中国开放指令生态(RISC-V)联盟
China RISC-V Alliance

开放指令集与开源芯片发展报告

中国开放指令生态（RISC-V）联盟

2019 年 2 月

版本修改历史

发布时间	版本号	内容修改
2019.1.11	v1p0	该版本为基础版本，主要内容如下： 1）介绍了开源芯片的兴起，以及 RISC-V 和 MIPS 等开放生态现状； 2）分析了芯片设计流程、敏捷芯片开发等开源芯片的发展现状； 3）总结了国内外学术界、工业界的开源芯片的发展动态； 4）展望了开源芯片面临的机遇和挑战。
2019.2.22	v1p1	1）增加 5.8 小节，Designless 设计模式； 2）增加部分引用； 3）修正部分文字错误。

目　录

联盟介绍

中国开放指令生态（RISC-V）联盟于 2018 年 11 月 8 日乌镇世界互联网大会正式成立，旨在以开放指令集 RISC-V 为抓手，联合各界推动开源芯片生态的建立与发展。联盟现状（截至 2019 年 1 月，详情请访问 http://crva.io）：

一、指导单位

中央网信办信息化发展局

工信部信息化和软件服务业司

中科院科技促进发展局

二、咨询委员会专家（按拼音序，下同）

方之熙（RISC-V 基金会中国顾问委员会主席）

卢　山（中国电子信息产业发展研究院院长）

孙凝晖（中国科学院计算技术研究所所长）

涂　强（长虹北美研发中心总经理）

严晓浪（浙江大学教授）

叶甜春（中国科学院微电子所所长）

三、依托单位

理事长（单位）：　倪光南院士（中国科学院计算技术研究所）

常务副理事长单位：中国电子信息产业发展研究院

四、副理事长单位

北京百度网讯科技有限公司

北京大学

北京紫光展锐科技有限公司

杭州中天微系统有限公司

华为技术有限公司

清华大学

四川长虹电器股份有限公司

腾讯科技股份有限公司

中国科学院微电子研究所

五、常务理事单位

鹏城实验室

睿思芯科（深圳）技术有限公司

上海交通大学

西安中科创星科技孵化器有限公司

中科创达软件股份有限公司

中国科学院上海微系统与信息技术研究所

致象尔微电子科技(上海)有限公司

六、单位会员

华米（北京）信息科技有限公司

江苏金羿智芯科技有限公司

浪潮电子信息产业股份有限公司

宁波中国科学院信息技术应用研究院

澎峰（北京）科技有限公司

青岛本原微电子有限公司

苏州国芯科技有限公司

天博电子信息科技有限公司

芯来科技（武汉）有限公司

中国科学技术大学

七、个人会员

陈铁军,宫晓利,郝沁汾,李诚,宋威,孙浩,魏继增,许冠斌,周平强

八、秘书处

设于中国科学院计算技术研究所

秘书长：包云岗

成　员：张科、唐丹、常轶松、王卅、解壁伟、赵然

九、联系方式

地址：北京市海淀区中关村科学院南路 6 号，邮编：100190

中国科学院计算技术研究所

电话：010-62601013/62601015　　邮箱：info@crva.io

1 前言

1.1 背景

芯片是信息技术的引擎，推动着人类社会的数字化、信息化与智能化。随着摩尔定律濒临终结，维持芯片技术创新面临挑战。开源芯片设计将是应对挑战的新思路。

如今芯片设计动辄需要上亿研发费用、投入上百人年，只有少数企业才能承担。反观互联网领域通过开源软件降低开发门槛，创造了繁荣的互联网产业。如果开源芯片设计能将芯片设计门槛降低几个数量级——3-5 人的小团队在 3-4 个月内，只需几万元便能研制出一款有市场竞争力的芯片，必将吸引大量人员投入芯片产业，重塑繁荣。

加州大学伯克利分校开发的开放指令集 RISC-V 朝着这个目标迈出了第一步，它希望像开源软件生态中的 Linux 那样，成为计算机芯片与系统创新的基石。但是只有 RISC-V 又远远不够，还需要开发基于 RISC-V 的开源工具链、开源 IP、开源 SoC 等才能形成完整的开源芯片生态，这需要更多支持开源芯片的力量参与和贡献。

历经九个月的研讨与筹备，在网信办、工信部、中科院等多个国家部委支持和指导下，中国开放指令生态（RISC-V）联盟于 2018 年 11 月 8 日浙江乌镇举行的第五届互联网大会上正式宣布成立。中国开放指令生态（RISC-V）联盟[1]旨在以 RISC-V 指令集为抓手，联合学术及产业界推动开源开放指令芯片及生态的发展，积极推动建立为全世界共享的开源芯片生态。

尽管开放指令集与开源芯片还处于初期发展阶段，但已经得到各界的越来越多的关注并付诸行动。除了 RISC-V 生态的快速成长，其他指令集也加入到开放开源队伍中，例如 Wave Computing 于 2018 年 12 月 17 日宣布开放其 MIPS 指令集架构。

为了更好地厘清当前开放指令集与开源芯片的发展态势，梳理开源芯片生态与芯片敏捷开发现状与未来面临的挑战与机遇，中国开放指令生态（RISC-

[1] 简称“RISC-V 中国联盟”，英文简称 CRVA，官方网址：http://crva.io

V）联盟成立调研工作组，结合联盟成员的最新实践，编制了《开放指令集与开源芯片发展报告》。

1.2 内容概述

本报告主要包括九方面内容：

- 第 1 章介绍开放指令集与开源芯片的背景以及报告基本内容概述；
- 第 2 章综述传统芯片设计的现状与开源芯片总体趋势；
- 第 3 章集中介绍 RISC-V 开放指令集生态现状；
- 第 4 章介绍 MIPS 开放生态现状，并与 RISC-V 进行了对比；
- 第 5 章分析开源芯片发展现状以及芯片敏捷开发案例；
- 第 6 章介绍业界动态，包括代表性企业与产品；
- 第 7 章列举几个主要国际在开源芯片领域的战略计划与部署；
- 第 8 章介绍建立开源芯片生态的挑战、机遇以及发展方向；
- 第 9 章为报告总结。

希望通过本报告能为社会各界更了解开放指令集与开源芯片有所帮助。在此，更希望呼吁更多的中国企业、工程师与学者投身开源芯片设计行动，携手推动开放指令芯片发展，共创开源芯片生态黄金时代！

2 开放指令集与开源芯片的兴起

2.1 芯片设计高门槛现状

芯片领域的创新门槛之高、投入之大业内公认。设计与制造一款芯片涉及多个环节，包括 EDA 开发环境搭建、外围 IP 模块选型、芯片前端逻辑设计、后端物理设计、流片与封装测试等，每个环节都需要巨额的资金与大量的人力投入。

以 28nm 工艺研制一款 SoC 芯片为例，比较完整的 EDA 工具版权费便超过 500 万元，购买内存控制器、PCIe 控制器等外围 IP 费用往往高达 500～1000 万元，流片费用由芯片面积大小而定，但往往也会达到 1000 万，封装相对便宜，大约需要 50 万左右。简单估算，研制这款芯片所需要的资金投入便已经超过 2000 万元。另一方面，芯片的研发往往需要数十位工程师，花上一年的时间来设计与验证，仅工资开销就需要上千万元。但是，芯片设计与验证时哪怕出现一个很小的错误，都有可能导致芯片最终无法工作。不光前期的投入打水漂，还不得不再花上千万元重新流片[2]。

芯片领域的高门槛客观上严重阻碍了创新。在互联网领域，得益于开源软件，拥有创新想法的互联网初创公司，往往只需几百万元甚至几十万元，便可以在几个月时间内推出原型产品进行迭代优化。相比而言，芯片领域的初创公司却需要数千万的资金，而且迭代优化的时间周期很长，所以很难想象风险投资人愿意把资金投入到这类初创公司。

事实上，半导体、大规模集成电路发展的黄金期是上世纪六七十年代，当时芯片规模都很小、成本较低，却又具有很高的收益，吸引了美国、日本、中国台湾等大量优秀的人才投入到半导体领域创办公司，也吸引了大量资本投入到这个领域，但中国大陆错过了这个黄金时代。经过半个世纪的发展，少数发达国家和地区通过市场机制自然地形成了技术积累与产业优势，同时构建了极高的创新门槛，不仅让后来者追赶无望，也让芯片成为他们“卡别人脖子”的利器。

---

[2] 引自美国 DARPA 的 ERI (Electronics Resurgence Initiative，电子复兴计划) 白皮书

如今中国的半导体产业要想改变现状，已经很难完全依靠初创企业与风险资本来追赶了，必须通过必要的产业政策来引导。

2.2 传统芯片设计模式

传统的芯片设计模式是芯片开发者提出芯片设计目标，将需求分解为各个功能模块。通用性比较强的功能模块，比如处理器核、内存控制器、PCIe 控制器、网络控制器、存储控制器、视频编解码器等，采取购买商用 IP 的形式获得。对于领域专用功能模块，可能没有商用 IP 可以采购或者出于技术保密考虑，采用自己研发的形式完成。在主要功能模块就绪之后，将所有功能模块采用总线形式进行互联。随后进入正常的芯片设计流程，即进行芯片的验证及后端设计开发。

传统芯片设计模式中包含大量的第三方商用 IP 并采用商用 EDA 工具开发，形成极高的开发门槛，极大限制了普通开发者的参与，不利于芯片开发的普及和创新。

2.3 开源软件的启示

中国的互联网公司在国际上是具有竞争力的。事实上，全世界十大互联网公司中国占了 4 家，而中国的互联网共享经济、移动支付甚至处于引领世界的地位。

2016 年美国移动支付金额为 1120 亿美元，而中国则达到了 5.5 万亿美元，是美国的 50 倍。2017 年 11 月 11 日，阿里巴巴“双十一”成交额达到惊人的 1682 亿（约为 260 亿美元）。相比之下，美国黑色星期五（2017 年 11 月 24 日）的网络销售成交额仅为 50.3 亿美元。根据阿里提供的数据，“双十一”零时 5 分 22 秒，支付宝的支付峰值达到每秒 25.6 万笔，其数据库处理峰值更是达到每秒 4200 万次操作，均刷新了全球纪录。相比之下，全球最大的清算组织 VISA 的交易峰值仅为 1.4 万笔 / 秒，支付宝是 VISA 的 20 倍。

中国互联网公司的成功离不开背后强大的技术支持。值得一提的是，中国的互联网公司普遍使用开源软件来构建其业务系统。开源软件，就是一种源代

码可以自由获取并在遵循开源协议的规定下，进行自行修改而无需付费的计算机软件。

对于互联网企业而言，开源软件为它们节省大量的时间与成本。据统计，常用的开源软件已经构成了一个价值超过 150 亿美元的开源软件生态，一些开源软件如 LAMP（Linux + Apache + MySQL + PHP）组合或 MEAN（MongoDB + Express + AngularJS + Node.js）组合，已经成为互联网公司（尤其是初创公司）构建其业务的标配，为企业节省大量的时间与成本，让其可以专注于业务创新上。

阿里云前总裁章文嵩博士曾在一次演讲中提到，阿里集团因为使用开源软件而节省了大量成本。有了开源软件，当人们想创办摩拜这样的新公司，不再需要从零开始造“轮子”，而是可以在庞大的开源软件库中寻找到合适的模块并根据需要进行合适的修改。比如利用 LAMP 组合，3～5 位开发人员就在几个月时间里快速搭建出一个业务原型。因此，开源软件很重要的意义在于大大降低了互联网创新的门槛。

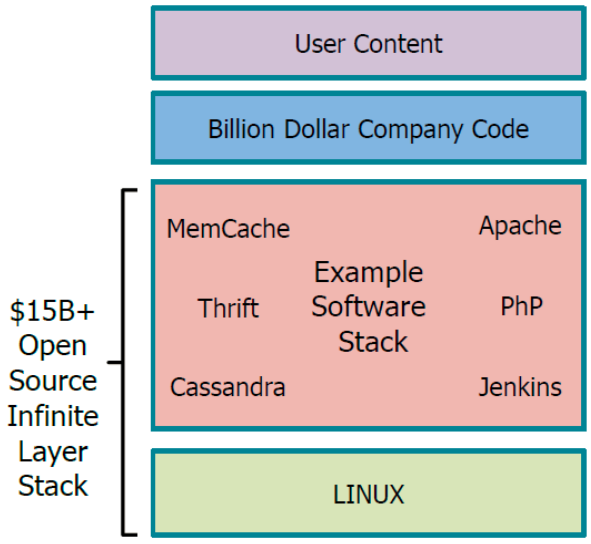


图 1. 价值上百亿美元的开源软件栈[3]

另一方面，开源软件也可以让中国的互联网产业，在软件技术方面不再面临“卡脖子”问题，从而让中国的互联网企业能与硅谷几乎在同一起跑线上竞

[3] 图片来源：Andreas Olofsson, Intelligent Design of Electronic Assets (IDEA),2017.

争，甚至在共享经济、移动支付等领域更具竞争力。当银行等传统企业还严重依赖于 IOE（IBM 的小型机、Oracle 的数据库、EMC 的存储）时，中国的互联网企业却已经用行动证明了“去 IOE”的可行性，这一切都得益于中国互联网企业积极拥抱开源软件的实践。

2.4 开放指令集与开源芯片趋势

和开源软件对于中国互联网产业的作用类似，开源芯片生态如果能形成气候，则会大幅降低芯片领域创新门槛，形成芯片领域的“大众创新”格局。这将会对中国乃至全世界半导体产业产生深远和积极的影响。

构建开源芯片生态是伟大的理想，加州大学伯克利分校开发的开放指令集 RISC-V 朝这个目标迈出了第一步。基于 RISC-V 开放指令集的处理器有可能像 Linux 那样成为开源芯片生态的基石。但是，只有 RISC-V 远远不够，还需要开源的 EDA 工具链、IP 模块、工艺库等协同合作，才能真正实现开源芯片生态。

这正如今天的开源软件生态除了 Linux 以外，还包括大量构建于 Linux 之上的其他开源工具与开源软件，例如 GCC、LLVM、MySQL、Apache 等。RISC-V 还只是星星之火，但却已展露出燎原之潜力。从当前发展势头来看，RISC-V 很有可能像 Linux 那样成为主宰世界的开放指令集标准。

2.5 开源芯片的意义与可行性

经过多年的发展，开源芯片已经具备良好的基础，已经有了比较丰富的开源芯片设计资源。例如，研制一款 180nm 工艺的芯片，可以使用开源的 Magic（包含 Xcircuit、IRSIM、NetGen、Qrouter 和 Qflow）EDA 工具链，可以使用兼容 WISHBONE 总线协议的开源 IP 模块，并有多种 180nm 开源工艺库供选择，流片费用也并不高。综合起来，研制一款 180nm 工艺的芯片可能只需要几千美元便可实现，门槛已经大为降低。

然而，对于中高端的芯片，还缺乏完整的开源芯片设计 EDA 工具链与工艺库资源。畅想一下，如果全世界也拥有了价值上百亿美元的芯片设计所需的开源 EDA 工具链、IP 模块、工艺库等，使得中高端芯片研制成本降低两个数量

级，从数千万元降至数十万元级别，那么芯片领域的创新将像今天的互联网那样层出不穷，这可能为解决中国半导体产业的卡脖子问题提供了一条新思路。

历史上降低芯片设计门槛的思路曾取得巨大的成功。1980 年代初，美国的半导体产业一度落后于日本。更为尴尬的是全美上千所大学中只有不到 100 位教授和学生从事半导体相关的研究，人才储备严重不足。

为了应对这场半导体之争，美国各界都积极采取行动。1981 年美国国防部高级研究计划局（DARPA）启动 MOSIS 项目，资助成立一个服务大学和科研机构的专业流片服务机构，隶属南加州大学信息学院。MOSIS 提出多项目晶圆 MPW (Multi Project Wafer)模式，实现将多个使用相同工艺的集成电路设计放在同一晶圆片上流片，大幅降低了芯片设计与制造成本。近 40 年来 MOSIS 为大学和研究机构流了 60000 多款芯片，培养了数万名学生，缓解了人才匮乏问题。更为重要的是，MOSIS 通过降低芯片开发门槛，直接催生了新的商业模式——无晶圆厂模式（Fabless）企业，如英伟达（NVIDIA）、高通（Qualcomm）、博通（Broadcom）和赛灵思（Xilinx）等，它们只需专注于芯片设计（如图 2[4]）；也启发张忠谋于 1987 年创办了台积电（TSMC），专注于芯片制造。

如今通过开源芯片降低芯片设计门槛，也有可能催生出芯片产业领域的新商业模式。因此，开源芯片虽然面临众多困难，但仍是一条值得探索的道路。

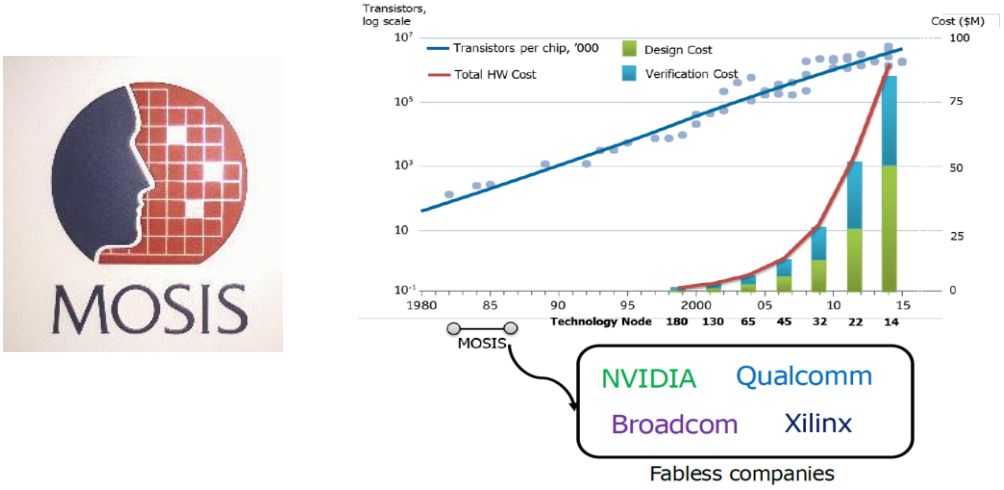


图 2. 1980 年代初 DARPA 资助 MOSIS 项目，降低芯片设计门槛，催生无晶圆厂模式（Fabless）企业的诞生

[4] 图片来源：Andreas Olofsson, Intelligent Design of Electronic Assets (IDEA),2017.

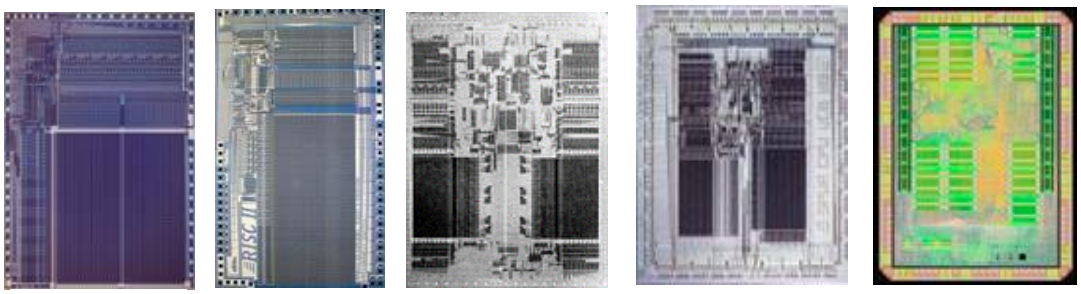
3 RISC-V 开放指令集生态现状

3.1 RISC-V 起源

2010 年，加州大学伯克利分校的 David Patterson 教授与 Krste Asanovic 教授研究团队正在准备启动一个新项目，需要选择一种处理器指令集。他们分析了 ARM、MIPS、SPARC、X86 等多个指令集，发现它们不仅设计越来越复杂，而且还存在知识产权问题。于是伯克利的研究团队临时组建一个四人小组，开展一个 3 个月的暑期小项目——从零开始设计一套全新的指令集！这个小项目的目标是新指令集能满足从微控制器到超级计算机等各种尺寸的处理器，能支持从 FPGA 到 ASIC 到未来器件等各种实现，能高效地实现各种微结构，能支持大量的定制与加速功能，能和现有软件栈与编程语言很好的适配。还有最重要的一点就是要稳定——不会改变，不会消失。

2011 年 5 月，第一版指令集正式发布。该指令集设计非常简单，采用了基础指令集与扩展指令集的方式。基础指令集只包含了不到 50 条指令，但已经可以用于实现一个具备定点运算和特权模式等基本功能的处理器。扩展指令集提供了一些常用的原子操作指令、浮点运算指令等，用户也可以需要自身需求进行自定义。这样，这套指令集既保留了“简单”这个大优点，又赋予了用户足够的灵活性。伯克利的研究团队在发布时还做了两个重大的决定：

一是将新的指令集命名为 RISC-V（读作 RISC-Five），表示为第五代 RISC（精简指令集计算机）。图 2 展示了此前的四代 RISC 处理器原型芯片。每一代 RISC 处理器都由 David Patterson 教授领导与参与。也正是他与学生



RISC-I 1981　RISC-II 1983　RISC-III (SOAR) 1984　RISC-IV (SPUR) 1988　RISC-V 2013

图 2. 五代 RISC 处理器

David Ditzel 在 1980 年发表的那篇经典论文“The case for the reduced instruction set computer”中创造了 RISC 一词。

二是将 RISC-V 指令集彻底开放，使用 BSD License 开源协议设计了开源处理器核 Rocket Core。伯克利研究团队认为，指令集 ISA 作为软硬件接口的一种说明和描述规范，不应该像 ARM、PowerPC、X86 等指令集那样需要付费授权才能使用，而应该开放（Open）和免费（Free）。他们选择的 BSD 开源协议给予使用者很大自由，允许使用者修改和重新发布开源代码，也允许基于开源代码开发商业软件发布和销售。因此 BSD 开源协议对商业集成很友好，很多的企业在选用开源产品时都会首选 BSD 开源协议。

于是，一套全新的开放指令集 RISC-V 诞生了——全世界任何公司、大学、研究机构与个人都可以开发兼容 RISC-V 指令集的处理器，都可以融入到基于 RISC-V 构建的软硬件生态系统，而不需要为指令集付一分钱。伯克利研究团队对 RISC-V 寄予厚望，希望它能被应用到各种场合，从微控制器到超级计算机；也希望它能像 Linux 通过开源成为全世界操作系统的事实标准之一，最终成为全世界处理器指令集的事实标准，为下一个 50 年计算机系统设计与创新做出奠基性贡献。

3.2 RISC-V 指令集特点

RISC-V 开源的特点保证了它的稳定性，因为它只属于一个开放的、非盈利性质的基金会。而商用指令集的盛衰往往与该商业公司的发展息息相关，历史上由于商业公司倒闭或者被收购而导致其商用指令集消失的例子比比皆是。

指令集设计者对以往指令集取其精华、去其糟粕后才得到了 RISC-V。例如，对于分支跳转指令，RISC-V 没有采用 MIPS 和 SPARC 等指令集中被广为诟病的分支延迟槽。零寄存器的存在，使得 RISC-V 可以省略很多在没有零寄存器的 ARM 和 x86 中所必需的指令，简化了指令集设计。

RISC-V 最为重要的一个特点是模块化。传统的增量指令集架构设计要求保持向后的二进制兼容，因此新处理器必须负重前行，在实现所有历史设计的基础上实现新的设计，这使得指令集架构的复杂度随时间持续增长。RISC-V 的做法是将指令集划分为几个标准的子集，称为扩展，并保持一些基础的扩展（例

如 RV32I）永远不变。这一约定给编译器和操作系统相关的开发人员提供了稳定的目标。并且这些扩展是可选的，处理器的设计者可以根据需求选择实现不同的扩展，这对于嵌入式应用至关重要。

RISC-V 主要的指令扩展如下：

- I 扩展：整数扩展（RV32I）为 RISC-V 的基础整数指令集，所有实现都必须支持。RV32I 极度精简，仅有 38 条指令，但是功能齐全，执行通用计算所必须的整数计算、访存、分支以及系统调用等指令一应俱全。处理器仅需支持 RV32I，便可以运行完整 RISC-V 软件栈。
- M 扩展：乘法扩展（RV32M）为 RISC-V 整数乘除法扩展指令集，M 扩展支持整数的有符号以及无符号乘除法运算。
- F 扩展/D 扩展：单精度浮点扩展（RV32F）和双精度浮点扩展（RV32D）为 RISC-V 的浮点指令集。共用一组独立于整数寄存器的浮点寄存器，拥有常规的访存和运算指令，也有一些包括乘加指令在内的融合运算指令，使得运算过程更精简而准确。另外为了有助于数学库的编写，还包括了有助于符号操作的符号注入指令和测试操作数属性的分类指令。F 和 D 扩展没有包括浮点分支指令，取而代之的是浮点比较指令，可以根据浮点数的比较结果设置寄存器的值，并用于条件分支。
- A 扩展：原子扩展（RV32A）为 RISC-V 的原子操作指令集，为同步操作提供了必要的支持。RV32A 扩展为不同的使用场景提供了两种对应的原子操作。其中加载保留（lr）指令和条件存储（sc）指令保证了原子的比较-交换（compare-and-swap）的实现；AMO 指令在多处理器系统中的可扩展性比 lr 和 sc 更好，可以实现 I/O 通信中的总线原子读写，从而简化设备驱动，提高 I/O 性能。
- G 扩展：通用扩展（RV32G）为 RISC-V 基础整数指令集 RV32I 加上标准扩展（M、F、D、A），统称 RV32G。
- C 扩展：压缩扩展（RV32C）为 RISC-V 的压缩指令集。包括了与标准 32 位 RISC-V 一一对应的短指令，它们只对汇编器和链接器可见，因此编译器编写者和汇编语言程序员可以忽略它们。以往的 ISA 设计在重

新设计短指令集时，会为处理器和编译器的设计增加负担，而 RISC-V 通过上述设计避免了这一缺陷。

- V 扩展：向量扩展（RV32V）为 RISC-V 向量指令集，与其它指令集中的单指令多数据流（SIMD）指令不同的是，RV32V 将内部向量寄存器的宽度与指令集解耦，解决了 SIMD 指令集每一代升级宽度时，带来的上层软件适配问题。向量指令集支持向量计算、向量 load/store、向量条件运算等操作。

RISC-V 的特权模式架构的特点：

与其它指令集一样，RISC-V 为操作系统和其它场景提供了更高的权限模式。除了通常的用户模式（U 模式）以外，RISC-V 架构还包括最底层的机器模式（M 模式）和为操作系统提供的监管者模式（S 模式）。M 模式是所有标准的 RISC-V 处理器必须实现的，拥有对硬件的完全控制权。简单的嵌入式系统只需要支持 M 模式即可，在此模式下可以处理异常和中断。M 模式和 U 模式的组合可以实现简单的基于地址寄存器比较的内存隔离，而更复杂的基于分页的虚拟内存方案需要依靠 S 模式来实现。默认情况下，所有异常都会交由 M 模式处理，但对于那些实现了 S 模式的系统，RISC-V 提供了一套异常委托机制，可以选择性地将中断和同步异常交给 S 模式处理，完全绕过 M 模式，从而避免了异常处理效率的降低。这两种权限模式各有一组控制状态寄存器（CSR），而嵌套中断需要配合软件用栈实现。

3.3 RISC-V 基金会

RISC-V 基金会（RISC-V Foundation）[5]是一个非盈利性组织，负责 RISC-V 指令集架构及其软硬件生态的标准化、保护和推广。RISC-V 基金会的会员可以参与 RISC-V 指令集规范以及相关软硬件生态的开发，并决定 RISC-V 未来的推广方向。

RISC-V 基金会每年举办全球性大会，以整合其广阔的生态系统，探讨 RISC-V 的最新项目与实现，并推动 RISC-V 指令集架构的未来发展。大会邀请顶尖科

[5] https://riscv.org/

技公司和学术机构，讨论 RISC-V 的架构、开源实现和商业实现、软件、芯片、向量和安全、应用和加速器、模拟器基础设施等等。

RISC-V 基金会成立于 2015 年 8 月，会员条约于 2016 年 12 月形成。RISC-V 基金会遵循的原则包括：

1）RISC-V 指令集及相关标准必须对所有人开放且无须授权；

2）RISC-V 指令集规范必须能够在线下载；

3）RISC-V 的兼容性测试套件必须提供源码下载。

为了保护 RISC-V 标准，只有基金会会员可以使用“RISC-V”及相关商标，并且只能用于已经通过 RISC-V 兼容性测试套件测试的产品。

基金会的董事会由来自 Bluespec、谷歌、Microsemi、英伟达、恩智浦半导体、加州大学伯克利分校和西部数据的七名代表组成。董事会成员的变更须由董事会投票决定。董事会可修改会员条约，须经过三分之二赞成票通过。董事会可决定建立委员会来处理关于 RISC-V 的具体事务，并任命下属委员会的主席。目前已经建立的委员会包括技术委员会和营销委员会。

图 3. RISC-V 基金会部分成员

RISC-V 基金会目前有超过 210 名成员，包括机构、学术和个人会员，覆盖 25 个国家，这些国家占据世界人口的 55%。基金会自 2015 年成立以来，会员数的年增长率超过 100%。当前的成员包括：中科院计算所、阿里巴巴、华为、谷歌、镁光、英伟达、高通、三星、西部数据、日立、IBM、联发科、希捷、海力士等。图 3 展示了 RISC-V 基金会的部分成员。

3.4 RISC-V 研讨会/峰会

RISC-V 研讨会（Workshop）一共举办了 9 届，举办了 1 届峰会（Summit），有来自全球 20 多个国家的共 1100 多名参会人员。往届研讨会的相关信息在表 1 中列出。

表 1.历届 RISC-V 研讨会的信息

届数	时间	地点
1	2015 年 1 月	美国加州蒙特里万豪酒店
2	2015 年 6 月	美国加州伯克利国际之家酒店
3	2016 年 1 月	美国加州红木海岸 Oracle 会议中心
4	2016 年 7 月	美国马萨诸塞州麻省理工学院
5	2016 年 11 月	美国加州 Google 山景园区
6	2017 年 5 月	中国上海交通大学
7	2017 年 11 月	美国明尼苏达州西部数据公司
8	2018 年 5 月	西班牙巴塞罗那
9	2018 年 7 月	印度金奈

RISC-V 研讨会的目的是向全社区通报全球各种 RISC-V 项目的最新活动，并就 RISC-V 项目的未来步骤达成共识，同时举办训练营提供从 RISC-V 开发团队了解 RISC-V 基础架构的机会。研讨会和训练营中，展示了多个 RISC-V 流片、FPGA 板设计和相关软件工具的演示。研讨会的特色是讲座和海报展示，介绍 RISC-V 社区当时在公开提交期间收集的活动。

其中，第二届研讨会就 RISC-V 项目的未来步骤（包括 RISC-V 基金会）达成共识。第三届和第四届研讨会与 RISC-V ISA 和 RISC-V 基金会的启动就未来

的步骤达成共识。第五届，第六届和第七届研讨会就指令集的未来达成共识。第六届研讨会是在北美以外举办的第一次 RISC-V 基金会研讨会，有超过 270 名注册参加者。研讨会为 RISC-V 新手和尚未接触 RISC-V ISA 的人员举行了为期一天的会议。会议包括 RISC-V 基金会的演示文稿，RISC-V ISA 的一些原始创建者以及 RISC-V 社区内供应商的产品演示。

第八届研讨会的主题演讲包括 NXP 软件工程研发副总裁 Robert Oshana，Western Digital 执行副总裁兼首席技术官 Martin Fink 以及巴塞罗那超级计算中心主任 Mateo Valero。会议第一天涵盖的主题包括基础 ISA 批准、BitManip、合规性、调试、正式规范、内存模型、操作码空间管理、权限规范、安全性、软件工具链和向量扩展。第二天和第三天举行了两天的 RISC-V 架构、商业和开源实现、软件和芯片、载体、安全、应用、加速器和仿真基础设施等的演讲。

第九届研讨会活动包括 RISC-V 架构的深入技术演示，演讲方包括领先技术公司和研究 RISC-V 生态系统中的机构。

RISC-V 的首届峰会于 2018 年 12 月 3 日至 6 日在美国硅谷圣克拉拉举行。对于 RISC-V 基金会来说是历史性的一次会议，吸引了来自全球 20 个国家的 1100 多人参会。超过全球 25 个国家的 150 家成员公司参加。峰会的展览厅共有 29 家参展商，两天内共有 53 场演出。展会上，存储产品供应商西部数据公司 CTO Martin Fink 在 RISC-V 峰会的主题演讲中宣布，将 WD 自己开发的 RISC-V 内核 SweRV 开源。SweRV 是一款采用 RV32IMC 指令集的 32 位、9 级流水线 RISC-V 内核，它采用双路超标量设计，可以同时加载和执行多条指令以缩短程序运行时间，时钟频率为 1.8Ghz，将采用 28mm CMOS 工艺制造。其测试性能可以达到 4.9 CoreMarks/Mhz，将用于 WD 内部的各种嵌入式应用，比如闪存控制器和 SSD 等。SiFive 首席架构师 Krste Asanovic 在其主题演讲中揭示了针对高性能计算的 RISC-V AI 平台，包括带向量扩展的 RISC-V 内核、HBM2 高带宽存储接口，以及 56Gb/s SerDes 接口。另外，Microchip 公司旗下的 Microsemi 在峰会上演示了一种基于 RISC-V 的 PolarFire SoC FPGA 架构，结合其低功耗 PolarFire FPGA 系列与 SiFive 的 1.5GHz U54-MC RISC-V 内核（性能与 ARM Cortex-A35 相当），将实时、确定性的非对称多处理(AMP)能力带到

Linux 平台。这一平台具有灵活的 2 MB L2 存储器子系统、单-双错误校正(SEC-DED)等可靠性和安全性功能，以及 SmartDebug 逻辑分析仪等调试功能。

3.5 RISC-V 软硬件生态

芯片技术是信息技术产业链中的重要基石，也是我国大幅落后于国际水平的技术领域之一。发展自主可控的芯片技术，并打破 Intel、高通、ARM 等芯片巨头的技术垄断，已成为当下亟待解决的问题。

处理器指令集是软硬件的接口，是构建芯片生态和发展芯片技术的核心部分，其重要性不言而喻。2011 年，加州大学伯克利分校发布了开放指令集 RISC-V，并很快建立起一个开源软硬件生态系统。由于 RISC-V 的指令标准清晰，并且开源免授权等特性，在芯片领域受到广泛关注。截止 2019 年 1 月，已有包括 Google、NVidia 等在内的 200 多个公司和高校在资助和参与 RISC-V 项目。其中，部分企业已经开始将 RISC-V 集成到产品中。例如全球第一大硬盘厂商西部数据（Western Digital）最近宣布将把每年各类存储产品中嵌入的 10 亿个处理器核换成 RISC-V；Google 利用 RISC-V 来实现主板控制模块；NVidia 也将在 GPU 上引入 RISC-V 等。此外，国内阿里巴巴、华为、联想等公司都在逐步研究各自的 RISC-V 实现；上海市将 RISC-V 列为重点扶持项目；印度政府也正在大力资助基于 RISC-V 的处理器项目，使 RISC-V 成为了印度的事实国家指令集。这表明 RISC-V 已经逐渐成为芯片设计领域的主流指令集之一，且广受各大厂商青睐。下面从 RISC-V 指令集、生态系统和已有 RISC-V 实现等几方面介绍 RISC-V 的发展现状。

RISC-V 指令集清晰且开源，与现有商用指令集相比，RISC-V 更加精简。此外，RISC-V 支持在标准指令集之外，自定义扩展指令集，兼顾了灵活性。目前，基于 RISC-V 的指令扩展已经发展得颇为丰富，如 SiFive、LowRISC、ORCA 等已经发展出了面向矢量化、虚拟化、硬件安全和特定领域加速等的多种扩展指令集，极大提升了 RISC-V 应对真实场景下应用需求的能力。

RISC-V 的生态系统日趋完善。良好的生态系统对发展芯片技术，以及形成良性可持续的芯片产业循环是至关重要的。与其它开源指令集(如 OpenSPARC 和 OpenPOWER)相比，RISC-V 在社区支持方面更完善，支持包括 Linux、SeL4、BSD 等通用操作系统，支持 FreeRTOS 和 RT-thread 等实时操作系统，支持 GCC、LLVM

等通用编译和调试工具链，支持 C/C++、Java、Python、OpenCL 和 Go 等主流编程语言。近年来国内外兴起的 RISC-V 的研究热潮也将持续完善其生态系统，借助该生态系统，可以使得优秀的 RISC-V 实现迅速落地并形成产业影响力。

目前已经出现了面向各种目标设计的开源 RISC-V 实现（如表 2 所示）。其中大部分 RISC-V 实现，如 Rocket Core 和 LowRISC 等，面向嵌入式和低功耗的场景应用，采用单发射顺序执行技术，获得良好的性能功耗比。而面向工业级处理器的 BOOM 采用超标量乱序执行技术，性能提升巨大。

鉴于 RISC-V 指令集开源、规范清晰、扩展灵活并已经具备完整的生态系统支持，已成为当下芯片技术研究的热点。虽然面向嵌入式场景的 RISC-V 实现已经百花齐放，但目前还没有一个开源且实用的高性能 RISC-V 实现来应对复杂场景下的应用需求。为填补此空白，占领未来芯片技术发展的先机，研究可靠实用的高性能 RISC-V 核心已经迫在眉睫。

表 2.部分开源的 RISC-V 实现

项目		**目标定位**	**实现语言**	**核心架构**
Rocket Core		ARM 顺序核	Chisel	RV32G/RV64G(五级流水/顺序执行)
LowRISC				
BOOM		工业级处理器	Chisel	RV64G (超标量/乱序执行)
ORCA		低资源占用	VHDL	RV32IM (5 级流水/顺序执行)
RISCY		MCU	System Verilog	RV32ICM (4 级流水/顺序执行)
Roa Logic		嵌入式		RV32G/RV64G (4 级流水/顺序执行)
SCR1		MCU		RV32I (4 级流水/顺序执行)
Mriscv		MCU	Verilog	RV32I (3 级流水/顺序执行)
VexRiscv		FPGA	SpiralHDL	RV32IM (5 级流水/顺序执行)
Hummingbird		嵌入式和教学	Verilog	RV32IMAC (2 级流水/顺序执行)
Shakti class	I 系列	通用处理	System Verilog	RV32IM/RV64I(8 级流水/乱序执行)
	C 系列	ARM 顺序核		RV64IMAFD (6 级流水/顺序执行)
	E 系列	MCU		RV32IM (3 级流水/顺序执行)
Riscy		FPGA	Verilog	RV32I (5 级流水/顺序执行)

3.6 RISC-V 在中国的发展

3.6.1 中国企业

RISC-V 不仅在国际领域广受关注，在中国也呈现风起云涌之势。截至 2018 年年底，可查询到的与 RISC-V 芯片、硬件、软件、投资、知识产权及生态相关的中国公司（含外资公司中国分公司）数量已接近一百家，在表 3 中列出（按首字母排序）：

表 3 与 RISC-V 相关的中国企业（名单陆续更新中）

序号	单位名称		序号	单位名称
1	杭州中天微系统有限公司		2	杭州秘猿科技有限公司
3	北京百度网讯科技有限公司		4	合肥格易集成电路有限公司
5	北京华大九天软件有限公司		6	华大半导体有限公司
7	北京集创北方科技股份有限公司		8	华米（北京）信息科技有限公司
9	北京君正集成电路股份有限公司		10	华为技术有限公司
11	北京联想核芯科技有限公司		12	华夏芯通用处理器技术有限公司
13	北京猎户星空科技有限公司		14	华芯投资管理有限责任公司
15	北京探诚科技有限公司		16	江苏金羿智芯科技有限公司
17	北京万宏兴业科技有限公司		18	晶晨半导体（上海）股份有限公司
19	北京先风广集管理咨询有限公司		20	晶心科技（武汉）有限公司
21	北京芯启科技有限公司		22	景略半导体（上海）有限公司
23	北京翼辉信息技术有限公司		24	钜泉光电科技（上海）股份有限公司
25	北京御芯微科技有限公司		26	浪潮电子信息产业股份有限公司
27	北京中科汉天下电子技术有限公司		28	乐鑫信息科技（上海）有限公司
29	比特大陆科技控股公司		30	摩尔精英网络科技南京有限公司
31	成都锐成芯微科技股份有限公司		32	澎峰（北京）科技有限公司
33	诚迈科技（南京）股份有限公司		34	青岛本原微电子有限公司
35	地平线（上海）人工智能技术有限公司		36	睿思芯科（深圳）技术有限公司
37	格芯（中国）有限公司		38	厦门半导体投资集团有限公司

39	广东高云半导体科技股份有限公司	40	厦门积微信息技术有限公司
41	广东汉为集成技术有限公司	42	上海埃瓦电子科技有限公司
43	国家电网有限公司	44	上海安路信息科技有限公司
45	海高汽车技术有限公司	46	上海聪链信息科技有限公司
47	杭州华澜微电子股份有限公司	48	上海复旦微电子集团股份有限公司
49	杭州秘猿科技有限公司	50	上海赋华网络科技有限公司
51	上海富瀚微电子股份有限公司	52	北京声智科技有限公司
53	上海富芮坤微电子有限公司	54	苏州硅岛信息科技有限公司
55	上海格易电子有限公司	56	苏州国芯科技有限公司
57	上海恒锐知识产权服务有限公司	58	台积电（中国）有限公司
59	上海慧存微电子科技有限公司	60	深圳市腾讯计算机系统有限公司
61	上海集成电路产业投资基金管理有限公司	62	天博电子信息科技有限公司
63	上海嘉韬实业有限公司	64	VMware（中国）有限公司
65	上海谨嵘电子科技有限公司	66	武汉晟联智融微电子科技有限公司
67	上海睿赛德电子科技有限公司	68	西安恩狄集成电路有限公司
69	上海晟矽微电子股份有限公司	70	西安优矽智芯电子科技有限公司
71	上海时芯电子科技有限公司	72	芯来科技（武汉）有限公司
73	上海小蚁科技有限公司	74	芯翼信息科技（上海）有限公司
75	上海芯铄投资管理有限公司	76	芯原微电子（上海）有限公司
77	上海兴橙投资管理有限公司	78	亚创（上海）工程技术有限公司
79	上海弋盛投资管理有限公司	80	Imagination（上海）有限公司
81	上海音航信息科技有限公司	82	四川长虹电器股份有限公司
83	上海岳芯电子科技有限公司	84	致象尔微电子科技(上海)有限公司
85	上海云从企业发展有限公司	86	中科创达软件股份有限公司
87	上海云间半导体科技股份有限公司	88	西安中科创星科技孵化器有限公司
89	上海兆芯集成电路有限公司	90	中芯国际集成电路制造有限公司
91	上海智百咖信息科技有限公司	92	中兴微电子技术有限公司
93	深圳市国微电子有限公司	94	中颖电子股份有限公司

95	深圳云天励飞技术有限公司	96	重庆华犇创业投资管理有限公司
97	紫光展锐科技有限公司		

3.6.2 中国的研究

在 2015 年之前，大多数微结构和芯片相关的研究受限于指令集的授权问题而难以开展。随着 RISC-V 开源开放理念的流行，越来越多的科研项目受益于 RISC-V 而得以开展。其中有代表性的研究成果介绍如下：

- **RISC-V 系统架构创新方向**

中国科学院计算技术研究所先进计算团队开展的标签化 RISC-V（Labeled RISC-V）属于国内 RISC-V 的首批研究之一，该项目基于 RISC-V 来对软件定义体系结构进行探索。该团队也基于 RISC-V 处理器开展了缓存预取的研究。

- **RISC-V 加速器和领域专用处理器领域:**

中国科学院计算技术研究所泛在计算团队开展了基于 RISC-V 核心的轻量级神经网络处理器的研究，探索了 RISC-V 核心在物联网设备中的应用，而上海交通大学的北斗导航与位置服务重点实验室则开展了基于 RISC-V 指令集的基带处理器扩展研究项目，甚至一些处理器爱好者也开展了基于 RISC-V 核心的 CNN 加速器项目。

- **RISC-V 安全领域**

清华大学计算机软件研究所团队开展了基于 RISC-V 的操作系统安全研究，中国科学院信息工程研究所信息安全国家重点实验室团队则开展了基于 RISC-V 的微结构安全研究。

据不完全统计，目前围绕 RISC-V 开展科学研究或领域生态调研的科研机构包括但不限于（按拼音排序）：北京大学、南京大学、南开大学、宁波中国科学院信息技术应用研究院、鹏城实验室、清华大学、上海交通大学、上海科技大学、天津大学、浙江大学、中国电子信息产业发展研究院、中国科学技术大学、中国科学院计算技术研究所、中国科学院上海微系统所、中国科学院微电子所和中国科学院信息工程研究所等。预计未来国内基于 RISC-V 的研究将越来越繁荣。

3.6.3 中国的教育

RISC-V 指令集和体系结构源于 UC Berkeley 的 CS152 课程。作为计算机体系结构界殿堂级的存在，UC Berkeley 在体系结构教学和教育方面的经验值得国内高校认真学习和借鉴。在国内，RISC-V 正被越来越多的高校教师和学生接触和学习。南京大学袁春风老师在开设的《计算机组成原理》课程中尝试使用 RISC-V 作为课程设计的基础。清华大学的向勇和陈渝老师在操作系统实验课上也把 RISC-V 作为目标系统。自 2016 年开始，北京大学的易江芳老师就在计算机体系结构实习课上介绍并尝试进行 RISC-V 的各级模拟器和 FPGA 实现。2019 年，《计算机组成与设计：软硬件接口》RISC-V 中文版即将出版，可以作为各大高校体系结构课程的教材选择，这将为 RISC-V 在中国高校中的推广起到重要的推动作用。不过，作为经典体系结构的 MIPS 在计算机相关课程体系中本就占有重要地位，并随着“龙芯”的推广其地位不断加强。在高校和教育界，RISC-V 如何能够获得更多的拥趸和青睐，还需要社会各界更多力量的加入和推动。

3.6.4 中国联盟与组织

随着 RISC-V 的知名度与关注度不断增加，2018 年在中国成立了三家与 RISC-V 相关的联盟组织。虽其侧重点略有不同，但都在为 RISC-V 在中国的宣传发展及生态建设凝聚力量。

1) 中国 RISC-V 产业联盟（简称 CRVIC 联盟）

2018 年 9 月 20 日，中国 RISC-V 产业联盟宣布成立，上海芯原控股有限公司担任联盟首任理事长单位。中国 RISC-V 产业联盟致力于集聚和整合国内 RISC-V 创新力量，助推 RISC-V 产业生态的建设，提升中国企业在 RISC-V 指令集创新、标准制定中的影响力，同时，加快 RISC-V 的市场推广和产业化应用

2) 中国开放指令生态(RISC-V)联盟（简称 CRVA 联盟，网址：http://crva.io）

2018 年 11 月 8 日，中国开放指令生态（RISC-V）联盟在第五届互联网大会上宣布成立，联盟理事长由倪光南院士担任。CRVA 联盟旨在召集从事 RISC-V 指令集、架构、芯片、软件和整机应用等产业链各环节企事业单位及相关社会团体，自愿组成一个全国性、综合性、联合性和非营利性的社团组织。此联盟将围绕 RISC-V 指令集，整合各方资源，通过产、学、研、用深度融合，力图推进 RISC-

V 生态在国内的快速发展。

3) RISC-V 基金会中国顾问委员会

2018 年 11 月 8 日，在中国乌镇举行的世界互联网大会(World Internet Conference)上,RISC-V 基金会(RISC-V Foundation)宣布成立中国顾问委员会，将就 RISC-V 基金会的教育和应用推广战略提供指导。半导体行业资深人士方之熙(Jesse Zhixi Fang)博士被任命为 RISC-V 基金会中国顾问委员会主席。RISC-V 基金会在中国的影响力不断扩大，覆盖超过 25 个组织机构与大学，在此基础上，中国顾问委员会将对 RISC-V 基金会的教育与应用推广战略提供指导意见，以进一步加速 RISC-V 生态系统在该地区的发展。

3.7 RISC-V 相关资源信息

本节按教材资料、社区动态和国际项目分别介绍 RISC-V 相关资源信息。

3.7.1 教材资料

采用 RISC-V CPU 作为国内高校软硬件系统类课程的学习研究对象，有助于高校老师和同学充分利用 RISC-V 生态环境，使用丰富的开源软硬件资源和信息，简化课程建设，有利于打通编译原理、汇编语言、操作系统、计算机组成原理、计算机系统结构等课程，让学生能够更加深入和全面的掌握计算机系统相关的知识，提供系统能力。

目前国内高校对 RISC-V 的教学和教育相对较少，在普及 RISC-V 方面落后于企业。在教科书、教材方面，胡振波撰写的《手把手教你设计 CPU——RISC-V 处理器》是一本介绍通用 CPU 设计的入门书，以相对通俗的语言系统介绍了 CPU 和 RISC-V 架构，可帮助读者理解 CPU 设计和计算机体系结构。中科院计算所勾凌睿等人翻译了《The RISC-V Reader: An Open Architecture Atlas》一书。此书讲述了 RISC-V 的设计缘由,即为什么它的设计师选择赋予它这些能力，再加上一些对其演化历史的深刻见解及与其它常见架构的比较，让教学过程变得更为生动。英文教材方面，《Computer Organization and Design RISC-V Edition: The Hardware/Software Interface》是一本经典教材。

在计算机组成原理等硬件课程方面，目前已经有一些大学，如清华大学、江南大学等，基于或部分基于 RISC-V 进行课程原理和实验内容建设。在编译原理、

操作系统等系统软件类课程中，清华大学把 RISC-V 作为可选的实验硬件平台，用于编译器后端实验和操作系统实验等。

未来应加强推广面向高校教师的基于 RISC-V CPU 的教学培训、会议、workshop、tutorial 等，与企业加强合作，提供丰富的 RISC-V 软硬件生态、非常便宜易用的 FPGA/ASIC 软硬件实验环境、详细的可参考的教学素材资源，使得 RISC-V CPU 能得到高校师生更广泛的认可与接受。详细信息如下：

- 《Computer Organization and Design RISC-V Edition: The Hardware/Software Interface》于 2017 年正式出版。由北京大学翻译的《计算机组成与设计：软硬件接口》RISC-V 中文版将在 2019 年出版。
- 图灵奖得主 David Patterson 和 Andrew Waterman 合著的《The RISC-V Reader》向嵌入式系统工程师、学生和开源指令集爱好者们详细介绍了 RISC-V 设计理念，详见：http://www.riscvbook.com/。由中科院计算所包云岗研究员的团队将《The RISC-V Reader》翻译成中文版本，下载地址：http://crva.io/documents/RISC-V-Reader-Chinese-v2p1.pdf
- 由芯来科技创始人胡振波编著的《手把手教你设计 CPU——RISC-V 处理器篇》介绍了 CPU 和 RISC-V 架构，并结合 RISC-V 开源实例进行教学。
- 维基百科上关于 RISC-V 的解释词条从历史、设计理念等角度介绍了 RISC-V，详见：https://en.wikipedia.org/wiki/RISC-V

3.7.2 社区动态

- CNRV 双周报：国内 RISC-V 爱好者利用 Github 协作方式以双周简报形式介绍 RISC-V 的最新信息，同时在微信公众号和 CNRV 网站发布。其内容覆盖 RISC-V 邮件列表、行业新闻、项目进展以及各类点评。详见 https://cnrv.io/
- 开源中国：社区开设了单独的板块总结了 Github 上部分 RISC-V 资源，详见 https://www.oschina.net/search?scope=project&q=RISC-V
- 开源 RISCV：https://www.zhihu.com/people/risc-v/activities

3.7.3 国际项目

- 美国国防部高级研究计划局（DARPA）于 2017 年 6 月启动了“电子复兴

计划”（ERI：Electronics Resurgence Initiative）。其中“一流开源硬件”（Posh Open Source Hardware，POSH）项目旨在引导学术及产业界开展开源 SoC 设计与验证系统方向的研究工作，具体包括：实现来源未知的电路组件自动化验证确保投片质量，以及初步建立开源芯片 IP 资源库。通过以上工作以期形成开源且低成本的 SoC 生态。此外，“电子设备智能设计”（Intelligent Design of Electronic Assets，IDEA）项目旨在实现数字芯片、模拟芯片、器件封装、印刷电路板多层次的复杂混合电路布局中的无人参与以及自动化系统综合生成。

- Libre Silicon Alliance 通过打造多个开源项目包括基于自由和开放的芯片制造流程，高质量的标准单元库，以及自由开源的 EDA 工具推动开源芯片设计，即使用完全自由和开放的工具使芯片设计和制造过程打破大厂商垄断以及绕过厂商的 NDA（保密协议），详见 https://libresilicon.com

4 MIPS 开放生态现状

4.1 MIPS 开放计划介绍

Wave Computing 于 2018 年 12 月 17 日宣布开放其 MIPS 指令集架构以便半导体公司、开发者以及大学能快速采用 MIPS 架构用于下一代的 SoC 芯片的设计开发。Wave Computing 是致力于开发 AI 和深度学习数据流芯片的美国硅谷公司，该公司于 2018 年 6 月收购了 MIPS 公司，开放 MIPS 指令集是该公司“All in AI”战略的重要组成部分。

该公司认为，虽然开源指令集以及开源芯片在最近几年发展迅速，但直到现在仍然缺乏达到工业标准的、受到专利保护以及经过硅验证的开源 RISC 架构。而工业标准的 MIPS ISA 代表了几十年以来的持续创新，拥有数以千计的设计实现，生产了超过 85 亿颗芯片，并且以每年 10 亿颗的速度继续增长。

MIPS 开放计划将会鼓励全球的半导体社区开发新的用于各行业市场的 MIPS 兼容芯片解决方案。MIPS 开放计划将由 Wave Computing 公司、工业界领先的 OEM 制造商、合作伙伴、大学以及杰出的技术人员组成的咨询委员会共同主持，将以社区驱动的形式来帮助指导在 MIPS 架构上的创新。MIPS 开放计划也会引入一些经过认证的验证合作伙伴用来帮助确认实现的兼容性以防止架构碎片化。MIPS 开放计划将会极大扩展包括全球数以千计的开发者、超过 100 个的学术机构的现有的 MIPS 生态环境，这些生态环境将会从第三方工具供应商、软件开发者以及大学获得创造创新的机会。通过 MIPS 开放计划，MIPS 架构的客户也将获得保证，MIPS 开发生态环境提供的开发工具、应用软件或者其它增值特性及服务将会与新的实现保持兼容。

MIPS 开放计划的细节，包括可下载的 MIPS 架构、授权细节、支持机制以及如何参与等信息，将会在 2019 年一季度公布。

4.2 MIPS 指令集发展历史及 MIPS 公司主要产品

4.2.1 MIPS 指令集发展历史

MIPS 指令集从 1985 年第一个版本发布至今发已有 30 多年历史，其中 2014

年发布的 MIPS Release 6 是最新的版本。每个版本的发布时间如表 4 所示。

表 4 MIPS 各版本发布时间

版本	发布时间
Release 1	1985 年
Release 2	2002 年
Release 3	2010 年
Release 5	2013 年
Release 6	2014 年

不仅如此，MIPS 架构也在不断演进，其演进路线如图 4 所示。

MIPS 指令集及架构在 2010 年以后发展迅速，至 2014 年的 5 年时间共发布了 3 个版本，在传统的整数浮点应用指令基础上逐步增加了多线程、DSP 模块、SIMD 模块以及虚拟化模块。这也与移动互联网应用的迅速发展时间相吻合，随着应用需求变化，MIPS 指令集以及架构也在迅速发展。

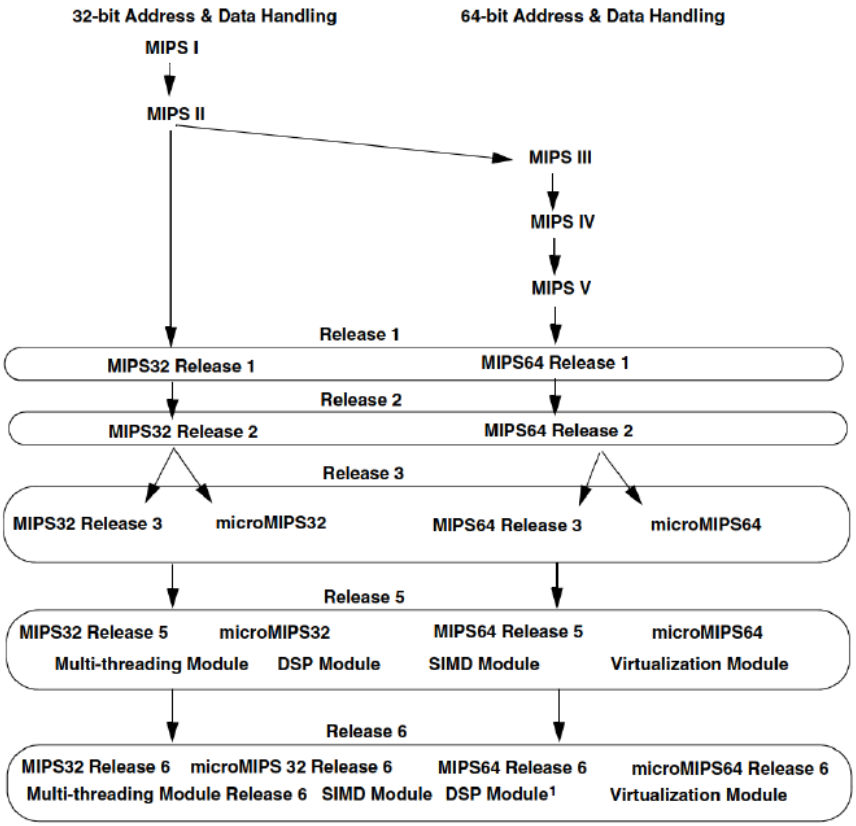


图 4.MIPS 架构演进路线

在 MIPS 指令集发展过程中，基本上严格保持新版本向后兼容的特点。比如 MIPS Release5 可以向下兼容之前的所有指令集版本。

但在最新发布的 MIPS Release 6 版本中，这种情况发生了变化。这个版本中增加了一些新的指令并对指令集进行了简化，删除了一些不常用的指令，重新排布了指令的编码，空余出了大量的指令槽用于将来的扩展。因此，可以把 MIPS Release 6 版本看成一个几乎全新的定义。

4.2.2 MIPS 公司主要产品

自 2014 年发布 MIPS R6 版本之后，MIPS 公司的 IP Core 产品全面转向 MIPS R6 版本，形成覆盖高中低不同性能和应用需求的 Warrior 产品系列，而把 R6 之前的产品全部归为 Classic 系列。Warrior 产品系列又分为 M-Class，I-Class 和 P-Class 三个级别。这三个级别的产品的特性如表 5 所示。

表 5 Warrior 三个级别的产品特性

级别	产品	产品特性
P-Class	P5600/P6600	● MIPS 64-bit R6 指令集 ● 16 级流水乱序执行 ● 4 指令取值宽度 ● 每拍同时发射 4 条整数和 2 条 SIMD 操作 ● BTB/RPS/JRC 等高级分支预测机制 ● 可选高性能 128 位双发射 SIMD 执行单元 ● 全硬件虚拟化 ● TSMC 28HPM 主频 2+GHz
I-Class	I7200/I6500/I6400	● MIPS R6 32-bit nanoMIPS 指令架构 ● 比传统 MIPS32 增加 40%的指令密度 ● 双发射 9 级流水线 ● 支持多线程技术 ● TSMC 16FF+工艺下 1.7GHz-2.1GHz
M-Class	M51xx/M62xx	● MIPS R6 32-bit microMIPS /MIPS32 指令集 ● 6 级流水线 ● 专用 DSP 和 SIMD 模块 ● TSMC 28HPM 12T SVt 工艺下 750MHz

MIPS 公司的 IP Core 产品主要被应用在人工智能、汽车电子、消费类电子、IoT 以及网络等 5 个市场。其中人工智能是 Wave Computing 公司既定的方向，

其技术方案将加速 AI 计算的数据流架构与 MIPS 嵌入式多线程 RISC 处理器核组合起来。

总体而言，MIPS 架构处理器在大部分行业应用中的市场份额处于下降趋势，即使在 MIPS 架构的传统优势领域，比如机顶盒、打印机控制器、WIFI 路由器、网络处理器等，也已经逐步被 ARM 架构的处理器所取代。

从已有信息来看，自 2014 年 MIPS Release 6 发布以来，除 MIPS 公司自己的 Warrior 产品系列之外，尚无第二个实现 MIPS Release 6 指令集的芯片。

4.3 MIPS 开放计划特点

MIPS 开放计划包含的内容包括以下 IP 和技术架构：

- 开源的 MIPS 32 和 64 位指令集，Release 6 版本
- MIPS SIMD 扩展
- MIPS DSP 扩展
- MIPS 多线程技术
- MIPS MCU
- microMIPS 架构
- MIPS 虚拟化技术

MIPS 公司对 MIPS 开放计划进行了以下澄清：

1) 此次开放计划仅仅针对 MIPS 指令集的最新版本，即 2014 年发布的 Release 6 版本。在开放计划中，并不包含早期版本（即 Release 5 或更早期）。不能把 MIPS 开放计划的授权与早期版本的授权混淆。
2) 如果一个公司开发了一个基于 MIPS 开放计划授权的实现，可以不需要开源这部分代码。
3) 合作伙伴如果加入 MIPS 开发计划并且通过认证，有机会把自己的服务放在 MIPS 开源计划中进行销售。

从以上信息可以知道，MIPS 开放计划并不是开放所有版本的 MIPS 指令集，而仅仅是最新的 MIPS Release 6 版本。从 MIPS Release 6 版本的文档可以知道，该版本并不与早期版本完全兼容，其中的修改甚至包括删除大量不常用指令，将回收的指令槽用于新增加的指令，在一定程度上几乎可以看成是一个全新的指

令集。由此可见，MIPS 开放计划中的 MIPS 指令集的初始生态环境并不乐观。这也是很少有产品采用基于 MIPS Release 6 指令集的 IP Core 的主要原因之一。

4.4 MIPS 开放计划与 RISC-V 开源计划对比

MIPS 开放计划与 RISC-V 的开源计划有相似之处，也有不同之处。相对自 2010 开始发展的 RISC-V 指令集，MIPS 开放计划的中开放的 MIPS Release 6（2014 年)指令集优势并不明显。与 RISC-V 指令集致力于全面的开源 ISA 不同，MIPS 开放计划仍然会存在一些商业上的顾虑和一些沉重的历史包袱，显然不如 RISC-V 的开源计划来得更彻底、更纯粹。

表 6 中，对 MIPS 开放计划和 RISC-V 开源计划进行了对比。

总体来看，MIPS 开放计划在商业化应用上具有一定优势，但其 License 形式相对 RISC-V 具有一定的不确定性，而 RISC-V 的开源更为彻底，采用较为宽松的 BSD License 形式。

表 6 MIPS 开放计划与 RISC-V 开源计划对比

项目	MIPS 开放计划	RISC-V 开源计划
指令集起始时间	2014 年(MIPS Release 6)	2010 年
主导方	Wave Computing 公司	RISC-V 基金会
SIMD/DSP 扩展	有	有
虚拟化技术	有	有
多线程技术	有	无
硅验证	有	有
批量商业化应用	有	无
开源实现	无	多个
License 形式	未知	BSD License

4.5 芯片商业模式对比

为了更好的理解开放指令集与开源芯片的内核，在此借用图灵奖得主 David Patterson 教授在清华大学的学术报告中展示一张关于不同处理器芯片商业模式的表格（见表 7）。

当讨论处理器芯片商业模式时，需从三个不同的设计层次表述：指令集规范、IP 核设计与 SoC 芯片设计。在每个层次都存在开源（免费）模式、IP 授权模式与完全私有模式。表 7 展示了现有主流的处理器各自的定位。

在指令集规范层次，RISC-V 和最新推出的 MIPS R6 都是开源可免费使用；ARM 和早期的 MIPS 指令集采用的是授权模式，用户需出资获取授权；Intel 对应的 X86 指令集则完全私有。

在 IP 核设计层次，伯克利早期开发的面向 RISC-V 指令集的 Rocket Core 是开源实现；而 ARM 与 SiFive 各自设计 IP 核，但用户需出资获取授权（或源码）；同样，Intel 设计的 IP 核只为自己使用。

在 SoC 芯片设计层次，目前尚无成熟的开源 SoC 实现，提供 IP 授权的 SoC 芯片设计也较少，仅有高通等少数几家企业；而 Intel、海思等企业有实力开发 SoC，但也并不提供授权服务。

虽然 RISC-V 和 MIPS 在开放指令集方面迈出了重要的一步，但是开源芯片的模式依然不是主流，其在开源的 IP 核设计和开源 SoC 上依然还有很长的路要走。

表 7. 处理器芯片商业模式对比[6]

商业模式 / 设计层次	开源/免费	IP 授权	完全私有
指令集规范	RISC-V/MIPS R6	ARM/MIPS	Intel
IP 核设计	Rocket Core	ARM/SiFive	Intel
SoC 芯片设计		Qualcomm	Intel/海思

[6] 摘自图灵奖得主 David Patterson 教授报告

5 开源芯片发展现状

5.1 芯片设计流程

芯片设计总体流程如图 5 所示，一般可分为前端设计和后端设计两部分，二者之间通过逻辑综合交换网表数据。除此之外，后端设计完成布线阶段以后生成带延迟的网表提供给前端进行后仿真。下面对芯片的前后端设计流程分别进行介绍。

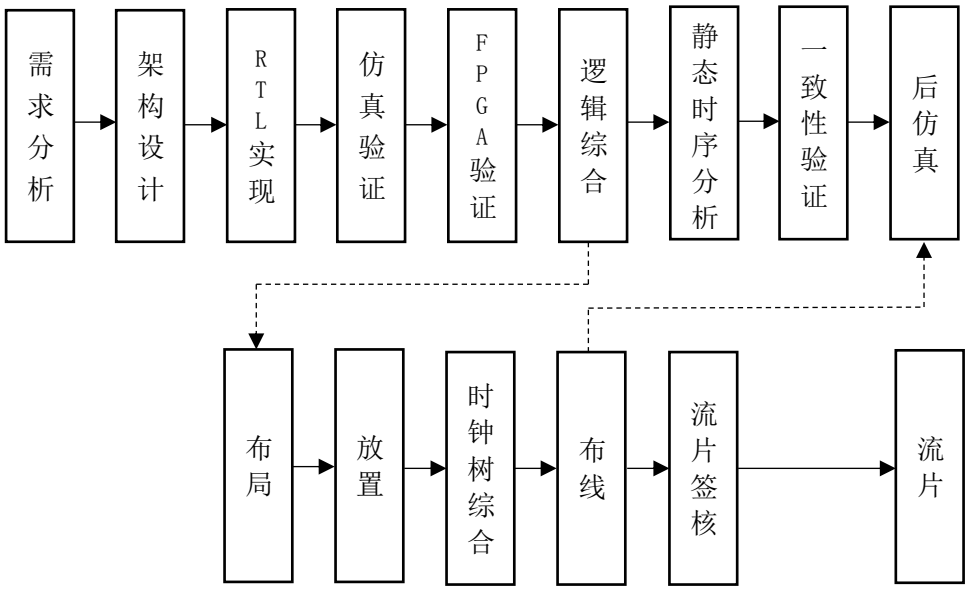


图 5.芯片设计总体流程

5.1.1 芯片前端设计流程

芯片前端设计流程主要有芯片需求分析、硬件架构设计与分析、RTL 实现、功能验证（仿真验证）、FPGA 验证、逻辑综合、静态时序分析、一致性验证以及时序仿真（后仿真）。

需求分析主要是了解芯片功能、性能、成本，对后续芯片设计产生指导作用。在确定需求以后进入具体实现流程。

硬件架构设计与分析完成芯片的高层次架构的分析与建模，为硬件提供一个正确的软件功能模型，更重要的是通过大量的高层次仿真和调试，为 RTL 实现提供总体性的设计指导。

RTL 实现根据硬件架构设计和分析的结果，完成从高层次描述到使用硬件描述语言（RTL）描述的实现过程。常用的硬件描述语言有 verilog、system Verilog 及 VHDL 等。

功能验证（仿真验证）是在无硬件延迟的理想情况下，通过大量的仿真，发现电路设计中的各种错误。在功能验证过程中将会使用到各种验证手段，其中形式化验证起到主要作用。功能验证的主要指标是功能覆盖率，覆盖率越大硬件出错的概率越小。

FPGA 验证是将 RTL 代码映射到 FPGA 硬件平台上进行验证的方法。因为功能验证的运行速度较慢，不能启动较为复杂的应用（比如操作系统），也不能外接较为复杂的硬件设备（比如显示控制器），即使经过了功能验证（仿真验证），其验证仍然不完全。FPGA 具有和 ASIC 类似的后端设计流程，运行速度相对较快，可以连接大量的外设，更接近实际的 ASIC 环境，成为芯片验证中的重要环节。

逻辑综合将 RTL 代码映射为与工艺库相关的门级网表。这个过程的输入文件为 RTL 代码、foundry 提供的工艺库文件（db）以及由逻辑综合工程师和 ASIC 工程师共同确定的约束文件（SDC）。逻辑综合完成后输出门级网表（netlist）、标准延时文件（SDF）、工程文件（DDC）以及各种报告。这些报告包括时序报告、面积报告、约束报告、时钟报告和设计违例报告等。经过逻辑综合以后，芯片后端设计可以介入开始芯片后端设计。

静态时序分析是了解芯片的设计是否满足芯片的时序性能设计要求。静态时序分析从逻辑综合步骤开始，每进行一次调整都要进行时序分析，以确认设计在时序上是否收敛。

一致性验证用于确认 RTL 代码和门级网表在功能上是否完全一致。需要进行一致性验证的主要原因是逻辑综合步骤会对生成的网表进行性能优化，有可能对实际电路进行调整，一致性验证用于检查调整后的电路与 RTL 表示的电路的一致性。

时序仿真（后仿真）与功能仿真（前仿真）类似，只是将要仿真的 RTL 换为网表，并且需要加载标准延时文件（SDF）和工艺库仿真模型。该步骤的目的在与观察电路延迟接近实际芯片的条件下，功能是否还能保持正确。

5.1.2 芯片后端设计流程

芯片后端设计也称为物理设计，将门级网表转换成有功能描述、物理大小定义和位置位置信息的硬件单元，完成所有单元之间的连线，在实现过程中满足芯片设计需求的面积、功耗以及性能等要求。

芯片后端设计的主要流程有布局、放置、时钟树综合、布线、signoff。

布局（Floor Plan）阶段需要定义芯片的长宽尺寸、pad 或者 pin 的位置、芯片的时序目标、电源环的宽度以及 strap 的长度或者宽度等内容。如果芯片中包含锁相环、RAM、寄存器堆等宏单元，还要确定这些宏单元的位置，以及如何摆放能使得走线更顺畅，芯片面积更小等。

放置（placement）的目的是将所有标准单元放入 core area 中，并能满足拥塞及时序要求。

时钟树综合的目的是为了生成满足芯片时序要求的时钟树。芯片中的时钟网络要驱动电路中所有时序单元，因此时钟源端门单元的负载很多，负载延时很大且不平衡，需要插入缓冲器减少负载和平衡延时。时钟网络及其缓冲器构成了时钟树。

布线是将所有需要连接的线在芯片中形成实际的金属走线。布线中需要满足 DRC 和 LVS 等规则检查，不会导致芯片时序变差也不会引入信号完整性问题。此外布线还要考虑可制造性设计（DFM），主要解决在制程中所遇到的一些问题，比如 fix antenna、add filler、slot wire、fill nbotch 和 fill dummy 等。

signoff（签核）是芯片后端设计的最后流程。主要完成一些最后的检查工作。检查芯片是否有时序违例，检查芯片静态功耗、动态功耗和 IR drop、检查设计规则如 DRC、LVS、ANT、ERC 和 DFM 等，最后还要检查布线完成后的网表与综合后网表的功能一致性。完成 signoff 检查后即可以进行流片工作。

5.2 以往开源芯片现状与分析

芯片开发的投入非常高，表现为投入人员多、开发周期长、芯片流片费用高等。芯片的整个开发流程对 EDA 软件的依赖极大，没有相应的 EDA 软件，芯片开发几乎无法开展。这些 EDA 软件种类繁多，专业性强，价格昂贵，普通个

人开发者无法负担。芯片的开发特别是后端设计对 foundry 的依赖极大，没有 foundry 的技术资料，几乎很难完成芯片的开发并正常流片。而 foundry 的技术资料也不会对个人开发者开放，导致熟悉开发流程的研发人员数量稀少。

除此之外，现在的芯片基本上都以片上系统（System-on-Chip，SoC）的形式出现，芯片上集成了非常多的功能部件，一个芯片开发团队很难把所有的功能部件独立开发出来，会集成很多第三方 IP。由于知识产权的问题，这些第三方功能部件可能不能进行开源，也在很大程度上限制了开源芯片的发展。

从以上分析来看，芯片开发的门槛极高，只有实力雄厚的大公司才能承担，普通个人开发者很难参与，导致开源芯片远不如开源软件发展迅速和普及。事实上，由于芯片开投入大、门槛高、周期长、知识产权复杂导致开源芯片发展极为缓慢。从某种意义上讲，目前并没有完全意义的开源芯片存在，基本上都是以开源 IP 的形式出现。

5.3 开源 IP

随着集成电路行业制造工艺的飞速发展，SoC 芯片集成度急剧增加，其功能和复杂度也大幅提高。从设计开发难度和上市时间上看，这些特征都对芯片设计人员提出了更大的挑战。目前，设计复用技术已成为解决上述问题的有效方法。根据业界经验，任何逻辑模块如果不作任何修改就可以在十个或更多设计项目中复用，都应以 IP（Intellectual Property）核的形式进行开发设计，以供更多开发者使用。基于 IP 核的设计已成为 SoC 芯片设计的必由之路。选择满足产品需求，又经过严格硅验证的 IP，可大幅降低 SoC 设计的复杂度和难度，提高投片成功率，加快产品的上市速度。

IP 核是指具有独立知识产权的电路核，用于实现特定的电路功能和结构，一般包括软核、硬核与固核。软核是以 FPGA 为代表的硬件可编程逻辑器件为目标载体，用硬件描述语言（如 Verilog HDL）设计可复用电路。硬核是针对集成电路制造而设计的电路结构掩膜。固核是完成综合后以网表形式存在的文件。以 IP 核形式设计的各种功能模块具有良好的兼容性和移植性，恰当地复用这些 IP 核可使设计效率大大提高，降低设计成本和开发周期。

目前各芯片设计厂商及逻辑模块开发商均提供商用的 IP 核设计。这些 IP 核往往集成于相应的芯片开发工具中或单独向用户提供销售，用户无法得到源代码。对于资金短缺的小规模设计团队或个人，这些闭源 IP 核的使用授权费用大幅增加了设计的成本负担。同时，由于缺少相应厂商的技术支持，完全闭源的 IP 设计也为系统集成带来了极大困难。即使可以获得技术支持，也需要付出额外的经济代价。

作为一种推广尝试，自 21 世纪初开始，国内外一些非盈利组织和个人致力于开源 IP 核的发展，为芯片设计人员提供遵循开源协议（如 GPL）的免费 IP 核，具有代表性的开源 IP 组织包括：

- OpenCores：http://opencores.org/projects
  著名的 OpenRisc 处理器核即是 OpenCores 组织提供的基于 GPL 协议的开源 RISC 处理器源码
- Design And Reuse：https://www.design-reuse.com/
- OpenHW：http://www.openhw.org
  OpenHW 社区是赛灵思（Xilinx）大学计划在中国资助的非营利学生社团组织，是 Xilinx 学生俱乐部的官方网站。

这些开源社区为中小公司、大学及科研院所提供了较为丰富的 IP 核开源 RTL 代码资源。上述机构在开源 IP 核的支持下，可以使用最小的人力和财力投入，完成小规模 SoC 芯片或科研样片的投片，并加速设计迭代速度。

由于支付了高额的使用授权费用并受到严格知识产权限制，各大芯片设计公司的高性能 IP 核还不能在开源社区公开出来。目前开源社区中开源 IP 核 RTL 代码一般是由一些科研机构或硬件设计爱好者贡献出来的，设计水平相对有限。因此，这些社区提供的 IP 核往往更加关注低端嵌入式应用领域，并主要提供 SoC 芯片中较为易于实现的低速 I/O 接口（如 UART、SPI、I2C、USB 等）。

另一方面，大学及科研院所受限于流片资金的限制，更多专注于基于 FPGA 硬件可编程逻辑器件的工程设计与实现。由于开发工具的通用性与设计语言的标准化，面向 FPGA 的 IP 核逻辑设计过程基本与器件的硬件结构无关。与此相反，由于使用的制造工艺不同，ASIC 设计需要重点关注底层的硬件结构及芯片

封装细节，仅凭开源 RTL 代码无法完成最终的设计。这就造成开源 IP 核，特别是复杂高速 I/O 接口 IP 核（如 DDR 控制器、PCIe 控制器、SATA 控制器、网卡控制器等），在被迁移到 ASIC 设计时，需花费使用者大量额外的时间进行修改和适配，大幅降低了开源 IP 核的可移植性。

缺少 SoC 芯片中最核心的高可用处理器 IP 核，也是制约设计人员广泛使用开源 IP 社区的重要因素。由于缺少对整体设计的考量，开源 IP 社区中发布的 I/O 接口 IP 核往往仅针对某一接口内部功能进行优化设计，并未充分考虑与处理器核的整体适配，进而缺少 SoC 集成支持（如 SoC 片上总线接口），造成移植开发过程中需要投入大量额外的设计精力。同时，这些开源 IP 核对应的驱动程序或应用程序软件也比较欠缺，对软硬件系统的集成开发也会造成极大的困难。

随着开放处理器指令集 RISC-V 的出现，上述问题已开始得到有效缓解。一大批围绕 RISC-V 指令集开发设计的开源处理器核心已逐步问世，其中的典型代表包括：

- 标量处理器 Rocket Core：Rocket Core 是美国加州大学伯克利分校设计的一款 64 位、5 级流水线、单发射顺序执行处理器。该处理器核支持 MMU 及分页虚拟内存，可移植 Linux 操作系统。兼容 IEEE 754-2008 标准的 FPU；支持分支预测。采用 Chisel 编写，已成功投片十余次；
- 超标量乱序执行处理器 BOOM：BOOM（Berkeley Out-of-Order Machine）是 UCB 设计的一款 64 位超标量、乱序执行处理器，支持 RV64G。采用 Chisel 编写，利用 Chisel 的优势，只使用了 9000 行代码。流水线可划分为六个阶段：取指、译码/重命名/指令分配、发射/读寄存器、执行、访存、回写；可支持参数化配置。已在 40nm 工艺下进行了验证投片；
- 国内第一款开源 RISC-V 处理器核蜂鸟：蜂鸟系列处理器核由中国大陆本土研发团队开发，专为 IoT 领域量身定做，其具有 2 级流水线深度，用户能够轻松获取与本土开发人员的交流和支持。该处理器所有代码均为人工编写，添加丰富的注释且可读性强，非常易于理解。功耗和性能指标均优于目前主流商用同类型处理器。

这些高质量的开源 RISC-V 处理器不仅提供处理器 IP 核本身的实现，还提供完整的配套 SoC、详细的 FPGA 原型平台搭建和软件实例，并支持完整的调试方案。这就使得二次开发用户可以按照其步骤重现出整套系统，轻松将这些处理器 IP 核应用到具体产品中，有效提升设计复用的效率。

受 RISC-V 指令集所引领的开源开放生态影响，其他处理器设计公司也纷纷将其内部的高水平处理器核 IP 以不同形式开放出来，供开发者使用。ARM 为帮助嵌入式设计开发者、初创企业以及 OEM 厂商能够快速获得其 IP 而推出了加强版的 DesignStart 计划。在该计划中，ARM 将开放 Cortex-M0、Cortex-M3 和 Cortex-A5 及相关 IP 子系统，并取消了预付授权或者评估费用，改以产品成功量产出货后才收取版税的模式运作，旨在降低开发风险。以此为基础，用户通过一个简单的可下载授权，即可在项目商业开发初期，进行定制化处理器的设计、评估和原型开发。而通过在线获取 ARM CoreLink SDK-100(一个已获证实的子系统和系统 IP 解决方案)、CoreLink SSE-050 子系统和已获验证的对 mbed OS 的支持，开发效率实现了 10 倍以上的提升。此外，作为生态系统型的公司，ARM 还在该项目中提供了来自 ARM 以及 ARM 认证的设计公司合作伙伴的设计辅助服务，以及数以千计的物理 IP 库，以期最快、最高效地实现芯片。虽然未完全开放设计的 RTL 源代码，但通过为用户提供大量经过验证的、可用的软件和中间件，以及非常容易获取的开源支持、工具以及欣欣向荣的生态系统，ARM 还是希望能够开启用户发掘定制化 SoC 潜能并且加速其上市时间。

MIPS 指令集的持有者 Wave Computing 公司于 2018 年底宣布将于 2019 年第一季度允许参与者自由访问最新版本（core R6）的 32 位和 64 位 MIPS 指令集，无需支付许可或使用费。该公司同时承诺 MIPS 将为开源社区带来具备商业水准和工业强度的指令集架构，并将为芯片设计人员提供根据经过验证且经过良好测试的指令集来设计自己处理器核心的机会。

通过上述分析可以看出，由自由开发者主导的开源 IP 核虽然可在一定程度上为芯片开发者提供设计参考，但因生态本身并不完善，从而在 SoC 芯片的二次开发过程中引入过多的额外设计开销。我们认为，以处理器核为核心，构建从处理器、内存、I/O 接口等开源 IP 核到相应操作系统、驱动程序软件；并打

通逻辑设计到物理设计的闭环设计链，依托有影响力的组织或公司打造完整生态，是开源 IP 核支撑未来开源芯片设计的必由之路。

5.4 开源工具链

在芯片的设计中，EDA(Electronics Design Automation，电子设计自动化)工具是非常重要的辅助设计软件，且价格昂贵，一整套 EDA 工具可达数百万元，已经成为发展开源芯片的巨大阻碍。故而，人们转而开始研究开源 EDA 工具，以降低芯片设计门槛，这也是美国的电子复兴计划中的主要目标之一。

EDA 工具贯穿着整个芯片设计流程，包括多个阶段：逻辑编译，综合，优化，布局，布线和仿真等，各阶段之间的松耦合性为开源 EDA 工具的发展提供了很大空间。目前，开源社区已经孕育出了诸多 EDA 工具，功能完善且互为补充，借助这些工具已经基本可以满足芯片设计的所有需要，不少创业公司已经在开始使用开源 EDA 工具链来研发低成本芯片。本文选择了其中较为活跃且知名者，列在了表 8 中。

这些开源 EDA 工具主要分为两类：功能专用和全流程设计。其中功能专用工具的功能较为单一，仅专注于芯片设计流程中的单个或数个阶段，且性能较好。以 Icarus Verilog 和 GTKWave 为例，可由 Icarus Verilog 对 RTL 代码进行仿真，再交由 GTKWave 来呈现波形图。此外，功能专用的 EDA 工具，如 Magic 等，支持自定义功能，为用户尝试更加高效的布局、布线和路由算法提供了很大空间。全流程设计工具则包含了芯片设计流程中所需要的几乎全部功能，如 Electric，QFlow 和 gEDA 等，其一般由多个功能专用的 EDA 工具整合而成，支持从 RTL 代码到版图的整个芯片设计流程。其中 Electric 和 gEDA 支持可视化 IDE 设计，进一步优化了设计效率和用户体验。

表 8. 知名的开源 EDA 工具链介绍

工具	功能描述
Xcircuit	提供可视化的电路原理图设计，且支持跨平台(Linux/Win)
Icarus Verilog	支持 Verilog 综合和仿真，且可生成指定格式网表
Verilator	综合和仿真工具，且仿真速度较快
GTKWave	查看仿真波形的工具
QRouter	路由工具
IRSIM	门级电路模拟工具
Magic	电路布局工具，性能稳定且应用广泛
Ngspice	历史悠久的电路仿真工具，孕育了诸多商用 SPICE 软件
Electric	可视化的全流程芯片设计工具
QFlow	完整的芯片设计流程工具(松散的工具组合)
gEDA	可视化的全流程芯片设计工具
PCB	PCB 布局和编辑工具

目前开源 EDA 工具的发展蓬勃向上，但仍然面临着如下挑战：其一，开源 EDA 工具在部分关键算法上与商用软件如 Cadence、Synopsys 等仍有差距，亟需性能更加优秀的算法，这将大幅提升开源 EDA 工具的可用性，从而吸引更多的使用人群；其二，目前开源的器件库较少且工艺较低，限制了最终设计的芯片的实用性，加强与芯片设计厂商的合作交流，提高器件库的工艺，对设计真实可用的开源芯片至关重要；其三，各开源 EDA 工具达数十个之多，且互相之间都是松耦合，这虽有利于各 EDA 工具的独立发展，但也限制了其易用性，故而构建完整高效，灵活可配置，且简单易用的开源 EDA 工具链非常重要。

5.5 开源芯片“死结”与突破口

传统芯片开发的门槛非常高。一次芯片设计流程需要投入 20~50 人，花费 1~2 年的时间，同时需要投入几千万元的资金。如果流片失败，投入的时间和成本将会付诸东流，需要承当相当大的风险。

同时，芯片设计的上述投入之间的关系还存在“死结”（如图 6）。具体地，

芯片设计本身就需要投入大量的成本，因此开发人员或企业不愿意开源其设计的芯片与 IP；这导致无高质量的开源芯片与 IP 可用，于是企业购买高价的 IP；这进一步推高了开发代价，导致企业希望充分验证设计来提高流片的成功率，这又需要投入更多的人力和时间进行验证……如此循环，最终形成了一个死结。

打破上述“死结”的一种可能的解决方案是芯片敏捷开发——如果芯片设计也能像软件开发一样，3~5 人的小团队在 3~4 个月内，只需几万元便能研制出一款有市场竞争力的芯片，就可以大大降低芯片开发的成本和风险。

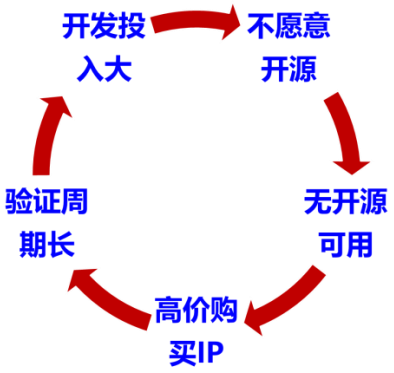


图 6 开源芯片“死结”

5.6 芯片敏捷开发

在学术界，目前大多数微结构相关的研究还是在模拟器上进行，这是因为基于 FPGA 的工作周期一般都比较长，例如中科院计算所的标签化 RISC-V 研究需要花费约半年的时间来完成 FPGA 原型系统的构建，基于真实芯片设计的研究工作则鲜有听闻。在工业界，芯片的开发周期长达 2~3 年，其中设计和验证工作需要花费 1~2 年，投片需要花费约 1 年。如果流片失败，投入的时间和精力将会付诸东流，风险相当大。因此，如果有办法加快芯片设计的效率，实现芯片的敏捷开发，那么将会对学术研究和芯片产业带来巨大的影响。

如果通过敏捷开发模式将芯片设计成本降低到几十万甚至几万、开发周期降低到几个月，那么用这种方式开发的 IP 模块成本也将大幅下降，开发人员将更愿意开源与共享 IP 模块。

当芯片开发周期也能从数年缩减为几个月，那将形成一种软硬件协同的敏捷开发模式（如图 7），这将颠覆现在的 IT 产品开发模式。如今，互联网应用开发

团队一般有负责手机 APP 的前端工程师，与负责服务器端的后端工程师配合起来一起开发。而在未来的软硬件协同敏捷开发模式下，开发团队将包括软件端工程师与硬件端工程师——软件端通过几个月开发出新的软件功能，硬件端则用几个月很快实现出相应的加速芯片。这正是两位图灵奖得主 John Hennessy 与 David Patterson 在图灵奖演讲中所推崇的领域专用体系结构 DSA（Domain-Specific Architecture ）的体现。

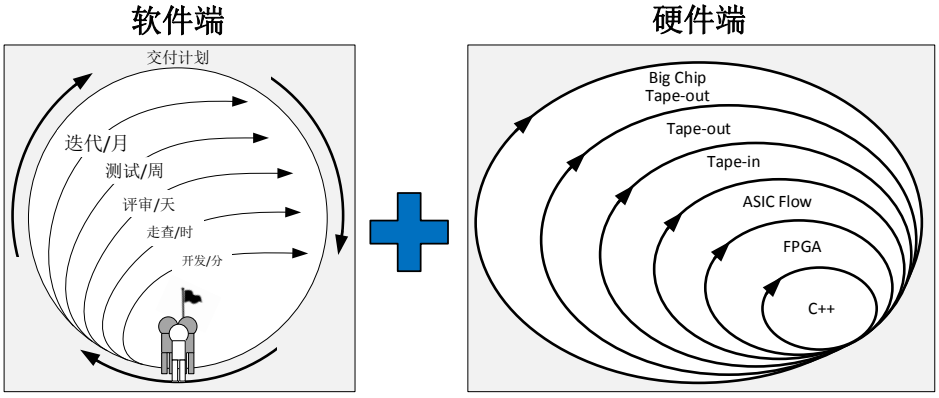


图 7. 软硬件协同的敏捷开发模式

伯克利研究团队在 2010 年的时候已经考虑到这方面的问题了，他们在 2012 年的DAC 会议上发表了一门新的编程语言 Chisel 来进行硬件的敏捷开发。Chisel 的主要目标是减少项目中的重复代码，提高代码密度，从而提升开发效率、代的可读性和易维护性。编写 Chisel 代码后，用 Chisel 编译器将其编译成底层的 Verilog 代码（网表），可用于标准的 ASIC 和 FPGA 流程。需要说明的是，虽然 Chisel 支持传统硬件描述语言不具备的很多高级特性，但 Chisel 还是一门硬件构建语言，而不是高层次综合语言。硬件构建语言用于描述电路具体如何构建，而高层次综合语言则用于描述算法的流程。表 9 中总结了 Chisel 和传统的硬件描述语言（Verilog 和 SystemVerilog）之间的对比。

表 9.不同硬件描述语言的特性总结

特性	Chisel	SystemVerilog	Verilog
信号整体连接	支持	支持，但有局限性	不支持
元编程	支持	部分支持且不可综合	不支持
面向对象编程	支持	部分支持且不可综合	不支持
函数式编程	支持	不支持	不支持

1） 信号整体连接

Chisel 丰富的类型系统使得我们可以很容易地修改类型的定义，在不修改引用该类型的代码(如通过该类型来定义的信号)的情况下，可以轻松对该类型的信号进行全局修改。此外，整体连接运算符“<>”会根据类型的定义，把相同类型的 2 组信号的相应成员信号一一连接，从而省去重复而且易出错的连线代码。在工程项目中，我们一般会在代码中的不同位置定义同种类型的信号，比如总线信号。在这种情况下，信号整体连接的特性能大幅减少项目中的重复代码。相比而言，使用 Verilog 语言进行开发时，若要对总线的成员信号进行改动，工程师只能对项目中所有用到总线的模块端口逐一进行改动，同时还需要手工添加或移除相应成员信号的 assign 语句，这对 Verilog 工程师来说是一件非常麻烦的事情。SystemVerilog 的 interface 特性一定程度上也能实现类似 Chisel 中信号整体连接的功能，但它仍然有一些局限性，例如 modport 不能嵌套定义，使得我们无法从不同的 interface 中将相同的部分进一步抽象出来。Chisel 中的信号整体连接特性比 SystemVerilog 还要强大，这使得我们可以进一步减少重复的代码，实现“一改全改”的效果，从而提升项目的开发效率。

2） 元编程

Chisel 支持基于 Scala 的元编程，可以借助 Scala 的特性抽象出多份相似的 Chisel 代码的共性部分，我们只需要维护一份共性代码，具体的 Chisel 代码可以通过对共性代码实例化得到，从而进一步减少冗余的代码。一个常见的例子是使用模板来实现队列原型。无论队列中的元素是何种类型，队列本身的功能都是一样的。使用模板可以将队列本身的功能抽象成一个带类型参数的队列原型，然后在实例化时给出队列元素类型，就可以得到一个该种元素类型的队列。通过

这种方式，我们不必分别实现不同元素类型的队列，只需要维护一份队列原型即可。更进一步地，需要修改远队列元素类型时，可以实现“一改全改”的效果。而为了在 Verilog 中实现这一需求，针对代码中的所有相关队列，我们要么修改队列元素的宽度，要么通过不同的队列来缓冲不同的元素，同时还需要维护不同队列之间的对应关系，十分繁琐。SystemVerilog 也支持模板，但相应的代码是不可综合的，更多的是用在测试激励的编写中。

3） 继承

面向对象编程中的继承特性允许我们将一些共同的代码特性通过一个父类抽象出来，达到减少冗余代码的效果，同时层次化的类型系统还可以使类型检查的过程更加严格，从而降低代码出错的可能性。具体地，我们在实现一个模块时，只需要从父类继承，就可以让该模块自动拥有父类定义的所有特性，从而避免在不同的模块中重复实现这些特性。在硬件项目中，不少模块之间都有一些共同的特性，例如 cache 的不同替换算法都需要读出并更新历史状态，而对于带有总线接口的模块，总线相关的参数也非常相似。在这些情况下，使用继承特性可以有效地节省项目的代码量，让代码功能一目了然，容易维护。相比之下，传统的硬件描述语言则难以实现类似的效果，例如若使用 Verilog 编写，最简便也要通过子模块的方式来实例化这些共同的特性，甚至要在不同的模块中重复声明相关的参数，使得 Verilog 代码难以一目了然。而 SystemVerilog 虽然支持继承，但功能有限，而且相应的代码不可综合，无法对硬件构建提供帮助。

4） 重载

面向对象编程的另一个特性是重载，重载可以提升代码的可读性。首先，Chisel 支持以函数的方式来对电路进行抽象(包括模块实例化)，以达到复用的效果，同时可以把函数返回值当做相应电路的输出，直接作为 Chisel 表达式的一部分，而无需额外定义函数的输出信号。此外，Chisel 支持函数名重载，允许定义带有不同参数的多个同名函数，而且可以设置缺省参数，在调用函数时，Chisel 会在多个函数定义中自动选择类型匹配的函数定义。运算符重载则是把运算符当作一个特殊的函数，可以根据运算符左右两侧的变量类型来决定运算符的具体行为。例如 Diplomacy 语言对运算符“:=”进行了重载，可以让运算符两侧节点的 AXI4 主从端口通过“<>”进行整体连接，代码功能一目了然，容易维

护。相比之下，Verilog 和 SystemVerilog 中的函数、任务和运算符均不支持重载，无法实现上述效果，只能使用 module 来实例化这些模块，从而引入大量的连线代码。

5） 函数式编程

Chisel 支持使用函数式编程来描述电路，可以编写更紧凑、可读性更好的代码。首先，Chisel 使用“容器”(collection)来抽象电路元素，容器中可以是信号、寄存器、端口、模块、映射等，或者是这些元素的复合。然后，Chisel 使用 map 算子对容器中的对象进行批量操作，操作可以是连接、归约、算术和逻辑运算、选择、实例化、函数调用、计算新映射等，或者是这些操作的复合，操作结果返回一个新容器。通过容器和 map 算子的组合，我们可以轻松地通过少量代码描述复杂电路。这说明了 Chisel 确实是一门硬件构建语言，而不是高层次综合语言：容器中的对象和 map 算子的操作，都是可综合电路中的概念。Chisel 只是使用高级特性来方便地描述电路，而不是通过它们来描述算法。而 Verilog 虽然有 for 和 generate 特性，但它们只能基于整数进行迭代，与 Chisel 中可以对任意对象批量进行任意操作的功能相比，Verilog 的迭代功能就非常有限了。例如 Chisel 可以在 map 算子中使用“<>”运算符，能实现进一步减少重复代码的效果，而 Verilog 和 SystemVerilog 均不支持函数式编程，难以实现类似效果。

综上所述，和传统的硬件描述语言相比，Chisel 的这些高级特性可以大量减少项目中的冗余代码，提高项目的开发效率，同时高密度的代码也提高了可读性，使得项目更容易维护。正是 Chisel 语言的这些特性，使得它成为硬件敏捷开发的利器。

5.7 敏捷开发案例

我们将通过两个开发案例，分别从编码效率（开发耗时和代码量）和编码质量（性能、功耗和面积）这两方面，对以 Chisel 和 Verilog 为代表的敏捷开发模式和传统开发模式进行对比。

5.7.1 Chisel 与 Verilog 编码效率对比

中科院计算所包云岗研究员团队曾经由于项目需要，期望尽快实现一个简单的共享二级缓存。该二级缓存无需实现一致性协议的功能，只需要具有缓存功能

即可，但需要集成到标签化 RISC-V 项目中并正确运行。团队中的两人分别进行独立开发，具体情况如表 10 所示。

表 10.二级缓存开发案例对比

	工程师	本科生
项目经验	熟悉 OpenSPARC T1，修改过赛灵思系统缓存	做过 CPU 课程设计，有 9 个月的 Chisel 开发经验
使用语言	Verilog	Chisel
时间	6 周	3 天
效果	仍然无法启动 Linux	可启动多核 Linux，并支持 DMA 模式的以太网
有效代码行数	~1700	~350
代码复用	否	是

参与开发的其中一位是团队中的工程师，他在标签化项目中早期阅读并理解过 OpenSPARC T1 的二级缓存源代码，也修改过赛灵思系统缓存，在其中成功添加基于标签的路划分功能，具有丰富的缓存设计经验。这位工程师使用传统开发模式，选择 Verilog 语言来开发这个二级缓存，并决定从零开始搭建测试环境，不复用任何代码。他主要开发了 6 周，编写了约 1700 行有效代码。遗憾的是，他开发的二级缓存模块在几个月后仍无法在标签化 RISC-V 项目上成功运行。

参与开发的另外一位人员是团队中的大四本科生，他做过 CPU 课程设计，并有 9 个月的 Chisel 开发经验，但从未设计过二级缓存。这位本科生使用敏捷开发模式，选择 Chisel 语言来开发这个二级缓存，并使用 Chisel 标准库来帮助设计，同时也复用标签化 RISC-V 项目的测试环境。经过了两天的设计和仿真验证，他就写出了一个可以支持多核 Linux 启动的二级缓存，有效代码量约 350 行，只有工程师编写代码的 1/5。不过这个二级缓存一开始并不支持不完整的突发读写，导致 DMA 模式的以太网模块不能正确工作。1 周后团队将这个情况反馈给他，他又额外花了 1 天时间改动了约 50 行有效代码，添加了对不完整突发读写的支持，并进行仿真验证，最终成功支持 DMA 模式的以太网模块并正确工作。

这个案例充分展示了敏捷开发在项目中的优势：运用语言的各种高级特性、复用标准库中已经经过验证的模块来编写易读、易维护、高密度的代码，可以大大提升项目开发的效率。具体到这个案例中，敏捷开发模式的效率是传统开发模

式的 14 倍！由于没有进行代码和环境的复用，这位工程师表示他花费了一半的时间在构建测试环境和编写并测试基本元件（如 RAM、队列、仲裁器等）中，而使用 Chisel 的本科生并没有在这些事情上花费任何时间。不过即使把工程师花费的一半时间排除在效率比较的范围之外，敏捷开发模式的效率仍然是传统开发模式的 7 倍。实际上代码重用是敏捷开发的一个基本理念，代码的重用率越高，项目开发的效率就越高。

有趣的是，这位工程师后来提到，当时为了编写一份端口数量可配置的总线连接代码，他在 generate 特性的基础上运用了一些特殊技巧实现了这一功能，编写了约 250 行 Verilog 代码。但是工程师在编写过程中，由于连线和数字下标太多，并且需要顾及总线握手协议,他曾经因疏忽而导致两个连线错误的 bug，花了约 3 天时间才发现并修复它们。这位工程师还表示，这部分代码的可读性其实并不好，使用 Verilog 实现这一功能实在太繁琐了，即使在代码中有相应注释，他在 1 周后也不能马上理解他使用的特殊技巧是如何工作的了。相比之下，若使用 Chisel 来实现类似功能，我们只需要编写两行代码即可，可读性好，而且几乎不会出现错误。此外，本科生实际上也是在 1 周后重新回头阅读并修改自己编写的二级缓存，但他仍然在 1 天内成功修复了问题，这说明代码的可读性对项目维护来说是非常重要的。

5.7.2 Chisel 与 Verilog 编码质量对比

考虑上述案例展示的两个设计，虽然它们的需求是一致的，但不同的开发人员可能会采用不同的实现方式，导致编码质量的可比性不强。为了进一步对比 Chisel 和 Verilog 的编码质量，研究团队安排另一位没有 Chisel 开发经验的大四本科生，让他来把上述 Verilog 代码中的部分关键模块翻译成功能等价的 Chisel 代码。在项目中提供了一些测试，用于验证翻译结果的等价性。

5.7.2.1 逐句翻译

由于这位本科生一开始并没有 Chisel 的开发经验，他需要从零开始学习 Chisel，并选择最简单的翻译方式：逐句翻译，而不使用 Chisel 的高级特性。这位本科生表示，他一开始觉得 Chisel 代码比较难读懂，但是学习并逐句翻译 Chisel 代码的过程中，他也逐渐感受到 Chisel 的方便之处，例如丰富的标准库、方便的数据类型系统及其转化机制、简洁的时序逻辑编码风格等，这些特性让他

对 Chisel 有了新的认识。

表 11.Chisel 和 Verilog 的性能、功耗、面积和对比

	Verilog	Chisel	Chisel-opt
WNS/ns	0.637	0.668	1.511
最大周期/MHz	135.814	136.388	154.107
功耗/W	0.770	0.749	0.749
LUT 逻辑	5676	6422	2594
LUT 存储	1796	1264	1492
FF	4266	3638	747
有效代码行数	618	470	155

我们用 Chisel 编译器把翻译后的 Chisel 代码编译成 Verilog 代码(网表)，然后对其以及工程师编写的 Verilog 代码分别进行评估。我们在 Vivado 2017.01 中，使用 xc7v2000tfhg1716-1 型号的 FPGA，在 125MHz 的时钟频率下进行评估，结果如 9 中第 2，3 列所示。为了评估设计的性能，我们展示了时序报告中的最差负时序余量（worst negative slack，WNS），并将其换算成可运行的最高时钟频率，结果显示 Verilog 和 Chisel 最高分别可运行在 135.814MHz 和 136.388MHz 的时钟频率下，性能非常接近。而两者的功耗则分别为 0.770W 和 0.749W，和 Verilog 相比，Chisel 的功耗节省了 2.73%。而为了展示面积开销，我们给出两份设计各自消耗的查找表(lookup table,LUT)和触发器(flip-flop，FF）数量。我们对 LUT 的消耗分成逻辑和存储两部分来统计，和 Verilog 代码相比，Chisel 代码多消耗了 13.14%的 LUT 逻辑，但节省了 29.62%的 LUT 存储，这是因为 Chisel 标准库提供了 RAM 相关的基本元件，它与工程师实现的 RAM 被 Vivado 分别映射成 RAM64M 和 RAM64X1D 这两种不同的原语，其中 RAM64M 会消耗更多的 LUT 逻辑，但节约大量 LUT 存储。对于 FF，Chisel 比 Verilog 节省了 14.72%,这是因为 Chisel 编译器对代码进行了更进一步的优化。从代码量上看，即使是逐句翻译，Chisel 也比 Verilog 节约了 23.95%的代码量，这是因为：1)和 Verilog 的 generate 特性相比，基于 Scala 的元编程特性编写出的代码更加

紧凑；2） 在 Chisel 中编写时序逻辑无须像 Verilog 那样声明 always 块；3）Chisel 代码不会产生锁存器（latch），无须像 Verilog 那样补全 if 和 switch 的所有分支。

逐句翻译方式的整体评估结果说明，使用 Chisel 开发不但节省了代码，编码质量也和 Verilog 非常接近，在部分指标上甚至优于 Verilog。

5.7.2.2 使用 Chisel 高级特性

这位本科生对 Chisel 上手之后，研究团队让他使用 Chisel 的高级特性对代码进行重构，包括使用 Chisel 标准库来实例化 RAM 和队列等基本元件，同时使用第 3 节提到的信号整体连接、元编程、面向对象编程和函数式编程来节省代码量。在使用这些高级特性的过程中，这位本科生逐渐认识到 Chisel 的更多好处，他表示：使用 Chisel 标准库之后，因为大部分异步通信被统一封装为 Decoupled 模板类，为了能使用“<>”进行信号整体连接，自然就会尝试从 Bundle 类继承来定义模块端口，进而也会考虑使用函数式编程中的 map 算子对 Vec 数组进行简便的连接。这些好处让他体会到“Chisel 高级特性的一以贯之，相辅相成”。

研究团队对重构之后的 Chisel 代码进行评估，结果如表 11 中第 4 列（Chisel-opt）所示。令人惊讶的是，和 Verilog 相比，Chisel-opt 的 LUT 逻辑节省了 54.30%，而 FF 则节省了 82.49%！这是因为和工程师编写的基本元件相比，Chisel 标准库提供的基本元件更成熟，能使用更少的资源实现相同的功能。由于资源的大幅节省，Chisel-opt 的性能也得到了进一步的提升。具体地，可运行的最高时钟频率提升到 154.107MHz，与 Verilog 相比提升了 13.47%。除此之外，Chisel 的高级特性使得 Chisel-opt 的代码量更加精简，和 Verilog 相比，节省了 74.92%的代码量。

这个案例很好地展现了一个敏捷开发的例子：一个本科生的 Chisel 新手，可以在更短的时间内编写更少的代码，编码质量就能达到和工程师相当的水平，甚至还可以超越工程师。即使编码质量与传统开发有 20%的差距，敏捷开发仍然展现了其节省人力和时间的价值：能快速构建一个可以工作的原型，对项目开展来说是非常有意义的。从这点来看，敏捷开发确实大大降低了硬件开发的门槛。

5.8 Designless 设计模式

今天的半导体行业的分工，最为大家熟知的是 Fabless(无晶圆厂)模式（如图 8 所示），即芯片设计公司负责定义、设计和设计定案，而制造则是在提供代工的 Fab 完成；三星、Intel 等半导体巨头也会走包含从定义到制造所有环节的 IDM(Integrated Device Manufacture, 集成设备制造商)模式。此外不少公司是混合 Fabless 和 IDM，即有的产品走 Fabless 模式，而有的产品走 IDM 模式，例如 Broadcom 在 CMOS 芯片产品线走的是 Fabless，而在射频前端模组产品线则是 IDM 模式有自己的 Fab。在今天的半导体行业分工中，设计服务公司提供模块设计 IP 以及协助一些公司做设计，主要起的是辅助作用。

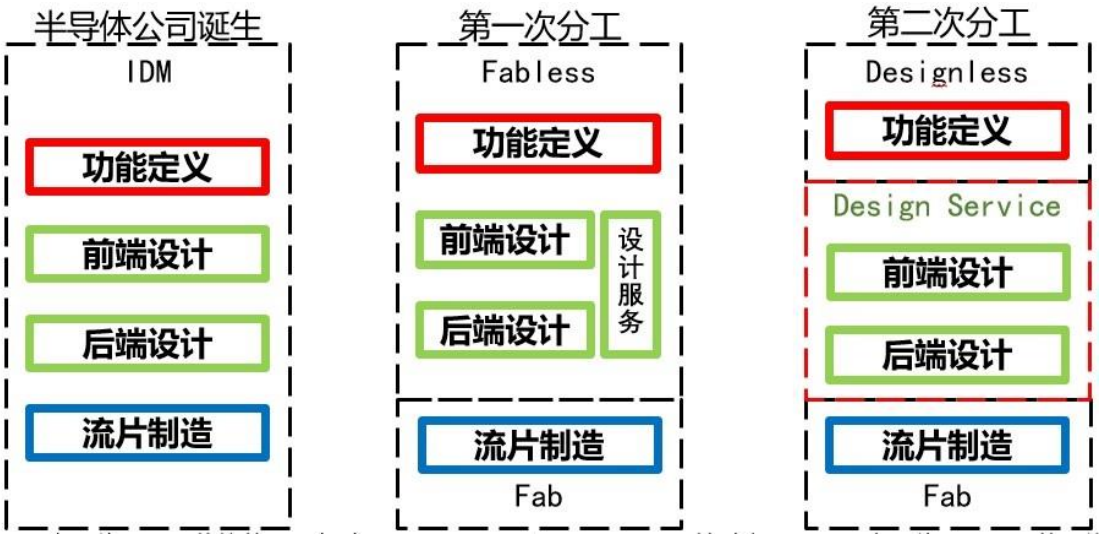


图 8　IDM、Fabless 和 Designless 的设计模式

在现在的 AI 时代，许多新的公司入局半导体行业，出现了一种新的模式，即 AI 硬件系统或算法公司负责定义芯片、完成小部分设计、并花钱完成设计定案流片，设计服务公司负责大部分设计，而代工厂负责芯片制造。这种新模式可以称为 Designless(无电路设计)模式（如图 8 所示）。历史上，半导体公司从传统的 IDM 走到 Fabless 模式，主要是因为 Fab 开销过高，成为了半导体公司发展的包袱，而代工厂则提供了一个非常灵活的选项。

未来的 AIOT 智慧物联网时代，大量碎片化的应用场景，导致不同应用场景下的系统公司和终端公司会入局半导体芯片，可能会出现 Designless 模式，把大量设计外包，以节省时间成本。系统公司做芯片，除了追求极致性能之外，还有快速的上市时间。对于他们来说，产品晚上市 3 个月，错过了产品上市周期节

点将会是灾难性的。如果要像传统半导体公司一样，需要从头开始培养自己的前端+后端设计团队，从头开始积累模块 IP，恐怕第一块芯片上市要到数年之后。这样的节奏，是跟不上 AIOT 时代电子产品的快速迭代的节奏。

那么如何实现高性能加快速上市呢？最佳方案就是这些巨头自己招募芯片架构设计团队做芯片定义，用有丰富经验的芯片工程师来根据需求定制架构以满足性能需求，而具体的实现，包括物理版图设计甚至前端电路设计都可以交给设计服务公司去做。半导体芯片的一个重要特点就是细节非常重要，ESD、散热、IR Drop 等一个小细节出错就可能导致芯片性能大打折扣无法达到需求。因此，如果把具体设计工作交给有丰富经验的设计服务公司，就可以大大减少细节出错的风险，从而减小芯片需要重新设计延误上市时间的风险。

事实上，我们已经在一些芯片中看到一些互联网巨头负责芯片架构+设计服务公司提供 IP/设计服务的模式了。一个例子就是微软使用在 HoloLens 中的 HPU，其中微软设计了多核架构以加速 AR 计算，而每个核则是使用了 Cadence 的 Tensilica IP。另一个例子，华米手环推出了全球可穿戴领域第一颗人工智能芯片“黄山 1 号”，其中就使用了 SiFive 的 E30 系列 IP。

历史上，半导体公司从传统的 IDM 走到 Fabless 模式，主要是因为 Fab 开销过高，成为了半导体公司发展的包袱，而代工厂则提供了一个非常灵活的选项，同样到了今天，芯片设计高昂的费用和时间成本，促使了半导体产业又一次分工，即由 Fabless 模式走向 Designless 模式。

6 业界动态

6.1 RISC-V 代表性企业与产品

表 12.RISC-V 代表性企业与产品

企业名称	代表性 RISC-V 产品
Google	BottleRocket RV32IMC Core
Qualcomm	N/A
NVIDIA	基于 RISC-V 指令集 Falcon Controller
Samsung	N/A
Western Digital	SweRV Core
Microsemi	PolarFire SoC FPGA 内置 U54-MC RISC-V 核

6.2 MIPS 代表性企业与产品

6.2.1 MIPS 处理器的应用领域

在处理器发展历史上，MIPS 公司的处理器曾经是唯一能覆盖从高性能计算到嵌入式领域的指令集架构，在市场上得到广泛应用。在 ARM 建立起移动互联网领域生态环境后，MIPS 指令集生态环境受到了极大冲击，即使在 MIPS 的传统优势领域也逐渐被 ARM 处理器取代。

MIPS 公司的 IP Core 产品目前主要被应用在人工智能、汽车电子、消费类电子、IoT 以及网络等 5 个市场。

人工智能领域，使用 MIPS 处理器 IP Core 的企业主要是 Wave Computing。该公司致力于基于数据流架构 AI 硬件加速技术，整合了 MIPS 的嵌入式处理器核。

汽车电子领域，MIPS 公司的 I6500-F 处理器 IP 刚刚通过 ISO 26262 和 IEC 61508 等汽车电子认证，有望顺利打入汽车电子市场。

消费类电子领域，MTK 和 Broadcom 公司分别将 MIPS 处理器用于 4G LTE 调制解调器芯片和家用无线路由器主控芯片中。

IoT 嵌入式单片机领域，代表性企业是美国 Microchip 公司。与全球绝大部分嵌入式单片机采用 ARM Cortex M 系列处理器核不同，该公司一直坚持采用 MIPS 的 32 位嵌入式处理器核。

美国 Cavium 公司（已被 Marvell 公司收购）生产的基于 MIPS 处理器的网络处理器在大吞吐率低延迟网络转发应用中占据一定市场份额。

值得一提的是在 2017 年的超级计算机 Green500 评比中，来自日本的 PEZY 公司采用 MIPS P6600 处理器核的超级计算机占据了榜单的第一、第二、第三以及第五名。

6.2.2 MIPS 指令集在中国的发展现状

基于 MIPS 指令集的处理器在中国有广泛应用。

在嵌入式领域主要有珠海炬力、北京君正和北京神州龙芯等。珠海炬力主要生产具有蓝牙及 WIFI 射频功能的多媒体 SoC 芯片，在消费类电子具有较大的市场份额。北京君正采用自主设计的 xburst 低功耗 MIPS 嵌入式处理器核，在电纸书、指纹锁以及智能硬件领域获得广泛应用。神州龙芯采用自主设计的 MIPS 嵌入式处理器核，在 IoT 领域有广泛应用。

在高性能桌面及服务器领域有龙芯中科技术有限公司，主要研发生产龙芯 2 号单核处理器系列以及龙芯 3 号多核处理器系列，产品广泛应用于桌面办公、服务器、工业控制、网络安全以及物联网等。

7 各国战略计划与项目部署

7.1 美国

美国国防部高级研究计划局（DARPA）资助了 RISC-V 基金会，同时于 2017 年启动了一项总值 15 亿美金的 ERI 计划，项目周期 5 年，以支持芯片技术的发展，重塑芯片开发制造流程，其中 POSH 子项目就是提供开源设计和验证框架，包括技术、方法、标准，从而实现具有成本效益的超复杂片上系统设计。DARPA 希望可降低复杂片上系统设计障碍的新工具能够打开一个特定应用设计创新的新时代。

7.2 欧洲

欧洲委员会于 2018 年启动了 EPI（European Processor Initiative）计划，拟开发面向欧洲市场的自主可控低功耗微处理器，其中重点关注 Exascale 级超算处理器，预计投资 1.2 亿欧元经费用于支持用户开发的超算处理器，而 RISC-V 和 ARM 都将作为此次计划的备选指令集。

7.3 印度

印度政府将 RISC-V 视为一次发展处理器的机会：（1）资助 9000 万美元开展的处理器战略项目（SHAKTI）目标为研制 6 款基于 RISC-V 指令集的开源处理器核，涵盖了 32 位的单核微控制器、64 核 64 位高性能处理器和安全处理器等多个应用领域。（2）2016 年 1 月，印度电子信息技术部资助 4500 万美元研制一款基于 RISC-V 指令集的 2GHz 四核处理器。（3）在印度政府支持的另一个关于神经形态加速器（neuromorphic accelerator）项目中，也将 RISC-V 作为计算主核心。过去几年，随着印度政府资助的处理器相关项目都开始向 RISC-V 靠拢，力出一孔，RISC-V 成为了印度的事实国家指令集。

7.4 以色列

2017 年以色列创新局（The Israel Innovation Authority）新成立名为 GenPro 工作组旨在开发基于 RISC-V 的快速、高效且独立处理平台，工作组主

要由来自学术界和工业界人士组成，计划为个体企业提供一整套软硬件解决方案，同时进一步发展前沿软硬件技术。

8 挑战、机遇与未来发展方向

8.1 面临挑战

开放指令集与开源芯片目前面临以下挑战：

- 开放指令集与开源芯片同样面临开发门槛高的问题；
- 开放指令集与开源芯片的稳定性及成熟度还需要通过硅验证及大量的市场应用来进行检验；
- 开放指令集与开源芯片会面临大量的商业指令集与商业芯片的竞争；
- 开放指令集与开源芯片还需要探索符合自身特点的商业模式，以便促进其健康发展；
- 公众对开放指令集与开源芯片的认识和重视程度还不够，一方面需要持续迭代开发来提供高质量的开放产品，另一方面还需要大力加强开放指令集与开放芯片的宣传，让更多的人参与进来。

8.2 机遇

开放指令集与开源芯片面临前所未有的历史性机遇。

首先，从市场需求角度来看，物联网、云计算以及 5G 的发展将会催生大量的 IoT 芯片应用需求，这些需求具有应用领域广泛、应用需求千变万化和使用量巨大的特点。除此之外，高速发展的人工智能技术还会推动这些应用需求进一步升级，对芯片的性能、功耗、灵活性提出更高的要求。传统芯片开发模式无论在开发的资金投入门槛、技术投入门槛以及产品开发复用性方面远远不能满足以上需求。在这种需求背景下，只有开放指令集与开源芯片这种全新的开发模式或者商业模式才能更好地满足要求，同时也将会极大推动物联网、云计算、人工智能以及 5G 应用的推广。可以设想在不久的将来会出现类似在互联网技术中大放异彩以及逐渐成为其发展基石的开源软件技术相对应的开放指令集与开源芯片，后者也将成为蓬勃发展的物联网、云计算、人工智能以及 5G 等的硬件基础，改变传统芯片的开发及应用模式。

其次，从技术发展的角度来看，目前已经完全具备发展开放指令集与开源芯片的技术条件。从芯片开发的整个流程来看，在各个关键点都已经有相应的

开放产品。其中尤为重要的是 RISC-V 这种开放指令集以及相应开发实现的出现和发展。除此之外，芯片开发中所需要的各种 EDA 工具也已经具有相应的开放版本，这些 EDA 软件目前虽然还达不到商用 EDA 软件的功能和性能水平，但对普通芯片开发已经足够，相信随着时间的推移以及参与的人越来越多，这些开源的 EDA 工具将会越来越成熟可靠。而且经过多年的发展，芯片开发中所需要的大部分 IP 也有相应的开源实现。

最后，芯片制造工艺的发展已经趋缓，其成本也会逐渐下降（如图 9 所示），这将会降低开源芯片的制造成本，缩小与商用芯片之间的工艺差距。

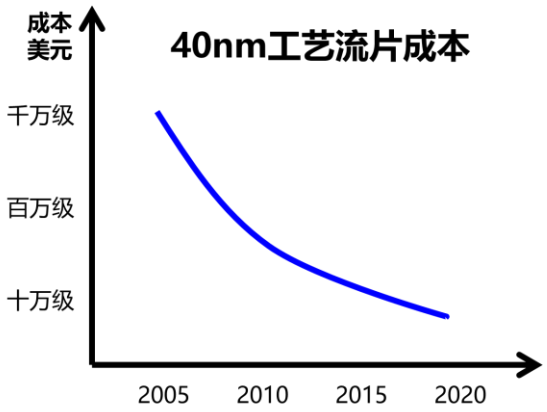


图 9. 成熟工艺制造成本会因摩尔定律放缓而自然下降

8.3 未来发展方向

开放指令集与开放芯片的主要发展方向有以下几点：

- 通过推动开放指令集、普及芯片敏捷开发模式、推广开源 EDA 工具链、开发云平台的应用、成熟工艺选择以及成熟 IP 积累等多个方向的努力来极大降低芯片开发门槛；
- 持续开发高性能、高质量的开源芯片，通过大量的硅验证及大量的市场应用持续检验开源芯片，促进其成熟发展；
- 探索并找到符合开放指令集与开放芯片特点的商业模式，形成可持续发展的模式；
- 通过宣传让更多的人了解开放指令集与开放芯片，让更多的人参与进来。

9 总结

在开源软件生态中，Linux 是整个生态的基石。基于 Linux，人们开发 Python、LLVM、GCC 等完整的工具链，创造 MySQL、Apache、Hadoop 等大量开源软件，实验各种创新思想与技术，逐渐形成一个价值超过 150 亿美元的开源软件生态。这对中国的互联网产业的意义尤为重大，不仅提升了 BAT 等互联网企业的技术研发能力，也大大降低了互联网产业创新的门槛，如今 3-5 位开发人员在几个月时间里就能快速开发出一个互联网应用。

在芯片设计领域，开源芯片越来越受到关注，涌现出 RISC-V、MIPS 开放指令集与开源处理器计划，它们有望像 Linux 那样成为计算机芯片与系统创新的基石。但是只有开放指令集又是远远不够的，更重要的是要形成一个基于开放指令集的开源芯片设计生态，包括开源处理器、开源工具链、开源 IP、开源 SoC 等等。

开源芯片还只是星星之火，却已展露出燎原之潜力。作为全世界最大的芯片用户，中国一直希望能把芯片产业做大做强，各方也都在努力。借鉴开源软件对于中国互联网产业发展的作用，开源芯片设计是一条值得尝试的道路。

致谢

本报告的编制得到中国开放指令生态（RISC-V）联盟理事长倪光南院士、咨询委员会专家的指导以及联盟成员的大力支持。

在此特别感谢（按单位拼音序）：

北京大学易江芳副教授，RISC-V 基金会中国顾问委员会主席方之熙博士，清华大学陈渝副教授，中国电子信息产业发展研究院集成电路研究所王世江所长，中科院计算所孙凝晖研究员、包云岗研究员、李华伟研究员、韩银和研究员、张科副研究员、唐丹高级工程师、常轶松助理研究员、王卅助理研究员、解壁伟助理研究员、赵然工程师、黄博文工程师、余子濠博士生、刘伯然博士生、颜志远博士生、刘志刚博士生、勾凌睿、李一苇、张旭、张亚杰，中科院上海微系统所郑云龙博士。