中国科学技术大学

工程硕士研究生学位论文

开题报告

论文题目：RISC-V 指令集模拟器的设计与实现

学 生 姓 名：王昊

学 校 导 师：汪增福

企 业 导 师：侯锐

工 程 领 域：软件工程

领 域 代 码：085212

研 究 方 向：应用系统设计

所 在 院 系：软件学院

实 习 单 位：中国科学院信息工程研究所

中国科学技术大学研究生院

填表日期：　　年　月　日

说 明

1. 工程硕士学位论文的开题报告是保证论文质量的一个重要环节，为了加强对工程硕士研究生培养的过程管理，规范其学位论文的开题报告，特制此表。
2. 工程硕士学位论文开题报告，应该在工程硕士学位授予点或培养单位组织的学术报告会上报告，听取意见，论证后再填写此表。
3. 此表一式两份经导师和培养单位负责人签字后，交培养单位研究生教学管理办公室存档。
4. 工程硕士研究生在申请学位论文答辩时，必须提交该学位论文开题报告。

一、 简况

<table>
<tr><td rowspan="5">研究生简况</td><td>学　号</td><td colspan="2">SA19225360</td><td>姓 名</td><td>王昊</td><td>姓名拼音</td><td>WangHao</td></tr>
<tr><td>性　别</td><td>男</td><td>身份证号</td><td colspan="2">321283199703212818</td><td>出生年月</td><td>1997.03.21</td></tr>
<tr><td>工程领域</td><td colspan="3">软件工程</td><td>研究方向</td><td colspan="2">软件系统设计</td></tr>
<tr><td>入学时间</td><td colspan="2">2019.09.06</td><td>录取方式</td><td>统考</td><td>培养方式</td><td>脱产</td></tr>
<tr><td>本科毕业时间</td><td colspan="2">2019.06.30</td><td>本科毕业学校</td><td>南京工程学院</td><td>本科专业</td><td>电气工程及其自动化</td></tr>
<tr><td rowspan="4">论文类型与性质</td><td rowspan="2">名称</td><td>中文</td><td colspan="5">RISC-V 指令集模拟器的设计与实现</td></tr>
<tr><td>英文</td><td colspan="5">Design and implementation of RISC-V instruction set simulator</td></tr>
<tr><td>类别</td><td colspan="6">工程设计或技术改造</td></tr>
<tr><td>形式</td><td colspan="6">1．工程设计 √
2．研究论文</td></tr>
<tr><td>性质</td><td colspan="6">1．应用技术研究 √
2．技术开发</td></tr>
<tr><td rowspan="4">论文内容和意义</td><td>摘要</td><td colspan="6">spike是RISC-V开源社区的指令集模拟器,能够模拟实际代码执行过程中的软硬件行为，提供指令级别的仿真，当前最新版spike仅支持少数设备。
本课题在spike的基础之上，添加一些在实际工作中迫切需要的功能模块，包括可视化调试界面的设计，模拟器设备的拓展，包括平台级中断控制器PLIC，串行外设接口SPI，mailbox，块设备等等，这部分拓展工作可以使spike支持更多的外设，大大降低系统软件和设备驱动程序移植过程中的成本，也对后续RISC-V架构芯片设计工作提供部分通用的模拟器接口。其次，本课题还对spike串口调试模块进行了优化，支持了更多的断点触发条件，并且提供实时的处理器状态检测，这部分优化工作丰富了调试手段，使得模拟器调试过程更加便捷高效。</td></tr>
<tr><td rowspan="3">主题词</td><td colspan="6">主题词数量不多于三个，主题词之间空一格（英文用“/ ”分隔）</td></tr>
<tr><td>中文</td><td colspan="5">RISC-V 模拟器</td></tr>
<tr><td>英文</td><td colspan="5">RISC-V/simulator</td></tr>
</table>

二、选题依据

1. 阐述该选题的研究意义，或工程设计的价值和意义，国内外概况和发展趋势，选题的先进性和实用性，技术难度及工作量。

1.1 选题的研究意义

RISC-V指令集于2010年在伯克利诞生，虽然至今才短短十年的时间，但是依靠其优越的设计理念，充分发挥了后发优势。再加上开源采用宽松的BSD协议，基于 RISC-V 指令集架构既可以去设计开源的处理器，也支持商业模式，开发周期也相对短暂，无论是对中小型公司和个人设计者，还是对知名的芯片公司和互联网企业，都可以作为开发产品的好选择。

众所周知，芯片产业是投入极高、回报极慢的领域，而 RISC-V 提供了免费开源、开发周期较短的解决方案。面对国外芯片的生态和专利壁垒，RISC-V 有望成为我国自主研制处理器芯片的极好的选择，或许能为国产芯片市场的繁荣提供机遇。

在芯片设计及验证的流程中，对于基础系统软件尤其是操作系统，底层驱动等的适配和验证往往是反馈硬件设计缺陷最频繁的部分，这部分的工作不仅是对于前期硬件设计的重要测试，也是后续用户态程序开发的基础。对于系统软件的移植和适配工作，目前有两种方式，一种是在模拟芯片硬件特性的FPGA开发板上仿真，另一种是通过软件模拟。两种方法各有利弊，FPGA开发板更加接近真实硬件环境，能够获取精确的仿真信号，但是速度相对较慢，并且能够提供的调试信息较少。而模拟器环境下的开发，其运行速度接近宿主机，并且调试方便，虽然信号精度与真实硬件有差异，但是能够在测试的前期反馈大部分的缺陷。所以真实的开发和测试流程一般是先使用模拟器验证，再上FPGA平台仿真，这样既能够提高开发效率，又不失精度。

目前RISC-V开源社区的指令集模拟器是spike，由SiFive公司维护，spike模拟实际代码执行过程中的软硬件行为，提供指令级别的仿真，本质上就是C++程序模拟每条指令执行过程中的软硬件行为，当前最新版本的spike模拟的设备包括总线，内存，时钟，处理器，和调试模块，这些部分能够使得spike运行一个简单的代理内核，并运行一些简单的RISC-V指令集架构的程序。但是对于真正进行RISC-V芯片设计工作的团队来说，spike模拟的内容还远远不够。真实的设计要复杂得多。

本课题在spike的基础之上，添加一些在实际工作中迫切需要的功能，包括可视化调试界面的设计，模拟器设备的拓展，比如平台级中断控制器 (Platform-Level Interrupt Controller, PLIC)，mailbox，串行外设接口 (Serial Peripheral Interface, SPI)，块设备等等，对于通用处理器设计过程中需要用到的主要设备添加模拟器支持可以大大降低系统软件移植工作的成

本，也对后续RISC-V架构芯片设计提供通用的模拟器接口。

1.2 RISC-V 指令集架构国内外研究历史与现状

1.2.1 ISA 指令集架构的定义

计算机只能识别和执行机器指令，也就是二进制代码，而现在大家都使用便于理解的高级语言，比如 C、JAVA 等，来编写程序，从而方便地实现我们想要的功能。想要将对人类来说可读性好的语言翻译成机器语言，让机器可以理解相应的操作，这就需要设计所谓的指令集，基于指令集去构建相应的指令集架构（Instruction Set Architecture，ISA）。由高级语言编写的程序，通过相应的某种编译器，根据相应的指令集架构，可以被翻译成能被机器识别的指令。而机器如何执行指令，一条指令如何在硬件上实现它所表述的操作，这就是所谓的微处理器体系架构。具体地从物理上实现某种指令有多种方式，也就是说，一种指令集架构可以对应多种微处理器结构

1.2.2 RISC-V指令集架构的起源和发展历史

最初的指令集的提出是基于当初的计算机性能不足、需要提高运算效率的需求，指令集的出现大大提高了设计者们的开发效率，也是出于这个原因，人们希望一条指令可以控制计算机处理尽量多的操作，这就使得指令变得越来越复杂，指令集的体量随之大幅增长。然而，复杂的指令越来越多，其使用率实则远低于简单的指令，过于全面的指令集反而成了一种负担。出于这种新的需求，人们设计了新的指令集，只保留了原来的指令集中被反复使用的简单指令，指令集被大大的简化了。为了区分这两种指令集，之前相对复杂的指令集被称为 CISC（Complex Instruction Set Computer，复杂指令集计算机），后来精简后的指令集被称为 RISC（Reduced Instruction Set Computer，精简指令集计算机）[1]。从硬件的角度上，为了实现相应的指令集，处理器的设计也基于这两个体系各自向前发展。 发展到今天，常见的处理器基本上被 x86 与 ARM 这两种指令集架构所占据。我们所使用的个人电脑绝大多数都是基于 x86 架构的，而 x86 架构其实归属于CISC。Intel 公司在 1978 年推出了第一个基于 x86 指令集架构的 CPU“8086”[2]。1981 年，IBM 公司推出了基于 Intel 的 x86 处理器的个人电脑[3]。20 世纪 80 年代后，基于 Intel 的处理器的个人电脑上普遍安装微软的 Windows 操作系统，二者几乎组成了固定搭配，硬件设计与软件开发齐头并进，迅速占据了绝大部分市场份额，如今，个人电脑的用户已经习惯了这个搭配，也习惯了四十年来发展起来的完善的生态环境[4]。但多年以来的发展也使得 x86 架构越来越复杂和冗余，为了保持兼容性，设计者们研发新的处理器时，不仅要实现现有的 ISA，还要去实现所有的过往的 ISA 扩展，哪怕那已经过时甚至是错误的。为此，基于 CISC 的 x86 处理器也会借鉴和引入一些 RISC 的思想，从而对自身做出改进[5]。

ARM 架构则广泛应用于移动通讯领域，目前我们所使用的手机基本都是基于ARM 架构。ARM 指令集架构属于 RISC 体系，ARM 公司于 1985 年推出了第一款ARM 处理器内核[6]。ARM 选择提供 IP 核，授权给其他芯片公司去生产处理器芯片的开放模式，逐渐形成基于 ARM 架构的生态圈，占据了服务器、嵌入式和移动终端的市场。谷歌推出的安卓系统，苹果、高通、三星公司推出的处理器，基本都是基于 ARM 指令集架构来实现的。

漫长的发展过程使 x86 和 ARM 架构成为非常成熟的架构，也使得它们难以避免地出现许多问题，不论是其指令集的复杂和冗余，抑或是昂贵的商业授权，都对设计者造成了极大的限制，更何况 x86 处理器和 ARM 处理器的源码是限制修改而且难以获取的。对免费和开放的处理器架构的需求，使各种开源架构进入了设计者们的视线。

Sun Microsystems 公司于 1986 年推出了 SPARC 处理器，SPARC 架构完全对外开放，成为经典的 RISC 处理器体系架构之一[7]。SPARC 处理器曾一度统治过服务器和工作站市场，但没能对 x86 处理器和 Windows 系统这个搭配挑战成功，后来逐渐消失在人们的视野当中。

开源社区 Opencores 基于 RISC 指令集架构推出了面向嵌入式系统的OpenRISC 处理器，其源代码开放，且提供了一套完整的开发工具链，许多商业公司和研究机构都将 OpenRISC 体系架构引入了自己的开发和研究中去[8,9]。但是作为一个社区推动的项目，OpenRISC 有些缺乏管理，各种文档的跟进和更新不够及时，再加上 GPL（GNU General Public License，GNU 通用公共许可证）协议下开源的项目无法做到商业化，都在一定程度上制约OpenRISC 的发展。

后来，全新的指令集架构 RISC-V 于2010年诞生在加州大学伯克利分校，Krste Asanović 教授等开发人员最初推出 RISC-V 架构其实是为了开展课程和研究，之后这个优秀的指令集架构渐渐吸引了外部人员的注意[10-12]。2015 年，RISC-V 基金会启动，旨在推动基于 RISC-V 架构的软件设计和硬件开发的协同创新[12]。2018 年RISC-V 基金会宣布与 Linux 基金会合作，得到了 Linux 这个庞大的开源软件生态的支持[13]。发展到今天，RISC-V 基金会的三百多个成员遍布全球各地，阿里巴巴、谷歌、高通、三星、西部数据等等都是其中的成员[12]。

RISC-V 为软件开发者提供了免费的指令集规范，硬件设计者根据规范去实现满足所需的处理器，这种通过开放标准和开放源代码来让广大设计者参与处理器创新的方式极大地降低了处理器设计的门槛，有望在新兴的 AI 与 IoT 领域中对ARM 的统治地位形成挑战，也为中国在芯片设计领域打破国外技术垄断提供了很好的机会。

1.2.3 RISC-V指令集架构的应用

RISC-V 指令集架构的官方处理器实现项目名为 Rocket Chip，一个完全开源的RISC-V 处理器生成器，由加州大学伯克利分校的 RISC-V 研发团队推出，也是最符合规范的硬件实现，源代

码托管在 GitHub 平台上。Rocket Chip 是一个开源的项目，使用开源许可证 BSD（Berkeley Software Distribution，伯克利软件套件），用 Chisel（Constructing Hardware In a Scala Embedded Language，嵌入 Scala 的硬件描述语言）编写，可以用来生成 Rocket 内核和 BOOM 内核。该团队还推出了基于 Rocket Chip 的开源 SoC 项目 freedom，其源代码同样托管在 GitHub 平台上。

2015 年，加州大学伯克利分校的 RISC-V 研发团队成立了商业公司 SiFive，推出了可由客户自定义的RISC-V IP核，以及同样支持定制的SoC平台，包括针对低功耗需求的 Freedom Aware 平台，面向 IoT 领域的 Freedom Everywhere 平台，针对高性能需求的 Freedom Unleashed 平台，以及面向 AI 领域的 Freedom Revolution平台[14]。

晶心科技推出了一系列嵌入式处理器和 SoC 平台，将 RISC-V 吸纳进自己的AndeStarTMV5指令集架构，推出了 AndesCore TM 产品线，包括 N25F，NX25/NX25F，A25 和 AX25 处理器内核，推出了 FreeStart AE250、Standard AE250 和 AE350 等SoC 平台，建立了一套完善的软件和硬件开发环境[15]。

越来越多的国家也开始对 RISC-V 重视起来，RISC-V 本身就是在美国国防高级研究计划局（Defense Advanced Research Projects Agency，DARPA）资助的项目中出现的，印度的几个自研处理器的政府项目都转向了 RISC-V 架构[16]。在中国，RISC-V 也引起了设计者们的极大兴趣。

2018年11月，中国开放指令生态联盟（China RISC-V Alliance，CRAV）成立，旨在基于 RISC-V 建立起中国的开源芯片生态，多个研究机构、企业、院校和个人设计者都是其成员[17]。

蜂鸟E203是中国第一个开源 RISC-V 处理器内核，其源代码托管在 GitHub平台上，为国内 RISC-V 处理器的研究者和院校提供了极好的实践案例，推动了国内 RISC-V 生态的发展[18]。 后来，蜂鸟E203的开发者胡振波创立了芯来科技，推出了 N100、N200、N300、N600、N900 等多个系列的 RISC-V 处理器，为多家企业提供了定制RISC-V处理器IP核。芯来科技与兆易创新合作，为其定制了 RISC-V 处理器内核 Bumblebee，兆易创新基于此推出了GD32VF103系列的量产通用微处理器。

阿里平头哥也致力于将 RISC-V 架构应用于自研的玄铁系列处理器上，推出了64 位 RISC-V 处理器内核 C910 和兼容 RISC-V 的 E902，在 AIoT 领域的生态中占据了重要位置[19]。

此外，华米科技推出了针对可穿戴设备的基于 RISC-V 的 AI 处理器黄山 1 号，于 2019 年 8 月宣布量产，并应用于商业产品 Amazfit 米动健康手环上。

上海交通大学与瓶钵信息科技合作研发了“蓬莱”安全架构，一个可应用于RISC-V 平台的开源 TEE（Trusted Execution Environment，可信执行环境）系统，同时也丰富了 RISC-V 的指令集扩展，现托管在 GitHub 平台上[20]。

1.3 选题的先进性和实用性

RISC-V在众多ISA中脱颖而出，不仅是因为其后发优势，提出模块化设计的概念，而且因为它开源，包容，提供了高度灵活的配置空间，在积极拥抱开源软件的今天，RISC-V已经成为未来的主流架构，因此对于RISC-V的学习和研究非常有意义,也具有一定的先进性。

此外，面向RISC-V的开源软件生态需要不断完善，包括各主流软件的RISC-V移植，这些工作大部分都需要在模拟器的环境中做，所以该选题对于spike模拟器的优化和拓展能够对于上述工作提供便利，具有很强的实用性。

1.4 选题的技术难度和工作量

该选题是对已有开源社区工具的优化和拓展，涉及要模拟的设备较多，不仅需要对于RISC-V指令集架构有较为深入的理解，而且需要对于RISC-V体系结构的各个功能模块，比如中断响应过程，多核并行处理等有细致的研究，具体到模拟器各功能模块:平台级中断控制器PLIC，串行外设接口SPI，块设备，并且处理模块间的交互，具有一定的技术难度。

对于spike调试模块的优化，包括调试界面的UI设计，寄存器，内存，指令格式等触发条件的支持，实时特权级状态监测，外部中断的主动触发选项等等，加之与上述各功能模块的集成和调试，该选题作为硕士毕业设计工作量适中。

2. 主要参考文献（列出作者、论文名称、期刊名称、出版年月）

[1] W.Stallings. Reduced instruction set computer architecture[J]. Proceedings of the IEEE, 1988, 76(1): 38-55

[2] S.P.Morse, W.B.Pohlman, B.W.Ravenel. The Intel 8086 Microprocessor: a 16-bit Evolution of the 8080[J]. Computer, 1978, 11(6): 0-27

[3] 黄绍平. 个人电脑:走向二十一世纪[J]. 机电新产品导报, 1995(07): 41-42+45.

[4] R. Casadesus-Masanell, D.B.Yoffie. Wintel:Cooperation and Conflict[J]. Management Science, 2007, 53(4): 584-598

[5] 郑飞. 超标量与超流水线混合结构微处理器 Pentium[J]. 微处理机, 1994(4): 13-16

[6] 丛欣. ARM 处理器在网络安全领域中的应用[J]. 信息安全与通信保密, 2011(5): 41-42

[7] 施蕾, 刘波, 周凯. 基于 SPARC V8 结构处理器的计算机系统设计[J]. 空间控制技术与应用, 2008(03): 48-52

[8] F.Conti, D.Rossi, A.Pullini, et al. PULP:A Ultra-Low Power Parallel Accelerator for Energy-Efficient and Flexible Embedded Vision[J]. Journal of Signal Processing Systems, 2015, 84(3): 339-354

[9] 陈曦，黄毅. 片上系统设计思想与源代码分析(附光盘)[M]. 电子工业出版社，2008，9-48

[10] D.A.Patterson, A. Waterman. RISC-V 手册[EB/OL]. http://crva.ict.ac.cn/documents/RISC-V- Reader-Chinese-v2p1.pdf, 2018, 3-7

[11] 雷思磊. RISC-V 架构的开源处理器及SoC研究综述[J]. 单片机与嵌入式系统应用，2017, 37-38

[12] RISC-V Foundation. RISC-V History[EB/OL]. https://riscv.org/risc-v-history/, 2020, 3-4

[13] The Linux Foundation. The Linux Foundation and RISC-V Foundation Announce Joint Collaboration to Enable a New Era of Open Architecture[EB/OL], Nov 2018, 13-16

[14] 邓亚威. 引领芯片定制化革命——SiFive2018上海技术研讨会圆满召开[J]。 中国集成电路, 2018(6): 17-18+26

[15] Andes Technology.RISC-V@Andes[EB/OL], 2020, 2-4+6

[16] 包云岗. 关于RISC-V成为印度国家指令集的一些看法[J]. 中国计算机学会通讯, 2018，14(1): 38-44

[17] China RISC-V Alliance. 中国开放指令生态（RISC-V）联盟于第五届互联网大会宣布正式成立[EB/OL]. http://crva.ict.ac.cn/?page_id=107, 2018，11月8日

[18] 胡振波. 手把手教你设计 CPU——RISC-V 处理器篇[M]. 北京:人民邮电出版社, 2018, 3-6

[19] Nuclei System Technology. 客户案例: 兆易创新GD32V系列RISC-V通用MCU[EB/OL]. https://www.nucleisys.com/product.php?site=lcxp

[20] 瓶钵信息科技. 重磅！RISC-V 平台的可信执行环境“蓬莱”正式开源[EB/OL]. https://baijiahao.baidu.com/s?id=1654577121362062020&wfr=spider&for=pc，2020

三、课题内容及具体方案

1．课题内容

（1）平台级中断控制器 PLIC 设备模块的实现

（2）外接设备模块(uart/mailbox/flash)的实现

（3）串口调试模块的实现

（4）各部分功能模块的集成

（5）与 FPGA 开发板的对照测试

2． 系统需求分析（应用软件工程专业描述工具描述）

2.1 概述

随着 RISC-V 开源社区的日益壮大,更多的芯片设计厂商选择 RISC-V 作为其指令集架构，在芯片的验证过程中，软硬件适配工作作为测试的重点，往往需要模拟器环境的支持，当前开源社区的指令集模拟器 spike 由 SiFive 公司维护，能够定期地更新 RISC-V 的最新特权级指令，是各厂商优先考虑使用的基础模拟器。厂商可以根据自身产品特性，拓展 spike 支持的设备，并在此基础之上进行软件移植工作和前期软硬件设配工作，以此来提高芯片验证与测试工作的效率。因此，模拟器的优化与拓展，应该在硬件层面上与待验证芯片的硬件特性相符合，又要在软件层面上满足调试功能易用性的要求。

2.2 功能需求

spike 模拟器的运行过程包括三个部分，模拟器自身部分，面向开发者(用户)的串口调试部分，以及目标程序部分。目标程序一般指的是移植到 RISC-V 平台的通用基础软件，如 linux 内核，处理器所支持外设的驱动程序等，这部分程序涉及到的资源就是模拟器应该支持的部分。因此，本次课题涉及到的需求分为两个部分，一个就是对上述功能模块提供模拟器支持的需求，另一部分是优化串口调试模块的需求。

2.2.1 平台级中断控制器 PLIC

RISC-V 核心(hart)包含 local 中断源和 global 中断源。只有 global 中断源可以被 PLIC Core 响应，通常为 I/O 设备，PLIC 负责将中断源(global interrupt sources)链接到中断对象(interrupt targets)，也就是处理器核心。spike 自身不支持 PLIC 设备，所以对于外部中断的支持只能通过轮询来实现，效率相当低下，并且不符合实际处理器的特性。因此，首先对 spike 的拓展便是 PLIC 的支持，对于 uart，spi 等串口控制器，可以在后续通过设备树直接挂载为 PLIC 的中断源。多个外设作为独立的中断源，通过 PLIC Core 仲裁，将外部中断信号传递给相应的核心。PLIC 中断控制流程如图 1 所示。

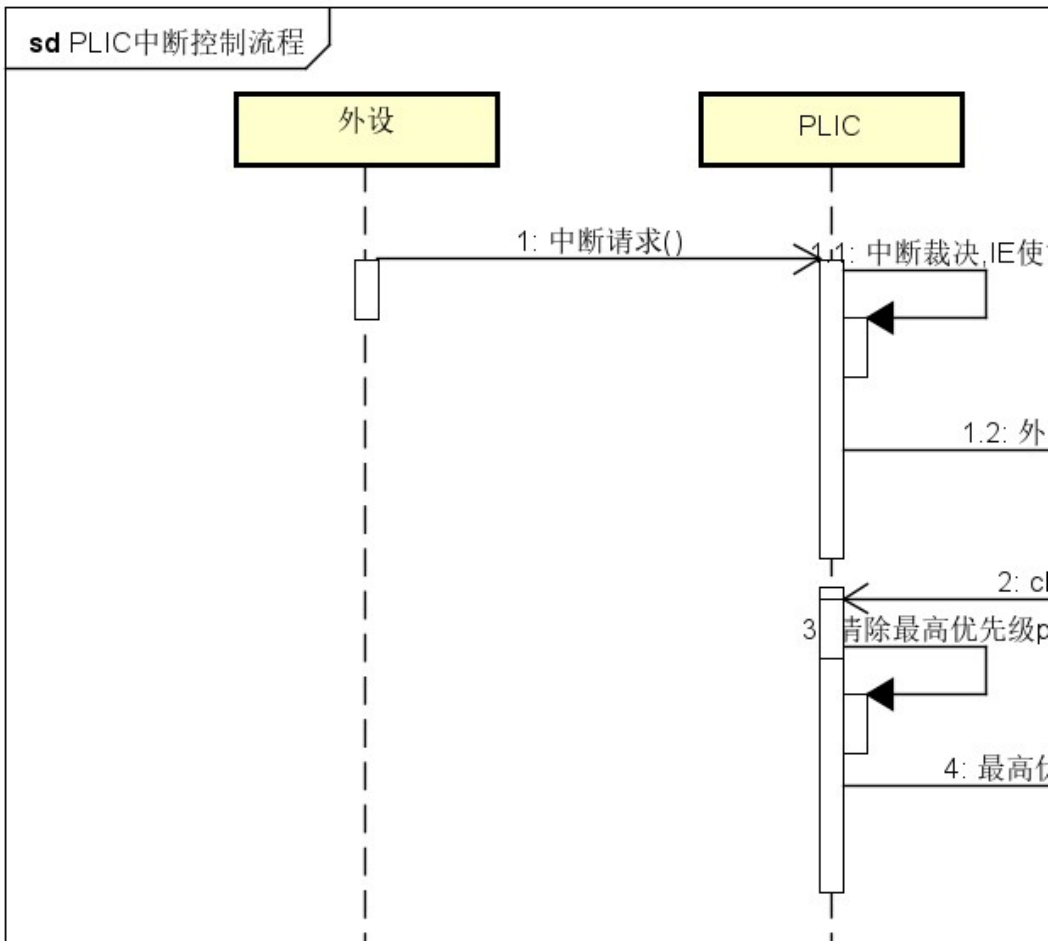


图 1 PLIC 中断控制流程

对 PLIC 设备模块的需求是，模拟器启动 linux 内核，并且 uart 以中断号 2 成功挂载到 PLIC，通过查看/proc/interrupts 进行验证，具体的设计见系统概要设计部分。

2.2.2 uart16550 串口控制器

spike 提供 HTIF 模拟串口通信控制器，HTIF 是伯克利处理器的非标准工具，因此没有文档。随着 RISC-V 平台规范的发布和内核的更新为自托管，HTIF 很快就会消失。HTIF 是主机/目标接口，它通过 riscv-fesvr 与目标设计（Sodor）通信。riscv-fesvr 通过 HTIF mem 端口将二进制文件加载到 Sodor 内存中，然后通过状态寄存器告知内核。程序完成后，Sodor 告诉 riscv-fesvr，它通过重置主机 CSR 完成交互，模拟结束。

在 bootloader 加载内核之前，串口通信控制器就必须能够工作，提供裸机的交互功能，是前期进行串口调试的重要工具，能够结合 spike 的单步调试功能完成前期的大部分调试工作，其实现的优先级很高。

总体来说,HTIF 提供了串口通信的功能,但是其本质上还是需要主机的轮询查询,效率很低,考虑到实际硬件设计中必不可少的串口通信控制模块(uart16550a 芯片),需要为 spike 添加 uart 模拟,通过外部中断的方式实现串口通信.也可以在模拟器上完成对 uart 的前期测试,包括波特率的设置,传输模式的选择等等.

2.2.3 mailbox 核间通信模块

由于本次设计涉及到的芯片是一款主动安全处理器芯片,通过和其他厂商计算核心的互联发挥作用,核间的 mailbox 通信部分至关重要. 区别于 ipi 处理器内部中断方式, mailbox 是一种框架，通过消息队列和中断驱动信号处理多处理器间的通讯.

对 mailbox 的模拟,需要满足高并发的异步通信需求,鉴于无法真实模拟与另一个计算核心的通信, 需要对模拟器核心间通信添加 mailbox 支持, 来模拟真实的交互环境.

2.2.2 串口调试模块

spike 本身具备单步调试的功能，但是在模拟真实硬件行为的复杂场景下，命令行式的单步调试显然难以使用，根据以往的调试经验，人性化的 UI 设计需要涉及到处理器状态寄存器，当前特权级模式，程序运行窗口等信息，还需要设置对应多核的寄存器触发条件，实现 mailbox 中断主动触发，保存快照方便复现 bug 等功能。

这部分的功能需求主要有:

(1) 对于多核心独立 csr 寄存器触发条件的支持

(2) 对内存/指令格式等触发条件的支持

(3) 能够查询历史指令序列执行情况

(4) 实时监测状态寄存器，当前特权级模式

(5) 主动发送 mailbox 中断信号

2.3 非功能需求

UI 设计的易用性, 体现在调试窗口简单明了, 串口调试时的常用指令能够以图标的形式设计, 简化命令行的操作, 指令流的显示窗口需要能够容纳至少 1000 条历史指令记录,以便后续调试定位.关于故障的复现, 需要能够为模拟器添加快照, 省去 reset 之后再复现故障的麻烦.

3. 系统概要设计（应用软件工程专业描述工具描述）

3.1 平台级中断控制器 PLIC 结构设计

PLIC Core 的结构如图 2 所示。

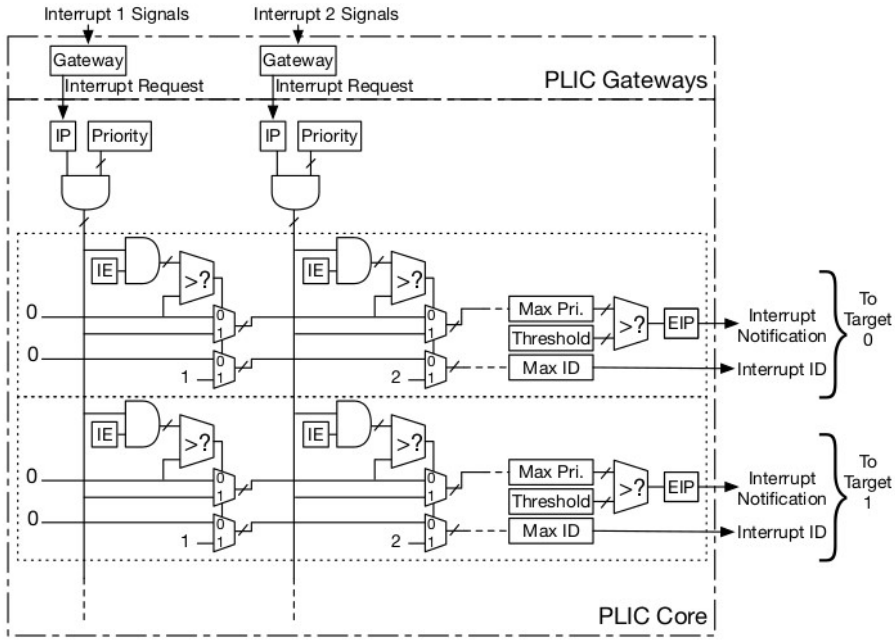


图 2 PLIC Core 结构图

主要的功能组件说明如下:

中断源(interrupt sources)，可以使用队列来储存多个中断信号。所有global interrupt sources都会被转为PLIC core能接受的标准形式，每个interrupt source都会被赋予一个从1开始的无符号整数作为标识（Interrupt Identifiers ID）。该ID也作为在多个sources具有相同优先级是的选择条件：数值较小的优先于较大的。每个interrupt source都会绑定一个与平台相关的优先级寄存器，数值不能为0。硬件也可以选择将优先级直接固定为硬连接。每个interrupt source都被赋予一个使能位(enable bit)IE，储存在平台相关的寄存器中。

中断处理终端(interrupt targets)，对应为RISC-V处理器核心的各个特权级模式。对应M/H/S/U四种模式，PLIC产生的interrupt notification会分别标示在target的mip/hip/sip/uip 寄存器的meip/heip/seip/ueip位。PLIC不负责处理中断抢占和嵌套，由interrupt target处理上述问题。中断处理终端对应平台相关的priority threshold寄存器，

只有高于该threshold的active interrupt才会发送给对应target。threshold必须支持0，表示没有中断被屏蔽；通常也需要支持max priority level，表示所有中断都会被屏蔽。

interrupt gateways，这个类似中断源和PLIC core的中继，负责将sources的各类interrupt转换为PLIC core的通用格式，同时控制向PLIC core发起request的整个流程。gateway在收到interrupt target完成中断服务的notification后，发送新的interrupt request 到 PLIC core。在收到interrupt completion之前，gateway不会再向PLIC core发送新的请求，如果收到interrupt completion后，source的request还拉高着，会被作为新的interrupt request。如果source request在PLIC core收到request后，target处理该request之前拉低，这个request仍然会储存在PLIC core的IP位中，即仍然会被target所响应。

PLIC Core，负责所有中断请求的仲裁和分发，任何时候，最多只能有一个pending interrupt request存在于PLIC Core中，由对应IP位保存PLIC Core对每个interrupt source都会被赋予一个独立的priority和ID标识，PLIC Core包含一个matrix of interrupt enable bits IE，用以控制中断使能。PLIC Core对每个interrupt target都赋予一个独立的平台相关的priority threshold寄存器，用以控制中断发生的门限中断请求(interrupt request)。

interrupt notification，由PLIC Core 发给各target的中断请求，PLIC Core为每个target赋予了一个(external interrupt pending bit) EIP，表示有待处理的中断。EIP的值可以被source，target或者其他源进行修改。EIP发送给target的过程被称为interrupt notification。PLIC Core只支持multicasting，即中断信号会发给所有符合条件的target，并不会选择其中之一发送。这样虽然响应较快，但会带来一些冗余处理。软件可以通过控制IE 位间接改善。首先claim的target会负责该中断的处理，PLIC Core只保证对于EIP的修改所有对应的targets都可见。

interrupt claim，由target返回给PLIC Core的响应信号，表示请求接受。PLIC Core收到claim后,会选出最高优先级的source ID,并将其对应的IP清除。这个ID会发送给target。如果ID为0，表示没有需要处理的中断。claim被PLIC Core接受后，次优先级的中断会显现出来，因此对应的EIP可能并不会被清0。因此target可以在退出中断服务程序前检查本地Xeip 位(X表示当前特权模式)，确认是否有pending 的EIP。PLIC支持target在EIP=0的情况下仍然发起claim。这是为了支持某些target的threshold设置为最高，不接受中断打断，但使用claim方式进行查询。

interrupt completion，由target 返回给PLIC Core的完成响应信号。

global interrupt handling，基本过程是，global sources首先发送给interrupt

gateway，由gateway负责产生interrupt request，发送给PLIC Core， PLIC Core将每个interrupt request储存在内部的(interrupt pending bits) IP中，如果该targets有enabled pending interrupt， 并且priority超过了per-target threshold ，PLIC Core将interrupt notification发送给一个或者多个targets。当target接受了该外部中断，会发送一个interrupt claim request给PLIC Core，用以取得对应该target的最高优先级的pending interrupt，同时将对应的IP位清零，当target完成了中断服务，需发送一个interrupt completion message给对应的interrupt gateway表示中断完成。接下来gateway可以发送另一个interrupt request给该target。

spike中提供了用于实现外部设备的抽象类abstract_device_t，内建的设备有mem_t内存设备，bus_t总线设备，clint_t中断控制器设备，在需求分析的基础上，设计出plic_t平台级中断控制器设备类型，如图3。

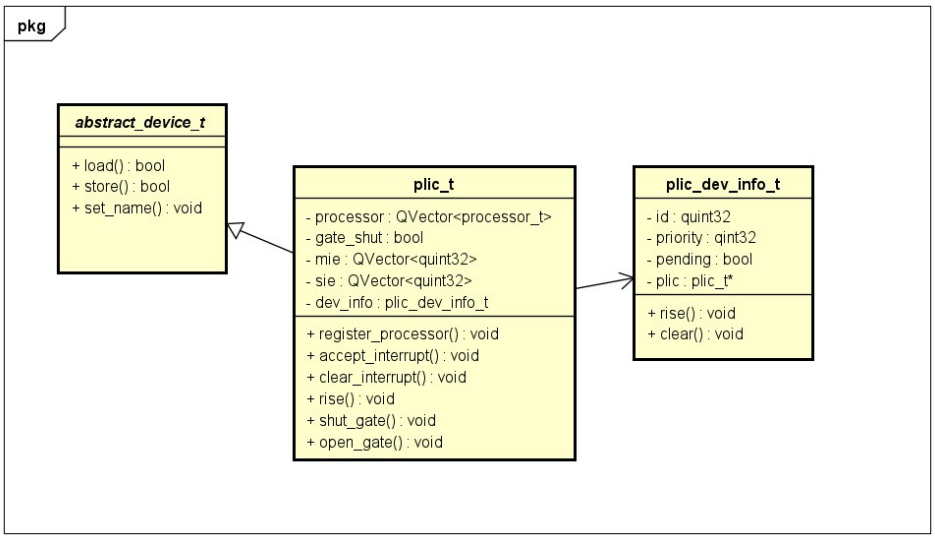


图 3 plic_t 设备类图

plic_t 设备类提供基本的中断控制器模拟，除了寄存器组的模拟，还包括 gateway 的打开和关闭，外部中断的收发，中断使能位的控制等，plic_dev_info_t 设备类模拟外部中断源设备信息，后续包括 SPI 控制器，uart，mailbox 等设备都需要经过 PLIC 中断控制器。

以 uart16550a 串口控制器为例，设备树中的节点信息实例如下:

```
uart16550@40000000 {
        compatible = "ns16550a";
        reg = <0x00 0x40000000 0x00 0x1000>;
```

```
            current-speed = <115200>;
            clock-frequency = <25000000>;
            reg-io-width = <0x04>;
            device_type = "serial";
            interrupts = <0x02>;
            interrupt-parent = <&plic_intc>;
        };
```

uart_t 设备类需要包含 fifo 深度信息，串口控制器寄存器信息，plic_dev_info_t 中断源设备信息，并且在设备树中设置为以中断号 2 挂载为 PLIC 的外部中断源(source)设备。

图 4 为 uart 串口控制器作为中断源挂载 PLIC 的类图。

同样的，mailbox 设备类在 spike 中的模拟也通过 plic 控制器，图 5 为 mailbox 挂载 PLIC 的类图。

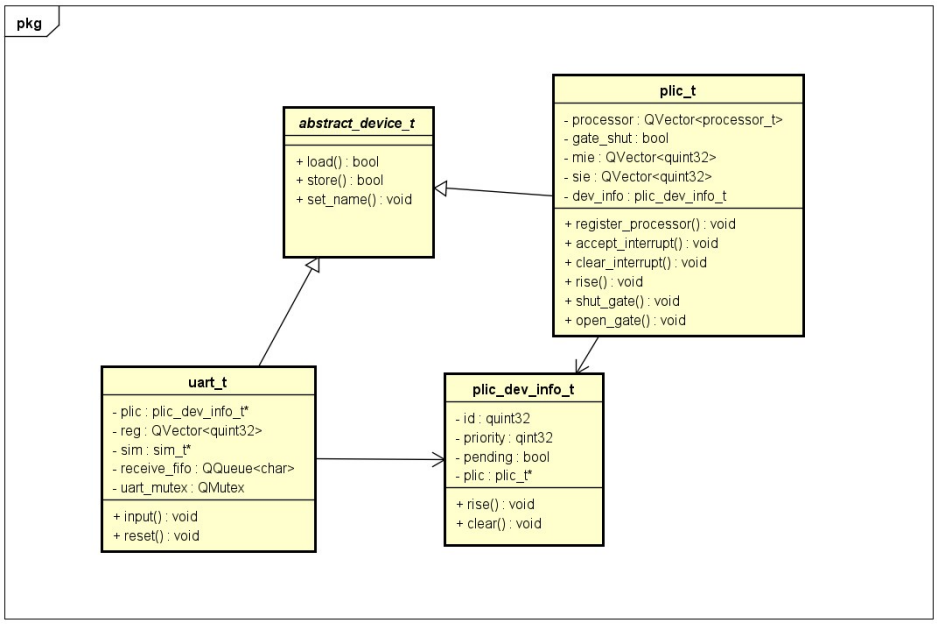


图 4 uart_t 设备类图

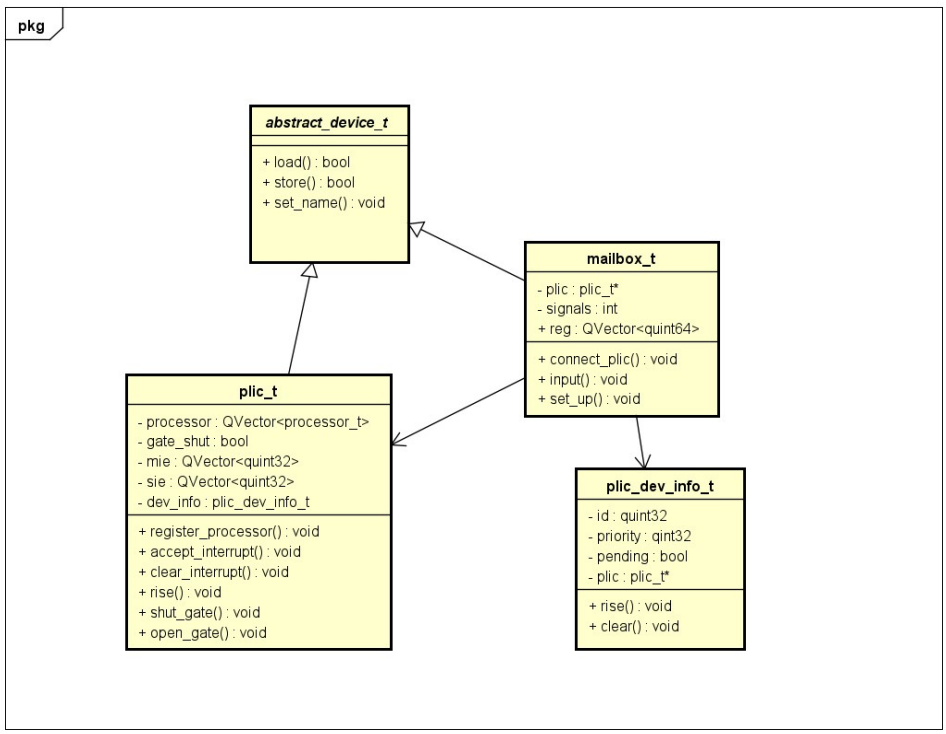


图 5 mailbox_t 设备类图

3.2 串口调试模块设计

spike 支持单步调试功能，支持 csr 寄存器查询，mem 内存查询功能，分别对应 spike 的相应人机交互指令，在实际工作中很难使用，参照一些主流 IDE 的调试界面设计 spike 的调试模块，整个调试模块的构成如图 6 所示。

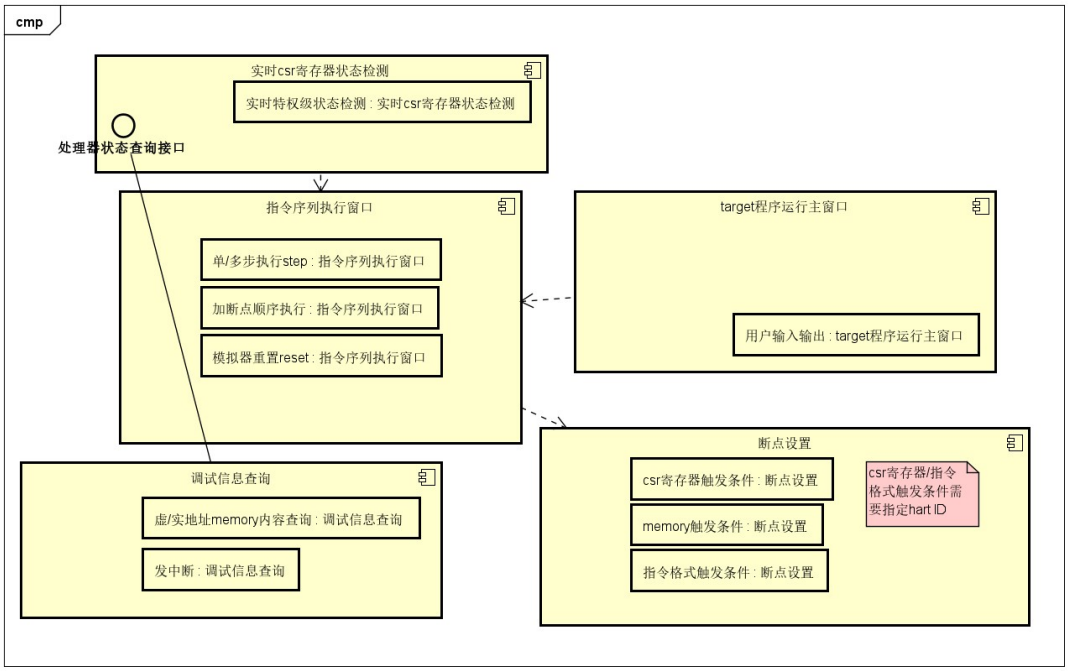


图 6 串口调试模块的结构

为了将调试模块与模拟器其他设备模块的耦合度降低，在设计调试模块时，对需要各部分功能模块都采用了接口设计。本课题采用的应用系统环境是 C++平台的 Qt 框架，需要实现的组件有：

处理器状态检测模块，target 程序运行窗口，反汇编指令序列执行模块，调试信息收发及断点设置模块。

4. 拟采用的开发方法、环境，测试方案等

本方案拟采用面向对象的设计方法，在 Ubuntu20.04 环境下使用 Qt Creator 的 UI Designer 模块和其他 Qt5 的类库文件，整体设计呈现开放式的结构，便于功能的扩展和持续的改进。

测试方案是在 x86 平台上运行模拟器主程序，target 程序为 riscv proxy kernel(包含 Berkeley Boot Loader)，RISC-V 架构的 linux-v5.10 版本内核，交叉编译的 busybox1.32 版本，

以及移植到 RISC-V 架构的 glibc 库。运行成功后在用户态对模拟器其他拓展设备模块进行测试。

5. 技术难度及特色分析

本课题涉及到较新的 RISC-V 指令集架构，其软件生态的发展还不够完善，对于 RISC-V 指令集模拟器的拓展需要深入了解体系结构的逻辑功能，并且对硬件设备进行建模，所涉及到的技术面较广，技术难度主要存在以下几个方面:

（1）平台级中断控制器 PLIC 的设计

平台级中断控制器是模拟器能够支持多种外设的桥梁，需要对不同的外部中断源提供普适的接口，由于处理器核心只接受来自 PLIC 的外部中断信号，主要的裁决，屏蔽，使能工作都交给 PLIC Core 来完成。并且 target 程序，主要是 boot loader 和 linux 内核对于 PLIC 的初始化工作也需要对具体的 PLIC 实现方式做相应的调整。

（2）外接设备的建模工作

由于真实的外接设备通常会有自带的控制器，包括主要的寄存器，固件代码和硬件逻辑电路，模拟器通过软件模拟以上的部件，需要参考特定设备厂商提供的产品说明，有时还需要自行设计驱动程序，这部分内容较为繁杂，因此本课题只挑选了几类具有代表性的通用外设进行模拟。

（3）软硬件难适配的问题

在真实的开发过程中，软件往往是易迭代的，而硬件设计部分可能还是参考的较旧版本的指令集手册，尤其是对于 RISC-V 这样迭代速度很快的新版指令集。这种情况会导致模拟器译码部分失效，以及随之而来的处理器行为异常，因此需要充分考虑指令集版本的影响。

6、本人主要工作描述

在本课题中，需要完成对 RISC-V 体系结构等相关理论的研究，完成系统总体架构的设计，对 RISC-V 处理器指令执行流程、中断控制器模块、外接设备等进行建模，并且集成到模拟器中，完成串口调试界面的设计和编写，最后在模拟器上进行 GCC 的 RISC-V 移植工作，并在 FPGA 开发板和模拟器上进行对照试验.

四、工作进度的大致安排

应包括文献调研，工程设计，项目开发和调试，实验数据的分析处理，撰写论文等。

时间	工作	阶段成果
2021.3 - 2021.5	完成对 spike 模拟器的调研，分析项目的可行性。	文献综述，开题报告
2021.6 - 2021.7	完成前期课题调研，研究分析相关技术，并掌握项目总体框架的设计	需求分析报告
2021.8 - 2021.11	掌握项目各模块具体的原理以及自己相关工作的重点和难点，熟悉系统架构的每个细节	概要设计报告
2021.11 - 2022.1	根据设计文档完成自己相关模块的工作的详细设计，部署好开发环境，完成代码框架的编写	详细设计报告
2022.1 - 2022.2	完成项目编码与测试工作	项目代码
2022.2 - 2022.3	整理相关文档，撰写毕业论文，进行毕业论文答辩	毕业论文初稿
2022.3 - 2022.4	修改和完善毕业论文	毕业论文定稿

五、预期成果

应包括软硬件产品、文档、模型、专利、论文等

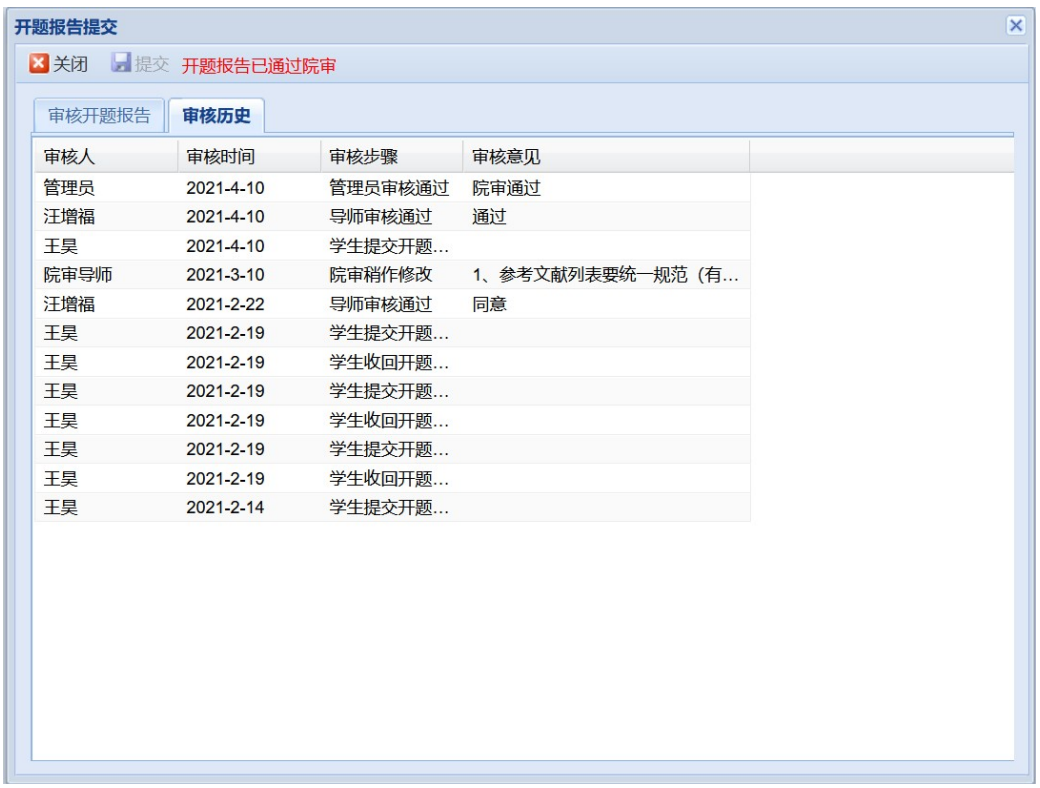
（1）设计出一个基于 spike 的功能更加丰富的 RISC-V 指令集模拟器，可以方便地进行新设备的拓展，提供易用的调试界面。给出模拟器使用说明和必要的文档。

（2）撰写工程硕士学位论文。

六、审核意见

导师意见 导师签名： 年　　月　　日
培养单位负责人意见 培养单位负责人签名： 年　　月　　日


开题报告提交
关闭
提交
开题报告已通过院审
审核开题报告
审核历史
论文题目*:
RISC-V指令集模拟器的设计与实现
英文题目*:
Design and implementation of RISC-V instruction set simulator
学生学号:
SA19225360
学生姓名:
王昊
学校导师:
汪增福
辅导师:
企业导师:
侯锐
所在单位:
软件学院
工程领域:
软件工程
领域代码:
430013
研究方向:
01_应用系统设计
类别:
工程设计或技术改造
形式:
工程设计
研究论文
性质:
技术开发
应用技术研究
报告附件:
浏览...
未选择文件。
请选择.doc,.docx,.pdf格式文件上传
下载开题报告
提交更改备注:
校内导师审核意见:
通过


开题报告提交
关闭
提交
开题报告已通过院审
审核开题报告
审核历史
审核人 审核时间 审核步骤 审核意见
管理员 2021-4-10 管理员审核通过 院审通过
汪增福 2021-4-10 导师审核通过 通过
王昊 2021-4-10 学生提交开题...
院审导师 2021-3-10 院审稍作修改 1、参考文献列表要统一规范（有...
汪增福 2021-2-22 导师审核通过 同意
王昊 2021-2-19 学生提交开题...
王昊 2021-2-19 学生收回开题...
王昊 2021-2-19 学生提交开题...
王昊 2021-2-19 学生收回开题...
王昊 2021-2-19 学生提交开题...
王昊 2021-2-19 学生收回开题...
王昊 2021-2-14 学生提交开题...