北京理工大学

本科生毕业设计（论文）开题报告

**学 院：** 计算机学院

**专 业：** 物联网工程

**班 级：** 07121601

**姓 名：** 贺 清

**指导教师：** 陆慧梅

**校外指导教师：**

二○二○年一月三日

# 毕业设计（论文）选题内容

基于Chisel语⾔实现的RISC-V处理器

# 研究方案

## 本选题主要任务

1. 在FPGA开发板⼩脚丫CYC10上⽤Chisel语⾔RISC-V CPU，实现基本的指令功能，支持实现RISC-V定义的中断异常等处理，希望⽀持MMU；
2. 替换计算机系统综合实验中的verilog语⾔的CPU实现，并测试其功能正确性；
3. 对⽐verlog和chisel的实现的性能和开销特征；
4. 撰写chisel CPU的实现帮助⽂档

## 技术方案的分析、选择

1. RISC-V 始于加利福尼亚大学伯克利分校，是一种全新并完全开源的精简指令集架构，具有设计简单，模块化强，工具链完整等一系列优点，同时拥有大量开源实现与流片案例，能够在开源社区得到有力支持。通过开源代码即可配置其相应软件开发过程所需的交叉编译工具链。
2. Chisel(Constructing Hardware In a Scala Embedded Language)语言是加州大学伯克利分校开发的一种基于Scala语言的**开源硬件构造语言**，它支持使用**高度参数化的生成器**和**分层特定领域的硬件语言**的高级硬件设计。它是一种硬件构造语言(非HLS，不是C到门级电路)，Java之父James Gosling曾赞美过Scala，认为该语言为除去Java后的首选语言。Scala相比Java并不逊色，可能会更加优秀，毕竟开发者的意图是创造一门比Java更加优秀、高效、好用的语言。Scala是一门面向对象的函数式语言，它的特点正如名字一样是可自由伸缩的特性，既能裁剪已有的类，也能扩展自己想要的库；可以简简单单开发设计一个简单的脚本，也可以开发出一个复杂庞大的软件系统。开发者们可以便利的使用Scala的库来发开新的东西，比如设计一门新的语言。
3. STEP-MAX10 是小脚丫平台基于Altera公司芯片开发的FPGA开发板。核心FPGA芯片采用了Altera公司MAX10系列下的10M08SCM153。板载资源包含4路轻触按键，4路拨码开关，8路用户LED，2路RGB\_LED三色灯，此外，板卡集成了下载器，通过MicroUSB数据线即可完成开发板的供电与下载。该实验板资源少，外部接口单一，但是却能完成基本的FPGA设计要求，此次毕业设计的要求就是如何用最少的资源完成对于课程实验的支持，以节约实验学习的成本。
4. Rocket是一款64bit的标量处理器，由加州大学伯克利分校编写，具有5级流水线，采用的是risc-v指令集，集成FPU，并有许多or1200没有的特性，比如：无阻塞缓存、分支预测、返回地址堆栈、硬件页表填充、cache支持ECC、支持多核等。它允许我们生成不同配置（自定义）的SOC，包括软件工具链。它包含了由Core、Cache以及互连等构成的模块库，以此为基础构建一个完整的SoC，并且可以生成可综合的RTL代码。还有其他很多利用Chisel实现的RISC-V架构处理器，比如BOOM，RV64乱序处理器；BottleRocket，RV32IMC微处理器。还有很多使用Verilog实现的RISC-V架构的处理器，如PicoRV32，Cliffod Wolf设计的RV32处理器，是本综合实验所使用的实验系统中原有的SoC系统。
5. 任何事物都具有两面性，Chisel语言在硬件设计上既有它的好处也有它的不足，Verilog语言也是如此。使用Chisel语言编写的CPU比较Verilog语言编写的CPU在编写过程上的优势表现为重复部分可利用生成器实现，利用参数的修改来修改对应部分，不用像Verilog修改大面积的代码，并且Chisel语言代码修改面积小，可以迅速知道修改范围，便于后续的修改或回滚。
6. 利用Chisel语言重新编写CPU需要重新设计Verilog语言CPU的寄存器、存储器、中断和异常等部分，对于各个CPU功能相差不大的ALU等模块进行保留设计。
7. 初级阶段先配置好IntelliJ IDEA中关于Chisel与RISC-V的环境，利用IDEA学习Rocket等RISC-V架构CPU的实现方法，了解他们设计的5级流水的思路，学习各个不同CPU在5个阶段做的相同工作，绘制出一幅大致的流程图，并规划好参数传递的方向。获得搭载处理器的综合实验系统软硬件代码，下载Quartus程序打开系统配置相关文件，编译运行搭载原有的Verilog语言处理器的实验系统，验证系统的完整性和正确性。此外还需和维护系统的往届同学进行交流，熟悉系统的使用，掌握系统各个模块所达成的功能，了解系统是否出现问题，处理器相关文件的位置等。
8. 完成CPU的编写过后，整理顶层文件的数据的输入与输出，与CPU之外的接口对应，比较与Verilog语言处理器在学习难易度、编写难易度、模块设计思路和后期代码维护的不同，根据最后实际设计的完成的处理器对性能相关指标进行测试。
9. 可复用性，又叫可重用性，指将一部分代码重复利用。使用复用的好处在于能得到较高的生产效率和之后维护花费的成本降低并提高软件质量。实现复用的方法有很多，简单的为复制粘贴代码，每块复制代码在工程各个地方，框架一样内容不同；复杂到使用队列、栈、链表等操作。除了第8点提到的相关对比测试外，还需从代码量相关部分进行对比，设计完成的Chisel语言处理器是否在设计开发上比Verilog语言处理器更加的简洁高效。
10. 最后需要将设计完成的Chisel语言处理器整合进原有的综合实验系统中进行功能测试。本实验所使用的实验环境由清华大学提供。综合实验系统的设计者构建了这基于RISC-V架构的实验平台，并对后续CPU的替换进行了便捷性设计，使替换处理器只需修改关于处理器的Verilog语言文件，整体框架并不需要改变。之前系统中处理器为一种集成IP核中的一部分，如果要替换处理器需要更改IP核设计，复杂且繁琐，并不利于实验。后续系统维护者对操作系统方面的功能进行了修改完善，修改了BBL后移植了32位rcore操作系统，实验系统运行操作系统的功能进行了验证，使得实验系统更加稳定。SoC，全称System-on-a-Chip，是将一些关键组件集成在一块的系统或芯片。使用SoC系统可以很好的减少空间开销，使空间利用率的到提高，以方便提供更多的空间给其他模块。本次使用的综合实验系统的SoC框架中RISC-V架构处理器通过Wishbone协议总线与其他设备进行数据交互，而SDRAM和Wishbone交互需要进行数据处理，需要将32位数据与16位数据进行转换。
11. CONFIG模块是为了兼容BBL，该模块中保存了一些硬件信息提供BBL查询设置，另外根据BBL要求，timer与cmp的地址也是通过内存地址访问的，这里也一并归于此模块中。当timer达到cmp的数值时会触发一个定时器中断，直接送往处理器。
12. ROM模块是一个封装好的IP核，是保存处理器运行指令的只读存储器。该IP核的配置大小为64Kb，实际使用大小为48Kb，即程序大小必须在48Kb以下，或者32Kb以下为最优，超出该大小将无法正常使用。可以利用Quartus程序反复修改ROM中的数据来完成重新烧录新程序。
13. RAM模块，随机存储模块，是处理器中进行数据交换的存储器，可以随时读写，临时存储操作系统或程序数据。该模块可以通过开关调用宏来开启或关闭。SDRAM模块，同步动态随机存取内存模块，同步是指存储器工作需要同步时钟，内部命令的发送与数据的传输都以它为基准，所有行动都基于一个时钟；动态是指存储阵列需要不断地刷新电容电量以保证数据不丢失；随机存取是指数据不是线性依次存取，而是随机地址进行读/写。
14. UART模块，串口模块，通过接受或者发送数据来与上位机进行交互。此模块设置为停止位1位，无校验位，波特率115200，在仿真的时候需要在接受端模拟一个串口。
15. LED模块，简单的开发板LED控制模块，通过变化LED的二进制码来进行简易的LED亮灭控制，通过LED的亮灭传播简易的信息。

## 实施技术方案所需的条件

1. 硬件：STEP-MAX10-小脚丫FPGA开发板
2. 软件：IntelliJ IDEA、Verilator、Quartus II

## 存在的主要问题和技术关键

1. 对Chisel语言了解较少，设计生成器会有些难度。
2. 现有的Chisel语言编写的RISC-V，如Rocket，功能大，逻辑严密，学习困难，需要花大量时间理解并设计一套适合自己的架构方案。
3. 需要掌握综合试验系统的使用，了解系统中各个模块的功能，根据所设计的Chisel语言处理器对相应的参数进行调整以让处理器适配系统。如果系统与处理器交互发生问题还需进行调试修改。

## 预期能够达到的研究目标

1. 实现部分RISC-V指令集：RV32I+M+A指令集(基本整数集+标准整数乘除法扩展集+标准操作原子扩展集)
2. 实现异常、自陷、中断的处理
3. 实现内存管理MMU
4. 整合进综合实验系统并工作正常

# 课题计划进度表

1. 一月：熟悉RISC-V架构，学习Chisel语言，配置Chisel开发环境，配置RISCV工具链和安装软件，搭建综合实验平台环境。
2. 二月至三月中旬：完成CPU的编写
3. 三月下旬至四月初：完成CPU的移植，并完成与Verilog语言CPU的功能对比工作
4. 四月下旬到五月：编写修订完成论文，准备答辩

# 参考文献

1. RISC-V: The Free and Open RISC Instruction Set Architecture
2. Reusability is FIRRTL Ground:Hardware Construction Languages, Compiler Frameworks, and Transformations
3. Lee Y, Waterman A, Cook H, et al. An Agile Approach to Building RISC-V Microprocessors[J]. IEEE Micro, 2016, 36(2): 8-20.
4. Sharat K, Bandishte S, Varghese K, et al. A Custom Designed RISC-V ISA Compatible Processor for SoC[C]. vlsi design and test, 2017: 570-577.
5. The RISC-V Reader: An Open Architecture Atlas David Patterson, Andrew Waterman Edition: 1st
6. 《Chisel Tutorials (Release branch)》
7. Chisel 3 wiki
8. Short Users Guide to Chisel
9. step-max10\_硬件手册v1.2
10. step-max10\_nios\_ii搭建教程