形状, 箭头

描述已自动生成

**32位MIPS单周期CPU设计方案**

（本科生）

**2022年春季学期计算机组成原理课程**

**负责人：韩旭达 联系方式：**

**刘文博 林骋楷 李清扬**

**‬班级：2020级数据科学四班**

**2022年4月9日星期六**

1. **项目简介**
2. **项目环境**
   1. **设计语言：**

Verilog硬件描述语言

* 1. **仿真环境：**

Vivado 2018.3

1. **任务目标**
   1. **课程设计要求：**

设计一个兼容32位MIPS或RISC-V指令集的单周期CPU（核）

* 1. **设计目标：**

1. 实现基于哈佛结构的单周期CPU
2. 兼容32位MIPS指令集（部分）
3. 覆盖R型、I型计算类、I型取数类、I型存数类、I型条件判断类、J型指令
4. 前仿成功
5. **参考资料**

[数字逻辑与组成原理实践教程.清华大学出版社[引用日期2022-04-09]](http://www.tup.com.cn/bookscenter/book_07600401.html)

1. **项目文件夹结构**

* Project.zip
  + Report.pptx
  + 设计方案.docx
  + 设计方案.pdf
  + 讲解视频
  + Materials #参考资料文件夹
  + Test #模块测试文件夹
    - IMEM\_Test #指令存储器测试文件
    - DMEM\_Test #数据存储器测试文件
    - ALU\_Test #算数逻辑单元测试文件
  + CPU\_PROJECT #Vivado项目文件夹
  + CPU #源代码文件夹
    - TOB\_TB.v #Testbench文件
    - TOB.v #顶层文件
    - CPU.v #CPU模块
    - ALU.v #算数逻辑单元
    - DECODER.v #译码器
    - DMEM.v #数据存储器
    - EXT5.v #位扩展器
    - EXT16.v …
    - EXT18.v …
    - IMEM.v #指令存储器
    - JOIN.v #位拼接器
    - MUX.v #多路选择器
    - NPC.v #PC+4
    - PC.v #程序计数器
    - REGFILE.v #寄存器
    - Instruction.txt #指令解释文件
    - imem.txt #指令文件

1. **实现细节**
2. **指令集选取**

* MIPS指令集可以分为三种类型（R I J）：

表格

描述已自动生成

R类型，该类型指令从寄存器堆中读取两个源操作数，计算结果写回寄存器堆。具体操作由op、func结合指定，rs和rt是源寄存器的编号，rd是目的寄存器的编号。

I类型，该类型指令使用一个16位的立即数作为一个源操作数。具体操作由op指定，指令的低16位是立即数，运算时要将其扩展至32位，然后作为其中一个源操作数参与运算。

J类型，该类型指令使用一个26位的立即数作为跳转的目标地址。具体操作由op指定，一般是跳转指令，低26位是字地址，用于产生跳转的目标地址。

* 完整MIPS指令集如下：

表格

描述已自动生成

表格

描述已自动生成

本设计我们选取MIPS指令集中的八条指令，分别覆盖R型、I型计算类、I型取数类、I型存数类、I型条件判断类、J型：

1. ADDU（R型，无符号加）；
2. SUBU（R型，无符号减）；
3. SLL（R型，逻辑左移）；
4. ORI（I型计算类，与立即数（零扩展）逻辑或）；
5. LW（I型取数类，读数据进寄存器）；
6. SW（I型存数类，写数据进存储器）；
7. BEQ（I型条件判断类，有条件跳转）；
8. J（J型，无条件跳转）
9. **模块设计**
   1. **存储器设计（MEMORY）**

指令存储器（2048MB）和数据存储器（2048MB）我们均采用Verilog代码的方式生成，其中，指令存储器使用Verilog文件操作‘$readfile’读取指令。

* + - 指令存储器如下：

图形用户界面, 文本, 应用程序

描述已自动生成

* + - 数据存储器如下：

图形用户界面, 文本, 应用程序

描述已自动生成

* 1. **寄存器设计（REGFILE）**

根据MIPS CPU设计要求，我们采用Verilog代码的方式定义一个寄存器组，该寄存器组包含32个32位寄存器。其中，0号寄存器总是返回0，为0这个有用常数提供了一个简洁的编码形式：

文本

描述已自动生成

对于寄存器组，在复位信号（rst）来临时初始化所有寄存器的值为0。在后续运算时，为方便指令编写，我们不考虑各寄存器的特殊用途。

* 1. **选择器设计（MUX）**

实现8指令MIPS CPU时，我们需要设计和两种不同规格的多路选择器，但是在实际使用时，所有的命令只要求我们在两条通路中进行选择。因此，我们可以只设计规格的多路选择器来减少不必要的工作：

文本

描述已自动生成

* 1. **ALU设计**

ALU采用组合逻辑结构，8指令MIPS CPU共涉及4种操作需在ALU中进行处理，分别是：addu、subu、ori及sll。J指令的位拼接运算我们放在JOIN.v模块中进行处理，在此不做过多讨论。ALU的核心代码如下：

文本

描述已自动生成

* 1. **程序计数器设计（PC）**

PC采用时序逻辑结构，复位信号来临时初始化PC的值为0，在后续运行中，每当时钟上升沿来临则更新为下一个PC值。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

* + - 注：这里有必要对指令存储器的操作进行说明。指令在指令存储器中顺序存放，而PC值却每次加4，因此我们无法读取到正确的指令。我们通过将PC在指令存储器中进行逻辑右移操作来解决此问题：



* 1. **NPC设计**

NPC用于实现PC+4操作：

文本

描述已自动生成

* 1. **译码器设计（DECODER）**

译码器通过‘与逻辑’判断指令类型和具体指令，通过‘或逻辑’给出相应控制信号，译码器主要功能：

1. 根据输入指令划分op码，func码；
2. 根据op码（I型、J型）或op码与func码（R型）判断指令；
3. 获取寄存器地址；
4. 获取立即数；
5. 根据指令类型给出控制信号；

图形用户界面, 文本

描述已自动生成

* 1. **位扩展器设计（EXTEND）**

本次8指令MIPS指令集单周期CPU设计项目主要涉及三种位扩展运算：5位偏移量扩展、16位立即数扩展及18位立即数扩展。其中，18位立即数扩展器用于执行beq指令操作，这里不再对beq指令的具体操作进行赘述，为防止误解，我们将对18位立即数扩展器做详细介绍。

* + - EXT5:

图形用户界面, 文本, 应用程序

描述已自动生成

* + - EXT16:

图形用户界面, 文本, 应用程序

描述已自动生成

* + - EXT18:

图形用户界面, 文本, 应用程序

描述已自动生成

BEQ指令需要先将指令中的16位立即数左移两位，再与PC+4进行相加，但本次项目没有在18位扩展器中设计相应计算模块，而是通过其他方式实现：



我们通过传参时的位拼接操作代替立即数左移，得到的实际效果相同。

* 1. **位拼接器设计（JOIN）**

位拼接器用于满足J指令相关功能需求，执行PC高位与地址位拼接操作：

文本

描述已自动生成

* 1. **CPU设计**

CPU模块用于实例化及封装除指令存储器、数据存储器外的其他模块，采用Verilog代码方式连接各个模块：

文本

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

* 1. **顶层模块设计（TOP）**

顶层模块用于实例化及封装CPU、指令存储器和数据存储器，采用Verilog代码的方式连接存储器和CPU：

文本

描述已自动生成

* 1. **时序设计**

时序模块用于产生时钟脉冲及复位信号。在CPU启动时时序模块给予CPU复位信号，CPU接收到复位信号后初始化寄存器组。CPU初始化后，时序模块持续产生时钟信号，CPU在每个时钟上升沿到来时读取指令，在下个时钟周期到来前执行结束：

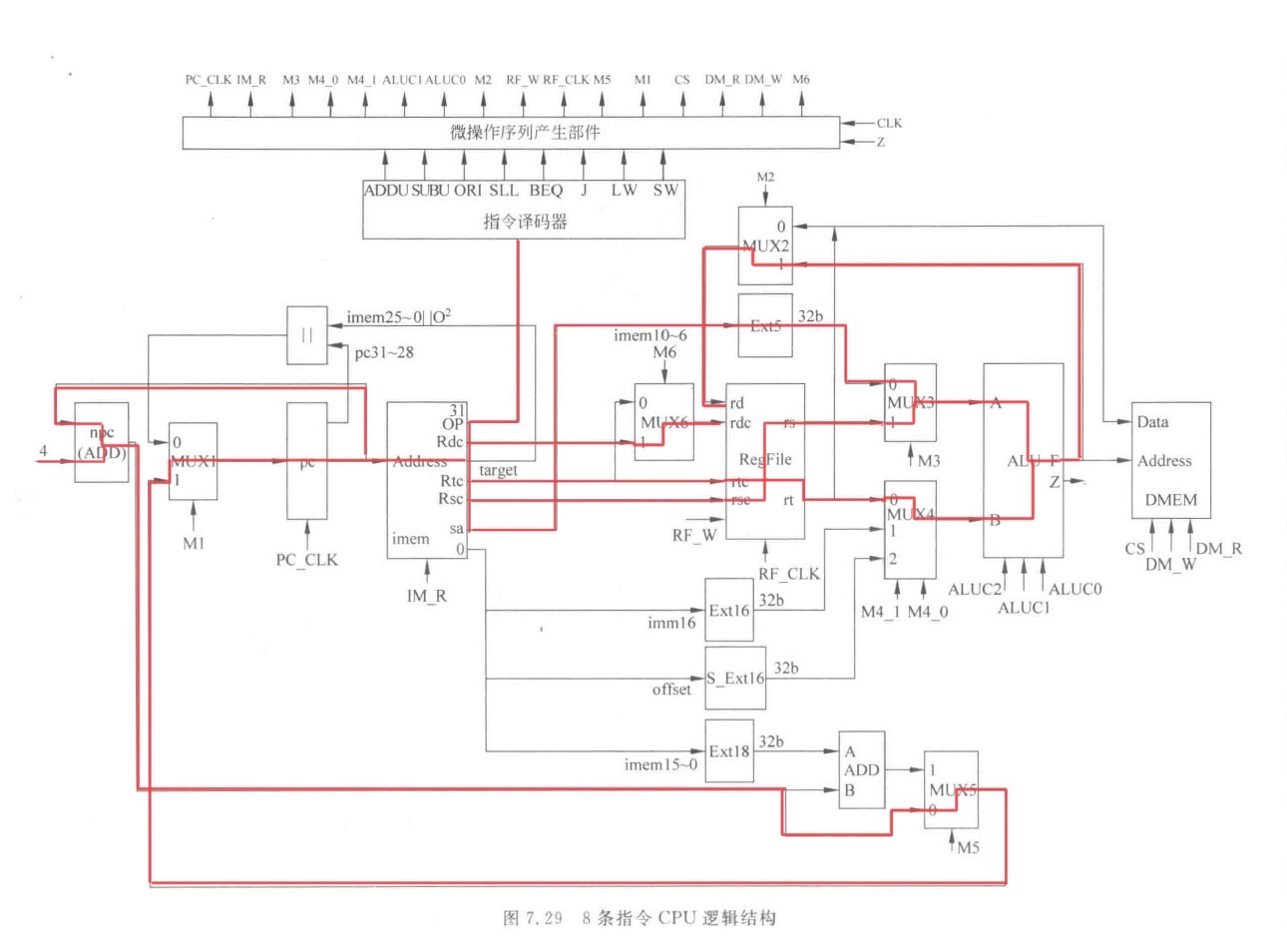
**图形用户界面, 文本, 应用程序

描述已自动生成**

1. **数据通路**

相同的指令类型具有相似的数据通路，不同指令类型数据通路如下：

* 1. **R型：**



* 1. **I型：**

**图示, 示意图

描述已自动生成**

* 1. **J型：**

图示, 示意图

描述已自动生成

1. **仿真测试**
2. **Testbench分模块测试**

在项目过程中，我们对ALU、DMEM、IMEM进行了单独测试：

* 1. **ALU - TestBench**

图片包含 应用程序

描述已自动生成

仿真结果如下：

图形用户界面, 应用程序

描述已自动生成

* 1. **IMEM - TestBench**

文本

描述已自动生成

仿真结果如下：

图形用户界面, 文本, 应用程序

描述已自动生成

* 1. **DMEM - TestBench**

文本

中度可信度描述已自动生成

仿真结果如下：

图形用户界面, 应用程序

描述已自动生成

1. **Vivado整体测试（覆盖8条指令）**

具体测试步骤如下：

1. 编写并翻译测试指令（Instruction.txt）

00000000 //初始化时给予空指令

**ori $1 $2 10**

001101 00010 00001 0000000000001010

3441000a

**ori $2 $3 5**

001101 00011 00010 0000000000000101

34620005

**sw $1 0($5)**

101011 00101 00001 0000000000000000

aca10000

**sw $2 1($5)**

101011 00101 00010 0000000000000001

aca20001

**addu $3 $1 $2**

000000 00001 00010 00011 00000 100001

00221821

**sw $3 2($5)**

101011 00101 00011 0000000000000010

aca30002

**subu $4 $1 $2**

000000 00001 00010 00100 00000 100011

00222023

**sw $4 3($5)**

101011 00101 00100 0000000000000011

aca40003

**sll $1 $2 1**

000000 00000 00010 00001 00001 000000

00020840

**lw $1 0($5)**

100011 00101 00001 0000000000000000

8ca10000

**lw $2 1($5)**

100011 00101 00010 0000000000000001

8ca20001

**j 00101**

000010 00000000000000000000000101

08000005

1. Vivado中仿真运行

测试指令中，首先使用两条ori指令为寄存器赋值（初始值为0），再通过addu、subu、sll指令进行无符号加、无符号减、逻辑左移运算，最终将计算结果写入数据存储器后跳转至首条运算指令重新开始。

仿真结果如下：（详细仿真结果见CPU\_PROJECT文件夹）

图形用户界面

描述已自动生成

1. **性能指标分析**

* 支持8条指令，单周期CPI=1；
* 时钟周期：200ns；
* 等效频率：5MHz；
* 单周期CPU中，每条指令在一个周期内完成，各部件大部分时间在等待，因此CPU的频率会受到制约，性能较低；

1. **总结**

通过本次课程设计任务，我们学习了单周期CPU的基本设计原理，对于CPU的构成有了更加深入的了解。

在设计数据通路的过程中，我们学习了各类型MIPS指令的执行过程，让我们对于精简指令集有了更加充分的认识，对精简指令集的优势有了更加直观的体会。

在编写测试指令的过程中，我们学习了有关MIPS指令集的基本汇编语言和各类型MIPS指令的算数逻辑功能、寄存器功能，掌握了有关MIPS指令集的基本知识。