华南理工大学

South China University of Technology

硕士学位论文

基于 RISC-V 的 DSP 微处理器及 SoC 实现与研究

作 者 姓 名	陈若晖
学 科 专 业	计算机科学与技术
指 导 教 师	赖晓铮 教授
所 在 学 院	计算机科学与工程学院
论文提交日期	2022 年 4 月

Implementation and Research of RISC-V DSP Microprocessor and SoC

A Dissertation Submitted for the Degree of Master

Candidate：Ruohui Chen

Supervisor：Prof. Xiaozheng Lai

South China University of Technology

Guangzhou, China

分类号：TP273 **学校代号：10561**

学　号：201720116273

华南理工大学硕士学位论文

基于 RISC-V 的 DSP 微处理器及 SoC 实现与研究

作者姓名：陈若晖　　指导教师姓名、职称：赖晓铮 教授

申请学位级别：工学硕士　　学科专业名称：计算机科学与技术

研究方向： 计算机体系结构

论文提交日期：2022 年 4 月 15 日　　论文答辩日期：2022 年 6 月 10 日

学位授予单位：华南理工大学　　学位授予日期：　　年　　月　　日

答辩委员会成员：

主席：_____

委员：_____

摘　要

计算机体系结构在过去几十年的计算机发展历史中不断的演化更替，催生出了种类繁多的计算机指令集架构。现今存在着多种主流的计算机指令集架构，如 x86、ARM、MIPS、OpenRISC 等。迈入 21 世纪以来，简洁明快的 RISC 指令集开始成为微处理器设计的主流，而现存的 RISC 指令集如 MIPS、SPARC、ARM 等则或多或少存在着历史遗留的缺陷，为了解决这一问题，伯克利大学在 2010 年启动了第五代的 RISC 项目，设计了全新的 RISC-V 指令集架构，其特点是免费开源以及提供了许多现代化微处理器设计的技术特性，包括可选的扩展指令集、自定义指令功能以及用户与特权架构分离等。基于以上的特性，RISC-V 微处理器正迅速占据着除桌面型微处理器以外的市场。与此同时，随着微处理器架构复杂程度的不断提升，对微处理器进行设计的工具与框架也面临着随之而来的效率与生产力的挑战，传统的集成电路设计工具与方法学如基于 Verilog/SystemVerilog 的瀑布设计模式开始变得捉襟见肘。随着基于敏捷硬件设计方法学设计工具如 Chisel、SpinalHDL 的兴起，基于高级编程语言并吸收了软件工程学方法的各类新兴硬件设计工具及框架开始逐步被学术界以及工业界所接受。本文将介绍基于 RISC-V 的开源面向嵌入式场景的 DSP 微处理器及其 SoC 及其设计，其实现了 RV32GCP 指令集架构，在性能上能够对标 ARM 专为嵌入式设备设计的微处理器 Cortex-M3。该微处理器能够填补当前国内对于开源 RISC-V DSP 微处理器的空白。同时，本文还介绍用于构建该微处理器的硬件设计与验证框架 PyHCL，PyHCL 是第一款基于 Python 的 RTL 级硬件构造语言，相比于 Chisel，PyHCL 提供更为简单易用的用户接口，较短的学习曲线以及完整的 Python 特性支持。其效率要优于 Chisel，且相比仅支持单元测试的 Chisel，PyHCL 具有完整的验证环境支持，包括单元测试、功能测试、差分测试等。本文由两大主要部分组成：一是基于 RISC-V 的 DSP 微处理器，二是敏捷硬件设计工具 PyHCL，在本文最后也会阐述用于本微处理器验证过程中所使用的 PyHCL 验证方法，并介绍用于本文实现的微处理器验证所使用的基于多级验证环境的差分测试框架。

关键词：计算机体系结构；微处理器；RISC-V；敏捷硬件设计；Python

Abstract

Computer architecture has evolved over the past decades of computer development history. Therefore, a wide variety of computer instruction set architectures have been created in the past few decades. Today, there are several mainstream instruction set architectures, such as x86, ARM, MIPS, OpenRISC, and so on. Entering the 21st century, the soncise and simple RISC instruction set has become the mainstream of microprocessor design, while the existing RISC instruction sets such as MIPS, SPARC, and ARM have more or less defects left over from history. In order to solve this problem, Berkeley University launched the fifth-generation RISC project in 2010, designed a new RISC-V instruction set architecture, which is characterized by free open source and provides many technical features of modern microprocessor design, including optional extensions Instruction sets, custom instruction capabilities, and separation of user and privilege architectures. Based on the characeristics mentioned above, RISC-V microprocessors are rapidly occupying the market other than desktop microprocessors. At the same time, with the continuous improvement of the microprocessor architecture complexity, the tools and frameworks for designing microprocessors also face the challenges of efficiency and productivity. Traditional integrated circuit design tools and methodologies such as The waterfall design pattern based on Verilog/SystemVerilog is starting to become stretched. With the rise of design tools based on agile hardware design methodologies such as Chisel and SpinalHDL, various emerging hardware design tools and frameworks based on high-level programming languages and absorbing software engineering methods have gradually been accepted by academia and industry. This article will introduce the design of a RISC-V open source DSP embedded microprocessor and SoC, it implements the RV32GCP instruction set, which can match the performance of the embedded microprocessor design by ARM: Cortex-M3. The microprocessor can fill the gap for open source RISC-V DSP microprocessors. At the same time, this paper also introduces the hardware design and verification framework PyHCL used to build the microprocessor. PyHCL is the first RTL-level hardware construction language based on Python. Compared with Chisel, PyHCL provides a simpler and easier-to-use user interface, short learning curve ,and full Python feature support. The efficiency of PyHCL is better than Chisel. Compared to Chisel, which only supports unit testing, PyHCL has complete verification environment

support, including unit testing, functional testing, differential testing, etc. This paper consists of two main parts: one is the design of the DSP microprocessor based on RISC-V, and the other is the agile hardware design tool PyHCL. At the end of this paper, the PyHCL verification method used in the verification process of this microprocessor will be described. Also, we would introduce the differential test framework based on the multi-level verification environment used for the verification of the microprocessor.

Keywords: Computer Architecture; Microprocessor; RISC-V; Agile Hardware design; Python

目　录

第一章　绪论

1.1　RISC 指令集架构与 RISC-V

1.1.1　研究背景

指令集架构（Instruction Set Architecture，ISA）是计算机体系结构中，介于微处理器架构（Microarchitecture）与操作系统之间的接口。指令集架构用于抽象微处理器底层的具体硬件实现以提供给操作系统一个简洁的硬件接口来管理计算机硬件系统。最早的指令集架构可以追溯到 1964 年 IBM 发布的 IBM System/360 指令集架构，并实现在同时发布的 IBM 360 计算机族当中 [1]。IBM System/360 的设计初衷是为了给 IBM 360 计算机族中不同型号的微处理器为操作系统提供一个统一的接口，而无需针对不同的微处理器实现而设计不同的接口。

指令集架构经过几十年的发展，现今已经出现了众多主流的指令集架构。下面对最为流行的三种指令集架构进行介绍：

1) x86。x86 是 1978 年由 Intel 公司推出的著名复杂指令集计算机（Complex Instruction Set Computer，CISC）指令集架构，作为桌面型个人计算机市场最主流的指令集架构，其成功来源于推出时对 IBM 计算机的兼容以及较好的向后兼容性。然而，从现今的眼光看来，x86 已经变得臃肿而复杂，自发展至今到现在 x86 已经添加了大量的复杂指令。截止 2015 年，x86 已经拥有 1300 多条指令，大量的寻址模式，大量的特殊寄存器以及多种转换寻址的策略。同时，x86 也有一些固有的缺陷，如贫瘠的寄存器数目以及多种隐式的技术特性，包括隐式的条件编码以及预测移动等，这在高性能的微处理器中是难以实现的 [2]。

2) ARM。ARM 是 1985 年由 ARM Holding 公司开发的精简指令集计算机（Reduce Instruction Set Computer，RISC）指令集架构。ARMv7 是 ARM 的 32 位版本，2011 年发布了基于 64 位的 ARMv8 版本。ARM 由于其低功耗、低成本的特性，被广泛应用于嵌入式系统当中。在 RISC-V 发布前，ARM 在嵌入式市场占据绝对主导地位，且在大多数的消费电子设备如便携式设备以及电脑外设等当中占有大部分的市场。然而，ARM 也存在着不少的缺点，如混合的特权级以及用户级架构、独立的压缩指令集 Thumb 等。

3) MIPS。MIPS 是 1985 年由 MIPS 科技公司开发的 RISC 指令集架构。MIPS 是最经典的 RISC 架构，其影响了后续大多数的 RISC 指令集架构的设计思想。MIPS 采用加载-存储的存储器访问架构，这意味着只有加载以及存储两种内存指令，其他所有的算术操作都是在寄存器上进行，这种设计降低了指令集以及实现硬件的复杂性，在编译器技术提升的基础上，可以促进更为简单的流水线处理器实现。虽然 MIPS 简单易行设计使得在学术领域青睐有加，但是该指令集架构还是有几个技术上的缺陷使得它对于高性能的实现来说不具有吸引力，如固定的分支延迟间隙（Delay Slot）、对动态链接的支持非常糟糕、乘除法使用了特殊的架构寄存器等等。

从上述三个典型的主流指令集的现状可以看出，目前的指令集架构发展趋势主要有以下方面：

1) RISC 指令集占据主导地位。进入 21 世纪以来，仍在广泛流行的 CISC 指令集仅剩下 x86。同时，随着 AMD K2 微处理器架构发布以来，所有的 Intel 的乱序执行的处理器引擎都会动态的将 x86 的指令转化成内部的更接近于 RISC 风格的微指令执行。x86 芯片的出货量自 2011 年达到峰值以来，每年下降近 10%，而采用 RISC 处理器的芯片出货量则飙升至 200 亿。

2) 主流指令集架构囿于设计年代久远而逐渐暴露历史遗留缺陷。x86 随时间不断膨胀而变得臃肿的指令集，ARM 低效的压缩指令集 Thumb，MIPS 固有的分支延迟间隙，这都是在当初指令集设计之初并没有预料到的缺点，伴随着异构体系结构的兴起，年迈的 ARM 以及 MIPS 在对现代高性能的微处理器设计中已经显得逐渐力不从心。

3) 开源、免费的指令集架构兴起。x86 以及 ARM 是闭源的指令集架构，闭源的指令集架构会阻碍成功的学术成果在商业化道路上的发展。如果想要设计一款基于上述指令集的微处理器，则需要缴纳大笔的授权费用。MIPS 面对 RISC-V 的推广后也宣布为开源指令集架构，在此之前 MIPS 同样受到知识产权的保护。免费开源的指令集架构意味着任何个人或者团体都可以基于该指令集设计微处理器，并自行决定开源亦或者是闭源该处理器实现。

为了解决上述的问题，伯克利大学在 2010 年启动了 RISC 第五代项目计划，开发了新一代的 RISC 指令集架构：RISC-V。RISC-V 吸取了上述提到了 RISC 指令集使用以

及开发过程的经验，且在设计过程中匹配了现代计算设备的性能与功耗需求，其主要特点包括有：

1) 将指令集划分为一个小型的基本指令集以及若干个可选的扩展指令集。RISC-V 包含 32 位以及 64 位的指令集系统。以 32 位指令集为例子，RISC-V 规定 32 位整型指令集（RV32I）为基本指令集架构，即所有基于 RISC-V 实现的 32 位微处理器都必须实现 RV32I 指令集。从另一方面来说，一个仅实现了 RV32I 指令集的微处理器，在不考虑性能的情况下，都可以执行任意的编译为 RISC-V 的程序。而根据微处理器面向的场景与功能需求，可以实现额外的扩展指令集，如乘除指令集（M）、压缩指令集（C）等等。目前，RV32I 以及 RV64I 指令集都已经处于冻结的状态，在未来也不会发生任何改变，这种模块化的设计吸取了 x86 的教训，为了保持二进制代码的向后兼容性，同时维持基本指令集的简洁不变，所有新加入的指令将会以扩展指令集的方式存在 [3]。

2) 指令集架构与具体微处理器架构的严格分离。对于指令集架构的设计者来说，在指令集当中对于特定微处理器实现添加某条指令会对性能以及成本带来提升，但是却对其他的实现或者未来的实现带来负担。以 MIPS 的分支延迟间隙为例子，该设计为了优化早期的 MIPS 流水线设计中的分支指令所带来的性能损耗问题，而定义为分支的操作固定在分支指令的下一条指令执行完后再执行，而程序员/编译器则需要往这一间隙中填充有用的指令即可。然而，这种设计对于后来更深的 MIPS 流水线设计以及更先进的分支预测策略来说成为了一种累赘，使用 MIPS 实现的微处理器为了保证指令集的向后兼容性，需要花费大量的时间适配该行为。RISC-V 为了避免这种情况的发生，在指令集设计过程中避免了对微处理器架构的过多设计，保持从低功耗的嵌入式微处理器到大型数据仓库式微处理器的一致性。

3) 开源指令集。RISC-V 属于一个开放的非盈利性质的基金会，其不受任何一个公司的兴起或者没落所影响。RISC-V 指令集可以供任何个人或者团体免费使用，包括开发基于 RISC-V 的芯片或者软件，而不需要支付任何授权的费用。这对于工业界来说 RISC-V 是一大优势，通过复用开源的基于 RISC-V 的 IP 软核设计，可以极大的降低微处理器实现的门槛。同时，对于学术界的基于 RISC-V 的微处理器实现也可以平稳的转移到商用微处理器上。

此外，RISC-V 还具有其他的特性，如适配从袖珍的低功耗嵌入式微处理器到高性

能计算机等各种规模的处理器；适配所有的芯片后端工艺，包括现场可编程逻辑门阵列（FPGA）以及专用集成电路（ASIC）；适配所有的微架构，包括顺序执行与乱序执行的流水线、单发射或超标量的流水线；支持异构体系结构，包括领域专用的加速器实现等等。基于以上的特性，RISC-V 现今在学术界以及工业界都吸引了极大的关注，近年来涌现了大量基于 RISC-V 的微处理器实现，本文将挑选数个典型的基于 RISC-V 的微处理器实现进行简要介绍。

1.1.2 国内外研究现状

本节将会对数款现今国内外主流的开源 RISC-V 微处理器进行简要的介绍。目前，国内外开源的 RISC-V 微处理器实现种类繁多，从低功耗的嵌入式微处理器到高性能的桌面型计算机处理器都有相应的开源实现。

Rocket[4] 是 RISC-V 基金会开发的标准 RISC-V 微处理器实现之一，使用 Chisel 硬件构造语言进行设计。Rocket 是一个 5 级的顺序单发射流水线处理器，通过配置可以实现为 RV32G 或 RV64G 内核，这两者的区别仅为使用不同的地址总线宽度以及增加了相应的双字操作。Rocket Core 包括有一个内存管理单元（MMU）来支持基于页级的虚拟内存实现，一个非阻塞的数据缓存以及一个包含分支地址预测的处理器前端。分支地址预测器的参数可以进行配置，其通过分支目标缓存（BTB）、分支历史表（BHT）以及返回地址栈（RAS）来实现。对于浮点实现，Rocket Core 直接复用了 Chisel 的标准浮点单元库进行实现。Rocket Core 还支持三级特权架构：机器级、特权级以及用户级。同时，Rocket Core 还开放了一系列的参数供用户配置生成不同大小的内核，包括对 M、A、F、D 扩展指令集的实现、浮点流水线的级数以及缓存和 TLB 的大小等。Rocket 的流水线结构如图 1-1 所示：

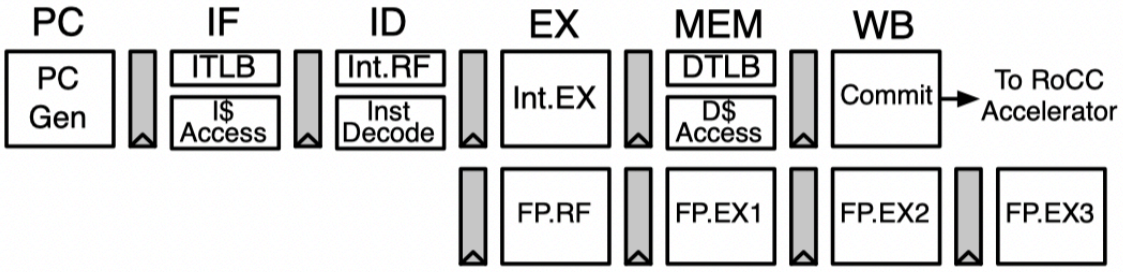


图 1-1 Rocket 的流水线结构

Rocket 面向的应用场景为低功耗优化下的高性能嵌入式设备，在同等 TSMC 40nm 工艺条件下，相比起对标的 ARM Cortex-A5 微处理器，Rocket 具有相近的性能表现

以及更小的面积以及更低的功耗：Dhrystone 评分 A5 为 1.57，Rocket 为 1.72（单位为：DMIPS/MHz）。面积上 A5 为 0.53mm^2，Rocket 为 0.39mm^2。动态功耗上 A5 小于 0.08mW/MHz，Rocket 为 0.034mW/MHz。

BOOM[5] 是 RISC-V 基金会开发的另一个标准 RISC-V 微处理器实现。与 Rocket 不同，BOOM 采用的是超标量乱序执行流水线实现，支持指令集为 RV64G。与 Rocket 相同，BOOM 也使用 Chisel 进行设计实现，因此在 BOOM 当中复用了大量 Rocket 中实现的模块生成器代码。BOOM 很大程度上受到 MIPS R10000[6] 以及 Alpha 21264[7] 实现的启发，与这两种实现相同，SonicBOOM 实现采用了显式寄存器重命名 [8] 的技术。BOOM 的流水线结构如图 1-2 所示，BOOM 采用了 10 级流水线结构，译码阶段采用 4 路译码，乱序执行算法在 Tomasulo 算法基础上加入重排序缓冲区技术改进，使指令最终提交时维持发射顺序，以达到维护精确中断的效果。

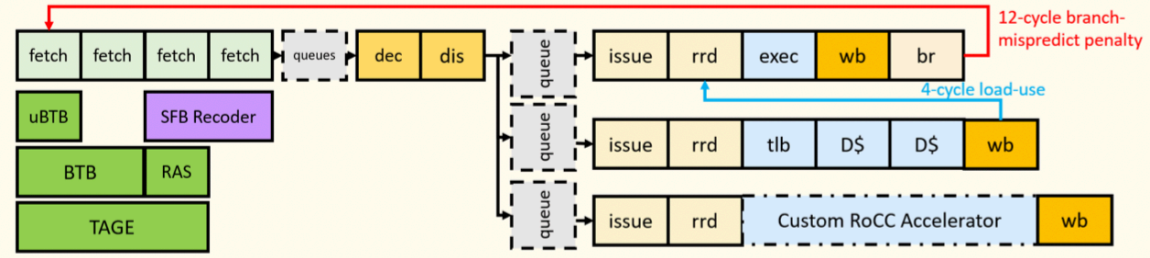


图 1-2 BOOM 的流水线结构

BOOM 面向的应用场景为兼具高性能以及高能效的消费类移动端/网络设备，在同等 TSMC 40nm 工艺条件下，相比起对标的 ARM Cortex-A9 微处理器，BOOM 具有稍好的性能表现，同时具有更小的面积以及更低的功耗：CoreMark 评分 A9 为 3.59，BOOM 为 3.91（单位为 CoreMarks/MHz）。面积上 A9 约为 2.5mm^2，BOOM 约为 1.00mm^2。功耗上 A9 为 0.5-1.9W，工作主频为 0.8-2.0 GHz，BOOM 为 0.25W，工作主频为 1 GHz。

Hummingbirdv2 E203[9] 是国内芯来公司所研发的面向超低功耗及面积优化的嵌入式设备 RISC-V 微处理器，支持 RV32IMAC 以及 RV32EMAC 指令集，两级顺序执行流水线结构，且仅实现了机器级特权模式。E203 的主要亮点在于其可配置的 NICE（Nuclei Instruction Co-unit Extension）系统，可以允许用户自定义指令，在保持低功耗的前提下提升与外部领域专用加速器交互的性能。E203 的 SoC 架构如图 1-3 所示。

E203 与其对标的 ARM Cortex-M0/M0+ 相比，在维持相近的逻辑门数量前提下，能够达到更优的性能表现，且包含了多周期的硬件乘法与除法器：Dhrystone Cortex-M0 与 M0+ 的分数分别为 0.84 与 0.93，而 E203 为 1.23（单位：DMIPS/MHZ）。CoreMark

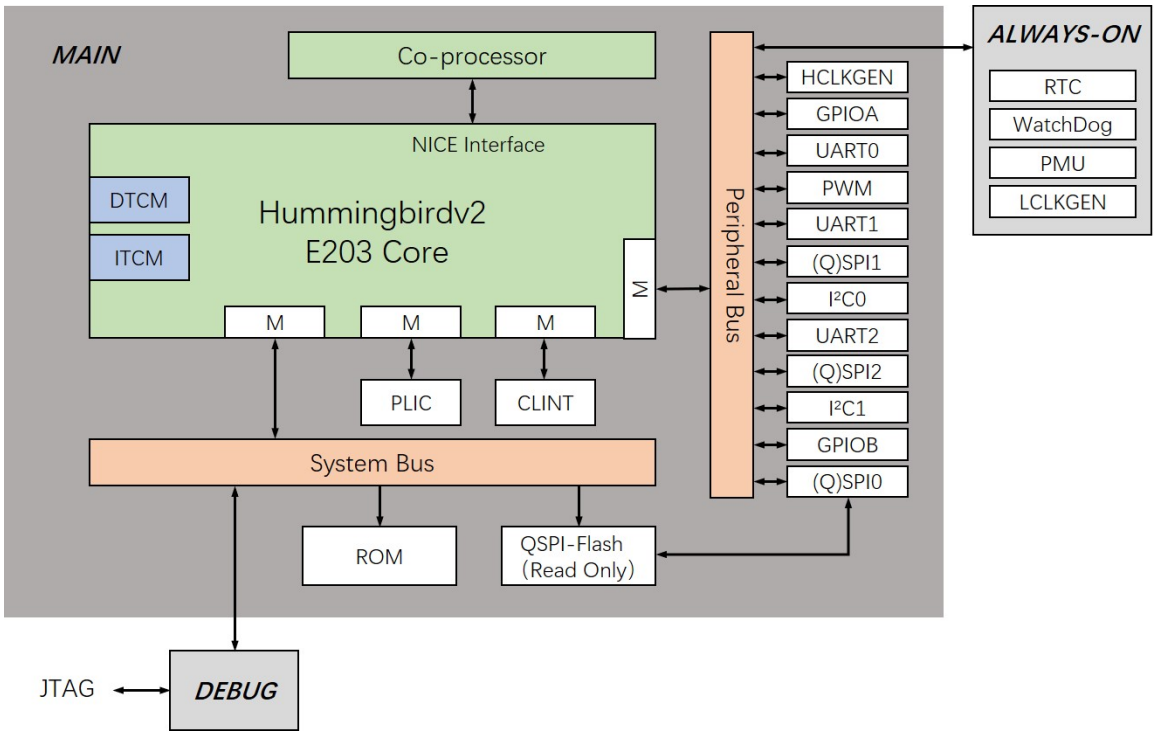


图 1-3 E203 SoC 架构图

Cortex-M0 与 M0+ 的分数分别为 1.62 与 1.77，而 E203 为 2.15（单位：CoreMarks/MHz）。三者逻辑门的数量都为 12K。

PicoRV32[10] 是由 Clifford Wolf 专为 FPGA 平台设计的面积深度优化的 RISC-V 微处理器实现。PicoRV32 可以配置为支持 RV32E，RV32I，RV32IC，RV32IM 以及 RV32IMC 指令集的内核，同时带有一个可选的内置中断控制器。PicoRV32 在 Xilinx 7-Series FPGA 开发板上仅占用 750-2000 LUTs，且能够达到 250-450MHz 的工作频率。鉴于其优秀的性能与面积以及可选的协处理器接口，可以很容易的将一些针对 FPGA 的特定应用协处理器（音视频加速器、神经网络加速器）等移植到包含 PicoRV32 的 FPGA 系统上。

RI5CY[11] 微处理器项目是瑞士苏黎世联邦理工大学（ETH Zürich）的综合系统实验室（IIS：Integrated Systems laboratory）和意大利博洛尼亚大学（University of Bologna）的高能效嵌入式研究组（EEES：Energy-efficient Embedded Systems）联合设计研发的 PULP[12]（Parallel Ultra-Low Power）项目中的其中一个子项目。PULP 的目的是实现一个开放、可扩展的 SoC，其包含的子项目如图 1-4 所示。其主要目的是解决 IoT 设备中日益增长的大量运算负载的场景。PULP 支持多种处理器核架构，RI5CY 是其中的一种处理器核实现。RI5CY 是一款 4 级流水线的 32 位 RISC-V 微处理器，其包含了 RV32IMC 以及可选的“F”单精度浮点数扩展指令集实现。同时，为了提高 DSP 处理性能，RI5CY

还包括自定义指令集如硬件循环（Hardware Loop）、乘累加（MAC）指令以及向量操作指令等。目前 RI5CY 项目已经迭代为 CV32E40P[13] 微处理器项目，在 RI5CY 的项目基础上添加了对标准 RISC-V 核内本地中断控制器以及高级调试模式的支持。

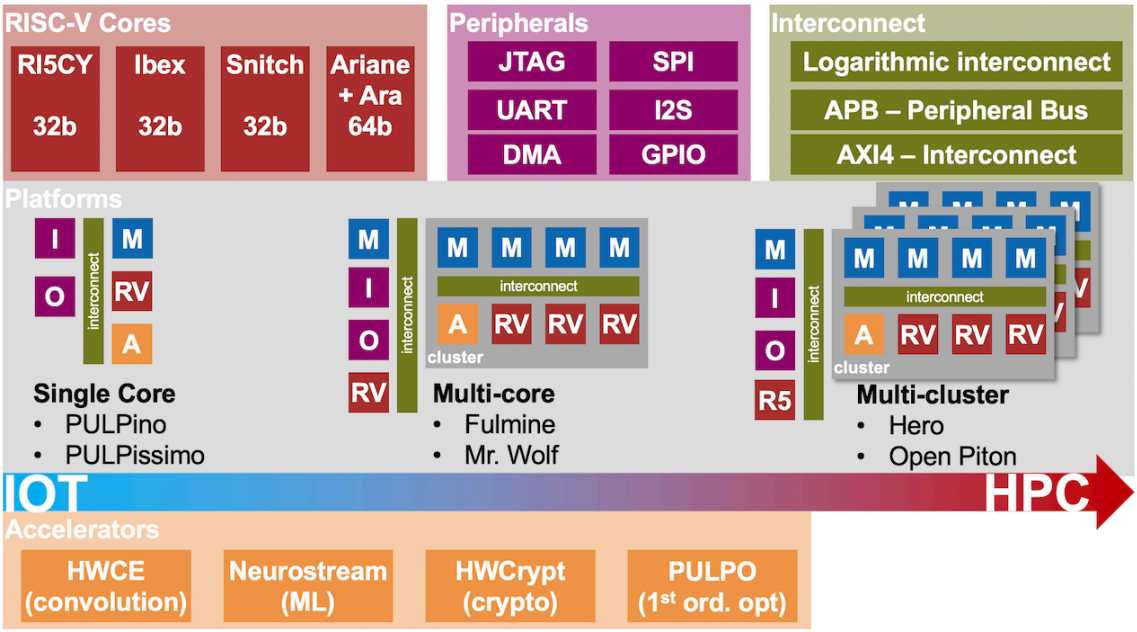


图 1-4 PULP 项目平台概览

RI5CY 在性能上与其对标的 Cortex-M4 微处理器基本持平，在面积以及动态功耗上都要优于 Cortex-M4:CoreMark Cortex-M4 的分数为 3.40，而 RI5CY 为 3.19（单位：CoreMarks/MHz）。在 65ns 的工艺下，Cortex-M4 的动态功耗以及面积分别为 23.2μW/MHz、0.062mm^2，而 RI5CY 则分别为 17.5μW/MHz、0.050mm^2。

香山 [14, 15] 是由中国科学院计算技术研究所在 2021 年发布的一款开源高性能 RISC-V 处理器，使用了包括 Chisel、Verilator、GTKwave 等在内的大量开源工具，实现了差分验证、仿真快照、RISC-V 检查点等处理器开发的基础工具，建立起了一套包含设计、实现、验证等在内的基于全开源工具的处理器敏捷开发流程。香山处理器实现了 RV64GC 指令集，采用乱序六发射结构设计，前端包括取指单元、分支预测单元、指令缓冲等单元，顺序取指。后端包括译码、重命名、重定序缓冲、保留站、整型/浮点寄存器堆、整型/浮点运算单元。香山处理器使用了分支预测器向前覆盖的机制来实现分支预测策略，主要目的在于使用使用容量大、预测算法更复杂、得到指令信息越多的预测器来保证更高的预测准确率，但缺点在于预测器得出结果的延迟比较高，因此香山还使用了一些小型的预测器进行先行预测。香山处理器的架构如图 1-5 所示。

香山处理器具有不俗的性能表现，香山的第一版架构“雁栖湖”在 28nm 工艺的频

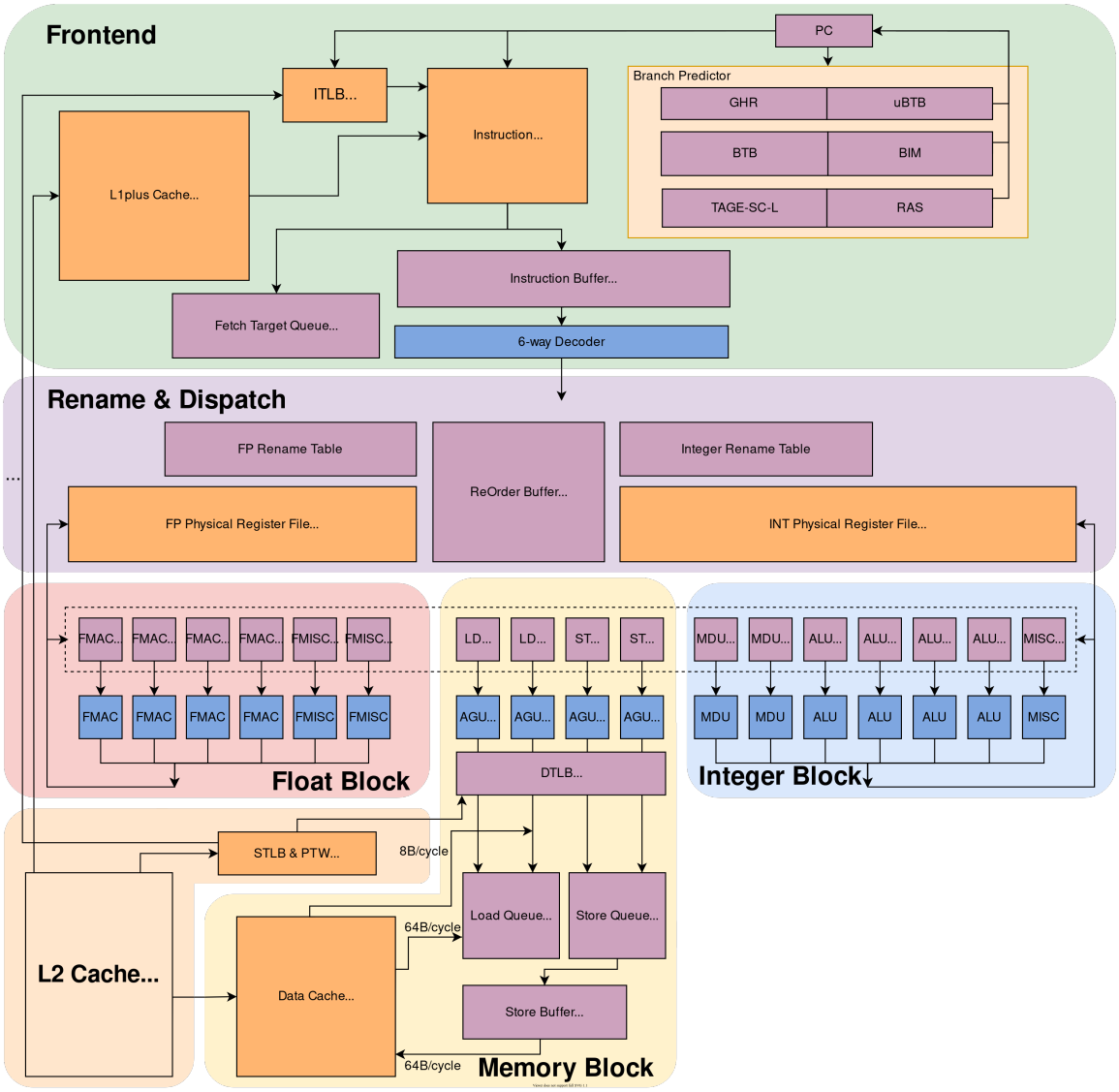


图 1-5 香山处理器架构图

率上能够达到 1.3GHz 的频率。而目前正在开发的第二版架构“南湖”则在 14nm 工艺上能够达到 2GHz 的频率。并且，香山处理器目前已经在 FPGA 部署上启动了 Linux/Debian 操作系统，成为第一个国内完全开源的支持 UNIX 系统的微处理器。

1.2 硬件设计方法学概论

1.2.1 研究背景

囿于丹纳德缩放定律以及摩尔定律的终结，现今计算机体系结构面临着数个显著的挑战。丹纳德缩放定律的失效使得功耗变为现今集成电路设计的首要关键因素，而摩尔定律的终结则减缓了对晶体管技术的进步。因此，计算机架构师必须要寻找其他的途径来提高计算机系统的整体性能，这其中的重要策略之一则是使用领域特定架构（Domain

Specific Architecture，DSA），也就是所谓的加速器来构成异构的计算机系统。然而，传统的硬件开发方法论缺乏在进行异构系统的设计时，缺乏相应的生产力以及效率。在过去的数十年中，瀑布模型一直是硬件开发流程所使用的设计模式，如图 1-6 所示。在瀑布模型的设计模式当中，开发者必须在当前阶段全部完成后才能够进入下一个阶段。举例来说，开发者只有对当前芯片所有功能的 RTL 设计完成后，才能够进入前端验证的阶段。如果在任意一个阶段中发现问题，则需要放弃该阶段中所有已经完成的工作而回退到上一阶段重新进行设计，且越进行到后续的阶段，解决问题的成本将会指数型的上升。

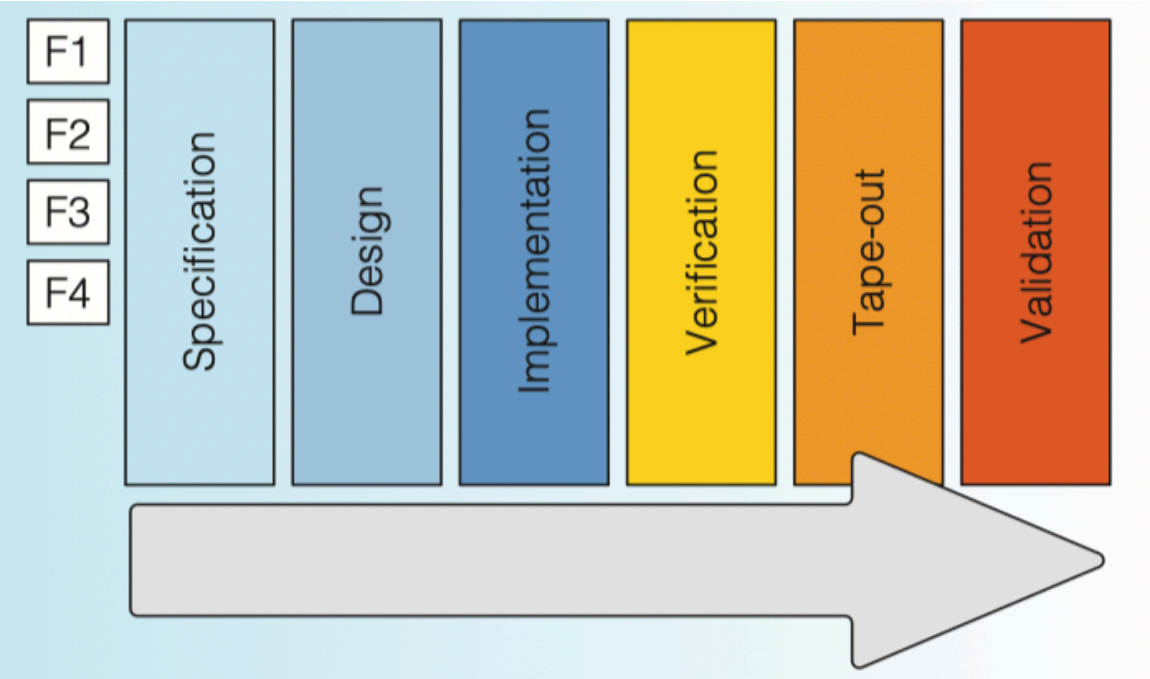


图 1-6 硬件开发流程中简易的瀑布模型示意，F1～F4 表示了该设计的功能单元。从左到右的阶段分别为设计需求分析、架构设计、（RTL）实现、验证、流片、样片功能测试。可以发现瀑布模型中设计所有的功能单元完成当前阶段的开发后才能进入下一个阶段。实际开发流程中每个阶段都包含了相当多的具体步骤。

为了解决这个问题，伯克利大学的计算机体系结构研究团队（BAR）提出了硬件敏捷设计的方法学 [16]。硬件敏捷设计的主要思想是通过快速迭代开发不完全功能的原型系统来达到敏捷设计的效果。举例来说，对于一个基于 RV32G 指令集微处理器的实现，硬件敏捷设计方法通过快速开发一个仅支持 RV32I 的芯片原型并进行流片，以获得在当前工艺下原型设计的具体参数，并以此为基础修正原先的设计。在此基础上，再添加乘除法、浮点等功能单元，并最终迭代得到成品。在硬件敏捷开发的设计模式下，当功

能 A 进入下一阶段时，负责当前阶段的团队可以立即着手对功能 B 的开发。硬件敏捷开发的设计模式如图 1-7 所示。

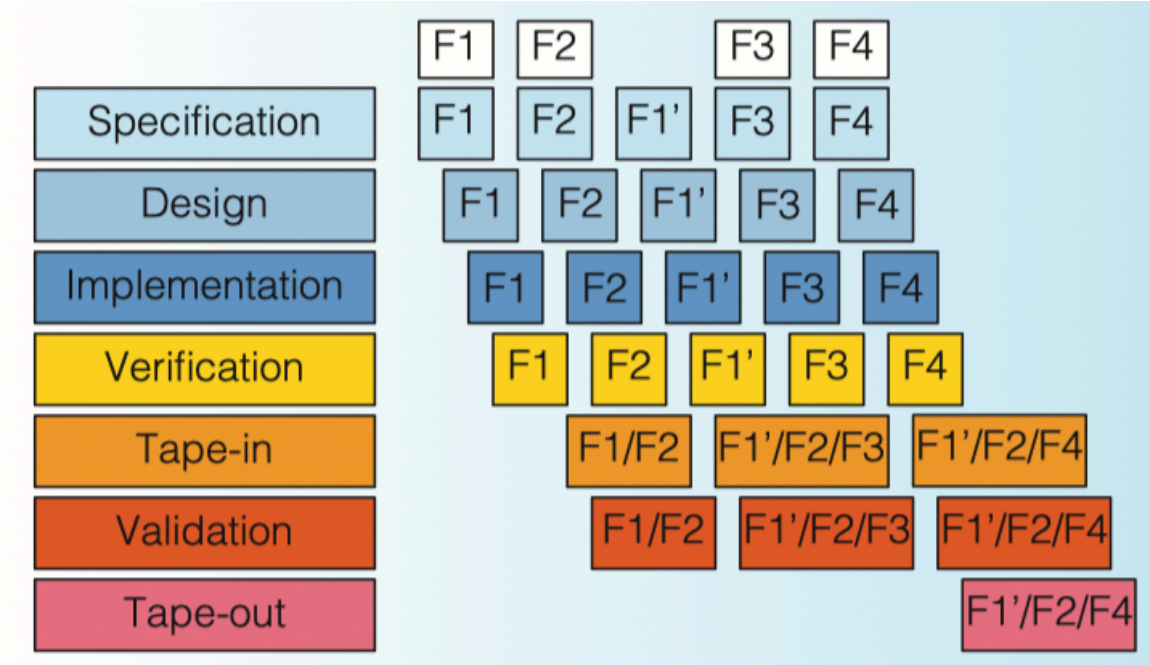


图 1-7 硬件敏捷开发设计模式示意图

为了在硬件开发中引入敏捷设计模式，还需要对硬件设计工具以及框架进行改进，传统的 Verilog/SystemVerilog 语言无法匹配敏捷设计的需求，因此，近几年来大量基于高层次语言的硬件生成框架（Hardware Generation Frameworks，HGFs）应运而生。除了通过引入敏捷设计模式来提高硬件开发的效率，还有一种可以显著加速异构计算体系设计的方法：通过高层次综合（High-Level Synthesis，HLS）[17] 的方法设计加速器。HLS 允许用户将所需要实现的加速器算法以特定的 C/C++ 语言描述出来，并通过专用的编译器生成优化过的 Verilog 代码。然而 HLS 对于架构师来说具有相当曲折的设计曲线，且目前 HLS 的设计优化仍然主要依赖于经验。因此，本文将重点主要放在基于高层次语言的硬件生成框架/语言当中，下一节将会介绍目前国内外对基于高层次语言的敏捷设计驱动的硬件生成框架/语言进行介绍。

1.2.2 国内外研究现状

早期对于硬件生成框架的研究项目有 Verischemelog[18]，它将 Verilog 语言混入 Scheme[19] 语言当中来描述硬件结构。早期的硬件生成框架项目还有 Genesis2[20]，它与 Verischemelog 的思路类似，但 Genesis2 是将 Verilog 语言混入到 Perl 语言当中。早期的硬件生成框架的设计都比较粗糙，一般仅使用字符串处理的方法来生成设计的 Verilog

代码，整体的编译过程都是静态的。因此，这些架构都缺乏可扩展性以及灵活性。

Chisel[21] 是现代硬件生成框架的先驱，其作为 RISC-V 的衍生项目而诞生。Chisel 并非简单的使用字符串处理的方式来将 Verilog 语言混入到某种高级语言当中，而是基于 Scala[22] 开发了一个完整的硬件开发库，其中包含了小到描述线网、寄存器、存储器、逻辑门的原语，大到算术逻辑单元、控制器、DSP 的生成器。同时，Chisel 还包含了一个介于 Chisel 以及 Verilog 之间的中间语言层 FIRRTL[23, 24]，它本质上是一个高度抽象化的 RTL 级语言。由于 Chisel 本质上属于使用 Scala 开发的第三方库，因此使用 Chisel 进行 RTL 级的硬件描述，并且支持 Scala 所有的语言特性，包括参数化、面向对象编程、函数式编程等等，这些特性在硬件敏捷开发模式当中将起到关键的作用。截至目前，已经有多个使用 Chisel 开发的微处理器、SoC 以及加速器项目，如上面提到的 Rocket、BOOM 以及 Raven-3[25]、FireSim[26]、ChipYard[27] 等等。然而，Chisel 在验证方面并没有给出比较完善的解决方案。目前 Chisel 仅支持使用基于 Chiseltest[28] 的单元测试验证方法，并不支持对于基于大规模的功能测试、系统测试的 UVM 验证方法学。Chisel 整体的框架如图 1-8 所示。

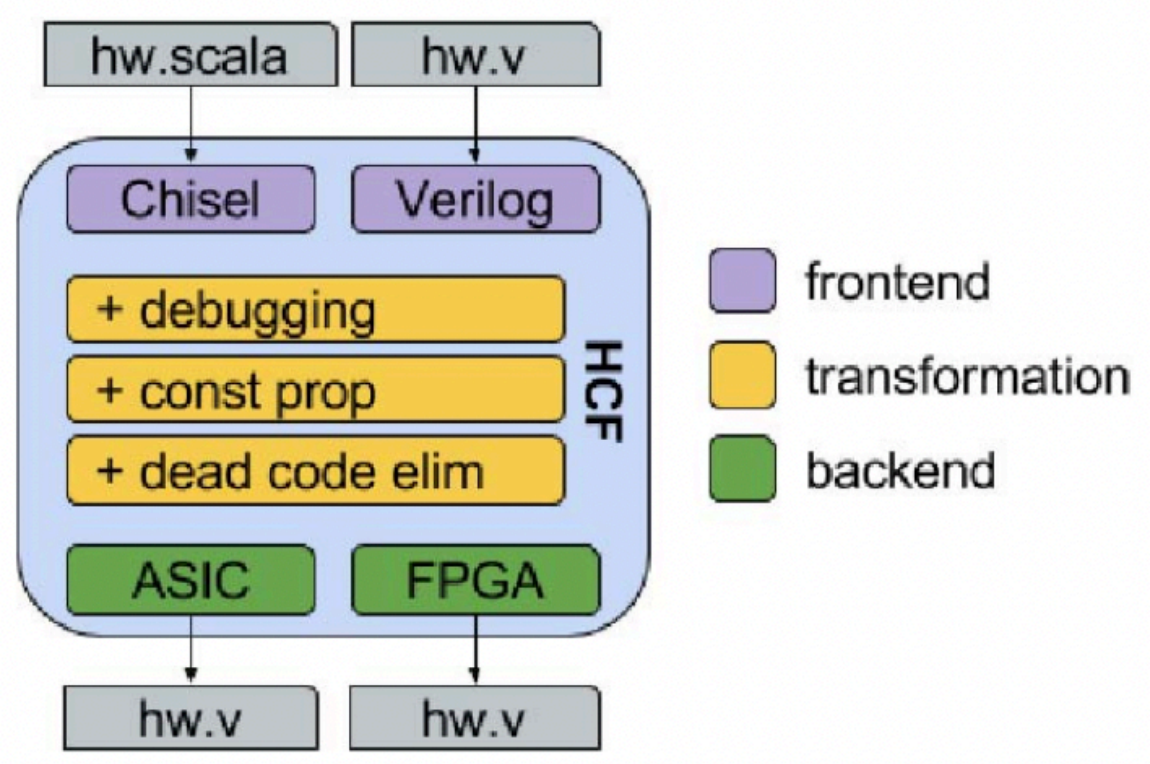


图 1-8 Chisel 整体框架图

与 Chisel 类似，使用 Scala 作为高层语言宿主的硬件生成框架还有 SpinalHDL[29]。SpinalHDL 最初是为了解决 Chisel 所缺乏的多时钟域功能所诞生的衍生性项目。与

Chisel 不同，SpinalHDL 在设计之初采用更偏向于实际工程开发的风格，支持常见的 EDA 工具链。SpinalHDL 不具有 FIRRTL 中间层，而是从 Scala 直接编译为对应的 Verilog 代码，且编译过程中不存在黑盒机制。并且，SpinalHDL 不会带来额外的面积或者性能上的开销。因此，SpinalHDL 相对于 Chisel 要更受工业界的欢迎，而 Chisel 则普遍受众于学术界研究。

相比较于近年来才开始流行基于 Scala 的硬件生成框架，基于 Python 的 HGF 设计在世纪初就开始有项目进行尝试。PyHDL[30] 是最初较为流行的基于 Python 的脚本式硬件描述语言，它使用 PamDC 以及 PAM-Blox 两个 C++ 硬件设计库扩展 Python 语言，其中，PamDC 提供了硬件描述语言所需要的原语如寄存器以及线网，而 PAM-Blox 则是使用 PamDC 原语构造的工具集，包含了参数化以及面向对象的硬件模块。MyHDL[31] 是一个免费开源的用于硬件描述以及验证的 Python 库。MyHDL 将 Python 代码编译为 Verilog 或者 VHDL 并提供基于 Python 的设计代码与主流 EDA 工具交互的方法。MyHDL 包含一个运行在 Python 解释器顶层的仿真器，并使用 PLI（Procedural Language Interface）来与仿真器进行交互。然而，基于 Python 的仿真器在性能上存在劣势，并且 MyHDL 并不支持基于 UVM 的验证方法。Stratus[32] 是一个基于 Python 的超大规模集成电路（Very Large Scale Integration，VLSI）描述语言，Stratus 通过一系列的方法以及函数对 Python 进行扩展，来生成 VLSI 设计的线网（Netlist）模型。PHDL[33] 同样使用 Python 作为硬件描述的语言，其主要亮点在于包含一个内置的模块库用于微处理器设计。然而 PHDL 不包含任何验证的工具或方法，仅提供了针对微处理器开发的设计模式。Migen[34] 与 Stratus 类似，也是基于 Python 的 VLSI 设计工具，Migen 基于硬件描述语言 FHDL，它包含了一个形式化描述的系统来描述电路中的组合电路信号或时序电路状态。同样，Migen 仅仅专注于对系统的描述，而缺乏对所设计电路的验证方法。PyRTL[35] 是一个新兴的基于 Python 的开源硬件设计语言，与 Chisel 类似，它主要的设计目标在于快速的硬件原型构建。PyRTL 与 PyHCL 一样，都是为了达到对硬件描述的简洁化。然而，PyRTL 依然缺乏对设计的验证方法或工具，仅提供了一个效率低下的内置仿真器。PyMTL[36] 及其后代版本 Mamba[37] 是一个基于 Python 的多层次的硬件建模工具，它的主要特点在于使用的是在三种层次：功能级（Functional Level）

时钟周期级（Cycle Level）以及 RTL 级上的建模。PyMTL 提供了基于 PyPy JIT（即时编译）解释器的仿真器，然而其效率相比业界基于 C/C++ 的开源仿真器 Verilator 仍然存在不小的差距。表 1-1 对上述提到的硬件生成框架与 PyHCL 进行对比。

表 1-1 PyHCL 以及其他硬件生成框架的对比

框架	宿主语言	敏捷硬件设计支持	宿主语言模式	内置仿真器	UVM 支持
PyHCL	Python	是	原语构造型	否	是
Chisel	Scala	是	原语构造型	是	否
PyRTL	Python	是	描述型	是	否
MyHDL	Python	否	描述型	是	否
PyHDL	Python	否	描述型	否	否
PHDL	Python	否	描述型	否	否
PyMTL/Mamba	Python	是	多层次建模	是	否

1.3 本文的创新点以及难点概述

通过 1.1 节对国内外基于 RISC-V 的开源微处理器实现以及 1.2 节对国内外开源硬件生成框架的概述，可以看出：

1) 在开源微处理器设计方面，目前开源的微处理器架构中，还不具备有使用 P 扩展指令集来实现的基于 RISC-V 的 DSP 微处理器。RI5CY 虽然是一款专用于 DSP 的微处理器架构，但由于其使用了大量的非标准化的指令，导致其整个上层的软件生态链都需要靠自定义的编译器实现，也缺乏足够的扩展性和稳定性。同时，基于 P 指令集实现的 RISC-V 开源微处理器现在还没有先例，目前大多数面向数字信号处理的 RISC-V 微处理器都是采用 V 指令集或者裁剪 V 指令集的方式来实现，这种方式对于芯片整体面积以及编译器实现都会带来不必要的麻烦。

2) 在开源硬件生成框架方面，某些的 HGF 仅支持设计功能，某些 HGF 包含有内置的仿真器，但是效率比较低下，且目前还没有 HGF 包含有内置的 UVM 验证方案。

针对以上的问题，本文所实现的基于 P 扩展指令的面向 DSP 的 RISC-V 嵌入式微处理器以及基于 Python 的开源硬件生成框架 PyHCL 具有以下的创新点：

1) 本文所实现的 RISC-V 微处理器是目前唯一的开源实现 P 指令集的 DSP 微处理器，P 指令集采用 SIMD 指令方式来实现 DSP 运算库中所需要的算法，其实现虽然比 V 扩展指令集缺乏一定的灵活性，但是在逻辑复杂度方面要简单许多，最终实现的芯片在逻辑门数、面积、功耗方面要比 V 扩展指令集占优。

2) 本文所实现的 RISC-V 微处理器使用并行数据通路的方式来实现 P 指令集，在性能上虽然要略逊于基于 RoCC 的协处理器实现方式，但能够得到更小的芯片面积，更适合应用在嵌入式场景当中。

3) PyHCL 同时包含了设计以及验证工具链，是首个基于 Python 的全栈硬件生成框架。PyHCL 允许集成电路设计以及验证的流程都在同一个框架内完成，而不需要分离为多个不同的工具链以及框架，加速硬件开发的迭代速度。

4) PyHCL 还提供了基于 UVM 的验证方法学的解决方案，是首个支持 UVM 验证方法学的开源硬件生成框架，且 PyHCL 还包含了灰盒测试及多级测试的验证策略，覆盖了从单元测试到集成测试的验证粒度。

本文的微处理器实现以及开源硬件生成框架 PyHCL 的实现存在以下的难点：

1) 目前业界缺乏基于 P 指令集的开源微处理器实现，且 P 指令集标准仍然处于草案状态，对于微处理器的设计、验证以及性能测试会带来一定的难度。

2) 使用多数据通路的并行实现方式，对微处理器的控制逻辑实现会带来比较大的负担，这是因为 SIMD 指令大多数需要多个周期来实现，则微处理器需要处理好处理器停顿以及数据冒险、控制冒险等问题。

3) 集成多种验证测试方法在同一个 Python 框架中需要保证对用户接口的统一性，否则会破坏 PyHCL 设计的初衷，即在同一框架模型中集设计与验证为一体。

1.4 本文的章节安排

本文分为五个章节，其余章节的内容作以下的组织安排：

第二章：基于 Python 的硬件开发框架 PyHCL。本章对 PyHCL 的架构进行概述，包括 PyHCL 原语、接口、中间表示层 IR 等。最后介绍了结合 FIRRTL 作为后端生成语言的数种优势。

第三章：面向 DSP 的嵌入式微处理器设计与实现。本章对实现了 RISC-V RV32GCP 指令集的微处理器进行介绍，包括对微处理器的基本流水线设计、特权级设计、多数据通道设计等等进行介绍。最后对包含了该微处理器的 SoC 平台进行介绍。

第四章：PyHCL 验证方法概述。本章对 PyHCL 支持的多种验证方法进行概述，包括基本单元测试使用的 PyHCL-Tester、基于 UVM 的大规模集成测试框架 PyUVM、针对 RISC-V 微处理器的差分测试框架等的介绍。

第五章：总结与展望。

第二章　模板简介

与很多外文杂志社不同，大部分中文期刊都不提供 LaTeX 模板给投稿者使用，也很少有学校给学生提供官方的毕业论文模板。目前 github 上的大部分模板都是由学生发起的非官方模板。在此感谢 Shun Xu 以及 yecfly 等人的工作，他们的无私贡献使得华南理工大学硕博士毕业论文也可以使用 LaTeX 撰写。

本模板是直接修改前人的模板得到的，更详细的介绍可到 [1, 2] 下载。本章仅从用户的角度简要介绍模板的使用，而尽量避免涉及 LaTeX 的模板制作细节（实际上是因为本人也不会）。正如我们使用手机并不需要了解麦克斯韦方程组，使用 LaTeX 写作也无需了解模板是如何制作的。

LaTeX 的源代码保存在后缀名为.tex 的文件中。当编写长篇文档时，例如当编写书籍、毕业论文时，单个源文件会使修改、校对变得十分困难。将源文件分割成若干个文件，例如将每章内容单独写在一个文件中，会大大简化修改和校对的工作。为方便，本文将 scutthesis.tex 文件称为主文件，而将 abstract.tex、chapter0x.tex、conclusion.tex 等文件称为章节文件。

值得注意的是，要每次编译时都更新参考文献著录，TeXstudio 软件的选项-> 设置中的构建并查看、编译器需要设置成如图2-1、2-2所示。此时只需在任意一个文件中点击构建并查看按钮即可编译文档。每次编译都更新参考文献会使得编译时间很长。

2.1　主文件

scutthesis.tex 文件相当于主函数，调用各章的内容。LaTeX 源代码以一个\documentclass 命令作为开头，它指定了文档使用的文档类。文档类规定了 LaTeX 源代码所要生成的文档的性质——普通文章、书籍、演示文稿、个人简历等等。

```
\documentclass[ options ]{ class-name }
```

其中 class-name 为文档类的名称，如 LaTeX 提供的 article, book, report，可在其基础上派生的一些文档类或者有其它功能的一些文档类。LaTeX 提供的基础文档类见文献 [3]。还可以自定义文档类，如华南理工大学硕博士论文文档类 scutthesis，其实现保存在后缀名为.cls 的文件中。可选参数 options 为文档类指定选项。

document 环境当中的内容是文档正文：

```
\begin{document}
正文内容
\end{document}
```

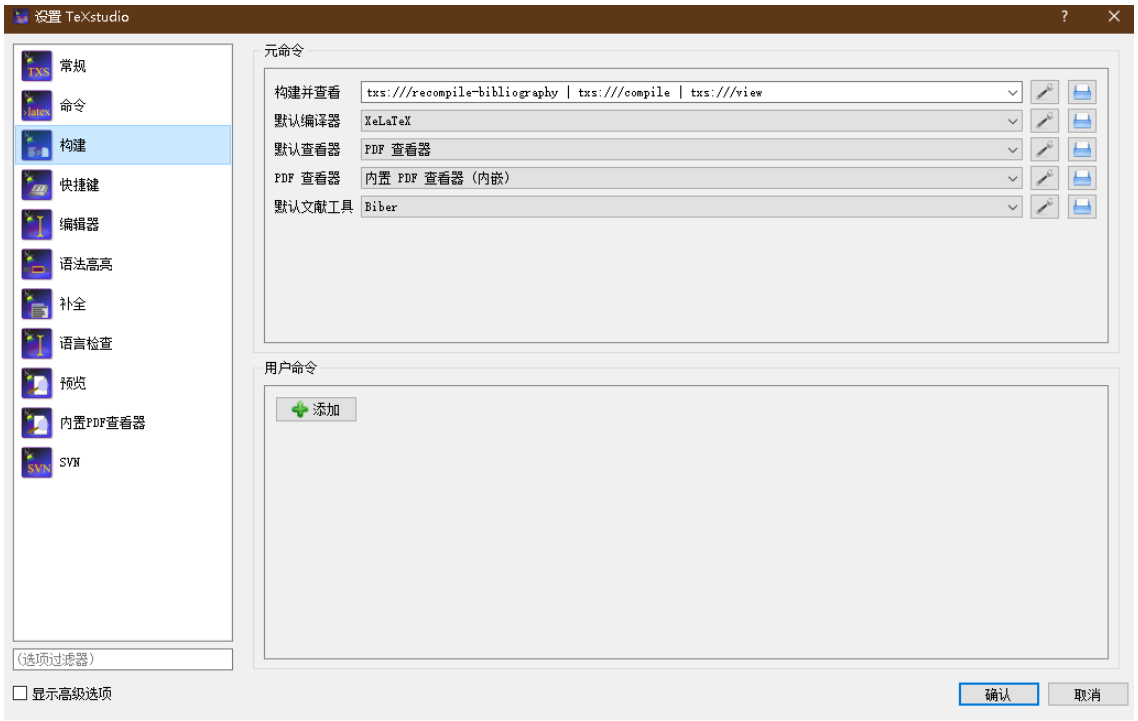


图 2-1 TeXstudio 环境

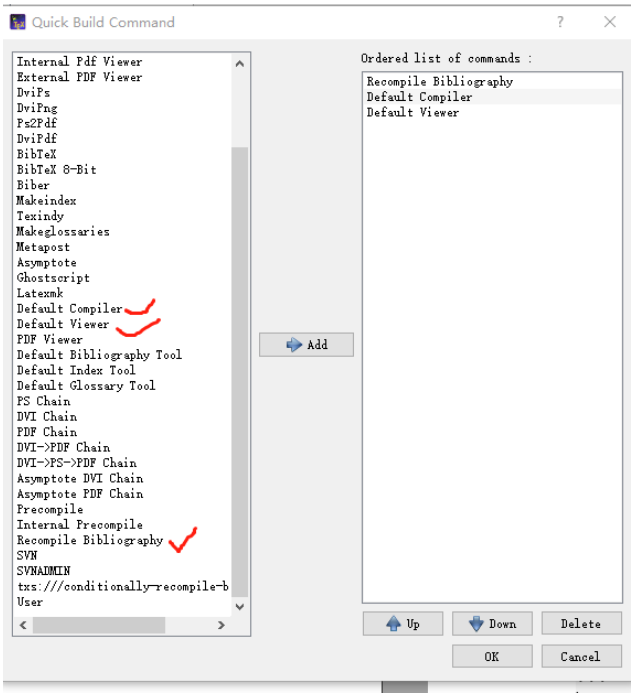


图 2-2 TeXstudio 编译选项

正文中包含各章节内容：

```
\include{abstract} % 中英文摘要
\tableofcontents    % 目录
\listoftables   % 表格目录（可选）
\listoffigures  % 插图目录（可选）
```

```
\include{symbols}   % 符号对照表(可选)
\include{abbreviation}  % 缩略词
...
\include{chapter01} % 第一章
\include{chapter02} % 第二章
\include{chapter03} % 第三章
% 自行根据需要添加章节。
...
\include{conclusion} % 结论
...
\printbibliography  % 参考文献著录
\include{appendix} % 附录
\include{pub} % 成果
\include{ack} % 致谢
```

其中 % 之后的内容为注释，... 表示省略其他代码，仅保留论文内容主体部分。\include{xxx} 指令用于包含 xxx.tex 文件的内容，各章节的内容主要在 xxx.tex 中保存。在\documentclass 和\begin{document} 之间的位置称为导言区。在导言区中一般会使用\usepackage 调用宏包，以及会进行对文档的全局设置。本模板的导言区除调用所需的宏包外，还进行了页眉页脚的设置。有的模板会把所有调用宏包的指令放到一个.sty 宏包文件中，页面的设置放在文档类文件.cls 文件中。因本人时间有限，就不做整理，欢迎有志之士加入完善。使用本模板并不需要了解导言区的指令，在需要时额外添加即可（要注意宏包冲突）。特别地，\includeonly{xxx} 指令用于使文档仅编译 xxx.tex 文件的内容，这就是分章节包含（include）的好处，可大大减少编译时间。

将封面打印保存为 thesis_cover.pdf 文件，硕士使用 master_cover.docx ，博士使用 doctor_cover.doc 。如果有更新版本的封面，可自行替换。文档类默认是博士论文，下面指令将控制添加封面与否：

```
\documentclass[unicode,master,pdfcover]{scutthesis} % 使用pdf文件封面的 硕士模板
\documentclass[unicode,master]{scutthesis}  % 不使用pdf文件封面的 硕士模板
\documentclass[unicode,pdfcover]{scutthesis}    % 使用pdf文件封面的博士模板
\documentclass[unicode]{scutthesis} % 不使用pdf文件封面的博士模板
```

不使用 thesis_cover.pdf 文件指定的封面时，将使用草稿封面。草稿封面也可以减少编译时间，因此可以在最终提交论文时再使用论文封面。草稿封面用以下指令设置：

```
%%%%%%%%%%%%%草稿封面设置%%%%%%%%%%%%%
\title{LaTeX模板}
\author{蒙超恒}
\supervisor{指导教师：裴海龙\ 教授}
\institute{华南理工大学}
\date{2020年5月20日}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2.2 章节文件

章节文件如 chapter0x.tex 等，其内容由\chapter{章名} 开头。新建一章可新建一个文件并由\chapter{新建章名} 开头填写内容即可。节及小节分别用\section{新建节名}、\subsection{新建小节名} 命令。

正文的的书写和 txt 文本文件的书写类似。LaTeX 源代码中，空格键和 Tab 键输入的空白字符视为“空格”。连续的若干个空白字符视为一个空格。一行开头的空格忽略不计。行末的回车视为一个空格；但连续两个回车，也就是空行，会将文字分段。多个空行被视为一个空行。也可以在行末使用\par 命令分段。在本模板中，英文之间的空格被保留，中文之间的空格被忽略。特别地，摘要，附录，结论等两个字的大纲级别为章的章名，中间使用空格隔开。对此论文撰写规范并没有明文要求，只是为了美观。也可以全部不加空格。一般情况下，在文本文字中添加空格使用\quad 命令，但由于文献 [4] 所述原因，直接使用\quad 命令会报警，因而使用\texorpdfstring{\quad}{}，其中最后一个 {} 里面可以加一个空格，不影响使用。目录二字之间添加空格在 scutthesis.cls 文件 317 行设置。

正文本环境中使用公式，即行内公式，需要用两个 $ 包围，如源码：$a+b=c$ 显示为 $a+b=c$。使用其他字符可自行百度或阅读参考文献。再次提醒，使用 LaTeX 撰写论文不需要研究其原理，在达到某种效果（图文显示、公式显示效果）时百度或查书寻找其代码即可。

综上，论文撰写只需要将自己的文本（包含行内公式）放到相应的章节处，并添加行间公式、图表环境并填写图表即可。行间公式、图表将在下一章介绍。

第三章　常用环境及参考文献设置

强烈建议在使用公式、表格、定理环境时进行百度，没必要研究各种用法，只需要知道自己需要什么。因本人的论文所用表格较少，因而对表格不是很熟悉，本章对表格的介绍相应的较少。本章仅介绍本人在论文撰写过程中常用的环境以及参考文献设置。

3.1　图

图的导入需要提前准备好图片文件，最好是.png、.eps、.pdf 或.jpg 文件。另外，如果是从 matlab 导出图片文件，可使用 print 函数或手动导出，print 函数的使用可参考 ICGNC2020plot.m 以及 PlotToFileColorPDF.m 文件等。手动导出（matlab 的 figure 界面的“文件”->“导出设置”设置好大小、分辨率和线宽等然后点击“应用于图窗”）主要用于观察效果，可设置某种样式名称后保存该样式，下次使用时加载，具体可百度“matlab 导出高清图片”。需要特别注意的是一定要 1:1 导入 matlab 生成的图片，并且图中文字设置好字体字号。否则缩放之后，图片的字号就变了，盲审老师一眼就能看出来字号不对，就很麻烦。这就是为什么要在 matlab 点击“应用于图窗“进行预览，观测效果后再 1:1 使用图片。

使用如下代码放置独立成行的图片，效果如图3-1所示

```
\begin{figure}[htbp]
    % 图片居中（列居中对齐）
    \centering
    % 包含当前路径下的Fig文件夹的图片文件DFUAV_f31.png
    \includegraphics[scale=1]{Fig/DFUAV_f31.png}
    % 添加标签one_DFUAV以及图标题“涵道风扇式无人机”，引用某图时使用\ref{xxx}，其中xxx就是标签，图编号是自动生成的。
    \caption{\label{one_DFUAV}涵道风扇式无人机}
\end{figure}
```

其中 figure 为环境名，[htbp] 表示将图片设置为浮动体，实际上这在.cls 文件已经设置过，因而可以省略。[scale=1] 表示安装 1:1 的比例导入图片，还可以按其他方式导入，需要时可自行百度。

使用如下代码划分页面并排放置图3-2、图3-3

```
\begin{figure}[htbp]
    \centering
    \begin{minipage}[c]{0.5\textwidth} % minipage将页面划分为0.5\textwidth
        \centering
        \includegraphics[width=6cm,height=6cm]{Fig/honeywell_t-hawk.jpg}
        \caption{\label{Hawk}T-Hawk}
    \end{minipage}%
    \begin{minipage}[c]{0.5\textwidth}
        \centering
```

图 3-1 涵道风扇式无人机

```
        \includegraphics[width=6cm,height=6cm]{Fig/GTSpy.jpg}
        \caption{\label{GTSpy}GTSpy}
    \end{minipage}
\end{figure}
```

其中 [c] 表示行居中对齐。当图片大小不一但又需要 1:1 导入时，图标题可能行不对齐，因此可以改为如下指令：

```
\begin{figure}[htbp]
    \centering
    \begin{minipage}[c]{0.5\textwidth}
        \centering
        \includegraphics[scale=1]{Fig/honeywell_t-hawk.jpg} %1:1导入
    \end{minipage}%
    \begin{minipage}[c]{0.5\textwidth}
        \centering
        \includegraphics[scale=1]{Fig/GTSpy.jpg}
    \end{minipage}\\[1pt]
    \begin{minipage}[t]{0.5\textwidth}  % 以下为新添加页面划分，[t]表示行顶部对齐
        \caption{\label{Hawk}T-Hawk}
    \end{minipage}%
    \begin{minipage}[t]{0.5\textwidth}
        \caption{\label{GTSpy}GTSpy}
    \end{minipage}%
\end{figure}
```

通常一个 figure 内含有其他小的 figure, 可以使用一些宏包，但最初本着简单的原则，本模板并没有使用这些子图包。后来应同学们要求在，把子图的功能加上，主要是修改了模板文件（scutthesis.cls 文件）的功能包参数。注意，很多网上拿到的代码不一定可以精确的调子图标题字体字号，因为此模板的子图标题字体字号是利用 subfig 宏包的选项进行设置的（在 scutthesis.cls 文件的“图表环境”中），而有些教程使用 subcaption

图 3-2 T-Hawk

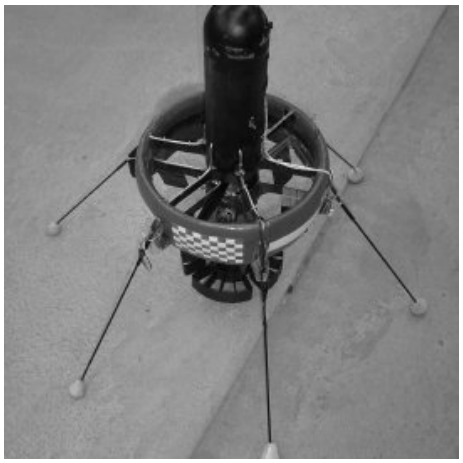


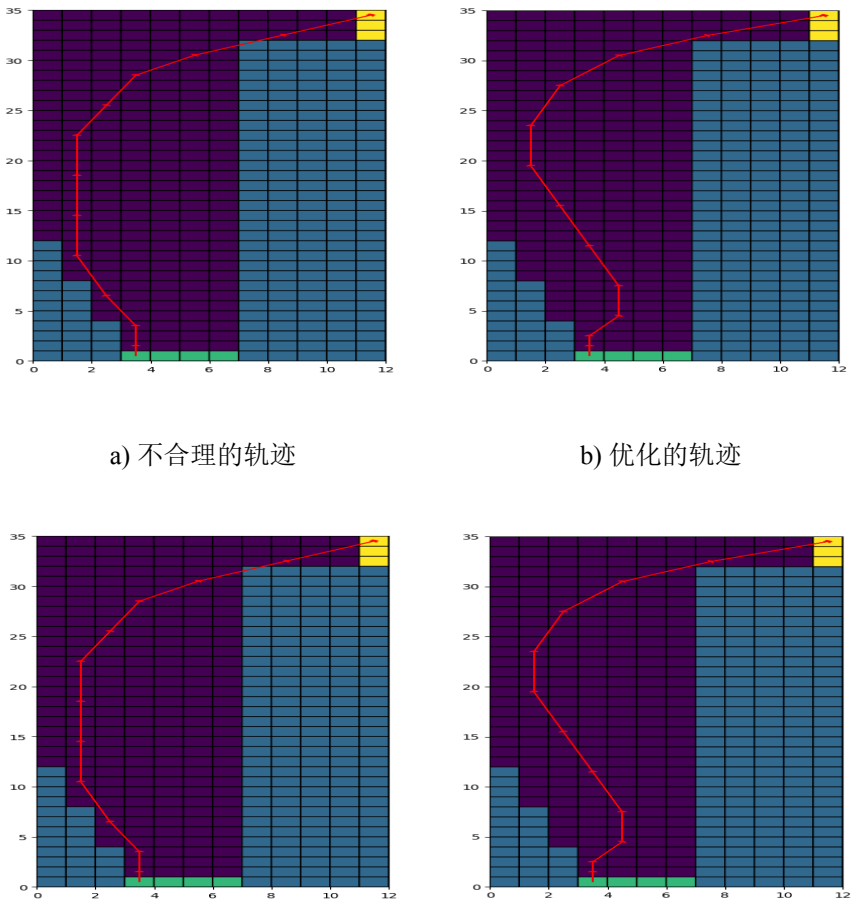
图 3-3 GTSpy

进行同样的设置，还需进一步验证可行性。另外图的排版方法很多，有些宏包已经被弃用，所以尽量使用本文给出的案例的格式进行排版图片。

常见的子图包有 subfigure 和 subfig。subfigure 是比较老的了，这里使用 subfig 包。两个包在使用的时候用法不同，千万不要混淆了，不然可能会报错。subfig 包的命令是\subfloat。这里给出一种使用 subfig 包的常用排版，如图3-4的子图3-4 b)，其中a)的试验并不好（这里测试了交叉引用\subref{xxx} 和\subref*{xxx}）。必要时也可以排版多行多列的图、调整图之间的间距，具体可百度。

```
\begin{figure}[!h]
    \centering
    \subfloat[不合理的轨迹]{\includegraphics[width=6cm,height=6cm]{Fig/Figure_1.png}%
        \label{Fig:1:a}}
    \subfloat[优化的轨迹]{\includegraphics[width=6cm,height=6cm]{Fig/Figure_2.png}
        \label{Fig:1:b}}
    \\ % 用 \\ 换行，也可以此处空一行进行换行，只有两个图的话下面就不需要了。
    \subfloat[不合理的轨迹]{\includegraphics[width=6cm,height=6cm]{Fig/Figure_1.png}%
        \label{Fig:1:c}}
    \subfloat[优化的轨迹]{\includegraphics[width=6cm,height=6cm]{Fig/Figure_2.png}%
        \label{Fig:1:d}}
    \caption{子图包使用测试}\label{Fig:1}
\end{figure}
-------------------------------------------------------------
% 引用某子图时使用\subref{xxx}，其中xxx就是标签Fig:1:a
子图的引用比较特殊，命令有：\subref{xxx}和\subref*{xxx}
注：在subfig包使用说明中，\subref{xxx}和\subref*{xxx}分别由参数listofformat和
    subrefformat控制，
并由如下定义，根据撰写规范需要定义为：
\DeclareSubrefFormat{empty}{}
\DeclareSubrefFormat{simple}{#1#2}
\DeclareSubrefFormat{parens}{#1 #2)}
\DeclareSubrefFormat{subsimple}{#2}
\DeclareSubrefFormat{subparens}{ #2)}
和
\DeclareCaptionListOfFormat{empty}{}
\DeclareCaptionListOfFormat{simple}{#1#2}
```

```
\DeclareCaptionListOfFormat{parens}{#1 #2)}
\DeclareCaptionListOfFormat{subsimple}{#2}
\DeclareCaptionListOfFormat{subparens}{ #2)}
```



a) 不合理的轨迹　　b) 优化的轨迹

c) 不合理的轨迹　　d) 优化的轨迹

图 3-4 子图包使用测试

3.2 表

本节仅展示使用常见的三线表

```
\begin{table}
    \caption{\label{TDF_para}涵道模型参数}  %表题在上
    \centering  % 表居中
    \small  % 表内字体小一号（即设置成和表题字号一致）
    \begin{tabular}{cccc}   % cccc表示4列并居中，若列之间需要分隔符则设置为|c|c|c|c|
        \hline  % \hline表示横线。列之间的元素用&分隔，\tabularnewline表示换行
        参数符号 & 数值 & 参数符号 & 数值 \tabularnewline
        \hline
        $I_x$ & $054593$           & $I_y$ & $0.017045         $ \tabularnewline
        $l_1$ & $0.0808\,\text{m}$ & $l_2$ & $0.175\,\text{m}  $ \tabularnewline
        $l_4$ & $0.2415\,\text{m}$ & $l_5$ & $0.1085\,\text{m} $ \tabularnewline
        \hline
```

```
    \end{tabular}
\end{table}
```

表 3-1 涵道模型参数

参数符号	数值	参数符号	数值
I_x	054593	I_y	0.017045
l_1	0.0808 m	l_2	0.175 m
l_4	0.2415 m	l_5	0.1085 m

3.3 公式

除了前面讲行内公式，常用的还有行间公式。公式中的数学符号可自行百度，本章仅介绍常用的几种公式环境。

单独成行的行间公式在 LaTeX 里由 equation 环境包裹。equation 环境为公式自动生成一个编号，这个编号可以用\label 和\ref 生成交叉引用，amsmath 宏包的\eqref 可为引用自动加上圆括号；如式(3-1)所示。

```
\begin{equation}
    a+b=c   \label{eq_1}
\end{equation}
```

$$a+b=c \tag{3-1}$$

若不需要编号则加星号，改为

```
\begin{equation*}
    a+b=c
\end{equation*}
```

其他环境类似。当使用 $ 开启行内公式输入，或是使用 equation 环境时，LaTeX 就进入了数学模式。数学模式相比于文本模式有以下特点：

1) 数学模式中输入的空格被忽略。数学符号的间距默认由符号的性质（关系符号、运算符等）决定。需要人为引入间距时，使用 \quad 和 \qquad 等命令。

2) 不允许有空行（分段）。行间公式中也无法用 \\ 命令手动换行。排版多行公式需要用到其他各种环境。

3) 所有的字母被当作数学公式中的变量处理，字母间距与文本模式不一致，也无法生成单词之间的空格。如果想在数学公式中输入正体的文本，简单情况下可用 \mathrm

命令。或者用 amsmath 提供的 \text 命令（仅适合在公式中穿插少量文字。如果你的情况正好相反，需要在许多文字中穿插使用公式，则应该像正常的行内公式那样用，而不是滥用 \text 命令）。

实际上更常用的的是多行公式，不需要对齐的公式组可以使用 gather 环境，需要对齐的公式组用 align 环境。长公式内可用 \\ 换行。

如果需要罗列一系列公式，并令其按照等号对齐，可用 align 环境，它将公式用 & 隔为两部分并对齐。分隔符通常放在等号左边：

```
\begin{align}
    a & = b + c \\
    & = d + e
\end{align}
```

$$a = b + c \tag{3-2}$$

$$= d + e \tag{3-3}$$

align 环境会给每行公式都编号。

如果不需要按等号对齐，只需罗列数个公式，可用 gather 环境：

```
\begin{gather}
    a  = b + c \notag \\
    f = d + e
\end{gather}
```

$$a = b + c$$

$$f = d + e \tag{3-4}$$

gather 环境同样会给每行公式都编号，如果某行不需要编号可在行末用\notag 仅去掉某行的编号。

align 和 gather 有对应的不带编号的版本 align* 和 gather*。

另一个常见的需求是将多个公式组在一起公用一个编号，编号位于公式的居中位置。为此，amsmath 宏包提供了诸如 aligned、gathered 等环境，与 equation 环境套用。以-ed 结尾的环境用法与前一节不以-ed 结尾的环境用法一一对应。我们仅以 aligned 举例：

```
\begin{equation}
    \begin{aligned}
        a &= b + c \\
        d &= e + f + g \\
        h + i &= j + k \\
        l + m &= n
```

```
    \end{aligned}
\end{equation}
```

$$\begin{aligned} a &= b + c \\ d &= e + f + g \\ h + i &= j + k \\ l + m &= n \end{aligned} \tag{3-5}$$

split 环境和 aligned 环境用法类似，也用于和 equation 环境套用，区别是 split 只能将每行的一个公式分两栏，aligned 允许每行多个公式多栏。

分段函数通常用 amsmath 宏包提供的 cases 环境，可参考文献 [3]

amsmath 宏包还直接提供了多种排版矩阵的环境，包括不带定界符的 matrix，以及带各种定界符的矩阵 pmatrix、bmatrix、Bmatrix、vmatrix、Vmatrix。其中中括号版的 bmatrix 最常用。这些矩阵环境需要在公式中使用，比如 gather 环境。

```
\begin{gather}
    A= \begin{bmatrix}
        x_{11} & x_{12} & \ldots & x_{1n} \\
        x_{21} & x_{22} & \ldots & x_{2n} \\
        \vdots & \vdots & \ddots & \vdots \\
        x_{n1} & x_{n2} & \ldots & x_{nn}
    \end{bmatrix}
\end{gather}
```

$$\boldsymbol{A} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nn} \end{bmatrix} \tag{3-6}$$

其中矩阵/向量加粗使用\bm{} 命令。另外还可以使用 array 环境排版矩阵，类似 tabular 环境，用 \\ 和 & 用来分隔行和列，这里不再赘述。

```
\begin{array }[外部对齐tcb]{列对齐lcr}
    行列内容
\end{array}
```

另外注意排版分式时，有两种方法：\frac 或者\dfrac，效果分别为 $\frac{1}{2}$ 和 $\dfrac{1}{2}$。以上介绍的数学环境中，空格可参考文献 [3]，例如常用\quad。

3.4 定理

在 scutthesis.cls 文件 536 行开始，已经用\newtheorem 命令定义了几种定理环境，包括：定义、假设、定理、结论、引理、公理、推论、性质等等，统称定理环境，关于\newtheorem 的用法，可参考[3] 或自行百度。要下面提供几个例子，在横线之间的深色区域是代码，效果在相应下方表示：

```
\begin{assumption}
    加权矩阵${{\bm{W}}_{1}}$和 ${{\bm{W}}_{2}}$ 是对称矩阵，且$ {{\bm{W}}_{2}}$非奇异。
    \label{assum_dca1}
\end{assumption}
```

假设 3.1: 加权矩阵 $\boldsymbol{W}_1$ 和 $\boldsymbol{W}_2$ 是对称矩阵，且 $\boldsymbol{W}_2$ 非奇异。

定理用法和假设类似：

```
\begin{theorem}
    如果假设\ref{assum_dca1}成立，$\bm{F}$满足式\eqref{eq_F}的定义，且${{\bm{W}}_{1}}$非奇异，则有$0\le e \left( \bm{F} \right) < 1$，其中$e \left( \bm{F} \right)$是 $\bm{F}$的特征值。    \label{the_dca2}
\end{theorem}
```

定理 3.1: *如果假设3.1成立，$\boldsymbol{F}$ 满足上式的定义，且 $\boldsymbol{W}_1$ 非奇异，则有 $0 \le e\left(\boldsymbol{F}\right) < 1$，其中 $e\left(\boldsymbol{F}\right)$ 是 $\boldsymbol{F}$ 的特征值。*

定理环境的编号可自定义，但通常不需要再进行设置，因为模板文件 scutthesis.cls 文件已经定义好。

3.5 参考文献

再次强调，使用其他参考文献管理软件的用户以及不使用任何软件的“裸奔”的用户不需要关注任何关于 zetero 的东西。

```
关于参考文献这块，很多同学有疑问。只有记住一点：不管用什么参考文献管理工具，最终目的是生成一个bib文件给TeXstudio使用，bib文件里是特定格式的文献信息。bib文件可以使用一个叫notepad++的软件打开（也可以用其他，当作文本打开）。
```

通常学位论文参考文献是基于 BibTeX 进行的，本模板最大的改进就是引入 BibLaTeX。关于这部分知识可参考文献 [3, 5] 的第六章，6.1 节参考文献和 BIBTEX 工具。

参考文献引用和著录是基于 ZOTERO 这个软件进行的。视频教程见 [6]。此外，为了符合毕业论文撰写规范，需设置参数。按照视频教程安装完必要的插件（如 Better BibTeX）后，在编辑-> 首选项进行设置。图3-5到图3-15所示的是我的 zotero 软件设置。其中最重要的是3-14的设置要排除的选项，多余的显示会让审稿人反感，按照论文撰写

规范进行即可。在毕业论文撰写时，在编辑-> 首选项->Better BibLTeX->Fields 中，Fields to omit from export 填 month,abstract,note,extra,file,keywords,type,url,doi，就是在参考文献著录中排除这些多余的项，避免过于复杂。而在写本模板使用说明时，没有排除 url，因为很多参考资料是网页。

使用zotero，科学上网很重要，通常我们使用谷歌学术搜索文献并利用chrome的zotero插件直接捕获文献著录信息。但我使用蓝灯，代理服务器均遇到过被谷歌学术封锁的情况。只能不断换科学上网方法。这里我现在用的chrome插件：谷歌上网助手，它可以轻松捕获谷歌学术的著录信息，注册一个账号即可使用。谷歌上网助手有可能和某些代理冲突。这些都是科学上网的问题，已经超出了本项目的范围，听说百度一下 v2ray 可发现新大陆，可惜我试了Vultr的服务器依然被谷歌封。知网捕获中文参考文献著录信息的话不需要考虑这个问题，直接在知网首页搜索文献然后点击插件既可以选想捕获的著录了。

在 zotero 软件点击文件-> 导出文献库，如图3-16所示，再在导出对话框图3-17选择导出格式为 Better BibLaTeX，同时勾选 Keep updated 选项保持自动更新，再点击 ok，在弹出的对话框图3-18确定保存路径和文件名，例如我的是 MyLibrary.bib，这也是我整个读书生涯的文献库 bib 文件。如果写小论文的话通常导出格式是 BibTeX 或者 Better BibTeX（这里按照期刊的要求来即可，文献管理软件的好处就是快速自动生成一个文件库）。关于 BibTeX 和 BibLaTeX 的区别这里不做展开。

得到文献库后，在 scutthesis.tex 文件第九行使用\addbibresource 命令，添加文献库。引用某文献时秩序在 zotero 选中某文献条目，然后按 Ctrl+Shift+C，复制引用关键字（Citation Key）到剪切板（快捷键可自定义）。然后在 tex 文件编辑界面直接粘贴，默认的时上标形式，若需要非上标形式，可以改为\parencite{xxx}，其中 xxx 是 Citation Key。这里的操作和认为设置的首选项参数有关，需要在编辑-> 首选项-> 导出界面的默认格式一栏选中相应的项，同时在编辑-> 首选项-> 高级-> 快捷键设置为默认值。

2020 年 12 月 2 日测试：下载最新 zotero，从知网和谷歌捕获文献（刚打开网页最好稍等一会再点击插件，谷歌可能需要现人机验证），对文献 [7]、[8] 进行引用。

2021 年 9 月 14 日测试：使用 endnote 的用户也可以利用导出的 bib 文件生成参考文献著录信息，导出选项是 bibTeX，貌似没有更多导出设置选项。导出设置没有 zotero 那么灵活丰富，得到 bib 文件后要引用某论文需要自行查找标签（label，也有软件叫引用关键字 Citation Key）{xxx} 然后手打\cite{xxx}。欢迎熟悉 endnote 的同学来信告诉我更好的办法。

另外有同学反映，换了电脑后重新导出的bib文件Citation Key值不同，记得设置好Better BibTeX之后，在著录条目界面全选著录（或仅选想更新的著录）然后右键选Better BibTeX更新refresh一下。然后在Automatic export选项点击Export now立即更新bib文件（按理说勾选了自动更新选项他会自动更新，但为了确保万无一失还是点一下）。

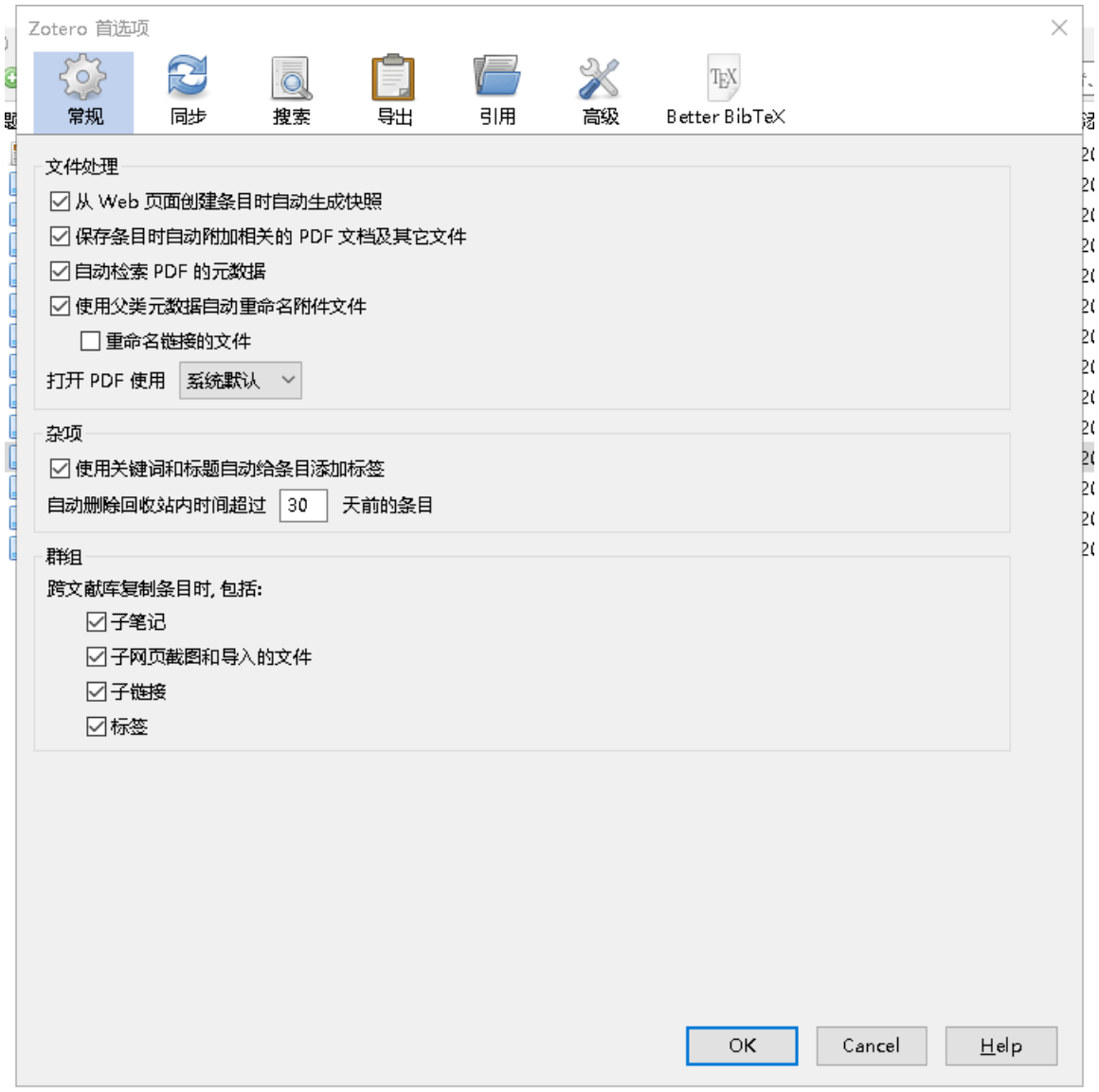


图 3-5 常规

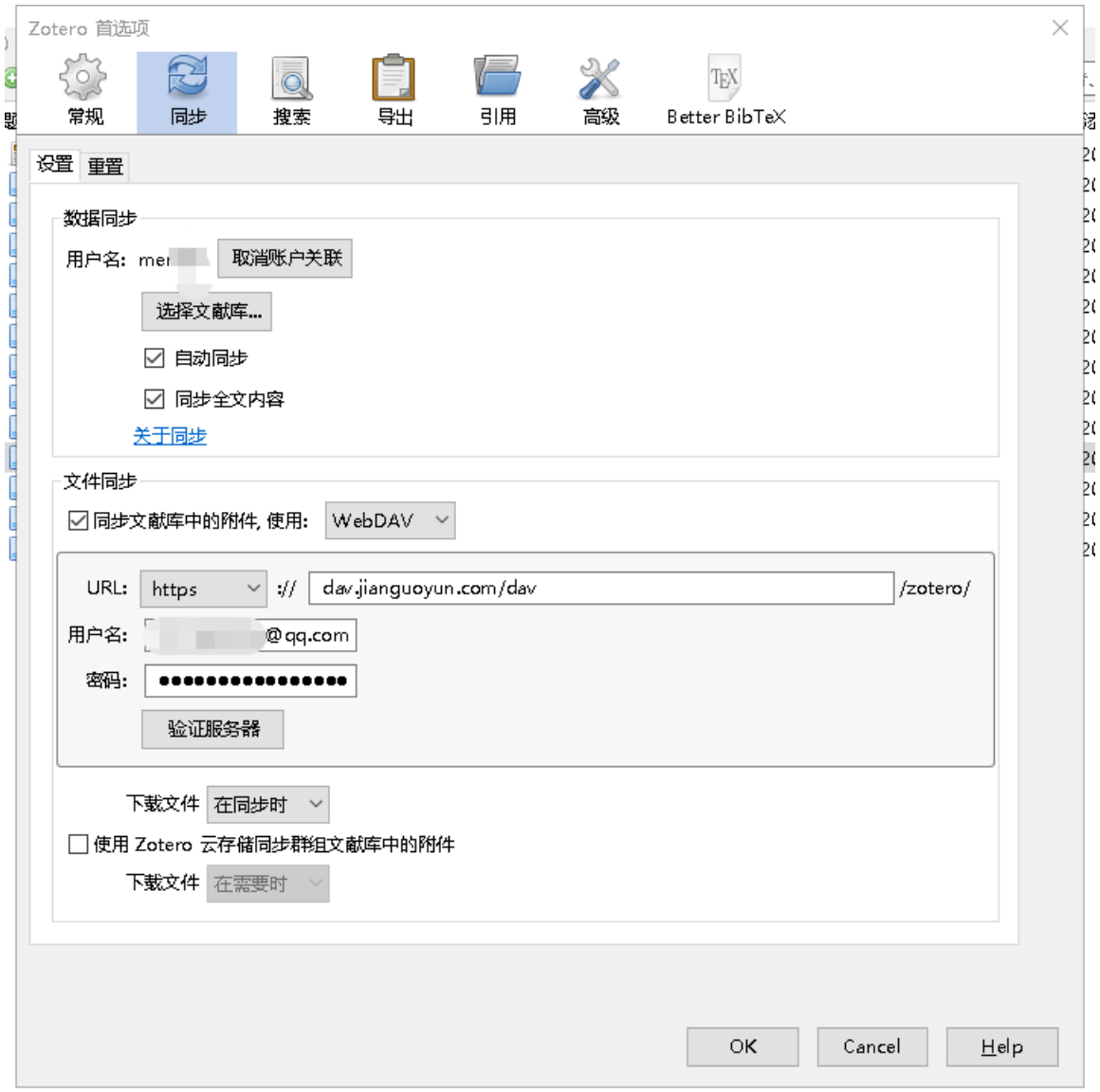

Zotero 首选项
常规
同步
搜索
导出
引用
高级
Better BibTeX
设置
重置
数据同步
用户名: mer
取消账户关联
选择文献库...
自动同步
同步全文内容
关于同步
文件同步
同步文献库中的附件, 使用:
WebDAV
URL:
https
://
dav.jianguoyun.com/dav
/zotero/
用户名:
@qq.com
密码:
验证服务器
下载文件
在同步时
使用 Zotero 云存储同步群组文献库中的附件
下载文件
在需要时
OK
Cancel
Help

图 3-6 同步 1

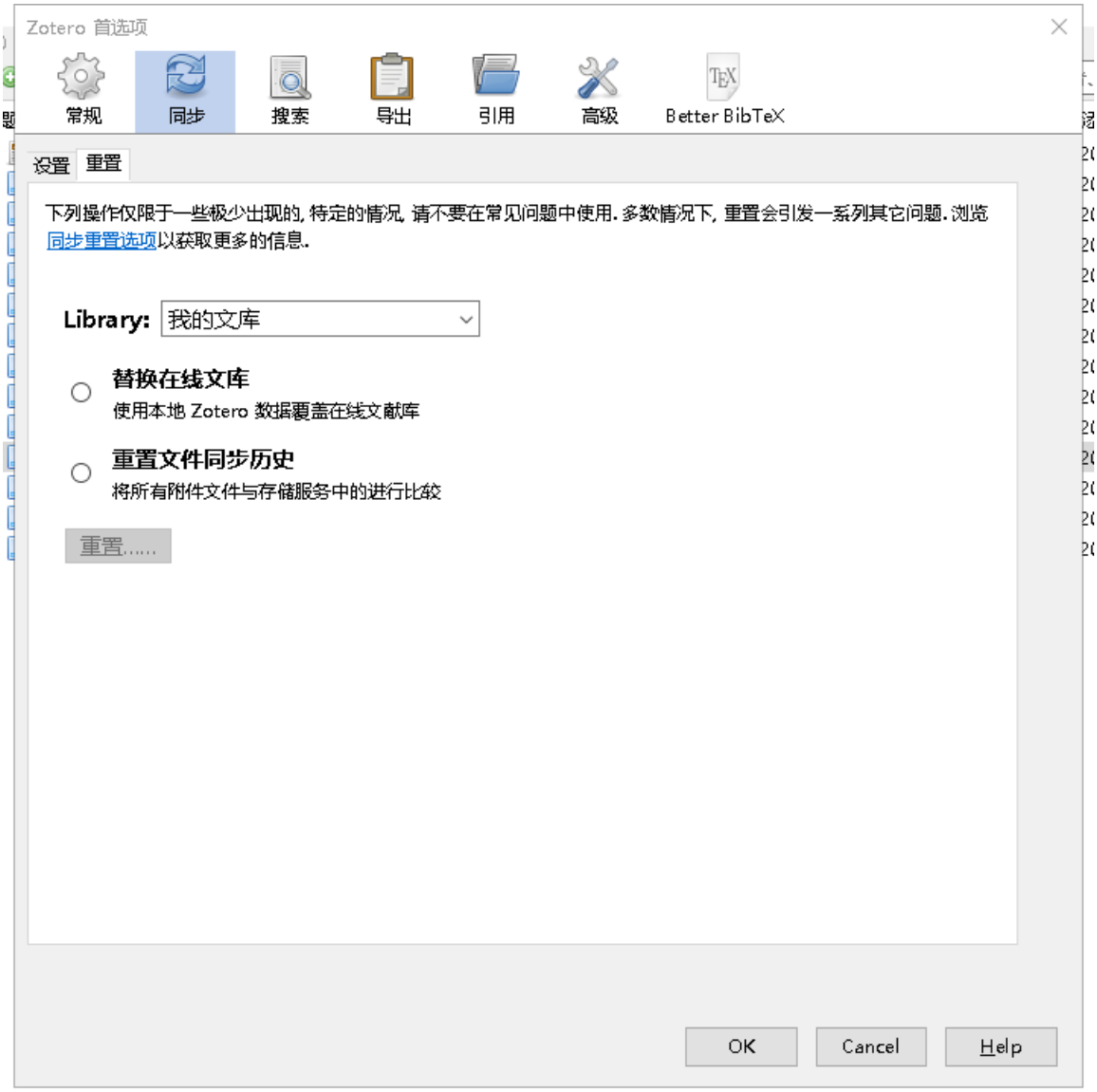

Zotero 首选项
常规
同步
搜索
导出
引用
高级
Better BibTeX
设置
重置
下列操作仅限于一些极少出现的，特定的情况，请不要在常见问题中使用。多数情况下，重置会引发一系列其它问题。浏览同步重置选项以获取更多的信息。
Library:
我的文库
替换在线文库
使用本地 Zotero 数据覆盖在线文献库
重置文件同步历史
将所有附件文件与存储服务中的进行比较
重置.....
OK
Cancel
Help

图 3-7 同步 2

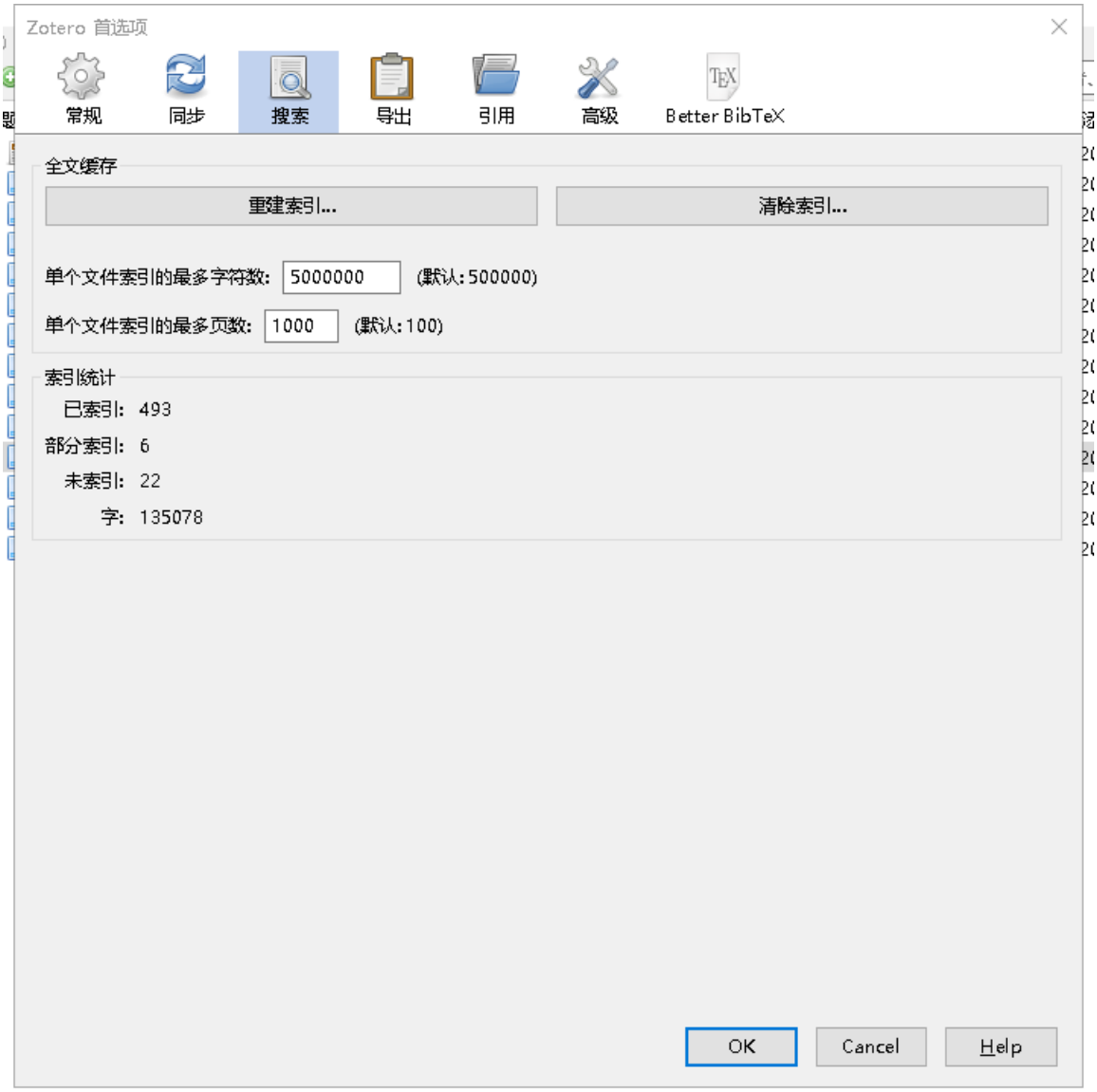

Zotero 首选项
常规
同步
搜索
导出
引用
高级
Better BibTeX
全文缓存
重建索引...
清除索引...
单个文件索引的最多字符数: 5000000 (默认: 500000)
单个文件索引的最多页数: 1000 (默认: 100)
索引统计
已索引: 493
部分索引: 6
未索引: 22
字: 135078
OK
Cancel
Help

图 3-8 搜索

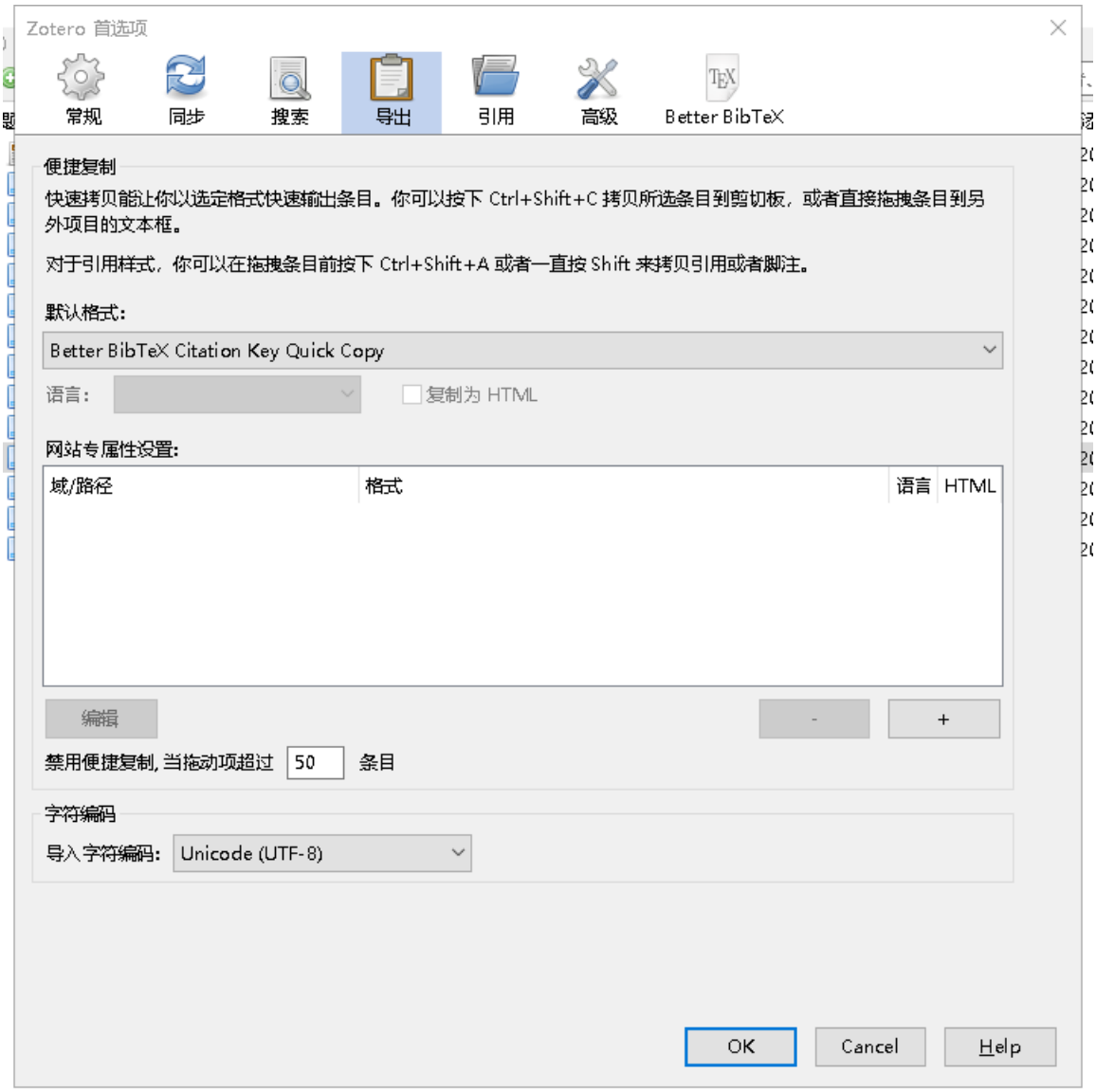

Zotero 首选项
常规
同步
搜索
导出
引用
高级
Better BibTeX
便捷复制
快速拷贝能让你以选定格式快速输出条目。你可以按下 Ctrl+Shift+C 拷贝所选条目到剪切板，或者直接拖拽条目到另外项目的文本框。
对于引用样式，你可以在拖拽条目前按下 Ctrl+Shift+A 或者一直按 Shift 来拷贝引用或者脚注。
默认格式:
Better BibTeX Citation Key Quick Copy
语言:
复制为 HTML
网站专属性设置:
域/路径
格式
语言
HTML
编辑
-
+
禁用便捷复制，当拖动项超过
50
条目
字符编码
导入字符编码:
Unicode (UTF-8)
OK
Cancel
Help

图 3-9 导出

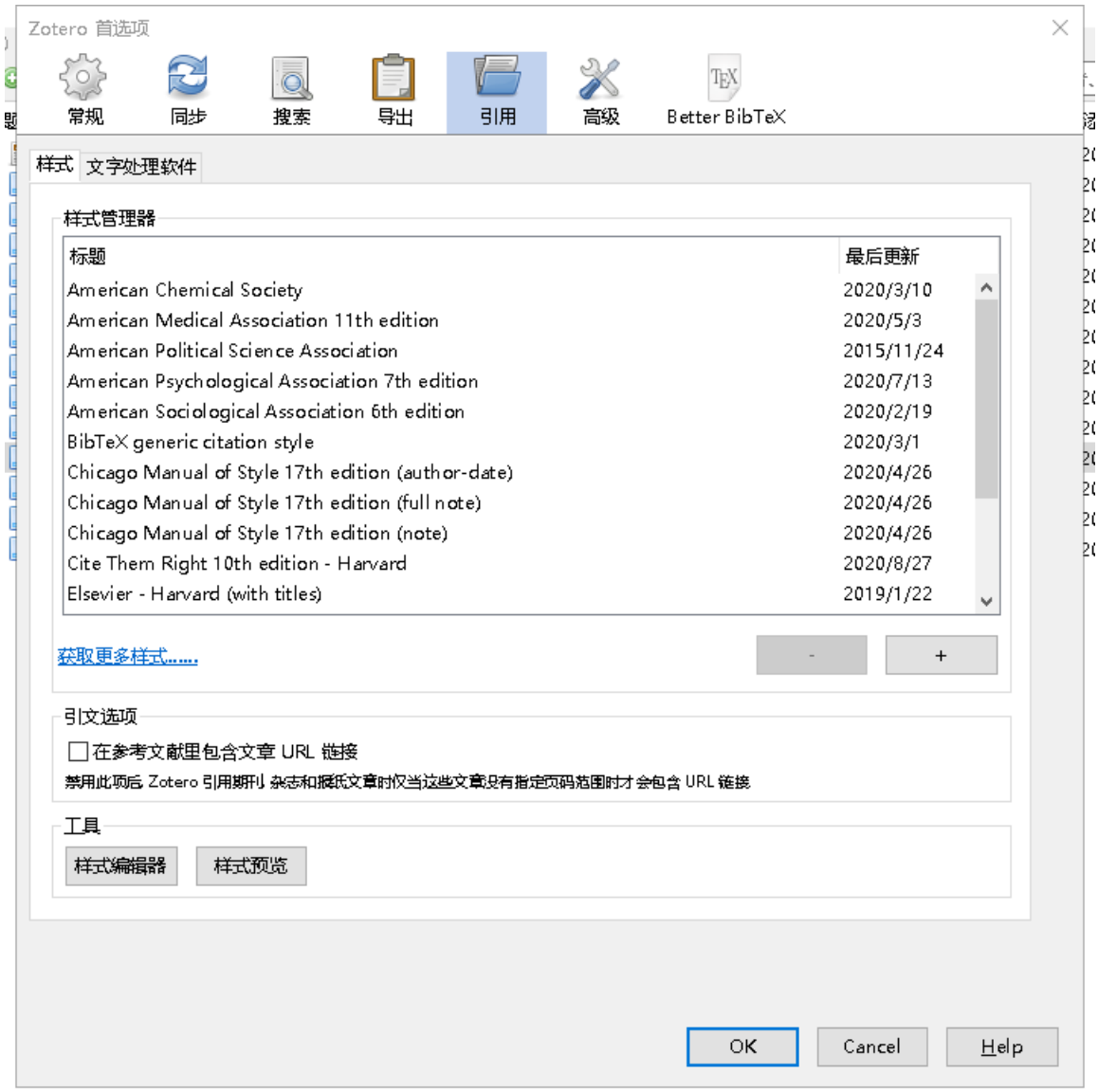

Zotero 首选项
常规
同步
搜索
导出
引用
高级
Better BibTeX
样式
文字处理软件
样式管理器
标题
最后更新
American Chemical Society 2020/3/10
American Medical Association 11th edition 2020/5/3
American Political Science Association 2015/11/24
American Psychological Association 7th edition 2020/7/13
American Sociological Association 6th edition 2020/2/19
BibTeX generic citation style 2020/3/1
Chicago Manual of Style 17th edition (author-date) 2020/4/26
Chicago Manual of Style 17th edition (full note) 2020/4/26
Chicago Manual of Style 17th edition (note) 2020/4/26
Cite Them Right 10th edition - Harvard 2020/8/27
Elsevier - Harvard (with titles) 2019/1/22
获取更多样式.....
-
+
引文选项
在参考文献里包含文章 URL 链接
禁用此项后 Zotero 引用期刊 杂志和报纸文章时仅当这些文章没有指定页码范围时才会包含 URL 链接
工具
样式编辑器
样式预览
OK
Cancel
Help

图 3-10 引用

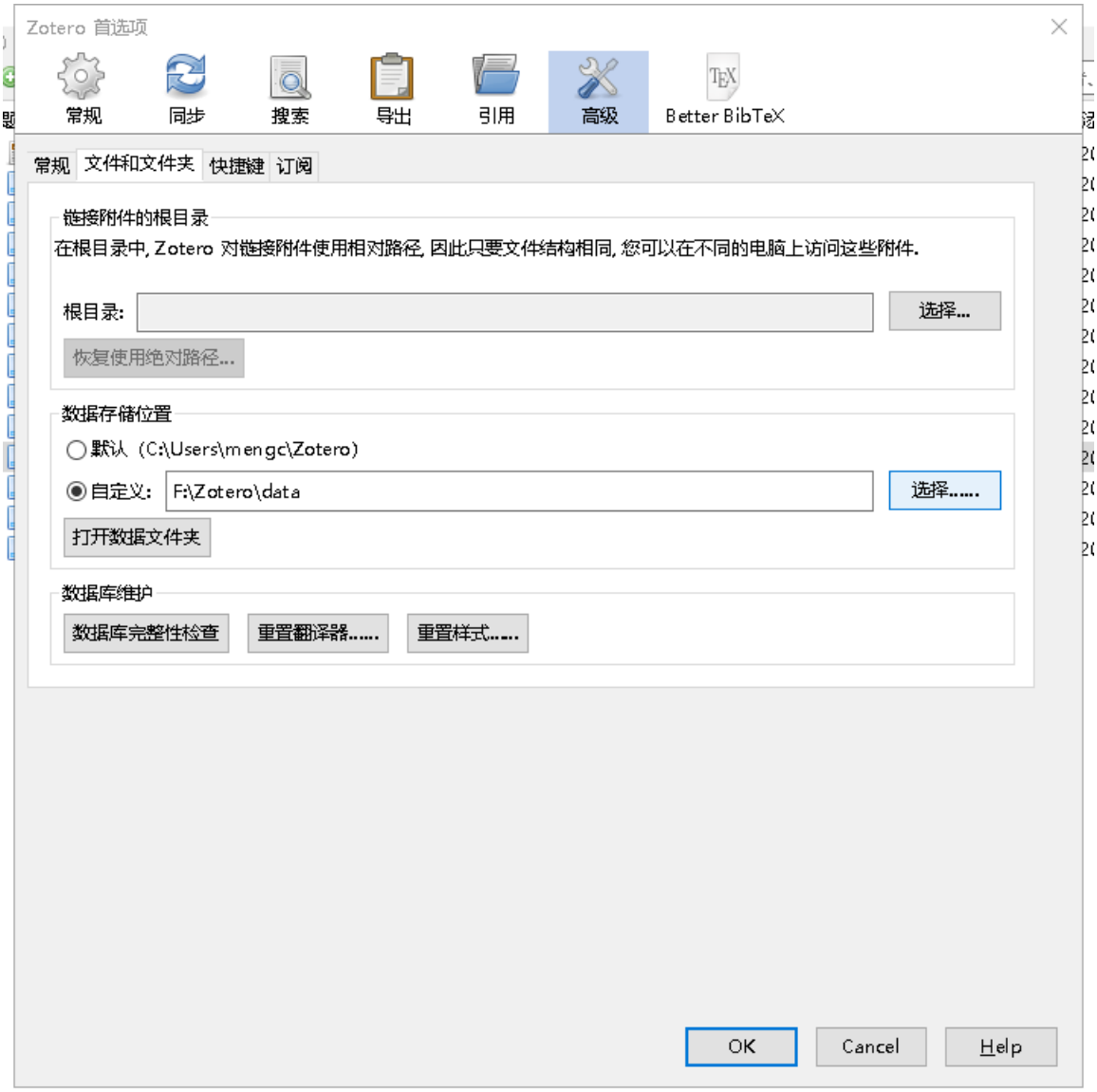

Zotero 首选项
常规
同步
搜索
导出
引用
高级
Better BibTeX
常规
文件和文件夹
快捷键
订阅
链接附件的根目录
在根目录中, Zotero 对链接附件使用相对路径, 因此只要文件结构相同, 您可以在不同的电脑上访问这些附件.
根目录:
选择...
恢复使用绝对路径...
数据存储位置
默认 (C:\Users\mengc\Zotero)
自定义:
F:\Zotero\data
选择.....
打开数据文件夹
数据库维护
数据库完整性检查
重置翻译器.....
重置样式.....
OK
Cancel
Help

图 3-11 高级 1

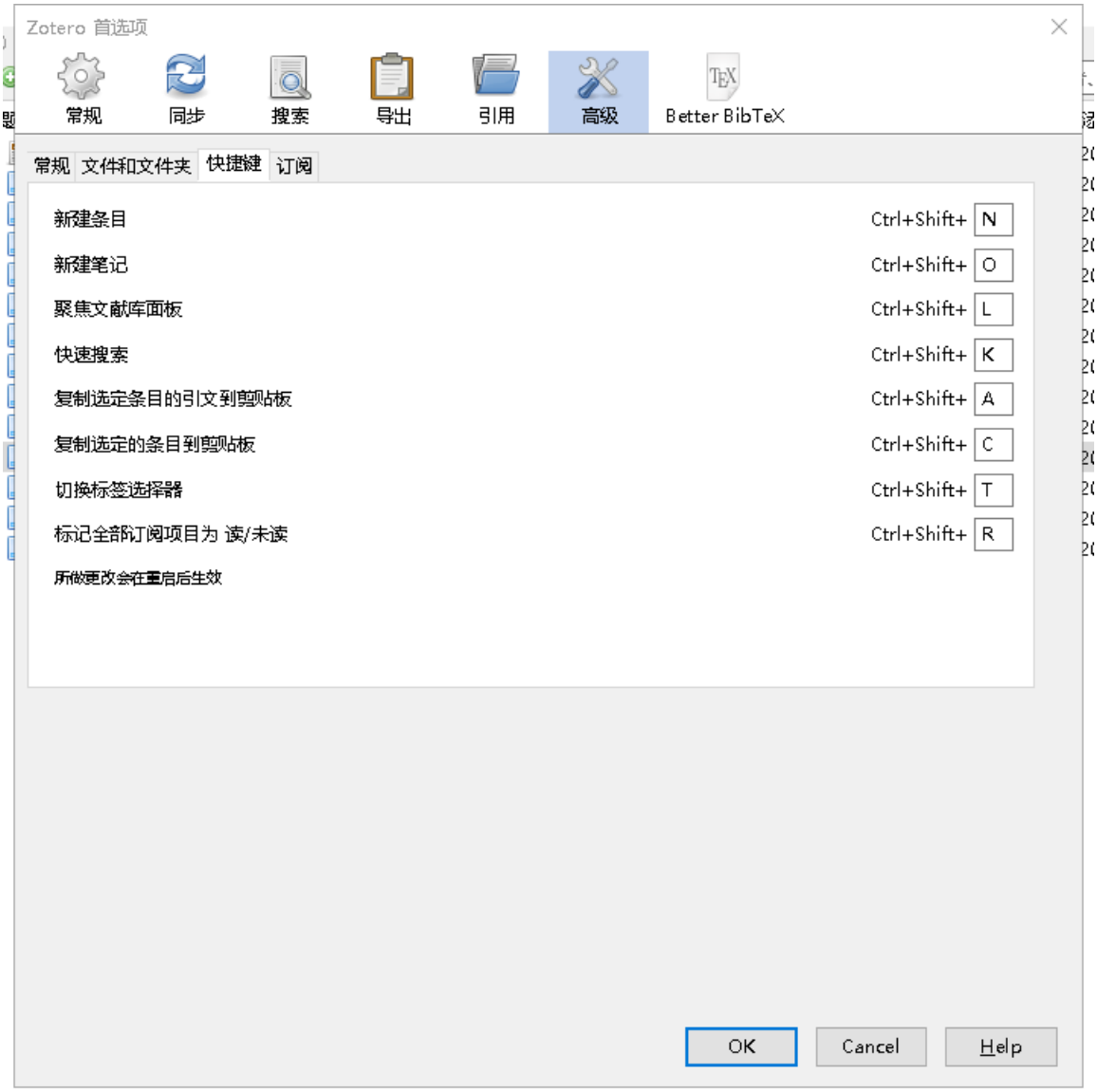

Zotero 首选项
常规
同步
搜索
导出
引用
高级
Better BibTeX
常规
文件和文件夹
快捷键
订阅
新建条目 Ctrl+Shift+ N
新建笔记 Ctrl+Shift+ O
聚焦文献库面板 Ctrl+Shift+ L
快速搜索 Ctrl+Shift+ K
复制选定条目的引文到剪贴板 Ctrl+Shift+ A
复制选定的条目到剪贴板 Ctrl+Shift+ C
切换标签选择器 Ctrl+Shift+ T
标记全部订阅项目为 读/未读 Ctrl+Shift+ R
所做更改会在重启后生效
OK
Cancel
Help

图 3-12 高级 2

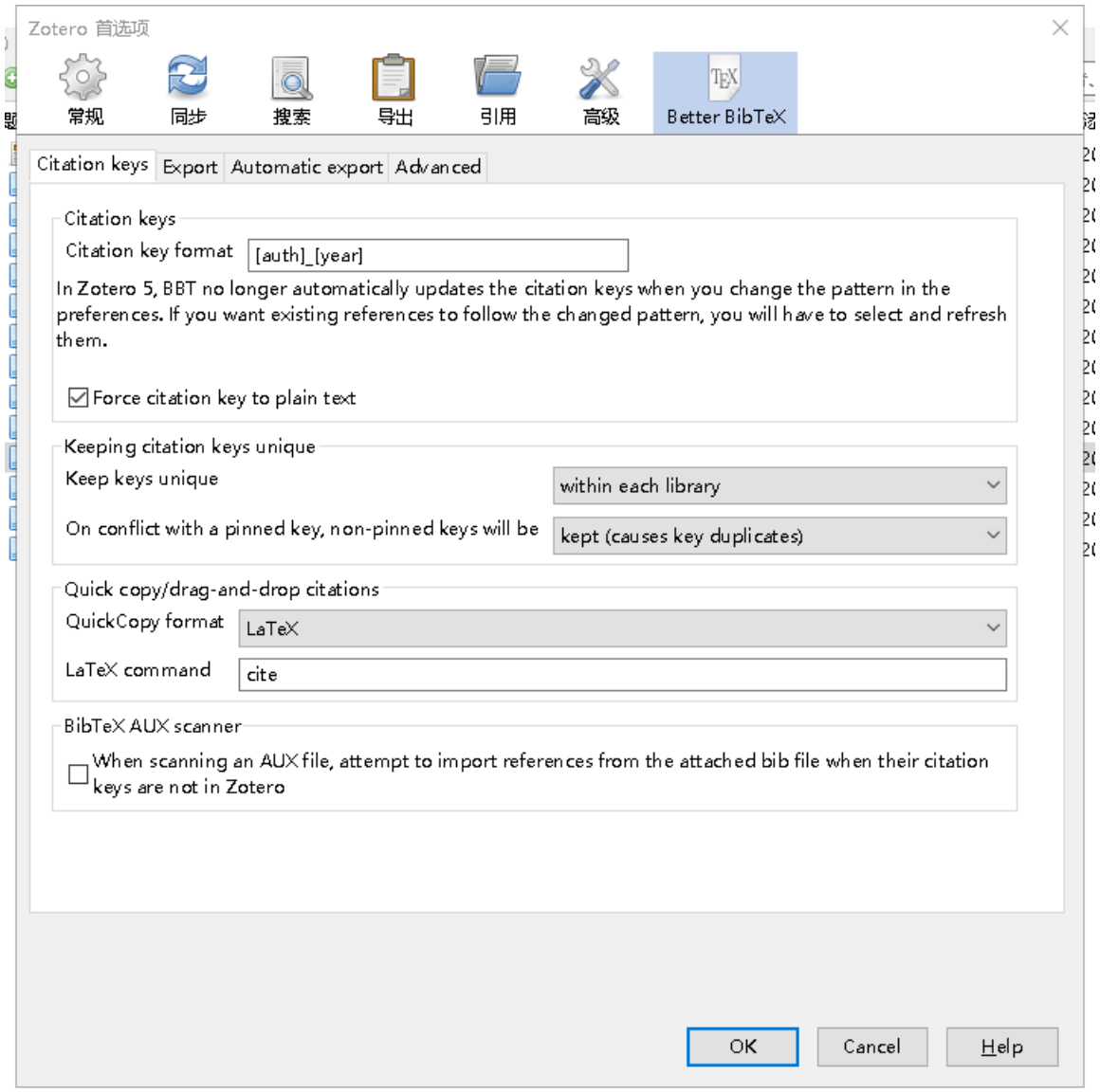


图 3-13 Better BibTeX1

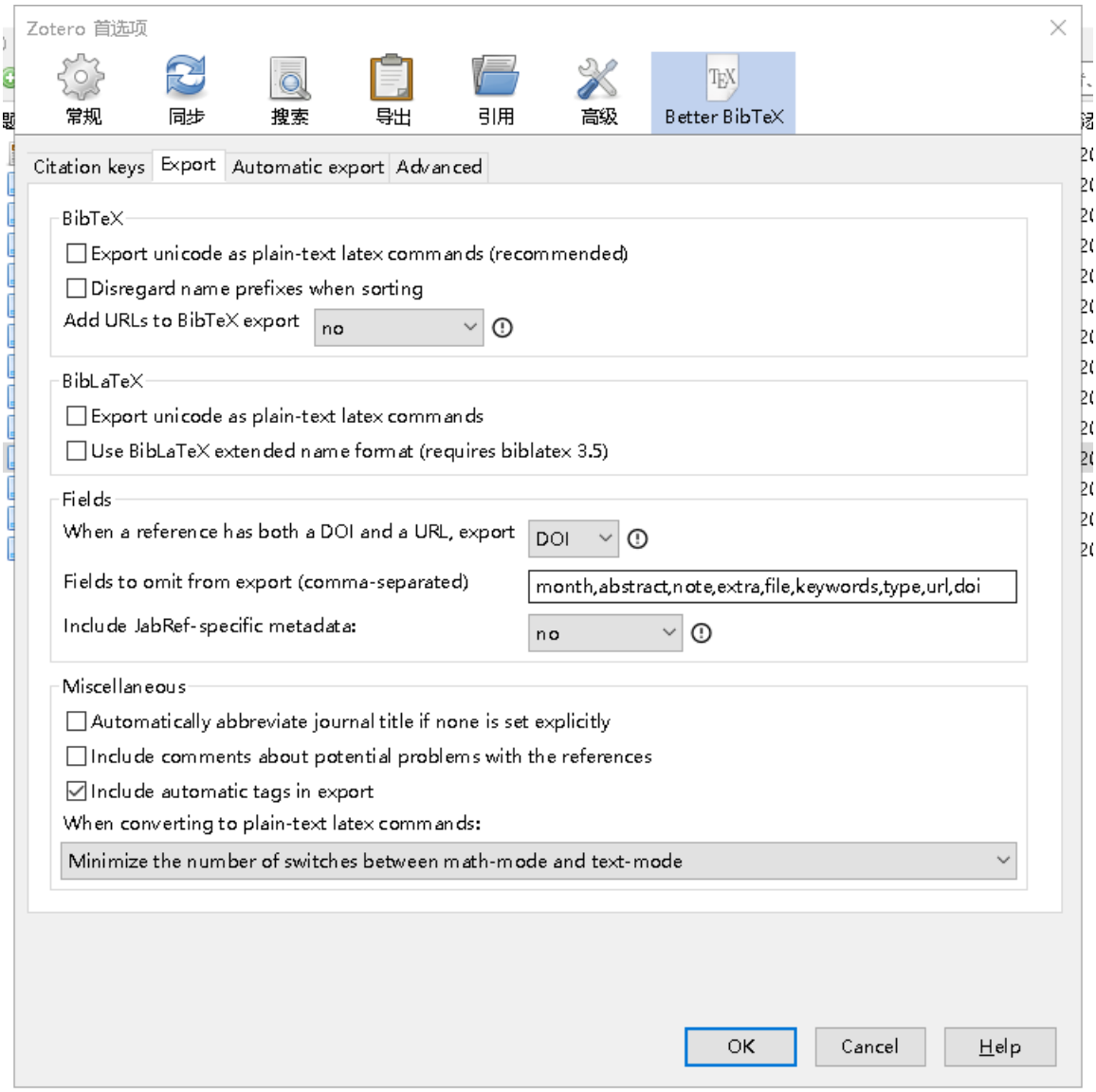


图 3-14 Better BibTeX2

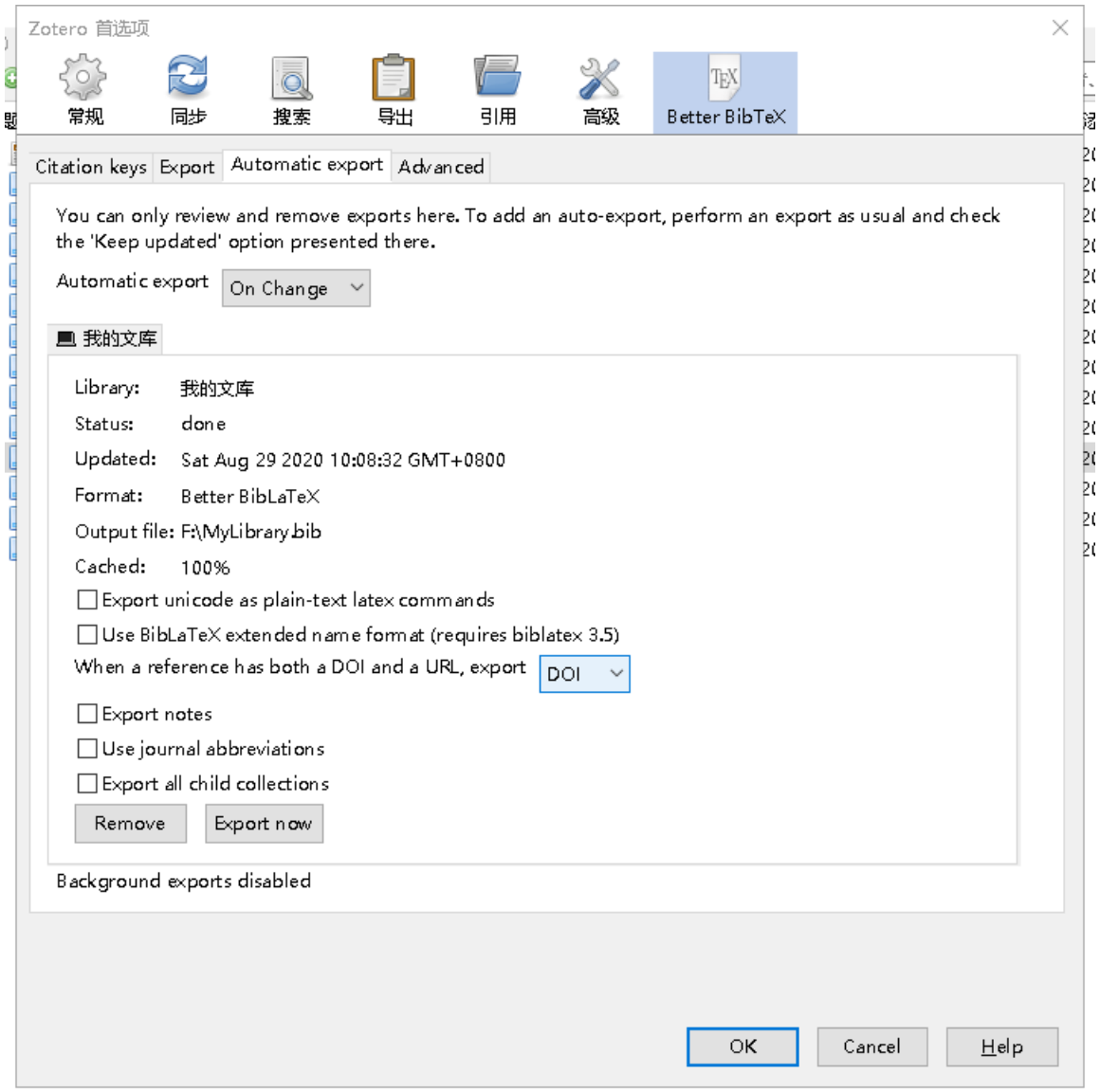


图 3-15 Better BibTeX3

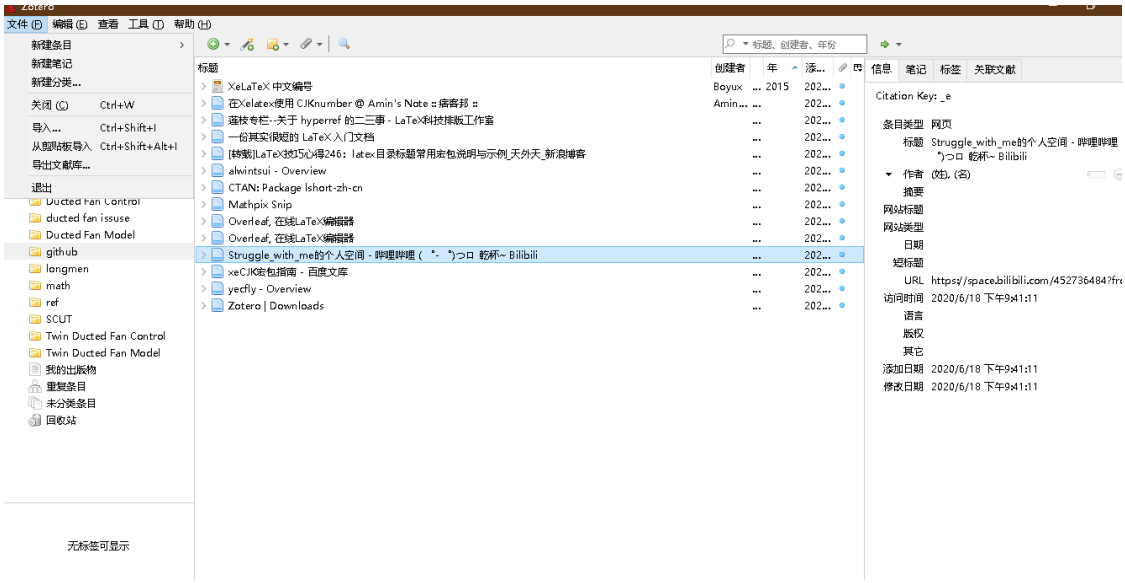


图 3-16 导出文献库

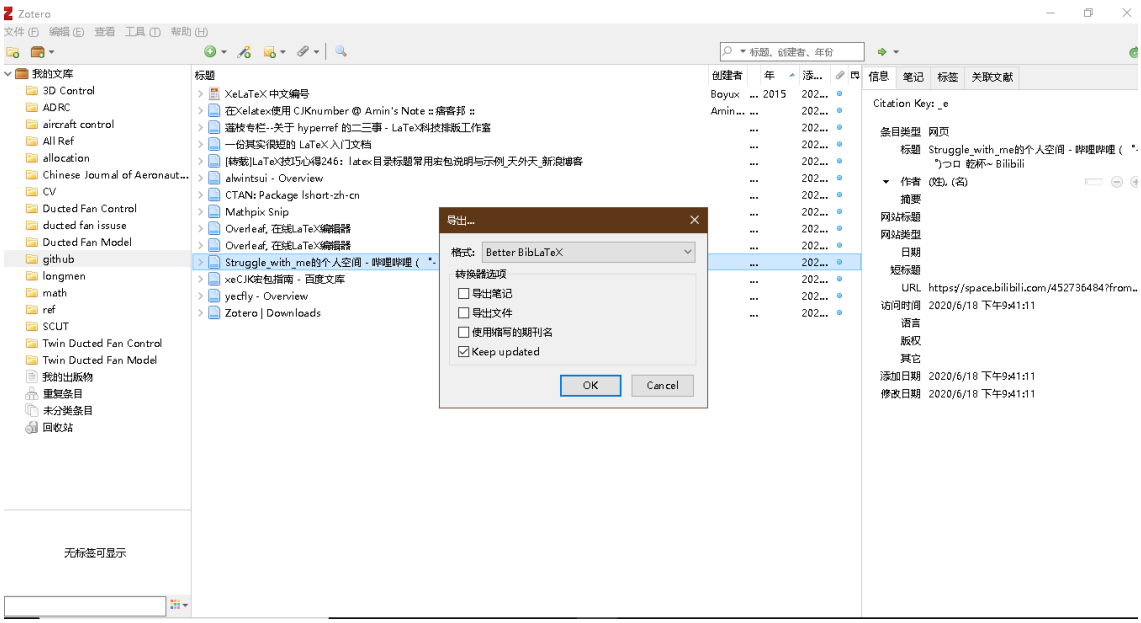


图 3-17 导出格式

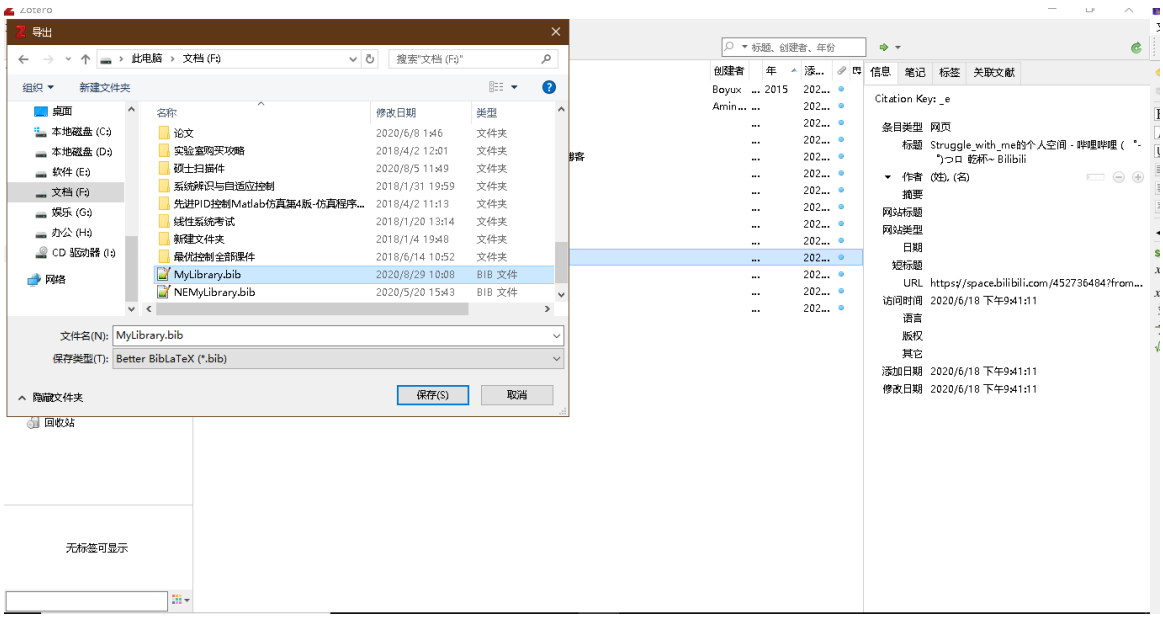

导出
此电脑 › 文档 (F:)
搜索"文档 (F:)"
组织
新建文件夹
桌面
本地磁盘 (C:)
本地磁盘 (D:)
软件 (E:)
文档 (F:)
娱乐 (G:)
办公 (H:)
CD 驱动器 (I:)
网络
名称
修改日期
类型
论文 2020/6/8 1:46 文件夹
实验室购买攻略 2018/4/2 12:01 文件夹
硕士扫描件 2020/8/5 11:49 文件夹
系统辨识与自适应控制 2018/1/31 19:59 文件夹
先进PID控制Matlab仿真第4版-仿真程序... 2018/4/2 11:13 文件夹
线性系统考试 2018/1/20 13:14 文件夹
新建文件夹 2018/1/4 19:48 文件夹
最优控制全部课件 2018/6/14 10:52 文件夹
MyLibrary.bib 2020/8/29 10:08 BIB 文件
NEMyLibrary.bib 2020/5/20 15:43 BIB 文件
文件名(N): MyLibrary.bib
保存类型(T): Better BibLaTeX (*.bib)
隐藏文件夹
保存(S)
取消
回收站
无标签可显示
标题、创建者、年份
创建者
年
添...
信息
笔记
标签
关联文献
Citation Key: _e
条目类型 网页
标题 Struggle_with_me的个人空间 - 哔哩哔哩 (゜- ゜)つロ 乾杯~ Bilibili
作者 (姓), (名)
摘要
网站标题
网站类型
日期
短标题
URL https://space.bilibili.com/452736484?from...
访问时间 2020/6/18 下午9:41:11
语言
版权
其它
添加日期 2020/6/18 下午9:41:11
修改日期 2020/6/18 下午9:41:11

图 3-18 导出文件名

结　论

本文主要是展示如何使用修改“祖传模板”得到的新模板，在使用时直接替换成自己的论文内容即可。总结下来最最最麻烦的是科学上网，只有科学上网才能获取文献信息生成 bib 文件，后面就好办了。

本模板难免有不足之处，主要是我本人的论文涉及的格式有限，有些地方没探索到自然就没去设置。比如附录，附录的图文并茂等等，我本人是没有研究的，这里仅仅做了一些初步的工作，不过对很多同学来说本模板是够用的。希望有能帮助到华工的同学们，有不足之处请多多理解，可以通过邮件联系我，上班之余我会尽量回复。

本模板会一直更新——2022-2-25

参考文献

[1] Alwintsui - Overview[EB/OL]. GitHub. [2020-06-18]. https://github.com/alwintsui.

[2] Yecfly - Overview[EB/OL]. GitHub. [2020-06-18]. https://github.com/yecfly.

[3] 一份其实很短的 LaTeX 入门文档[EB/OL]. 始终. [2020-06-18]. https://liam.page/2014/09/08/latex-introduction/index.html.

[4] 莲枝专栏–关于 Hyperref 的二三事 - LaTeX 科技排版工作室[EB/OL]. [2020-06-20]. https://www.latexstudio.net/archives/4800.html.

[5] CTAN: Package Lshort-Zh-Cn[EB/OL]. [2020-06-18]. https://ctan.org/pkg/lshort-zh-cn.

[6] Struggle_with_me 的个人空间 - 哔哩哔哩 (゜- ゜) つロ乾杯～Bilibili[EB/OL]. [2020-06-18]. https://space.bilibili.com/452736484?from=search&seid=12208069428001748893.

[7] Renduchintala A, Jahan F, Khanna R, et al. A Comprehensive Micro Unmanned Aerial Vehicle (UAV/Drone) Forensic Framework[J]. Digital Investigation, 2019, 30: 52-72.

[8] 蒙超恒, 裴海龙, 程子欢. 涵道风扇式无人机的优先级控制分配[J/OL]. 航空学报, 2020, 41(10): 327-338 [2020-12-02]. https://kns.cnki.net/kcms/detail/detail.aspx?dbcode=CJFD&dbname=CJFDLAST2020&filename=HKXB202010026&v=H33nFWoKPiMVe8lDuZG26q9EEFHrc40qK0CS6t2FdhaWKR8ppb9it6SMeovM2l8e.

攻读硕士学位期间取得的研究成果

已发表（包括已接受待发表）的论文，以及已投稿、或已成文打算投稿、或拟成文投稿的论文情况（只填写与学位论文内容相关的部分）：

序号	作者（全体作者，按顺序排列）	题目	发表或投稿刊物名称、级别	发表的卷期、年月、页码	相当于学位论文的哪一部分（章、节）	被索引收录情况
1	蒙超恒、裴海龙、程子欢	涵道风扇式无人机的优先级控制分配	航空学报	已录用，2020 年 5 月	2.1、2.2、3.4、4.1、4.2、5.1 和 5.3 节	EI
2	蒙超恒、裴海龙、程子欢	Dynamic Control Allocation for A Twin Ducted Fan UAV	2020 International Conference on Guidance, Navigation and Control	已录用，2020 年 8 月	2.3、4.3 和 5.2 节	EI

注：在“发表的卷期、年月、页码”栏：

1. 如果论文已发表，请填写发表的卷期、年月、页码；

2. 如果论文已被接受，填写将要发表的卷期、年月；

3. 以上都不是，请据实填写“已投稿”，“拟投稿”。

不够请另加页。

二、与学位内容相关的其它成果（包括专利、著作、获奖项目等）

致　谢

这次你离开了没有像以前那样说再见, 再见也他妈的只是再见
我们之间从来没有想象的那么接近, 只是两棵树的距离
你是否还记得山阴路我八楼的房间, 房间里唱歌的日日夜夜
那么热的夏天你看着外面, 看着你在消逝的容颜
我多么想念你走在我身边的样子, 想起来我的爱就不能停止
南京的雨不停地下不停地下, 就像你沉默的委屈
一转眼, 我们的城市又到了夏天, 对面走来的人都眯着眼
人们不敢说话不敢停下脚步, 因为心动常常带来危险
我多么想念你走在我身边的样子, 想起来我的爱就不能停止
南京的雨不停地下不停地下, 有些人却注定要相遇
你是一片光荣的叶子，落在我卑贱的心
像往常一样我为自己生气并且歌唱
那么乏力, 爱也吹不动的叶子

蒙超恒
2020 年 7 月 10 日
于华南理工大学