ercesiMIPS 实验报告

实验一、单周期处理器

姓 名: 计 春 生
学 号: 168668
班 号: xxxxx

CS 11007 计算机组成与体系结构
(春季, 2017)

西北工业大学
计算机学院
ERCESI

2017 年 5 月 1 日

摘 要

请在这里输入摘要内容.

版 权 声 明

目 录

1 概述

这一节请主要描述组成原理课程（cs11007）中所介绍的单周期处理器的基本结构及工作机理。Latex 中文添加了 ctex 包，请使用 xelatex 编译该文件，即在对应文件夹命令行输出，xelatex ReportTemp_ch_xelatex.tex 可使用分列项目（itemize）的方法说明不同的属性介绍。例如：

- **处理器支持的指令**包括：*sub, add, or, ori, lw, sw, lst, beq, j*。
- **所有指令在一个周期内完成**在目前阶段，单周期处理器模型讲计算、控制通路分开设计，为理解处理器结构提供了良好的支持。
- **本设计包含处理器数据通路、控制逻辑以及存储结构**为了说明计算机系统结构的特点，我们推荐奖处理器的数据通路、控制通路分开设计。
- **使用 Chisel3 描述结构** Chisel 是一种强大的结构化硬件描述语言，可以对模块、接口以及操作等进行高效率的描述，与 Verilog 语言相比较，对电路的结构属性具有更好的封装，对系统的描述更加简化。我们选择 Chisel3 是因为在 Linux 系统中，使用 Chisel3 完全不再依赖工业 EDA 软件实现电路的逻辑仿真，更有利于学术研究和教学应用。虽然 Chisel 语法在不同版本差异不大，但是部分修饰符的描述方式仍然具有差别，详细区别请参考: https://github.com/ucb-bar/chisel3/wiki/Chisel3-vs-Chisel2.

2 系统设计

2.1 System Overview

请在这一节描述单周期处理器的系统设计。推荐使用图表进行说明的方法，Latex 中插入图并进行索引的方式如下：图1. 如果需要索引参考文

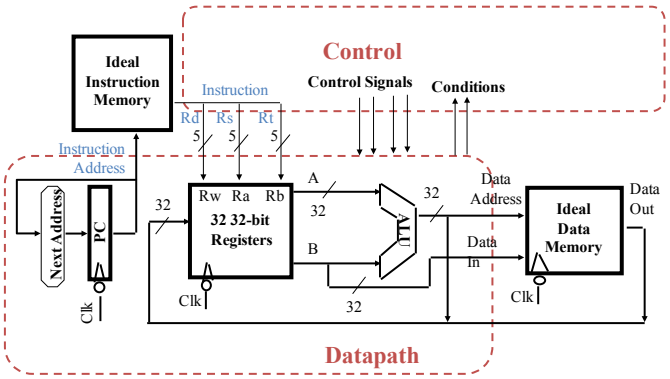

图 1: 单周期处理器结构图

献，请使用 [1], 同时已经将参考文献的项目模版在文末写出。

2.2 接口定义

请详细描述系统顶层接口信号的名称、逻辑特征、时许特性。可以建立类似表1这样的列表进行说明。

2.3 接口的（时许）逻辑

请在这个小节描述顶层结构中接口信号使用的逻辑、时序关系等。同样请利用图、表的方法进行描述。特别是时序，请画出时许图。

3 模块详细设计

这一节，主要描述各个模块的功能、接口、逻辑控制方法（状态机控制方法）等。

表 1: 测试模式信号定义

信号名	方向	位宽	功能描述
boot	Input	1-bit	触发测试模式，当处理器正常工作时被置为 0
test_im_wr	Input	1-bit	Instruction memory write enable in test mode,set to 0 in CPU regular process mode. In test mode, it will be set to 1 when if writing instructions to imem, otherwise it is set to 0.
test_im_re	Input	1-bit	Instruction memory read enable in test mode,set to 0 in CPU regular process mode. In test mode, it will be set to 1 when if reading instructions out, otherwise it is set to 0.
test_im_addr	Input	32-bit	Instruction memory address
test_im_in	Input	32-bit	Instruction memory data input for test mode.
test_im_out	Output	32-bit	Instruction memory data output for test mode.
test_dm_wr	Input	1-bit	Data memory write enable in test mode,set to 0 in CPU regular process mode. In test mode, it will be set to 1 when if writing data to dmem, otherwise it is set to 0.
test_dm_re	Input	1-bit	Data memory read enable in test mode,set to 0 in CPU regular process mode. In test mode, it will be set to 1 when if reading data out, otherwise it is set to 0.
test_dm_addr	Input	32-bit	Data memory address
test_dm_in	Input	32-bit	Data memory input for test mode.
test_dm_out	Output	32-bit	Data memory output for test mode.
valid	Output	1-bit	If CPU stopped or any exception happens, valid signal is set to 0.

3.1 ALU

3.1.1 功能描述

3.1.2 接口定义

3.1.3 逻辑控制

3.2 控制模块

3.2.1 功能描述

3.2.2 接口定义

3.2.3 逻辑控制

3.3 数据通路

3.3.1 功能描述

3.3.2 接口定义

3.3.3 逻辑控制

4 实验过程记录

记录实验的过程，完成了什么样的工作，存在的问题包括哪些，解决方案如何等。subsubsection 名称自行设定。

附录 A 代码

请在附录A中添加代码。请使用如下 Scala 的语法高亮描述方法。

```
class TopIO extends Bundle() {
  val boot = Input(Bool())
// imem and dmem interface for Tests
  val test_im_wr    = Input(Bool())
  val test_im_rd    = Input(Bool())
  val test_im_addr  = Input(UInt(32.W))
  val test_im_in    = Input(UInt(32.W))
  val test_im_out   = Output(UInt(32.W))

  val test_dm_wr    = Input(Bool())
  val test_dm_rd    = Input(Bool())
  val test_dm_addr  = Input(UInt(32.W))
  val test_dm_in    = Input(UInt(32.W))
  val test_dm_out   = Output(UInt(32.W))
```

```
  val valid     = Output(Bool())
}
class Top extends Module() {
  val io    = IO(new TopIO())//in chisel3, io must be wrapped in IO(...)
  //...
  when (io.boot & io.test_im_wr){
    imm(io.test_im_addr) := io.test_im_in
    } .elsewhen (io.boot & io.test_dm_wr){
    // please finish it
    } //...
}
```

参考文献

[1] P. Erdős, *A selection of problems and results in combinatorics*, Recent trends in combinatorics (Matrahaza, 1995), Cambridge Univ. Press, Cambridge, 2001, pp. 1–6.