计 算 机 研 究 与 发 展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202221036
60(3): 476−493, 2023

香山开源高性能 RISC-V 处理器设计与实现

王凯帆[1,2] 徐易难[1,2] 余子濠[1] 唐 丹[1,3] 陈国凯[1,2] 陈 熙[1,2] 勾凌睿[1,2] 胡 轩[1,2] 金 越[1,2]
李乾若[1,2] 李 昕[1,2] 蔺嘉炜[1,2] 刘 彤[1] 刘志刚[1] 王华强[1,2] 王诲喆[1,2] 张传奇[1,2] 张发旺[5]
张林隽[1,2] 张紫飞[1,2] 张梓悦[1,2] 赵阳洋[1] 周耀阳[1,2] 邹江瑞[5] 蔡 晔[5] 郇丹丹[6] 李祖松[6]
赵继业[6] 何 伟[3,4] 孙凝晖[1,2,3] 包云岗[1,2]

[1](处理器芯片全国重点实验室(中国科学院计算技术研究所)北京 100190)
[2](中国科学院大学计算机科学与技术学院 北京 100049)
[3](北京开源芯片研究院 北京 100080)
[4](鹏城实验室 广东深圳 518055)
[5](深圳大学计算机与软件学院 广东深圳 518060)
[6](北京微核芯技术有限公司 北京 100190)
(wangkaifan@ict.ac.cn)

XiangShan Open-Source High Performance RISC-V Processor Design and Implementation

Wang Kaifan[1,2], Xu Yinan[1,2], Yu Zihao[1], Tang Dan[1,3], Chen Guokai[1,2], Chen Xi[1,2], Gou Lingrui[1,2], Hu Xuan[1,2], Jin Yue[1,2], Li Qianruo[1,2], Li Xin[1,2], Lin Jiawei[1,2], Liu Tong[1], Liu Zhigang[1], Wang Huaqiang[1,2], Wang Huizhe[1,2], Zhang Chuanqi[1,2], Zhang Fawang[5], Zhang Linjuan[1,2], Zhang Zifei[1,2], Zhang Ziyue[1,2], Zhao Yangyang[1], Zhou Yaoyang[1,2], Zou Jiangrui[5], Cai Ye[5], Huan Dandan[6], Li Zusong[6], Zhao Jiye[6], He Wei[3,4], Sun Ninghui[1,2,3], and Bao Yungang[1,2]

[1](*State Key Lab of Processors*(*Institute of Computing Technology*, *Chinese Academy of Sciences*), *Beijing* 100190)
[2](*School of Computer Science and Technology*, *University of Chinese Academy of Sciences*, *Beijing* 100049)
[3](*Beijing Institute of Open Source Chip*, *Beijing* 100080)
[4](*Peng Cheng Laboratory*, *Shenzhen*, *Guangdong* 518055)
[5](*College of Computer Science and Software Engineering*, *Shenzhen University*, *Shenzhen*, *Guangdong* 518060)
[6](*Beijing VCore Technology Co.*, *Ltd.*, *Beijing* 100190)

Abstract In recent years, the open-source instruction set architecture represented by RISC-V, has led the trend of open-source processors. However, the performance of these open-source processors is not enough to meet the needs of researchers in academia and developers in industry. To fill in the gap, we launch XiangShan project. XiangShan is an open-source high performance RISC-V processor. It features six-width-issue superscalar out-of-order design. XiangShan has earned over 3 200 stars and 400 forks on the famous open-source hosting platform GitHub, becoming one of the most popular open-source hardware projects in the world. The project also receives great support from companies and researchers all around the world. XiangShan project has been developed for over two years with two stable generations. The first generation called Yanqihu micro-architecture has been successfully taped out, and the performance of the real chip is in line with our expectation. The second generation called Nanhu micro-architecture

收稿日期: 2022-12-28; **修回日期**: 2023-02-17
基金项目: 中国科学院战略性先导科技专项(C 类)(XDC05030200); 国家自然科学基金重大项目(62090022)
This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences (XDC05030200) and the Major Program of the National Natural Science Foundation of China (62090022).
通信作者: 包云岗(baoyg@ict.ac.cn)

has entered the final optimization stage. It will be taped out soon. To the best of our knowledge, Nanhu micro-architecture achieves the highest performance among all open-source processors. This paper mainly introduces the first two generations of XiangShan processor, focusing on the design details and evolution of micro-architecture. The challenges and experiences during the development of XiangShan are discussed systematically.

Key words RISC-V; high performance processor; open source; chip design; agile development

摘 要 近年来以 RISC-V 为代表的开源指令集引领了开源处理器的设计潮流. 然而, 目前国内外的开源处理器性能尚未满足学术界和工业界的需求. 为填补空白, 香山处理器项目启动. 香山是一款开源高性能 RISC-V 处理器, 采用 6 发射超标量乱序执行设计, 目前在著名开源项目托管平台 GitHub 上获得超过 3 200 个星标（Star）, 形成超过 400 个分支（Fork）, 成为国际上最热门的开源硬件项目之一, 得到国内外企业和研究者的积极支持. 香山处理器在近两年时间中历经两代版本演进, 第一代“雁栖湖”微架构已经成功流片, 回片性能符合预期; 第二代“南湖”微架构已进入最后的优化迭代阶段, 即将投片, 据已知消息, 其仿真评估性能在当前开源处理器中排名第一. 主要讨论香山前两代微架构的实现细节与设计演进, 并系统介绍开发香山过程中的各类挑战与经验.

关键词 RISC-V; 高性能处理器; 开源; 芯片设计; 敏捷开发

中图法分类号 TP302

处理器芯片是驱动信息时代发展的核心引擎. 随着大数据和物联网时代的来临, 各行业对数据处理的需求越来越高. 然而, 近年来摩尔定律濒临终结[1], 先进制造工艺节点研发成本上升, 昂贵的流片费用使得芯片开发成本也相应提升, 投入风险越来越高. 如今, 研发一款通用处理器芯片往往需要投入上百人年, 耗费上亿元研发经费, 只有少数大企业才有能力开发处理器芯片. 因此, 业界正在寻求降低芯片开发门槛的方案.

开源芯片为上述问题提供了一个方向, 成为近年来业界关注的新热点. 基于开源模式, 多方可联合开发共性技术, 构建支持芯片开发的基础设施, 从而降低开发门槛. 2011 年, 由加州大学伯克利分校发布的新型指令集 RISC-V[2] 首先实现了指令集层面的开源. 区别于常见的 x86 和 ARM 等指令集, RISC-V 的特点是开放和自由, 由非营利的 RISC-V 国际基金会维护, 任何人无需获得授权即可免费使用. 近年来, 基于 RISC-V 指令集的开源硬件不断涌现, 包括加州大学伯克利分校的 Rocket-Chip[3] 和 BOOM[4]、中国科学院大学和中国科学院计算技术研究所的 NutShell[5] 等. 然而, 不少学者和企业反映, 国内外现有的开源 RISC-V 处理器项目尚未满足学术界和工业界的高性能需求.

香山处理器项目于 2020 年 6 月启动. 香山是一款开源高性能 RISC-V 处理器, 以 Linux 在操作系统领域的影响力为目标, 期望建立一个在学术界和工业界广泛应用的体系结构创新开源平台. 2021 年 6 月 22 日, 香山处理器在第一届 RISC-V 中国峰会首次公开亮相, 引起各界广泛关注. 随后香山稳步发展, 第一代“雁栖湖”微架构成功流片, 性能达到预期, SPEC CPU2006 实测分值为 7.01 分/GHz, 同频性能达到 ARM Cortex A73 水平. 第二代“南湖”微架构正在进行最后的迭代优化, 即将流片. 仿真评估结果显示, “南湖”微架构 SPEC CPU2006 分值约为 10.42 分/GHz, 同频性能达到 ARM Cortex A76 水平, 单核性能为当前开源处理器之最, 目前已在企业支持下开展产品化改造工作. 第三代“昆明湖”微架构将持续优化性能, 目前得到众多企业的支持, 已建立联合开发团队启动架构设计工作.

香山项目采用新的开源模式, 开源内容不仅包括所有源代码和设计文档, 还包括支持香山开发的各类基础设施和工具. 因此, 香山项目能够成为联合企业开发的基础平台, 赋能多方合作.

本文主要介绍香山处理器前两代微架构的实现细节与设计演进, 并系统阐述实现开源高性能处理器过程中面临的各种挑战和汲取的经验, 内容包括微架构性能调优、功能验证、物理实现和多方协同开发等. 其中, 第 1 节介绍香山处理器的微架构总览; 第 2~5 节分别介绍香山处理器前端供指单元、乱序执行单元、访存子系统和缓存子系统的设计考量与微结构细节; 第 6 节介绍香山处理器的敏捷开发验证方法与性能评估结果; 第 7 节介绍香山处理器的后端物理实现; 第 8 节介绍香山联合开发模式与下一代展望.

1 香山设计总览

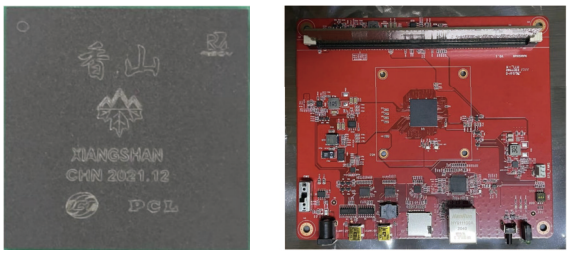
香山是一款超标量乱序执行的 RISC-V 通用处理器，最新版架构支持 RV64GCBK 指令集，兼容 RISC-V 软件工具链与生态. 作为一款开源处理器，香山使用 Chisel[6] 硬件描述语言实现，设计代码约 6 万行，验证代码约 3 万行. 香山的微架构设计经历了两代，分别为“雁栖湖”微架构和“南湖”微架构. 两者的参数如表 1 所示. 香山实现了灵活可配的参数系统，可以根据给定的时序、面积和功耗约束配置大多数设计参数.

Table 1 Tape-out Parameters Comparison of the Two Generations of XiangShan Processor

表 1 两代香山处理器流片参数对比

参数	“雁栖湖”微架构	“南湖”微架构
指令集	RV64GC	RV64GCBK
流片工艺/nm	28	14
频率/GHz	1.3	2
处理器核数	1	2
取指宽度	8×4B/周期	8×4B/周期
译码与重命名宽度	6	6
ROB/LQ/SQ 项数	192/64/48	256/80/64
整型/浮点物理寄存器堆项数	160/160	192/192
uBTB 项数	32	32
BTB 项数	2 048	4 096
其他分支预测器	TAGE-SC-L, RAS	TAGE-SC, RAS, ITTAGE
L1 指令缓存	16KB, 4 路	64KB, 4 路
L1 数据缓存	32KB, 8 路	64KB, 4 路
L1+ 缓存	128KB, 8 路	
L2 缓存	1MB, 8 路, 包含式	1MB, 8 路, 非包含式
L3 缓存		6MB, 6 路, 非包含式
ITLB 项数	32+4	32+4
DTLB 项数	32+4	64+16
共享 TLB 项数	4 096	2 048

2021 年 7 月，“雁栖湖”微架构首次以 28 nm 工艺投片，于次年 1 月回片并成功点亮，最高工作频率为 1.3GHz. 性能测试负载采用基准程序集 SPEC CPU2006, 以每 GHz 分数为性能指标，该指标与处理器的 IPC(instruction per cycle)成正比. 实测数据表明，第一代芯片运行 SPECint 2006 达到 7.03 分/GHz, 运行 SPECfp 2006 达到 7.00 分/GHz, 符合设计预期. 图 1 为香山第一代芯片与配套板卡.



(a) 第一代芯片 (b) 第一代板卡

Fig. 1 Chip and board of the first generation of XiangShan processor

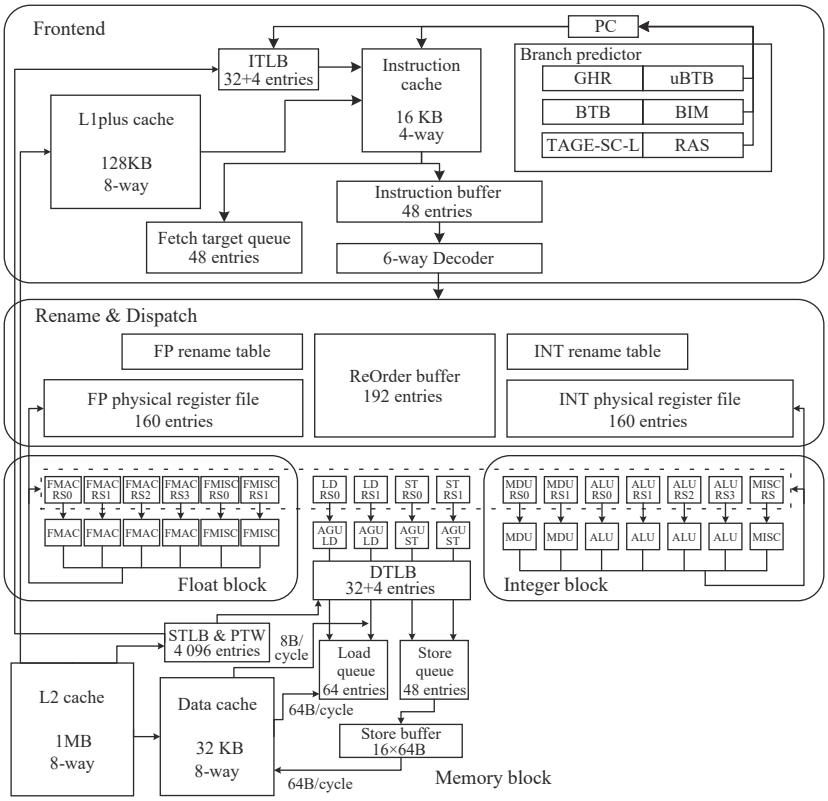
图 1 香山处理器第一代芯片与板卡

香山第一代“雁栖湖”微架构如图 2(a)所示. 前端供指单元采用 4 级流水线紧耦合设计，每周期最多可以供给 8×4B 的指令流. 分支预测采用多级前向覆盖机制，实现了简化版的 TAGE-SC-L 分支预测器. 指令从指令缓冲部件取出后分别流经译码级、重命名级和分派级，每级均可并行处理最多 6 条指令. 香山实现了分离的通用寄存器堆和浮点寄存器堆，在进入保留站之前读取数据. 保留站是完全分布式的，单个容量为 16 项，支持高效的乱序发射与执行. 在访存子系统中，香山包含 2 条 load 流水线和 2 条 store 流水线，配备高性能内存管理单元，支持乱序访存. 流片版本采用单核设计，配置 L2 缓存，并通过 L1+ 缓存缓解 L1 指令缓存容量小导致的性能损失.

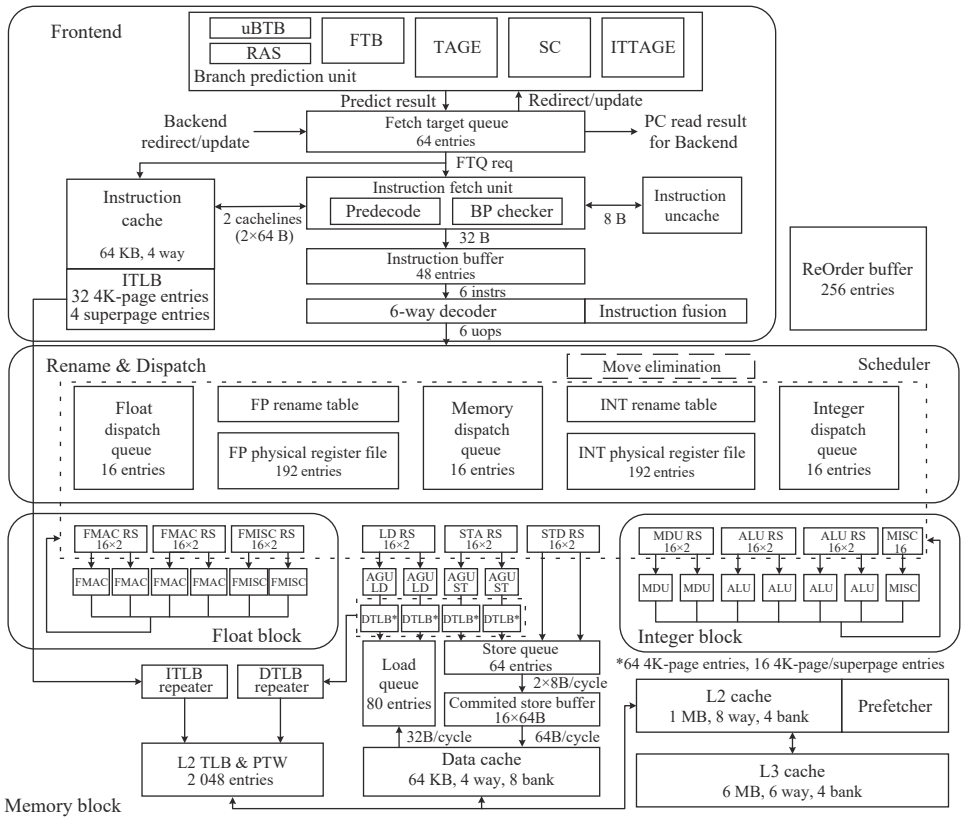
香山第二代“南湖”微架构如图 2(b)所示，它相比第一代“雁栖湖”微架构的主要变化包括：1) 前端设计采用解耦的取指单元和分支预测架构，提高指令供给吞吐和分支预测准确率；2) 功能部件新增 RISC-V 位操作(RVB)和标量加密运算(RVK)指令集扩展，并实现了兼容 IEEE-754 标准的高性能浮点运算部件；3) 访存子系统新增 RISC-V 物理内存保护和自定义的可配置物理内存属性，并优化了访存流水线结构；4) 缓存子系统实现了非包含式的高性能 L2/L3 缓存，提升并发度并降低延迟. 流片版本采用双核、配备 3 级缓存.

在两代微架构的设计迭代中，香山团队深刻体会到高性能处理器的设计是一门平衡的艺术，具体体现在 2 个维度：

1)宏观上. 微架构设计需要平衡前端指令供给、后端乱序执行与访存数据供给 3 个方面，任何一方的性能短板都会因木桶效应导致整体性能下降，因此盲目增大配置参数是不可取的，需要针对性能瓶颈优化才能提升整体性能.



(a)“雁栖湖”微架构



(b)“南湖”微架构

Fig. 2 Micro-architecture of the two generations of XiangShan processor

图 2 香山处理器两代微架构

2)微观上. 流水线设计需要平衡每一级的功能和时序. 处理器性能由 IPC 和频率共同决定, 流水级设计不平衡会导致频率下降从而影响整体性能. 因此需要预先评估设计复杂度, 规划各级逻辑功能并与后端团队紧密协作调整时序.

实践表明, 高性能处理器的微架构性能优化过程复杂而漫长, 可总结为 2 条原则: 一是挖掘硬件结构的并行度, 包括指令级并行、访存级并行等; 二是优化设计和算法以降低串行任务的延迟, 例如设计更流畅的数据通路、采用投机与推测等. 上述 2 条原则均在香山的设计与演进中得以体现.

2 前端供指单元

前端供指单元由取指单元和分支预测器 2 部分组成, 负责从内存中取指令以供给后端执行, 并基于分支预测结果实现指令流跳转. 超标量处理器每周期可提交多条指令, 故要求前端供指单元具备并行的高吞吐取指和分支预测能力. 为此, 第一代“雁栖湖”微架构采用紧耦合前端设计.

2.1 紧耦合前端设计

“雁栖湖”微架构的流水线以前端供指单元为起点, 每周期最多能取 8 条 4B 的普通指令或 16 条 2B 的压缩指令. 前端供指单元分为 4 个流水级(F1~F4):

1) F1 级. 选择下一个新取指 PC(next-PC)的来源, 包括指令序顺序的下一行、前端冲刷请求和后端冲刷请求. 同时并行访问 L1 指令缓存和多级分支预测器.

2) F2 级. 单周期延迟预测器返回预测结果, 包括跳转方向和目标地址.

3) F3 级. L1 指令缓存返回 32B 指令行, 将其拆分为指令并进行预译码; 2 周期延迟预测器返回预测结果.

4) F4 级. 根据 F3 锁存的预译码信息计算 jal 指令和条件分支指令(如 bne 指令)的目标地址, 判断预测结果是否正确; 3 周期延迟预测器返回预测结果, 并选出第一条预测结果为“跳转”的指令, 将其目标地址作为后续取指 PC.

前端供指单元的 F4 级会将打包的指令发送给指令缓冲部件. 指令缓冲部件用于解决处理器前端指令供给和后端执行速度的不平衡问题. 由于前端最大取指宽度大于译码宽度, 大多时候指令缓冲部件中的指令是充盈的, 因此在 L1 指令缓存发生缺失或因前端冲刷而产生气泡时, 指令缓冲部件仍可向后端供给一定数量的指令.

分支预测器是前端的重要部件. 处理器在执行过程中, 分支预测错误需要冲刷流水线, 清除错误路径上的指令, 故分支预测准确率对性能影响很大. 然而, 分支预测存在延迟和准确率之间的矛盾, 越准确的预测器往往越复杂, 产生预测结果需要耗费越多的周期数, 从而引入额外的前端气泡, 降低供指效率. 为解决该问题, 香山采用前向覆盖方法来预测分支方向和跳转目标, 其思想是部署简单和复杂的子预测器进行并行预测, 先投机采用前者的预测结果, 再在后续流水级中使用后者的结果矫正前者. 仅当简单子预测器预测错误时, 该方法才会暴露复杂子预测器的实际延迟, 且有效降低了需要流水线后端发起冲刷的次数. “雁栖湖”微架构的分支预测器分为 3 级, 如图 3 所示. 每一级的预测结果都会与前一级进行对比, 若不一致, 则采用后级的预测结果, 并冲刷前几级流水级, 同时更新 PC.

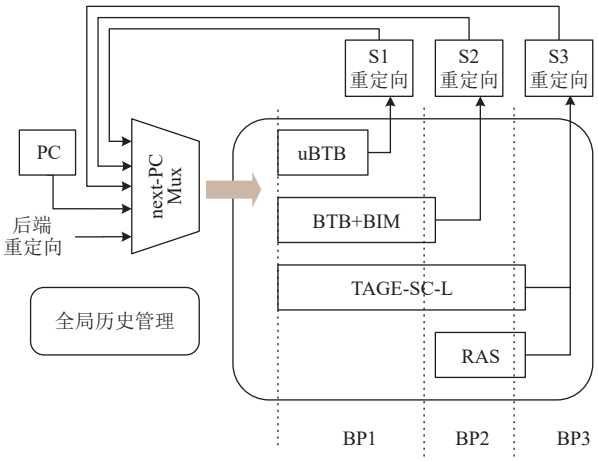


Fig. 3 Branch predictor in Yanqihu micro-architecture

图 3 “雁栖湖”微架构分支预测器

“雁栖湖”微架构的分支预测器主要包含 5 个子预测器:

1) uBTB. uBTB 是一个使用寄存器实现、单周期产生预测结果的小容量分支目标缓冲器(branch target buffer, BTB), 可提供无气泡的预测流. uBTB 采用全相联结构, 通过 PC 的低位索引, 所得结果与 PC 的高位拼接得到目标地址. 此外, uBTB 通过两位饱和计数器来预测条件分支指令的跳转方向.

2) BTB+BIM. BTB 和 BIM 分别是容量更大的分支目标缓冲器和更准确的跳转方向预测器, 均使用 SRAM 实现, 2 周期产生预测结果. BTB 和 BIM 采用 2 路组相联结构, 存储内容与 uBTB 类似.

3) TAGE-SC-L. TAGE[7] 是目前准确率最高的分支预测算法之一. “雁栖湖”微架构实现了该算法的

简化版，延迟为 3 周期. 预测器使用 64 b 分支历史和指令 PC 索引，利用 6 个历史表获取预测分支方向的初步结果，该结果经由统计纠正器（statistical corrector, SC）部件纠正. 此外，内部还设有循环预测器（loop predictor）部件预测循环退出条件，最终综合产生方向预测结果.

4）RAS. 返回地址堆栈（return address stack, RAS）会在执行函数调用指令时将 PC 压栈，然后使用栈顶结果预测函数返回指令的跳转 PC. RAS 单周期即可返回预测结果，但需要得到预译码信息后才能预测.

2.2 “南湖”微架构解耦前端设计

紧耦合前端设计虽然实现了超标量取指与多级分支预测的结合，但前端冲刷将给流水线引入大量气泡，降低供指吞吐. 为解决该问题，“南湖”微架构的最大改变是采用解耦前端（decoupled frontend）设计，即从流水线逻辑的角度将分支预测器与取指单元解耦，将两者组织为生产者-消费者关系，如图 4 所示. 该设计允许分支预测器在取指单元阻塞时继续工作，预测更超前的指令流，从而隐藏更多取指气泡，有效提升取指带宽，对“南湖”微架构的性能提升贡献很大.

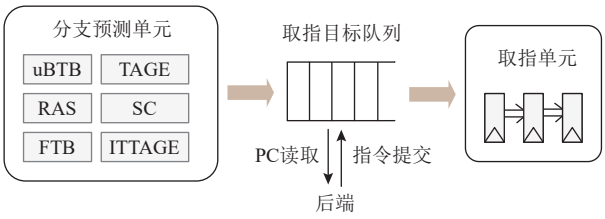


Fig. 4 Structure of decoupled frontend

图 4 解耦前端结构

基于解耦前端设计，“南湖”微架构引入新模块取指目标队列（fetch target queue, FTQ），作为分支预测器与取指单元之间的缓冲. 取指单元会从 FTQ 中读出待取指令块的信息，向指令缓存发送读取请求. 缓存返回结果后，会根据指令块范围进行指令切分和预译码. 预译码能检查一些简单的分支预测结果，例如是否将无条件跳转指令预测为不跳转等，相关信息会提早反馈给 FTQ 用于更新 PC. 同时，FTQ 中的指令流信息还可以用于指导指令缓存的预取，从而降低取指延迟.

此外，“南湖”微架构还全面优化了分支预测器，包括提升部件规模、降低预测器延迟，以及优化预测机制. BTB 部件被替换为以预测块为处理单位的取指目标缓冲（fetch target buffer, FTB），此处预测块指长度不超过 32B 且最多包含 2 条分支指令的指令块. FTB 不仅记录了预测块中的分支预测信息和起始地址预测信息，同时还预测该预测块的结束地址. 新增的间接跳转预测器采用了目前已知最先进的 ITTAGE 算法[7]，使每千条指令中间接跳转指令预测错误次数大幅降低. 更多变化细节如表 2 所示. 实验结果显示，针对 SPEC CPU2006，解耦前端设计使取指气泡平均降低约 50%，每千条指令中条件分支指令预测错误次数降低 20% 以上.

Table 2 Frontend Feature Evolution of the Two Generations of XiangShan Processor

表 2 两代香山处理器前端特性演进

功能点	“雁栖湖”微架构	“南湖”微架构	提升
BTB（FTB）	2 048 条分支, 2 路	4 096 条分支, 4 路	2x
TAGE 延迟	3 周期	2 周期	1.5x
分支历史	压缩, 64 b	准确, 119 b	1.85x
间接跳转预测		ITTAGE	新增
RAS	16 项	32 项	2x
FTQ	48 项	64 项	1.33x

3 乱序执行单元

乱序执行单元是处理器执行指令的核心，主要包含调度和运算两方面. 高性能处理器通过寄存器重命名技术消除“读后写”和“写后写”两种伪依赖，并允许在指令窗口中动态乱序调度，避免执行时间长的老指令阻塞处理器，从而提升指令执行的吞吐. 这部分的设计难点是如何在满足面积与时序约束的条件下尽可能提升指令执行的并行度，同时降低计算延迟.

3.1 “雁栖湖”微架构基础设计

基于物理实现的考量，香山将乱序执行单元划分为 3 个模块，包括控制模块、整型模块和浮点模块. 控制模块包含译码、重命名和分派等逻辑，如图 5 所示.

首先，译码阶段接收前端指令缓冲发出的指令，将其译码为结构化信息并送入重命名阶段. 香山采用统一物理寄存器堆重命名方式，通过映射表维护体系结构寄存器堆和物理寄存器堆之间的关系，每周期可分别分配和释放最多 6 个物理寄存器. 此外，还可以通过改变寄存器映射关系来消除 move 指令，实现零延迟执行.

分派阶段分 2 个流水级：第 1 级将重命名后的指令按照类型分别送入整型、浮点和访存分派队列中；第 2 级将指令按照具体操作类型分发到各个保留站中. 此外，分派阶段也会进行部分重命名工作，以缓

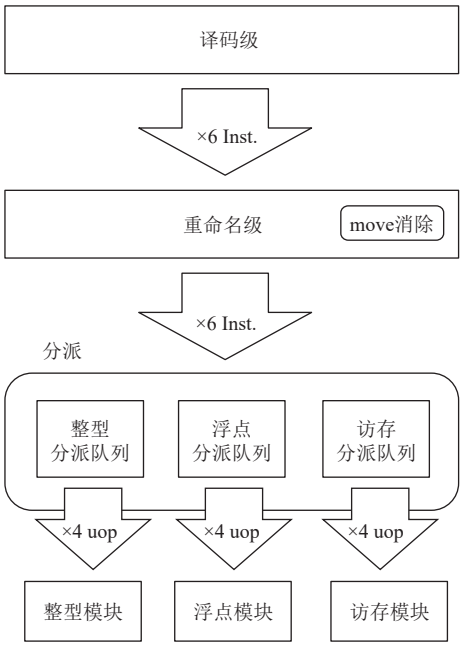


Fig. 5 Control module of out-of-order execution unit

图 5 乱序执行单元控制模块

解重命名阶段的时序压力.

“雁栖湖”微架构采用发射前读寄存器堆和完全分布式的保留站，以降低设计复杂度. 考虑物理寄存器堆最大读写口数量的限制，整型模块中包含 7 个保留站，与功能部件一一对应. 功能部件包含 4 个 ALU、2 个 MDU 和 1 个 Misc 部件. ALU 部件负责指令基本算术和逻辑运算，可提前唤醒其他保留站，实现背靠背的指令执行；MDU 部件负责执行乘除法运算，其中乘法采用 3 级流水的华莱士树实现，除法采用 SRT4 算法；Misc 部件负责执行跳转指令和 CSR 指令. 考虑到面积和时序约束，每个保留站有 16 项.

浮点模块的保留站组织结构与整型模块类似，包含 6 个保留站，分别对应 4 个 FMAC 和 2 个 FMISC 部件. 其中 FMAC 部件负责执行浮点乘法、加法和乘累加，延迟为固定 4 周期；FMISC 部件负责执行除法、开方等运算，均采用 SRT4 算法. “雁栖湖”微架构的浮点功能部件基于 Hardfloat[8] 开源实现进行了少量改进和时序优化，浮点寄存器堆均采用 Recode 格式存储浮点数. Recode 格式将浮点数的指数扩展 1 位，从而简化浮点部件的运算和舍入逻辑. 但该格式扩展位数的做法不利于向量部件的添加，因此“南湖”微架构不再采用.

为实现指令按序提交以支持精确异常，所有指令会在分派阶段并行送入重排序缓冲(re-order buffer, ROB)部件以按程序顺序记录指令流. 当指令经由保留站发射进入功能部件并执行完毕后，运算结果经过写回模块的仲裁器写回 ROB，并标记指令执行完成等待提交. ROB 每周期可接收来自分派阶段最多 6 条指令，并提交最多 6 条指令. 当分支预测错误或触发异常中断时，ROB 会标记出错指令，然后冲刷流水线，并通知前端供指单元根据新 PC 重新取指. 同时，ROB 从尾部开始回滚状态，并根据其中记录的重命名信息恢复重命名映射表.

3.2 “南湖”微架构的演进

“南湖”微架构整体沿用“雁栖湖”微架构中的乱序执行单元框架，同时通过若干优化提升整体性能.

首先，“南湖”微架构通过新增指令集扩展和指令融合特性来降低处理器的动态指令数，从而减少相同程序的执行时间. “南湖”微架构实现了 RISC-V 规范的 RVB 扩展和 RVK 扩展，不仅使部分程序的动态指令数降低 10% 以上，还能减少分支指令数量，降低误预测造成的整体性能损失. 此外，“南湖”微架构根据大量运算负载中总结出的常见指令组合添加了指令融合特性，提升了各个队列的存储效率，增大了乱序执行的指令窗口.

其次，“南湖”微架构优化了功能部件的运算延迟. “南湖”微架构实现了兼容 IEEE 754 标准的高性能浮点计算单元，使用双通路浮点加法器和级联式浮点乘加单元，降低了浮点乘法和加法的延迟. 此外，“南湖”微架构将整型和浮点的除法算法升级为 SRT16，除法运算的延迟大多时候可降低一半.

最后，“南湖”微架构还优化了发射单元的时序，使用最老优先的发射策略，同时合并了部分保留站，提升其利用率. 这些改进在满足指令背靠背执行的前提下有效提升了乱序执行单元的效率与时序表现.

4 访存子系统

访存子系统负责处理保留站发出的 load 和 store 请求，与乱序执行单元耦合，对缓解处理器访存瓶颈、提高数据供给效率十分关键. 访存子系统的设计挑战包括如何提升访存并行度、如何设计平衡的访存流水线等.

香山访存子系统支持乱序访存，符合 RVWMO (RISC-V weak memory ordering)内存一致性模型，包含 2 条 load 流水线和 2 条 store 流水线，可支持最多 4 条访存指令并行执行.

4.1 乱序访存机制

乱序访存机制指尽早发射没有依赖的访存指令，从而提升访存并行度，但仍要求访存指令按序提交.

为实现该机制，现代处理器大都通过专用访存队列，包括读取队列（load queue）和存储队列（store queue），记录访存指令的顺序，同时检查访存指令之间的数据依赖关系，若违背内存一致性，则回滚处理器状态. 为正确实现 RVWMO 一致性模型，需要检查访存违例，即确认访存地址相同的 load-load 指令对和 store-load 指令对必须按序执行，访存地址不同的访存指令对则可乱序执行. 香山处理器借助访存队列检查访存违例，通过流水级冲刷机制让处理器从违背依赖关系的访存指令开始重新执行.

访存队列大小决定了访存并行度的上限. “雁栖湖”微架构的 load queue 为一个 64 项的循环队列. 每周期从分派阶段接收最多 6 条 load 指令，从 2 条 load 流水线接收指令的执行结果，每周期最多写回 2 条发生缓存缺失并重填后的指令，写回操作会与正常的访存流水线共享写回端口. 当数据缓存发生重填时，会将重填的整个缓存行广播到 load queue 的所有项，从而唤醒等待该缓存行数据的访存指令. 类似地，store queue 共 48 项，每周期接收最多 6 条 store 指令，从 2 条 store 流水线中接收指令的执行结果，并最后写回.

store 指令的最后一步是将数据写回数据缓存. 但若每次提交 store 指令时均占用数据缓存的写端口，则会严重阻塞后端执行. 为缓解该问题，香山设计了 committed store buffer 部件，用于记录已提交但未写回数据缓存的信息，作用类似 L0 缓存，并以缓存行粒度合并 store queue 传来的数据. 缓冲项数超过阈值后，会使用 PLRU 替换算法选出一项进行换出，每周期至多向数据缓存写入 1 个缓存行. 这一设计符合 RVWMO 一致性模型，将数据核间同步的需求转移给软件同步指令以提升访存性能.

为减少访存违例导致的流水线冲刷与回滚，提升乱序访存性能，“雁栖湖”微架构添加了基于 load wait table[9] 的访存违例预测器，当预测到一条 load 指令可能违背依赖关系时，保留站将阻塞该 load 指令的发射，直到比它年老的所有 store 指令都发射后才能解除阻塞.

4.2 访存流水线

当访存指令从保留站发射后，就会进入 load 或 store 流水线执行，如图 6 所示.

为平衡各级时序，“雁栖湖”微架构 load 流水线分为 3 级，并通过前递机制解决依赖：1）第 1 级计算虚拟地址，访问 TLB 进行虚实地址转换，同时用虚拟地址索引数据缓存；2）第 2 级收到 TLB 翻译后的物

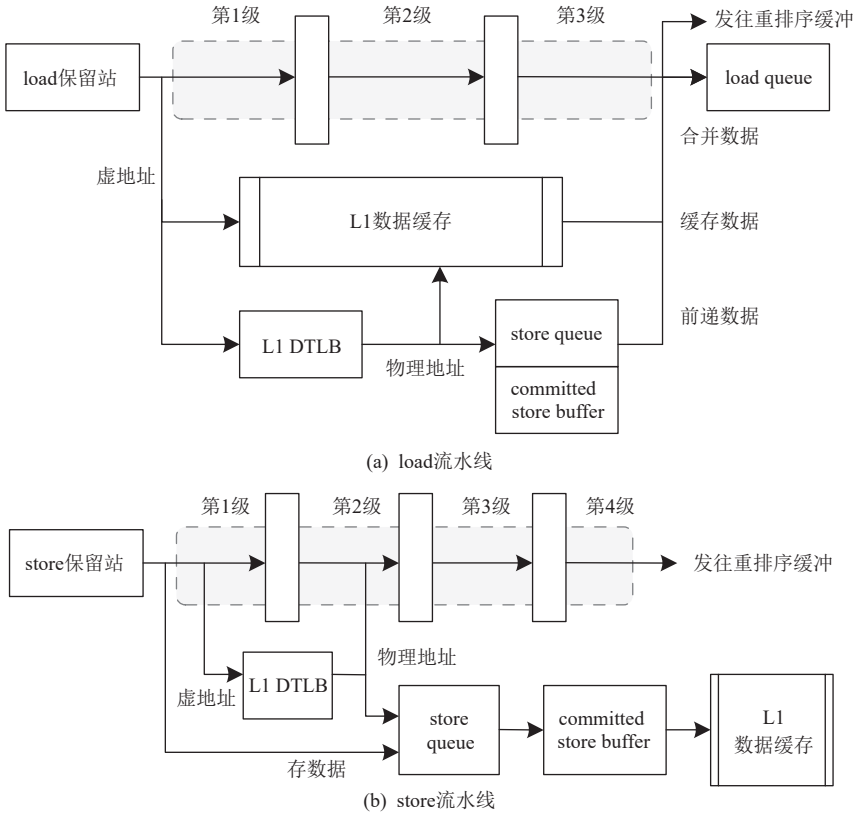


Fig. 6 load/store pipe of Yanqihu micro-architecture

图 6 “雁栖湖”微架构访存流水线

理地址,将其送入数据缓存比较标签,同时送入 store queue,查询是否需要将 store 结果前递到 load,此外还会产生数据缓存的命中信息,反馈至保留站唤醒后续指令;3) 第 3 级根据数据缓存的读取结果和 store queue 的前递结果选择实际读取的数据,将其写回到公共数据总线或者浮点模块,同时更新 load queue 中对应项的状态.

store 流水线略有不同,它分为 4 级流水:第 1 级与 load 流水线类似,计算地址并访问 TLB 和数据缓存;第 2 级则会将物理地址写入 store queue,同时检查访存违例;检查操作持续第 3 级和第 4 级流水,并将检查结果通知 ROB.

4.3 内存管理单元

内存管理单元(memory management unit, MMU)是访存子系统的重要部分,负责将虚拟地址翻译到物理地址并检查页权限,以支撑现代操作系统的页式内存管理. 香山处理器使用硬件遍历页表,实现了 RISC-V 规范的 Sv39 分页机制,支持 3 种不同大小(4KB, 2MB, 1GB)的页. 此外,香山还实现了物理内存属性(physical memory attribution, PMA),用于检查访存操作的权限.

地址翻译位于访存的数据通路中. 为了提升地址翻译效率,香山利用访存局部性原理,使用多级 TLB 缓存页表项. “雁栖湖”微架构的 L1 TLB 包括 ITLB 和 DTLB,分别与指令缓存和数据缓存的流水线耦合,均采用全相联结构,包含 32 项 4 KB 页和 4 项 2 MB/1 GB 页. L1 TLB 缺失时,会向 L2 TLB 发送请求. L2 TLB 是更大的页表项缓存,由 ITLB 和 DTLB 共享,缓存了 Sv39 机制中三级页表的页表项. L2 TLB 缺失时,将触发页表遍历器(page table walker, PTW)访问内存中的页表项. 由于两级 TLB 之间的物理布局距离较远,故在两者中间新增了 repeater 部件,用于缓冲请求并过滤重复请求,避免 L1 TLB 中出现重复项而降低命中率.

4.4 “南湖”微架构的改进

除增大访存队列外,“南湖”微架构对访存子系统的改进还包括增强 MMU 和优化访存流水线.

对于 MMU,“南湖”微架构首先优化性能关键的 DTLB,采用 16 项全相联和 64 项直接映射并行查询的设计,增大了有效容量,替换算法也从随机升级为 PLRU. 考虑面积和时序,“南湖”微架构缩减了 L2 TLB 的容量. 为弥补容量缩减带来的性能影响并提升 PTW 访存效率,“南湖”微架构将 PTW 部件拆分为 2 个状态机,一个负责访问前 2 级页表,另一个负责访问末级页表,并将后者的访存并行度从 1 提升至 8. 此外,“南湖”微架构还增强了PMA 部件的可编程性,新增兼容 RISC-V 规范的物理内存保护(physical memory protection, PMP)部件,为软件提供内存保护的抽象,使操作系统可与硬件协同提升香山的安全性.

对于访存流水线,“南湖”微架构将 store 指令的地址与数据计算分离,将“雁栖湖”微架构中 2 条 4 级 store 流水线拆分为 2 条 2 级存储地址流水线和 2 条 2 级存储数据流水线. 该设计使保留站不必等待 store 指令的地址和数据皆就绪才发射,提升了指令执行并行度,同时更有效地规避访存违例. “南湖”微架构的访存违例预测算法采用 store set[10],提升了预测准确率.

“雁栖湖”微架构中的 load 流水线是时序瓶颈之一,为提升主频,“南湖”微架构将 load 流水线的长度从 3 级调整为 4 级,并通过若干优化弥补由此带来的性能损失,其中有 2 个关键优化点:

1) load to load 优化. 指针追逐是程序中常见的访存模式,在指令中表现为将前一条 load 指令的读出结果作为后一条 load 指令的访存地址. 为优化该场景,经观察,一段时间内 load 指令的有效地址通常对应相同的缓存索引,因此“南湖”微架构推测访问数据缓存的标签. 若推测成功,则 load to load 的延迟可从 4 周期降为 3 周期.

2) load 前递机制优化. “南湖”微架构推测地使用虚拟地址比较前递时的地址,再使用物理地址判断前递结果是否正确. 该机制可在地址比较的数据通路中排除 DTLB 地址转换带来的延迟,优化关键路径的时序. 若物理地址检查失败,则清空访存流水线并重新执行. 统计结果显示,虚实地址前递结果不匹配的概率很低.

5 缓存子系统

香山处理器充分挖掘访存的时间与空间局部性,设计多级缓存以提升访存效率.

5.1 香山缓存结构

考虑面积约束,“雁栖湖”微架构采用单核 2 级缓存结构. L1 指令缓存和 L1 数据缓存均采用虚拟索引物理标签(virtual index physical tag, VIPT)模式,两者共享 L2 缓存. L1 指令缓存大小为 16 KB, 4 路组相联;L1 数据缓存大小为 32 KB, 8 路组相联. 为弥补指令缓存容量偏小带来的性能影响,“雁栖湖”微架构

额外设计了一个 L1+缓存，位于 L1 指令缓存与 L2 缓存之间，大小为 128 KB.

“雁栖湖”微架构 L2 缓存大小为 1 MB，8 路组相联，采用包含关系，硬件不维护指令缓存的一致性，由软件显式执行 Fence.i 指令来维护. “雁栖湖”微架构的 L2 缓存基于 InclusiveCache 开源项目[11]进行若干功能增强和时序调整，以满足时序约束.

“南湖”微架构采用双核 3 级缓存结构. 每个核心的 L1 指令缓存与 L1 数据缓存共享其私有的 1 MB L2 缓存，双核共享一个 6 MB、6 路组相联的 L3 缓存，硬件维护核间数据一致性. 此外，“南湖”微架构还支持指令缓存和数据缓存之间的一致性，软件执行 Fence.i 指令后无需冲刷指令缓存. “南湖”微架构重新设计了 L2 缓存和 L3 缓存的内部结构，这也是“南湖”微架构的缓存子系统相对“雁栖湖”微架构改动最大之处.

5.2 “南湖”微架构 HuanCun 缓存设计

“雁栖湖”微架构使用的 InclusiveCache 虽然设计精巧，但存在若干缺陷：1) 不支持下一级缓存的一致性，即无法处理下游发送的 snoop 请求；2) 频率难以提升，受其结构限制较大；3) 配置不够灵活，仅支持包含式的缓存结构，在处理器核较多、缓存结构复杂的场景下，采用包含式会损失较多有效容量. 为提升缓存性能和缓解香山处理器的访存瓶颈，“南湖”微架构开发了代号为 HuanCun 的缓存模块.

HuanCun 基于 TileLink 总线开发，使用目录维护一致性，支持包含式和非包含式，以及多种替换策略. 在流片版本中，“南湖”微架构的 L2 和 L3 缓存均采用非包含式和 PLRU 替换策略，以最大化有效容量.

HuanCun 总体结构分为目录、数据存储、未命中状态处理寄存器（miss-status handling registers, MSHR）、通道控制器和预取引擎 5 部分，如图 7 所示. 目录模块负责维护缓存元数据，同时保存上层数据与本层数据的元数据，包括权限、脏位等. 数据存储模块负责存储具体数据，可通过参数配置 bank 数量从而提升读写并行度. MSHR 是非阻塞缓存的核心部件，负责处理具体的请求控制逻辑，包含多组状态机以维护一致性. MSHR 的项数决定了缓存最大的处理并行度；通道控制模块负责与标准 TileLink 总线接口交互，包括将外部请求转换为缓存内部信号，以及将缓存内部请求转换为 TileLink 请求向外发送. “南湖”微架构的预取引擎实现了高性能的 BOP 算法和 SMS 算法，有效提升缓存命中率. 此外，HuanCun 可将请求地址的低位作为索引分片（slice）以提升并行度，每个分片的逻辑相互独立，负责具体的任务管理.

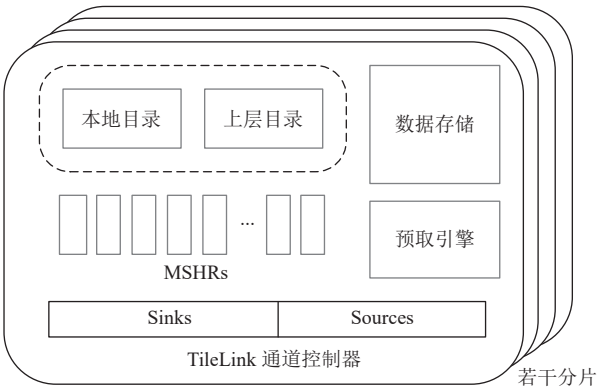


Fig. 7 Micro-architecture of HuanCun

图 7 HuanCun 微结构

HuanCun 的总体工作流程为：1）通道控制模块接受 TileLink 请求，将其转换为缓存内部请求；2) 仲裁和分配逻辑为内部请求分配 1 个 MSHR；3) MSHR 根据不同请求类型发起各类任务，包括数据读写、向上下层缓存发送新请求或返回响应、更新预取器状态等；4) 当 MSHR 完成一个请求的全部操作时，MSHR 将被释放，等待接收新请求. HuanCun 的设计存在不少挑战：

1）死锁问题

由于请求独占 MSHR 且 MSHR 资源有限，故在复杂的多级缓存子系统中，总线请求间容易造成循环等待，导致死锁. 为规避死锁，HuanCun 采取的设计包括更改请求分配策略，引入仅处理嵌套请求的 MSHR，以及引入请求的打断和跨层处理机制.

2）状态爆炸问题

非包含式缓存的状态空间很大，需处理多种异步的总线请求，以及请求打断与跨层处理等机制. 为此，HuanCun 摈弃了传统的状态机建模方式，而是采用若干标志寄存器联合表示状态. 同时，构造合理的抽象，尽可能将请求处理的状态机解耦.

3）缓存别名问题

由于“南湖”微架构的 L1 缓存（指令缓存和数据缓存）使用 VIPT 索引方式，且索引位与行内偏移的总位宽超过了页内偏移（4KB, 12b），这导致同一个缓存块可能在 L1 缓存中存在 2 个以不同虚拟地址索引的副本，造成别名问题. “南湖”微架构采用硬件抗别名（anti-alias）机制，在 L2 缓存的目录中存放 L1 缓存块的虚拟地址别名位；若 L2 缓存检测到潜在的别名问题，则向 L1 缓存发送命令使旧数据块无效，保证 L1 缓存只存放一个副本.

HuanCun 还采取了若干降低访问延迟和增大访

存并行度的设计. 在延迟优化方面, HuanCun 重点优化了通道控制模块的流水线, 以尽早释放资源; 新增的重填缓冲(refill buffer)部件负责将重填数据旁路传输给上级缓存. 在并行度优化方面, HuanCun 增加了请求缓冲(request buffer)部件, 缓解总线的阻塞情况; 同时重点优化了 MSHR 向通道控制器的仲裁策略, 使得所有的通道控制器大多时候均可并行工作. 实验结果表明, 针对部分敏感程序片断, HuanCun 与 InclusiveCache 相比性能提升约 30%.

6 敏捷设计验证方法与性能评估

6.1 基于 Chisel 语言的敏捷设计

香山项目启动时, 团队选择使用硬件描述语言 Chisel. Chisel[6] 由加州大学伯克利分校团队在 2012 年发布, 相比于传统的 Verilog 语言, Chisel 能够充分运用现代软件工程中面向对象编程、函数式编程和元编程等技术对硬件进行抽象, 提高编码效率和代码可读性, 使得硬件敏捷开发成为可能. 笔者所在团队曾对比两者的编码效率和代码质量[12], 结论是 Chisel 不仅在开发效率上远高于 Verilog, 生成的硬件电路也具有更小的面积和更低的延迟. 此外, 笔者所在团队在 2020 年研发了基于 Chisel 语言的标签化体系结构原型系统并成功投片, 打通了 Chisel 语言和后端物理实现的全流程, 给香山项目的开发语言选择提供了重要参考.

目前, “南湖”微架构的 Chisel 代码超过 6 万行. 据分析, Verilog 的代码密度约为 Chisel 的 1/5[12] 因此“南湖”微架构的设计规模约为 30 万行 Verilog 代码量. 此外, 香山项目积累的验证代码超过 3 万行.

图 8 展示了自 2020 年 6 月至 2022 年 10 月两代香山研发过程的 4 个角度: 1) 截止到“南湖”微架构代码冻结, 代码提交共 7498 次, 高峰期达到每周 169 次; 2) 香山在微结构特性的驱动下持续演进, 在主体框架搭建完成后逐步添加各类优化特性, 关键性能优化点在图 8 中列出; 3) 香山采用时序优化与微架构迭代并行前进的开发流程, 在设计早期就通过时序评估流程指导微结构设计; 4) 香山研制了各类基础设施与开发工具, 并不断完善其功能性和易用性. 这些设施与工具有效支撑香山的设计、验证和性能评估全流程, 是香山快速迭代的关键因素.

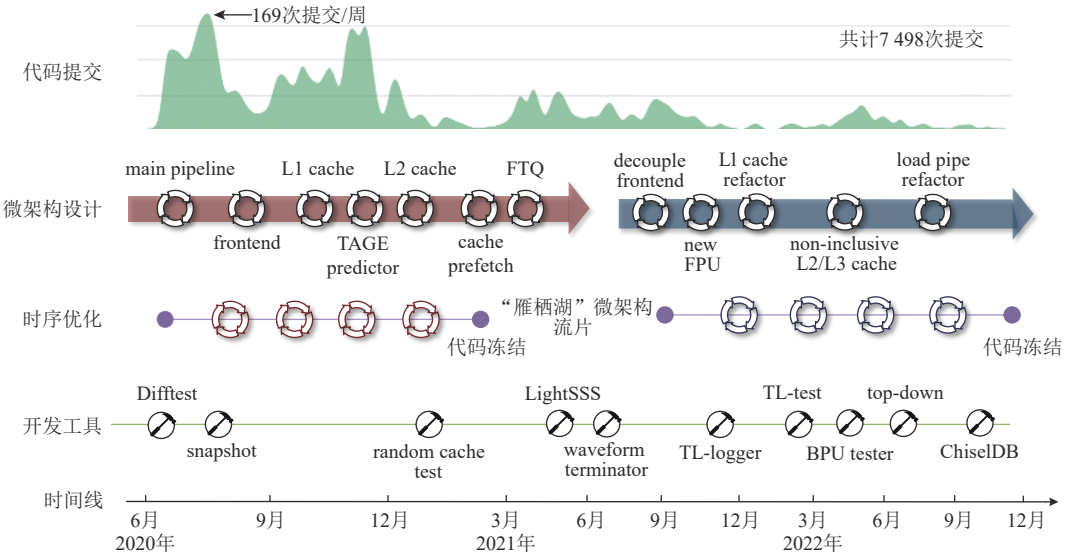


Fig. 8 Development timeline of XiangShan

图 8 香山开发时间轴

基于 Chisel 语言优秀的建模和描述能力, 香山构建了灵活的参数化系统, 成为一个“处理器生成器”, 可根据用户指定的性能、面积、功耗需求进行定制, 满足开发者的各类需求.例如, 香山只需改动一个参数, 即可由 6 发射修改为 4 发射, 并成功通过各种测试. 此外, 处理器的核数、各种队列的大小均可配置.

6.2 敏捷验证方法

处理器的设计效率提升后, 验证成为开发流程的新瓶颈. 高性能处理器结构复杂、状态繁多, 验证工作是一个公认的难题. 即使香山“雁栖湖”微架构在 3 个月内成功在仿真环境下启动 Linux 操作系统, 却仍需额外投入 8 个月和大量人力才完成更全面的验证工作. 为缓解验证瓶颈, 香山团队积极探索敏捷

验证的新方法和新工具，于 2022 年的体系结构国际顶会 MICRO 上发表相关成果[13]. 敏捷验证方法重点聚焦在仿真验证环节，将处理器验证问题看作一个循环，循环体包括发现错误、保存出错现场、定位并解决问题. 敏捷验证的思想是加快每一轮验证循环的过程，从而提升验证效率. 以下从 3 个方面介绍敏捷验证的工具和方法.

1）高效发现错误

传统的验证方法通常对比待测设计（DUT，通常是 RTL 实现的硬件电路）与参考模型（REF，通常是 C/SystemVerilog 实现的行为模型）输出结果的一致性. 然而，这需要耗费大量人力搭建参考模型，且一旦待测设计迭代更新，参考模型也需同步更新. 为提高验证效率，香山团队提出基于合法规则匹配的指令级在线差分验证框架 Difftest. Difftest 框架定义了一套精简通用的 API，捕捉处理器仿真时的指令提交和其他状态更新，驱动指令集模拟器执行相同的指令，在线对比两者状态，并基于自定义的规则判断处理器仿真状态是否正常. 该框架还支持多核系统验证，可检测缓存一致性、内存一致性的违例错误.

2）低成本保存出错现场

仿真出错后，开发者需要抓取出错现场信息以分析和定位问题. 传统验证方法要么在调试模式下全程仿真，产生大量的无用调试信息占用磁盘带宽和空间；要么出错后重新仿真复现错误，时间开销较大. 香山团队受软件容错技术的启发，提出定期快照和出错回滚结合的方式来保存和复现出错现场. 为解决大量快照占用磁盘空间的问题，香山团队提出 LightSSS 工具，利用操作系统的写时复制机制，定期通过 Fork 系统调用保存仿真过程的进程状态，实现仿真快照的轻量生成与便捷恢复.

3）快速定位并解决问题

恢复出错现场后，传统的错误定位方法通常是基于波形调试. 然而，波形信息密度较低，调试费时费力. 特别地，复杂队列和缓存相关的错误因状态更新间隔长，难以通过出错现场附近的波形直接定位问题. 为解决上述挑战，敏捷验证方法提出一套调试框架 ChiselDB，可获取设计代码中指定的数据结构，然后在仿真过程自动收集调试信息并导入数据库，后续可通过数据库 API 查看和分析，从而提升调试效率.

6.3 性能评估

香山实现了一套基于仿真的快速性能评估方法，通过大规模并行仿真具有代表性的程序检查点来评估架构设计的性能. 具体流程为：1) 使用高速的指令集模拟器对程序负载进行切片和采样，生成可在其他模拟和仿真环境中恢复的检查点；2) 使用机器学习聚类方法 SimPoint[14] 分析检查点，选出具有代表性的检查点并计算对应权重；3) 让香山处理器在多台服务器上并行仿真运行选取的检查点；4) 根据仿真结果和权重综合计算出香山处理器的性能评估数据.

通过该性能评估方法，可在 3~5 天内评估“雁栖湖”微架构的 SPEC 分值，评估结果与硅后的性能测试的对比如图 9 所示. 实践表明，该方法平均误差为 5%~10%，部分误差来源于芯片 DRAM 与仿真模型的不一致.

“南湖”微架构目前尚未投片，其单核仿真性能评估如图 10 所示. “南湖”微架构的仿真评估分值达到了 SPECint 9.550 分 /GHz、SPECfp 11.085 分 /GHz，在 2 GHz 下的总分为 20.844 分. 截至目前，该性能在开源处理器中排名第一，同频性能还超过了 Sifive P550 等商业处理器，达到 ARM Cortex A76 水平.

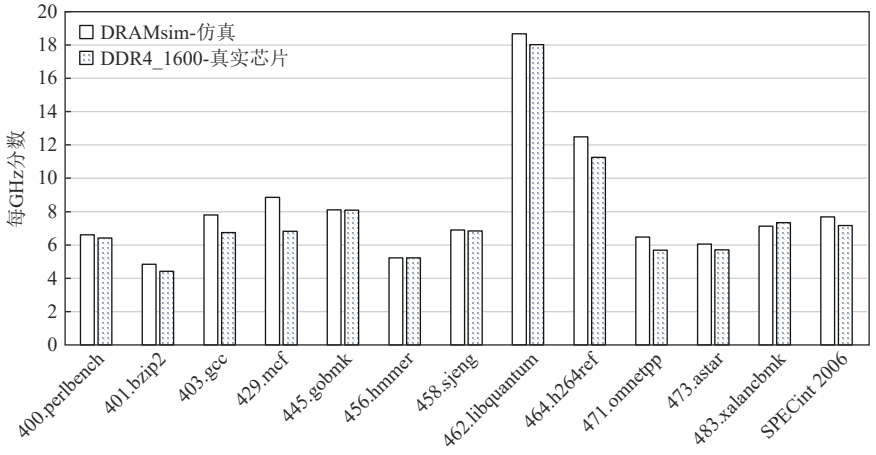
此外，该性能评估方法有助于快速探索设计空间. 例如，RVB 扩展的相关实验结果显示，基于 GCC 10.2.0 版本工具链，默认配置下开启 RVB 扩展使 SPECint 性能提升 10.8%，SPECfp 则基本保持不变. 性能提升的原因是 RVB 指令可替代部分分支指令，从而减少分支预测错误导致的回滚开销. 因此，优秀的指令集扩展对于处理器性能提升十分关键.

7 物理实现

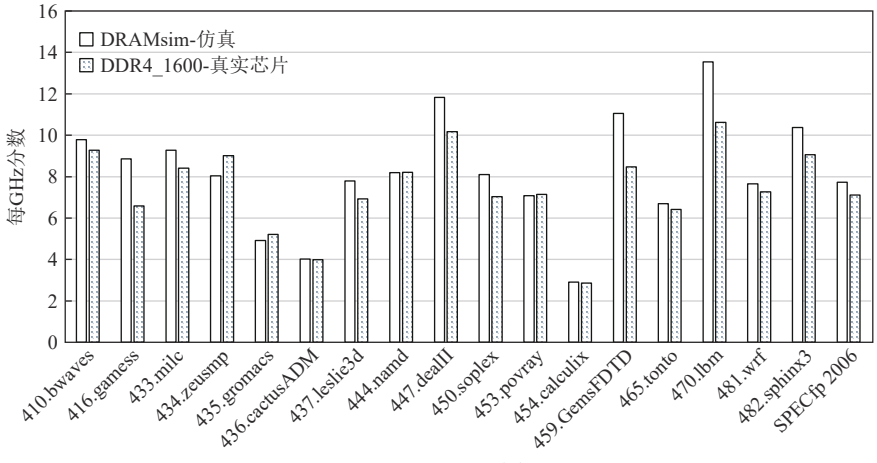
香山两代微架构均采用前后端紧密协作的开发流程，迭代优化处理器主频，表 3 为“雁栖湖”微架构最终流片的参数列表.

“南湖”微架构目前已进入后端物理实现阶段. “南湖”微架构的目标主频是 14 nm 工艺下达到 2 GHz，考虑到设计通用性，未采用定制电路. 后端团队根据物理评估结果多次调整布局布线方式，从开始的粗放式模块摆放演进到后期的细节调优，不断平衡时序与布线之间的冲突. “南湖”微架构后端物理实现已迭代优化 40 余次，最新一版如图 11 所示. 此外，前端团队也针对后端难以处理的时序关键路径改动了大量相关逻辑.

除香山外，目前暂未有开源高性能处理器达到 2 GHz，后端团队在相同条件下评估 BOOM 处理器[4]，其主频约 1 GHz. 大量开源处理器存在前端架构设计与后端时序优化割裂的问题，往往表现为：1) 寄存器



(a) “雁栖湖”微架构仿真与真实芯片SPECint分数对比



(b) “雁栖湖”微架构仿真与真实芯片SPECfp分数对比

Fig. 9 Performance comparison of Yanqihu micro-architecture between simulation and real chip

图 9 “雁栖湖”微架构仿真与真实芯片性能对比

扇入过大，逻辑级数过长；2) 寄存器扇出过大，需要插入驱动电路影响延迟；3) 模块间逻辑耦合过深，信号布线距离过长. 经过“雁栖湖”微架构的流片实践后，香山团队深刻认识到前后端协同优化需尽早开展，在设计规模大、设计功能复杂、不同功能的模块数量多的情况下，如何通过协同优化芯片的功耗、性能和面积尤为关键，这也是业界常说的“左移”. 在“南湖”微架构开发中，前后端团队积极沟通，主要优化措施为：

1) 重新组织时序紧张模块的流水线结构，对一些关键部件，例如大容量队列，可通过数据预读取等通用策略优化时序.

2) 在设计阶段分离控制流和数据流，规避大驱动逻辑. 另外可通过复制寄存器、切分扇出等方法解决部分大扇出问题.

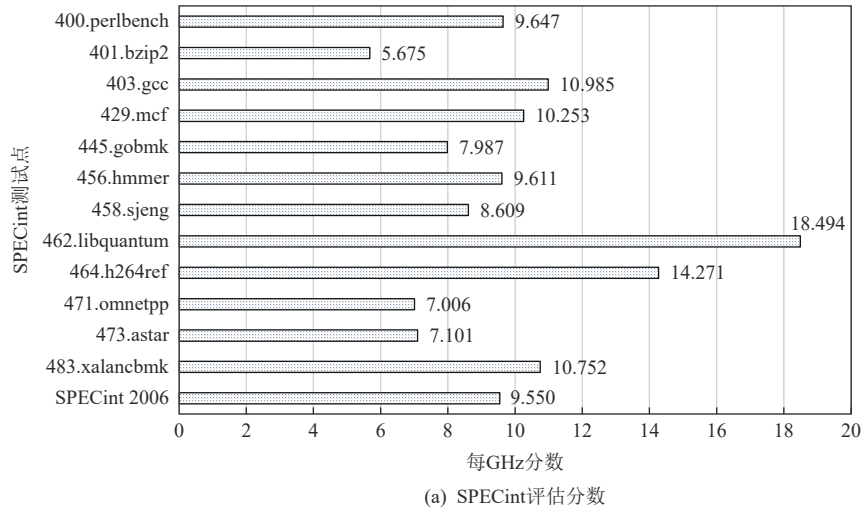
3) 降低模块间功能耦合程度，结合芯片布局情况进行跨模块优化. 设计处理器要有全局视野，而不能仅关注单个模块.

无论是前端设计和验证，还是后端物理实现，抑或是前后端协同优化，香山遇到的挑战在开源处理器项目中都是前所未有的. 香山团队希望在开发过程中尝试解决难点问题，并将成果开源开放，降低行业门槛. 未来规划包括 3 方面.

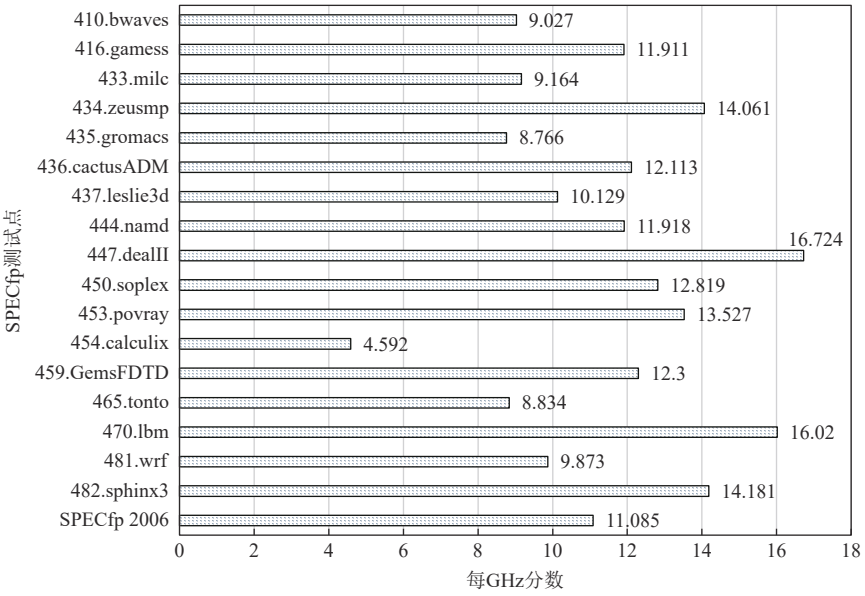
1) 在大规模复杂处理器芯片设计流程中，建立前后端协同优化迭代机制并积累设计经验，在此基础上开放流程并分享方法论.

2) 在开发过程中探索构建前后端紧密结合的芯片开发工具链，形成一系列支持高性能处理器芯片开发的基础工具，并开放前后端开发流程，开源基础设施.

3) 建立跨层优化能力体系之后，将提升流程的自动化程度，实现自动跨层优化的目标.



(a) SPECint评估分数



(b) SPECfp评估分数

Fig. 10 Performance evaluation of Nanhu micro-architecture

图 10 “南湖”微架构性能评估

Table 3 Tape-out Details of Yanqihu Micro-architecture

表 3 “雁栖湖”微架构流片指标

参数	数值
芯片面积/mm^2	8.6
标准单元数量	5 053 679
标准单元面积/mm^2	4.27
SRAM 单元数量	261
SRAM 单元面积/mm^2	1.7
密度	66%
标准单元比例	ULVT 1.04%, LVT 19.32% SVT 25.19%, HVT 53.67%
功耗/W	5
频率/GHz	1.3

8 联合开发模式与下一代展望

香山项目自 2021 年 6 月发布以来，截至目前在开源项目托管平台 GitHub 上已有超过 3200 星标（Star）和 400 分支（Fork），成为国际上最活跃的开源芯片项目之一[15]. 国内外企业界和学术界开始使用香山，有的参与产品化工作，有的开展前沿研究. 在北京市和中国科学院的支持下，18 家企业联合发起了北京开源芯片研究院，围绕香山开展产品化改造和后续架构探索工作，和企业共同建立了联合开发团队.

代码开源、平台开放的开源理念是香山联合开发模式的基石. 香山处理器的主线设计，从“雁栖湖”“南湖”到后续的“昆明湖”等微架构，都是开源开放

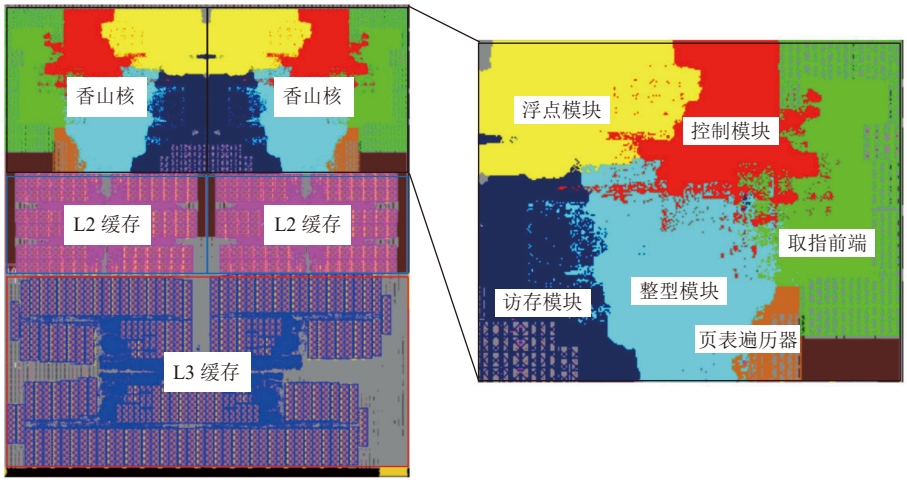


Fig. 11 Latest floorplan design of Nanhu micro-architecture

图 11 “南湖”微架构最新版图设计

的, 开放模式与现有模式不同. 以 x86 为代表的封闭模式只出售芯片产品, 数家核心公司垄断 x86 生态, 不利于多方协作与创新; 而以 ARM 为代表的授权模式虽然允许 IP 授权, 企业可付费获得设计代码, 但其开发平台仍是封闭的. 香山不仅开源设计代码和文档, 还开源各类开发基础设施, 实现开发平台的开放. 基于该平台, 企业可实现“竞争前合作”, 联合开发处理器高性能架构和基础设施, 亦可基于香山的设计代码和开发工具做二次开发, 满足各类定制化需求, 一定程度上解决芯片的碎片化难题.

香山有望成为连接工业界和学术界的桥梁. 首先由研究人员提出创新想法, 在香山代码上实现, 每年 1~2 次机会进行流片验证; 此后, 联合开发团队将会对初步验证的架构开展产品化改造工作, 最终形成稳定的高质量工业级 IP.

香山在 2022 年 8 月 24 日迎来了发展历程中的一个里程碑——中国科学院计算技术研究所、北京开源芯片研究院、腾讯科技、阿里巴巴集团、中兴通讯、中科创达、奕斯伟科技集团、算能科技等单位和企业共同建立了联合研发团队, 联合开发第三代香山——“昆明湖”微架构, 标志着香山及其开源模式得到了产业界的初步认可. “昆明湖”微架构将会以更高的性能为目标, 并增加向量、虚拟化等扩展.

9 总 结

本文首先介绍香山开源高性能 RISC-V 处理器项目的总体情况, 深入分析了前两代代号分别为“雁栖湖”和“南湖”的微架构设计, 分别从前端供指单元、乱序执行单元、访存子系统和缓存子系统 4 个方面系统地介绍香山团队遇到的设计挑战和开发经验. 同时, 本文还介绍了香山团队在敏捷设计和验证方向的探索成果, 通过一系列基础设施高效实现香山处理器的快速迭代和演进.

如今, 香山团队正在探索处理器开发多方协作模式, 欢迎各界企业与专家参与指导. 香山开源社区的 GitHub 网址为 https://github.com/OpenXiangShan. 项目在 Gitee, GitLink, iHub 等开源托管平台上均有镜像.

作者贡献声明: 王凯帆完成部分实验开发任务、数据整理分析并撰写论文; 徐易难和余子濠完成部分实验开发任务、数据整理分析并提供论文修改意见; 唐丹负责实验技术路线指导; 第 5 作者陈国凯到第 24 作者邹江瑞均完成部分实验开发任务和数据整理分析; 蔡晔、邮丹丹、李祖松、赵继业、何伟、孙凝晖、包云岗提供实验开发思路, 给予工作支持和指导意见.

参 考 文 献

[1] Eeckhout L. Is Moore’s law slowing down? What’s next?[J]. IEEE Micro, 2017, 37(4): 4–5

[2] Waterman A, Lee Y, Patterson D, et al. The RISC-V Instruction Set Manual. Volume 1: User-Level ISA, Version 2.1[R]. Berkeley, CA: University of California, Berkeley, 2016

[3] Asanovic K, Avizienis R, Bachrach J, et al. The rocket chip generator, UCB/EECS-2016-17[R]. Berkeley, CA: University of California, Berkeley, 2016

[4] Zhao J, Korpan B, Gonzalez A, et al. Sonicboom: The 3rd generation Berkeley out-of-order machine[C]// Proc of the 4th Workshop on

Computer Architecture Research with RISC-V. 2020[2022-12-28]. http://people.eecs.berkeley.edu/~krste/papers/SonicBOOM-CARRV2020.pdf

[5] Wang Huaqiang, Zhang Zifei, Zhang Linjuan, et al. OSCPU/NutShell: RISC-V SoC designed by students in UCAS GitHub repository [EB/OL]. [2022-12-28].https://github.com/OSCPU/NutShell

[6] Bachrach J, Vo H, Richards B, et al. Chisel: Constructing hardware in a Scala embedded language[C]// Proc of the 49th Annual Design Automation Conf. Piscataway, NJ: IEEE, 2012: 1216−1225

[7] Seznec A, Michaud P. A case for (partially) TAgged GEometric history length branch prediction[J]. Journal of Instruction Level Parallelism, 2006, 8(1): 1−23

[8] Waterman A, Hauser J, Liu Jiuyang, et al. Berkeley-Hardfloat GitHub repository [EB/OL]. [202-12-28].https://github.com/ucb-bar/berkeley-hardfloat

[9] Kessler R E. The Alpha 21264 microprocessor[J]. IEEE Micro, 1999, 19(2): 24−36

[10] Chrysos G Z, Emer J S. Memory dependence prediction using store sets[J]. ACM SIGARCH Computer Architecture News, 2002, 26(3): 142−153

[11] Chen A, Schmidt C, Zhao J, et al. Sifive/block-inclusivecache-sifive [EB/OL]. [2022-12-28]. https://github.com/sifive/block-inclusiveca-che-sifive.

[12] Yu Zihao, Liu Zhigang, Li Yiwei, et al. Practice of chip agile development: Labeled RISC-V[J]. Journal of Computer Research and Development, 2019, 56(1): 35−48 (in Chinese)
(余子濠, 刘志刚, 李一苇, 等. 芯片敏捷开发实践: 标签化RISC-V[J]. 计算机研究与发展, 2019, 56(1): 35−48)

[13] Xu Yinan, Yu Zihao, Tang Dan, et al. Towards developing high performance RISC-V processors using agile methodology[C]// Proc of 2022 55th IEEE/ACM Int Symp on Microarchitecture (MICRO). Piscataway, NJ: IEEE, 2022: 1178−1199

[14] Perelman E, Hamerly G, Biesbrouck M V, et al. Using SimPoint for accurate and efficient simulation[J]. ACM Sigmetrics Performance Evaluation Review, 2003, 31(1): 318−319

[15] Xu Yinan, Wang Huaqiang, Gou Lingrui, et al. OpenXiangShan/XiangShan: Open-source high-performance RISC-V processor[EB/OL]. [2022-12-28].https://github.com/OpenXiangShan/XiangShan

Wang Kaifan, born in 1997. PhD candidate. His main research interests include agile development of processors and computer architecture.
王凯帆, 1997 年生. 博士研究生. 主要研究方向为处理器敏捷开发与计算机体系结构.

Xu Yinan, born in 1997. PhD candidate. His main research interests include agile development of processors and computer architecture.
徐易难, 1997 年生. 博士研究生. 主要研究方向为处理器敏捷开发与计算机体系结构.

Yu Zihao, born in 1991. PhD. His main research interests include computer architecture and operating system.
余子濠, 1991 年生. 博士. 主要研究方向为计算机体系结构和操作系统.

Tang Dan, born in 1976. PhD, senior engineer. His main research interests include computer architecture and low power SoC design.
唐　丹, 1976 年生. 博士. 高级工程师. 主要研究方向为计算机体系结构和低功耗 SoC 设计.

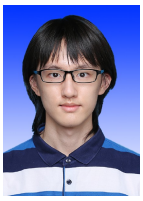
Chen Guokai, born in 1999. Master candidate. His main research interests include computer architecture and operating system.
陈国凯, 1999 年生. 硕士研究生. 主要研究方向为计算机体系结构与操作系统.

Chen Xi, born in 1999. PhD candidate. His main research interest is computer architecture.
陈　熙, 1999 年生. 博士研究生. 主要研究方向为计算机体系结构.

Gou Lingrui, born in 1998. PhD candidate. His main research interests include branch prediction and instruction fetch.
勾凌睿, 1998 年生. 博士研究生. 主要研究方向为分支预测和取指.

Hu Xuan, born in 1997. Master candidate. His main research interest is computer architecture.
胡　轩, 1997 年生. 硕士研究生. 主要研究方向为计算机体系结构.

Jin Yue, born in 1997. PhD candidate. His main research interests include high performance processor design, computer architecture security and processor verification.
金　越, 1997 年生. 博士研究生. 主要研究方向为高性能处理器设计、计算机体系结构安全以及处理器验证.



Li Qianruo, born in 2001. PhD candidate. His main research interest is computer architecture.

李乾若, 2001 年生. 博士研究生. 主要研究方向为计算机体系结构.

Li Xin, born in 1999. Master candidate. His main research interest is computer architecture.

李　昕, 1999 年生. 硕士研究生. 主要研究方向为计算机体系结构.

Lin Jiawei, born in 1998. Master candidate. His research interest is high-performance computer architecture.

蔺嘉炜, 1998 年生. 硕士研究生. 主要研究方向为高性能计算机体系结构.

Liu Tong, born in 1997. PhD candidate. His main research interests include SoC (system-on-chip), low-power design and processor design space exploration.

刘　彤, 1997 年生. 博士研究生. 主要研究方向为片上系统、低功耗设计和处理器设计空间探索.

Liu Zhigang, born in 1995. Master, engineer. His research interests include high-performance computer architecture, parallel processing, and operating systems.

刘志刚, 1995 年生. 硕士, 工程师. 主要研究方向为高性能计算体系结构、并行处理和操作系统.

Wang Huaqiang, born in 1998. Master candidate. His main research interests include processor micro-architecture and open-source hardware design.

王华强, 1998 年生. 硕士研究生. 主要研究方向为处理器微体系结构和开源硬件设计.

Wang Huizhe, born in 1994. PhD candidate. His research interests include microarchitecture design, QoS management on SoC and hardware description language design and methodology.

王诲喆, 1994 年生. 博士研究生. 主要研究方向为微架构设计、SoC 的 QoS 管理以及硬件描述语言设计与方法论.

Zhang Chuanqi, born in 1996. PhD candidate. His main research interest is data-center architecture.

张传奇, 1996 年生. 博士研究生. 主要研究方向为数据中心体系结构.

Zhang Fawang, born in 1997. Master. His main research interests include computer architecture and formal verification.

张发旺, 1997 年生. 硕士. 主要研究方向为计算机体系结构与形式化验证.

Zhang Linjuan, born in 1998. Master candidate. Her main research interest is high-performance computer architecture.

张林隽, 1998 年生. 硕士研究生. 主要研究方向为高性能计算机体系结构.

Zhang Zifei, born in 1997. PhD candidate. His main research interests include computer architecture and agile development.

张紫飞, 1997 年生. 博士研究生. 主要研究方向为计算机体系结构、敏捷开发.

Zhang Ziyue, born in 1998. PhD candidate. His main research interest is computer architecture.

张梓悦, 1998 年生. 博士研究生. 主要研究方向为计算机体系结构.

Zhao Yangyang, born in 1990. PhD, engineer. Her main research interests include memory system, accelerator and secure processor.

赵阳洋, 1990 年生. 博士, 工程师. 主要研究方向为内存系统、加速器和安全处理器.

Zhou Yaoyang, born in 1995. PhD candidate. His main research interests include CPU ILP enhancement, scalable CPUs design, workload sampling, and performance evaluation methods.

周耀阳, 1995 年生. 博士研究生. 主要研究方向为 CPU ILP 提升技术、可扩展 CPU 设计、负载采样和性能评测方法.

Zou Jiangrui, born in 1997. Master, engineer. His man research interest is high-performance computer architecture.

邹江瑞, 1997 年生. 硕士, 工程师. 主要研究方向为高性能计算机体系结构.

Cai Ye, born in 1974. PhD, associate professor of Shenzhen University. His main research interests include computer architecture and computer system.

蔡　晔, 1974 年生. 博士, 深圳大学副教授. 主要研究方向为计算机体系结构和计算机系统.

Huan Dandan, born in 1979. PhD, senior engineer. Her main research interests include high-performance computer architecture and microprocessor design.

郇丹丹, 1979 年生. 博士, 高级工程师. 主要研究方向为高性能计算机体系结构和微处理器设计.

Li Zusong, born in 1977. PhD, associate professor. His main research interests include high-performance computer architecture and microprocessor design.

李祖松, 1977 年生. 博士, 副研究员. 主要研究方向为高性能计算机体系结构和微处理器设计.

Zhao Jiye, born in 1977. PhD, senior engineer. His main research interests include high-performance chip physical design and high-speed digital circuit design.

赵继业, 1977 年生. 博士, 高级工程师. 主要研究方向为高性能芯片物理实现, 高速数字电路设计.

He Wei, born in 1987. Master, engineer. His main research interests include high-speed digital circuit design and low power design.

何　伟, 1987 年生. 硕士, 工程师. 主要研究方向为高速数字电路设计、低功耗设计.

Sun Ninghui, born in 1968. PhD. Academician of Chinese Academy of Engineering. Fellow of CCF. His main research interests include computer architecture and high performance computing.

孙凝晖, 1968 年生. 博士, 中国工程院院士. CCF 会士. 主要研究方向为计算机体系结构、高性能计算.

Bao Yungang, born in 1980. PhD, professor. His main research interests include data-center architecture, agile design methodology of processor chips and ecosystem of open-source processor chips.

包云岗, 1980 年生. 博士, 研究员. 主要研究方向为数据中心体系结构、处理器芯片敏捷设计方法论、开源处理器芯片生态.