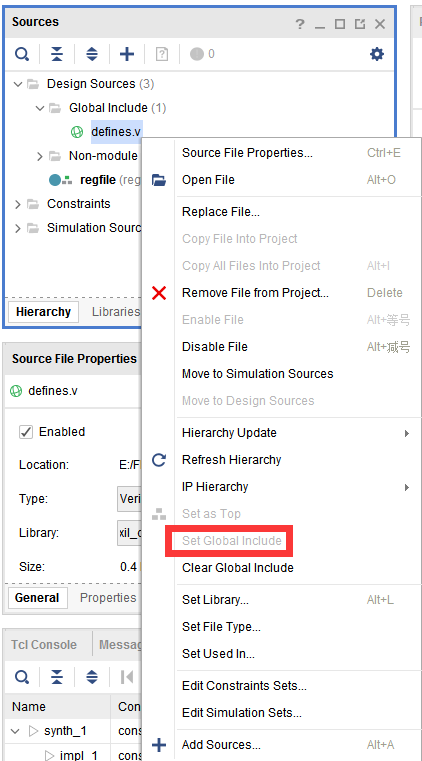


图4.1 defines.v设置成global include

### 4.2 调整仿真时间

将仿真时间由1000ns调整为1000000ns。如图4.2所示。

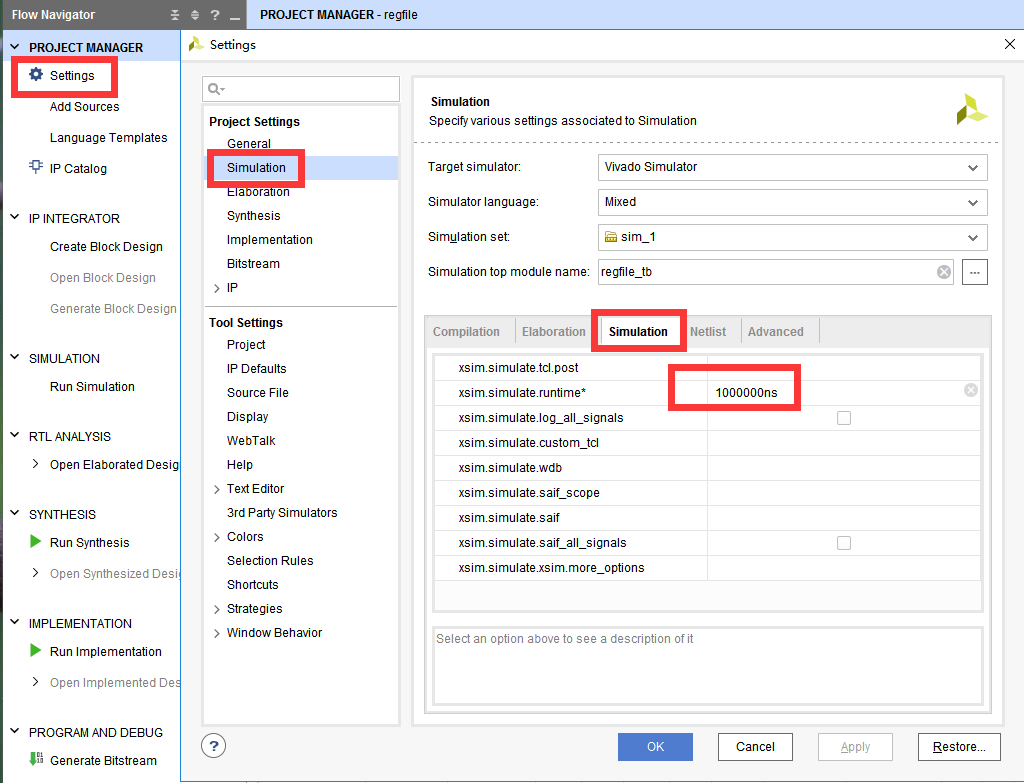


图4.2 调整仿真时间

### 4.3 verilog建模规则

原则1：时序电路建模时，用非阻塞赋值。

􀁹 原则2：锁存器电路建模时，用非阻塞赋值。

􀁹 原则3：用always 块写组合逻辑时，采用阻塞赋值。

􀁹 原则4：在同一个always 块中同时建立时序和组合逻辑电路时，用非阻塞赋值。

􀁹 原则5：在同一个always 块中不要同时使用非阻塞赋值和阻塞赋值。

􀁹 原则6：不要在多个always 块中为同一个变量赋值。

􀁹 原则7：用$strobe 系统任务来显示用非阻塞赋值的变量值

􀁹 原则8：在赋值时不要使用 #0 延迟

### 4.4 integer类型

integer类型变量用于循环变量和计数