计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20190044
56(12):2702-2719,2019

体系结构模拟器在处理器设计过程中的作用

张乾龙[1,2] 侯 锐[3] 杨思博[4] 赵博彦[3] 张立新[1]
[1](中国科学院计算技术研究所 北京 100190)
[2](中国科学院大学 北京 100049)
[3](中国科学院信息工程研究所 北京 100093)
[4](北京大学软件与微电子学院 北京 100871)
(zhangqianlong@ict.ac.cn)

The Role of Architecture Simulators in the Process of CPU Design

Zhang Qianlong[1,2], Hou Rui[3], Yang Sibo[4], Zhao Boyan[3], and Zhang Lixin[1]
[1](*Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)
[2](*University of Chinese Academy of Sciences, Beijing* 100049)
[3](*Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093)
[4](*School of Software & Microelectronics, Peking University, Beijing* 100871)

Abstract As Moore's law goes to an end, the improvement of CPU performance is increasingly dependent on the optimization and improvement of CPU microarchitecture which heavily relies on the assistance of architecture simulator. Therefore, CPU architecture simulator plays an increasingly important role in the design of high performance CPUs, for example, architecture simulator can be helpful in exploring the CPU microarchitecture, verifying the logic design before actual tape-out, building post-silicon test environment and starting to develop firmware, operating system and hypervisor before CPU is ready. In this paper, we summarize the experience of academia and industrial CPU vendors in developing and using architecture simulators, by which we clarify and summarize the important role of architecture simulators in the CPU design process and how to develop and optimize architecture simulators. First, we introduce the relationship between open source architecture simulators and CPU design, then we summarize and analyze the methodologies and experience of how to do well-known industrial CPU vendors develop and use architecture simulators in the process of CPU design. Second, we summarize the methodologies of how to calibrate and optimize architecture simulators, after that, some suggestions on the design and usage methodology of architecture simulators are put forward. Third, we summarize the scale-up and scale-out optimization methods of architecture simulators and introduce some new architecture simulators. At the end of the paper, we summarize the paper and put forward some problems in developing new architecture simulators.

收稿日期:2019-01-16;**修回日期**:2019-05-30

基金项目:国家重点研发计划项目(2016YFB1000201);国家自然科学基金项目(61771325,61420106013,61702480);中国科学院青年创新促进会项目(2013073);中国科学院前沿科学重点研究项目(QYZDB-SSW-JSC010);国家自然科学基金优秀青年科学基金项目(61522212)

This work was supported by the National Key Research and Development Program of China (2016YFB1000201), the National Natural Science Foundation of China (61771325, 61420106013, 61702480), the Youth Innovation Promotion Association of Chinese Academy of Sciences (2013073), the Key Research Program of Frontier Sciences of Chinese Academy of Sciences (QYZDB-SSW-JSC010), and the National Natural Science Foundation of China for Excellent Young Scientists (61522212).

Key words CPU design; simulator; performance analysis; performance modeling; high performance computing; heterogeneous simulator; simulator calibration; quantitative analysis

摘 要 随着摩尔定律趋于终结,处理器性能的提升越来越依赖于处理器微体系结构的优化改良,而处理器微体系结构的优化改良离不开体系结构模拟器的辅助,因此体系结构模拟器在现代和未来的高性能处理器设计中的作用越来越重要.具体地,体系结构模拟器可以辅助进行处理器微结构探索、芯片逻辑验证、硅后验证环境搭建、系统软件开发等工作.首先,介绍了开源模拟器与处理器设计的关系,并指出开源模拟器在辅助进行处理器设计方面的不足,同时对处理器厂商使用模拟器辅助进行处理器设计的方法和经验进行了分析总结.其次,对用于处理器微结构优化和改进的性能模拟器的校准方法进行了总结,然后对模拟器的纵向和横向优化方法进行了总结.最后,对新型异构模拟器进行了总结,并对未来模拟器的发展和基于模拟器进行处理器设计的方法进行了总结和展望.

关键词 处理器设计;模拟器;性能评估;性能建模;高性能计算;异构模拟器;模拟器校准;量化分析

中图法分类号 TP302

模拟器是体系结构量化分析的重要手段,对架构设计、芯片开发有重要的指导作用.基于模拟器辅助进行集成电路设计可以追溯到 1980 年代[1],自此模拟器便一直是处理器设计过程中不可或缺的工具.在芯片开发过程中,体系结构模拟器可以缩短处理器的设计时间,降低开发成本,其具体作用如图 1 所示:

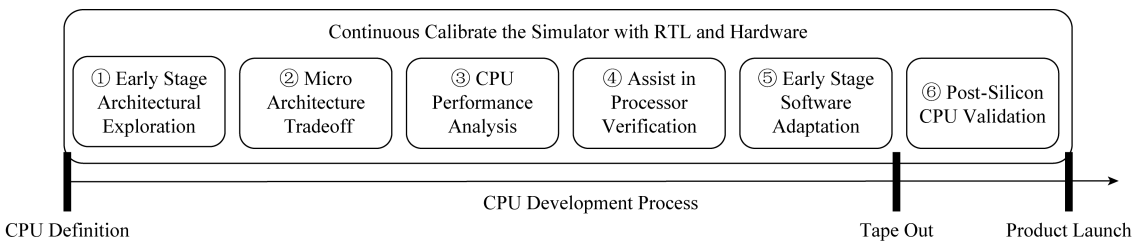


Fig. 1 The role of simulators in overall CPU design process

图 1 模拟器在整个芯片设计过程中所起的作用

由图 1 可知:

1) 在芯片开发早期,基于模拟器可以进行微结构探索和粗粒度微结构定义,此时模拟器的开发抽象层次较高.

2)随着处理器设计的不断推进和模拟器的不断完善,基于模拟器可以持续对芯片微结构进行评估、修改和取舍.

3) 当模拟器趋于成熟,可以对微结构、多核互联系统、一致性协议等进行详细性能分析,基于分析结果对微结构进行微调.

4)在对处理器逻辑设计进行验证的阶段,模拟器可以作为参考模型辅助进行验证,可以快速定位逻辑设计错误.

5) 在未流片之前基于模拟器就可以开展系统软件开发和适配工作,这样可以在芯片流片结束后以最快速度启动系统软件.

6) 流片结束后,基于模拟器可以辅助进行芯片硅后验证环境的搭建以及测试用例编写工作.为了保证模拟器可以顺利辅助进行处理器设计,在整个芯片开发过程中,需要持续对模拟器进行校准,通过持续对比模拟器和寄存器传输层(register-transfer level, RTL)之间的差别,可以互相校准并发现模拟器或者 RTL 的设计错误.

1 开源模拟器与处理器设计的关系

处理器设计的很多创新思想来源于体系结构学术研究,而学术研究使用的模拟器大多是开源模拟器,因此开源模拟器对于处理器设计有重大的意义.开源模拟器主要用于对处理器进行抽象建模并验证创新思想,因此开源模拟器对于学术研究有 2 种很大的价值:1)不同的开源模拟器具有不同特点,因此研究人员可以根据所研究的内容选择合适的开源模拟器平台进行实验,这给学术研究提供了很高的灵

活性,有助于快速实现原型系统;2)开源模拟器的社区一般比较活跃,在原型系统实现过程中,如果遇到问题可以很快找到解决方案,从而快速构建实验用的原型系统.

但是从辅助进行处理器设计的角度看,开源模拟器有一定的局限性.如果处理器厂商基于开源的模拟器辅助进行处理器设计,需要首先解决2个问题:

1)各处理器厂商的微结构一般与开源模拟器的微结构差别很大,因此修改模拟器的代价较大.例如gem5模拟器中乱序执行流水线的设计主要参考的是Alpha 21264[2],基于此进行微结构修改的工程量比较大.此外,由于模拟器开发是一项较大的软件工程,而软件工程的框架一旦确定,修改起来也会比较棘手.

2)开源模拟器一般不会与具体某款处理器进行校准工作,因此一般无法直接基于某款开源模拟器进行处理器验证.文献[3]指出:gem5等开源模拟器有很多未知错误,例如流水线中不合理的写回机制、生成微指令过程错误等.另外由于gem5对于访存子系统模拟精度不够,大小为512×512的矩阵运算在gem5模拟器上运算误差可以达到23.38%,并随着访存通信量越大误差越大[4].此外文献[5]通过对比真实的ARM平台,指出gem5全系统模拟误差也非常可观.因此基于该模拟器二次开发用于辅助处理器设计的模拟器需要首先修复上述开源模拟器误差和错误.

但是,基于开源模拟器二次开发用于辅助进行处理器设计的模拟器也有好处.开源模拟器中提供的工具可以直接使用,降低工具及模拟器相关库开发成本,例如SimpleScalar工具集提供了功耗模拟器WATTCH[6],用于选择模拟样本的SimPoint[7]等.

2 模拟器辅助处理器设计的方法学

模拟器是体系结构量化分析的重要手段,从辅助进行处理器设计的角度看,计算机体系结构模拟器可以分为3类,如图2所示.

由图2可知:1)微结构探索和性能优化,为了精准模拟处理器微结构并提高模拟器运行速度,同时为了方便开发和维护,一般会继续细分为单核性能模拟器和多核互联性能模拟器;2)辅助进行系统软件开发;3)辅助进行处理器验证.现代处理器设计厂商普遍自研适配于自家处理器架构的模拟器.本节根据处理器厂商公开发表的通过模拟器辅助进行处理器设计的经验进行分析并总结规律.

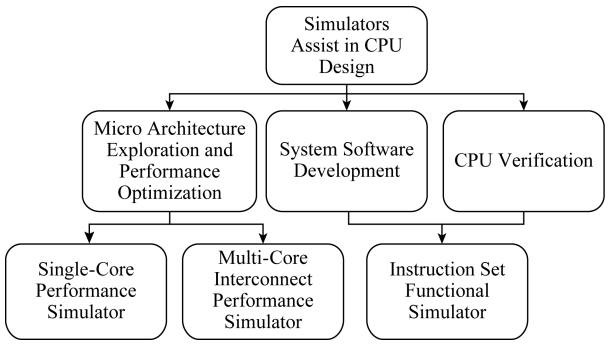


Fig. 2 Classification of simulators used in CPU design

图2 辅助处理器设计的模拟器分类

2.1 龙 芯

龙芯中科技术有限公司在使用模拟器辅助处理器设计的文献中提到[8-10],龙芯2号处理器研发过程中,最早开发的模拟器ICT-Godson由于对硬件模拟过于详细,导致其速度和灵活性不足[10].因此,龙芯基于Simple-Scalar开源模拟器[11]框架设计实现了Sim-Godson处理器核模拟器[8].相比于ICT-Godson模拟器,Sim-Godson具有运行速度快、灵活性高、支持大程序评估等优点[10].Sim-Godson可以支持功能模拟和时序模拟,由二进制可执行程序作为输入,主要用于处理器核的微结构性能探索.Sim-Godson虽然借用了SimpleScalar的基础模块,但龙芯中科技术有限公司对其进行了大量修改定制工作,使其达到高精准模拟龙芯2号处理器核微结构的目的.基于SimpleScalar中自带的指令集仿真器和I/O仿真器进行执行加速,其速度可以达到0.5MIPS(million instructions per second).经过详细校准,Sim-Godson模拟器与ICT-Godson的误差平均不到5%[8].类似地,基于Simple-Scalar进行二次开发的模拟器还有Sim-alpha[12].

基于Sim-Godson模拟器可以对单核微结构进行性能评估和分析,但是由于SimpleScalar的结构本身不支持对多核互联系统进行模拟,因此龙芯中科技术有限公司基于SimOS[13]全系统开源模拟器开发了SimOS-Goodson模拟器[9].借用SimOS模拟器的全系统组件,龙芯中科技术有限公司把处理器核模拟器嵌入其中,并解决了因为功能模拟和时序模拟并行执行导致的存储一致性问题,同时添加了全系统相关功能和调试功能.定制修改后的SimOS-Goodson配置灵活、执行迅速、模拟准确,同时支持用户态和全系统模拟,模拟速度可达0.3MIPS,模拟误差在15%以内.类似地,DEC和

IBM 也曾基于 SimOS 开发全系统模拟器 SimOS-Alpha[14]和 SimOS-PPC[15].

2.2 IBM

IBM 于 2012 年某研讨会中做了题为“IBM 使用模拟器的经验”的报告[16],对于 IBM 如何在处理器设计过程中使用模拟器进行了介绍.

在处理器早期设计研究期间,IBM 使用 Mambo[17]模拟器的时钟精确模式进行微结构探索和粗粒度微结构定义.Mambo 模拟器对微结构主要模块和结构进行了模拟,该阶段 Mambo 由踪迹(trace)驱动,主要运行和研究用户态应用,对处理器的产品竞争力进行横向比较研究[16].

在微结构设计实现期间,IBM 使用基于公司内部专用“T”语言编写的时钟精准模拟器 M1 进行详细模拟处理器微结构[16],如图 3 所示.M1 是以 Mambo 模拟器或者硬件上抓取的踪迹作为输入,并且可以收集非常详细的微结构数据进行性能评估.为了加速 M1 模拟器的执行速度,需要对所抓取的踪迹进行取样,同时为了方便调试,M1 支持微结构性能数据可视化功能.

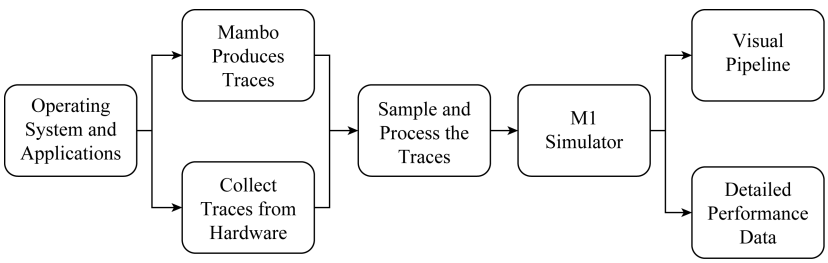


Fig. 3 IBM CPU simulator framework

图 3 IBM 模拟器框架

在处理器验证阶段,IBM 使用 Mambo[17]作为处理器验证参考模型辅助进行验证,此阶段 Mambo 可以为处理器功能正确性提供参考结果.Mambo 模拟了所有处理器的功能特征,把某些性能相关的微结构维护操作(例如 cache 维护类指令)翻译成空(nop)操作,对于计算类指令产生准确的结果,并精准追踪处理器寄存器的状态变化,同时支持指令撤销操作,为处理器验证提供参考.IBM 基于 Mambo 模拟器曾发现 PowerPC CPU 的一个控制寄存器存在竞争条件,使得该设计错误在流片之前就被发现并修改[17].在该阶段,IBM 还使用自研的由多个 FPGA(field programmable gate array)组成的 VHDL(very-high-speed integrated circuit hardware description language)仿真加速器 Twinstar[18]进行处理器综合验证.Twinstar 是时钟精准的仿真加速器,其推进方式是事件驱动模式,可以对整个处理器芯片进行仿真,以二进制程序作为输入,还支持详细的指令踪迹和处理器状态的实时追踪.该平台运行速度可以达到 4 MHz 并可以运行未经修改的系统软件.类似 Twinstar 的验证平台还有帕拉丁[19]等.

在系统软件开发方面,IBM 基于 Mambo(加速模式)[17],Simics[20],BGLsim[21]等多种平台,在流片之前就开始进行固件、操作系统、虚拟机管理器等软件的早期开发.IBM 基于 Mambo 模拟器曾开发了 K42 操作系统,在芯片可用之后 1 周内就启动了操作系统[17].IBM 基于 BGLsim-multi[21]平台和基于 OMNeT++[22]开发的 MARS(message passing interface application replay simulation)模拟平台还可以对机群网络相关的功能进行模拟,模拟器由可执行程序或者踪迹驱动,其中 MARS 平台还可以对 MPI(message passing interface)类应用进行调优.

2.3 AMD

AMD 于 2007 年 ISPASS 讨论会上做了题目为“AMD 性能建模和分析:游历指南”的主旨发言,介绍了 AMD 如何使用模拟器辅助进行处理器设计和验证[23].此外,在 2018 年 AMD 做过题为“现代服务器 CPU 性能分析”的报告[24],其中也提到了模拟器在 AMD 服务器处理器设计中的辅助作用.

在处理器研发早期,AMD 通过模拟器对体系结构进行粗粒度定义[23],还使用功能模拟器 SimNow[25]辅助进行处理器设计,例如产生程序执行踪迹、为 CPU 性能模拟器提供对比结果等.

在微结构设计实现期间,AMD 开发了 CPU 核模拟器和多核互联模拟器(由于 AMD 没有公开其命名,本文分别称之为 AMD-Core,AMD-NB 模拟器)用于对处理器性能进行评估.AMD-Core 模拟器框架如图 4 所示.

首先从真实硬件或者 SimNow 模拟器上直接

抓取程序执行踪迹,然后对踪迹进行分析获取初步微结构信息,例如 cache 缺失率等.同时把踪迹作为输入文件灌入时钟精准的 AMD-Core 模拟器进行详细分析,此时从真实硬件上获取的程序执行结果可以作为 AMD-Core 模拟器的参考,分析后可以产生详细微结构数据.

AMD-NB 模拟器可以模拟多核互联系统,如图 5 所示,由从真实硬件抓取的事务(transaction)踪迹驱动,因此可称其为事务精准模拟器.为加快执行速度,对互联模拟器中 CPU 进行抽象处理.基于该模拟器可以对于多核设计进行取舍,例如可以用于决定互联系统、内存系统的设计.此外,AMD 还开发了以指令执行踪迹作为输入的 TagSim cache 模拟器,可以对 cache 微结构(如替换算法、相连度等)进行分析.为提高准确性,上述模拟器都需要同 RTL 进行校准.

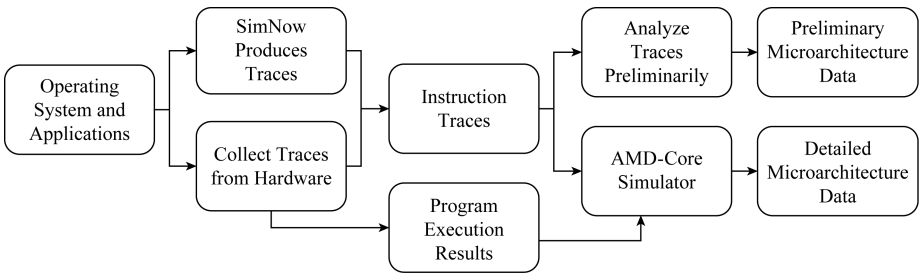


Fig. 4 AMD CPU core simulator framework

图 4 AMD 处理器核模拟器框架

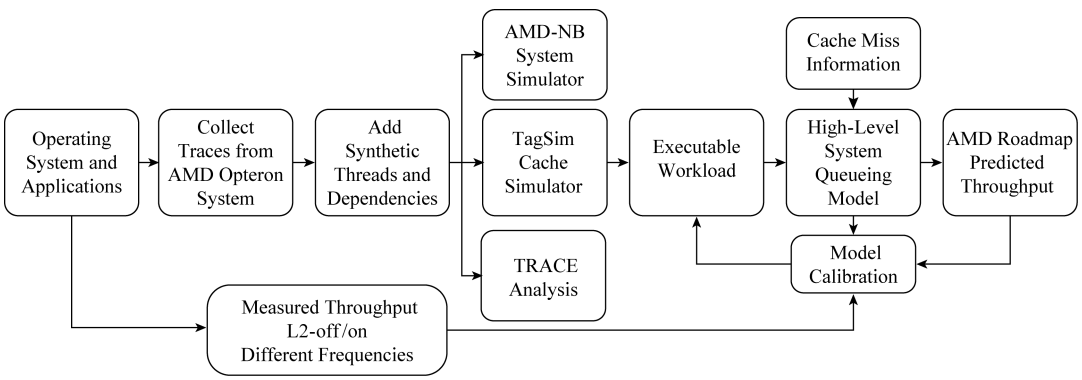


Fig. 5 AMD multi-core interconnect simulator framework

图 5 AMD 多核互联模拟器框架

AMD 的互联系统模拟器中还包含高级系统队列模型,该模型可以对资源利用率、cache 一致性协议等进行建模,并可以基于预定义的参数定量产生流量和 cache 缺失等信息,例如预先设定 L3 cache 缺失率 10%,该模型可以基于该参数产生 10% 的 L3 缺失率,为 L3 下游子系统产生输入请求.其输入是从 TagSim cache 模拟器中获取的可执行负载的参数、cache 参数等.该模型经过与真实硬件系统进行校准后,可以用于预测包括吞吐率和带宽利用率在内的 AMD 服务器性能路线图(roadmap).

在处理器验证期间,AMD 使用 SimNow 功能模拟器为 AMD-Core 和 AMD-NB 提供执行参考结果,此时的 SimNow 角色和 IBM 的 Mambo 相同.此阶段性能模拟器和 RTL 进行互相校准,可以通过模拟器发现并修改 RTL 的逻辑错误,反之也可以通过 RTL 对 AMD 性能模拟器进行校准.AMD-Core 和 AMD-NB 模拟器的开发目标是跟 RTL 一样时钟精准,并且保证执行正确性以方便进行不同层次的性能分析.AMD 模拟器相较于 RTL,在未校准之前每 1 000 条指令的绝对误差是 22%,经过校准后可以达到 2%~3%.

在系统软件开发方面,AMD 基于 SimNow 平台可以在流片前就开始 BIOS、硬件驱动代码、操作系统、软件编译器的开发.

2.4 Qualcomm

美国高通(Qualcomm)公司基于 ARMv8 架构研发的多核服务器芯片 Centriq 2400 已经发布[26-27],文献[26]对互联系统硬件架构进行了描述.此外文

献[28-31]作者均来自美国高通公司,该文献中实验评估部分对高通使用的多核互联模拟器框架进行了描述[28],从文献[26,28-31]可以得到高通公司所使用的互联系统模拟器架构如图 6 所示.高通的互联系统模拟器前半部是运行在开源 QEMU[32]虚拟机上的操作系统及真实应用程序,在运行时动态产生指令执行踪迹.模拟器后半部运行真正的互联系统模拟器,模拟多级 cache、内存控制器、互联总线等模块.为了加速互联模拟器的仿真速度,该后端模拟器的推进方式不是时钟精准而是时钟近似的.模拟器在运行时,前端 QEMU 运行速度比后端模拟器运行稍快,动态产生程序执行踪迹暂存到缓冲区中,后端模拟器读取该缓冲区数据进行模拟.缓冲区中会暂存一部分访存请求,在不违背访存之间依赖关系情况下,可以乱序发射以此对 CPU 乱序执行进行模拟.该模拟器是已跟 RTL 进行校准的企业级模拟器,支持 ARMv7/v8 指令集,并由高通 CPU 研究部门和开发部门共用[31].

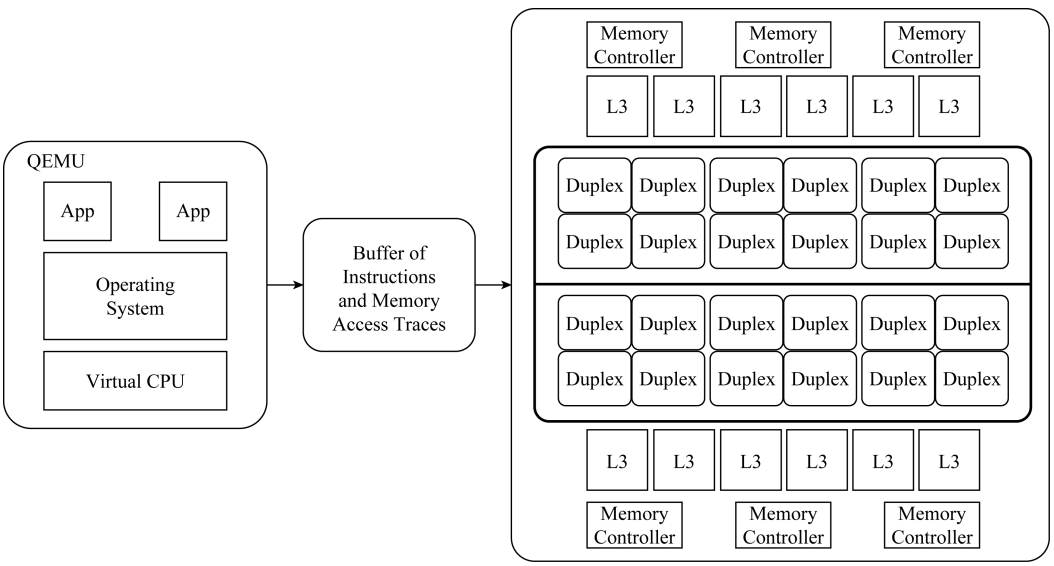


Fig. 6 Qualcomm multi-core interconnect simulator framework

图 6 高通多核互联模拟器框架

上述模拟方法中,QEMU 作为生产者动态产生指令踪迹,互联模拟器作为消费者消耗指令踪迹进行互联系统模拟,这种方法的优点在于可以不需要对踪迹文件进行存储,从而节省大量存储空间,并且还可以一定程度上模拟 CPU 乱序执行.但其缺点也比较明显,因为该模拟器由踪迹驱动,会导致其无法精确模拟处理器中如分支预测、推测执行等细节.

2.5 处理器厂商使用模拟器经验小结

第 2 节分别介绍了各处理器厂商如何使用模拟器辅助进行处理器开发,具体地,模拟器信息如表 1 所示,对于未公开的数据用“—”标示.

上述不同的模拟器有不同的作用并且运行速度不同,文献[23]对模拟器的运行速度及其作用进行了总结,如图 7 所示.

1) 1~10 Hz.RTL 仿真运行速度.对性能模拟器和 RTL 进行互相校准时需要运行 RTL 仿真,通过校准可以找到 RTL 中的逻辑错误和性能模拟器的模拟错误.

2) 1~10 kHz.时钟精准的单核性能模拟器的运行速度.Intel[33]和 AMD[23]的时钟精确模拟器运行速度都是这个级别,基于性能模拟器可以进行处理器微结构分析和探索.

3) 10~100 KIPS.踪迹分析模拟器和处理器抽象模型的运行速度.此时基于模拟器可以初步对踪迹进行分析,获取微结构信息,还可以对处理器进行抽象,基于抽象模型对未来性能进行估算.

4) 100~500 KIPS.这是大多经过优化加速(例如通过翻译执行)后时钟精确模拟器的执行速度,例如 SimpleScalar 的 sim-outorder 模式和 PTLSim 速度分别是 740KIPS 和 270KIPS[34].

5) 1~100 MIPS.指令集功能模拟器的运行速度.基于功能模拟器可以对指令集进行模拟和研究,作为对比模型进行处理器验证等工作.

文献[16]中曾提到,模拟器的开发是一个大型软件工程,开发周期长,需要持续与 RTL 进行校准,本节结合表 1 及文献[8-10,16,23-24]总结了处

Table1 Simulator Summary

表 1 模拟器总结

Company	Simulator	Programming Language	Target Architecture	Input	Functional/ Cycle Accurate	Forward Mode	Deviation /%	Performance
Loongson	Sim-Godson	C	MIPS	Executable Binary	Cycle	Cycle Accurate	5	0.5 MIPS
	SimOS-Goodson	C/Tcl	MIPS	Executable Binary	Multi-Core	Functional/Cycle Accurate	15	0.3 MIPS
IBM	M1	T	Power	Trace	Core	Cycle Accurate	—	—
	Mambo	C/Tcl	Power	Executable Binary/Trace	Multi-Core	Cycle Accurate	—	4 MIPS
	BGLsim	—	Power	Executable Binary	Cluster	Cycle Approximate	—	2 MIPS
	Simics	—	Power	Executable Binary	Machine	Functional	—	>100 MIPS
	Twinstar	FPGA-based	Power	Executable Binary	Multi-Core	Cycle Accurate	—	4×10^3 kHz
AMD	AMD-Core	C++	X86	Executable Binary/Trace	Core	Cycle Accurate	1-2	1-10 kHz
	AMD-NB	C++	X86	Executable Binary/Trace	Multi-Core	Transactional Accurate	1-2	1-10 kHz
	SimNow	C++	X86	Executable Binary	Core	Functional	—	>100 MIPS
Qualcomm	Qualcomm-Simulator	—	ARMv8/ARMv7	Trace	Multi-Core	Cycle Approximate	—	—

Note: "—" means unpublished information.

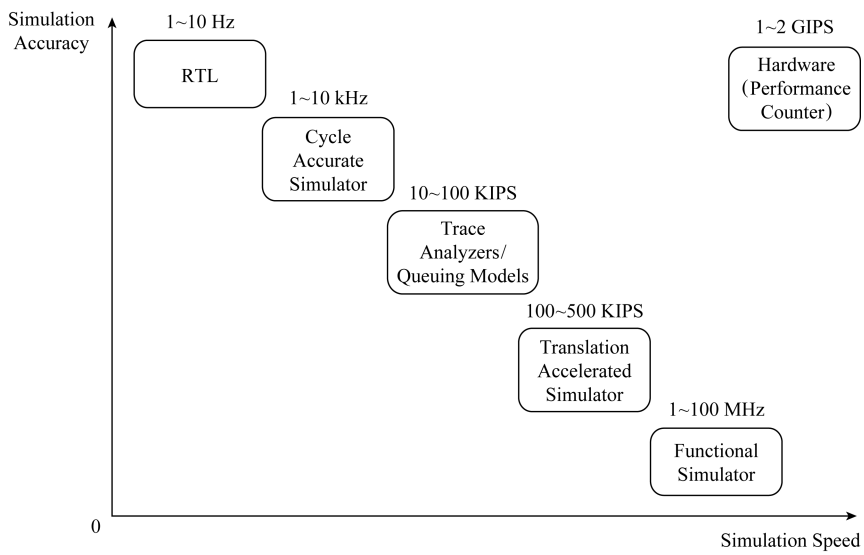


Fig. 7 Simulator speed and role

图 7 模拟器速度及作用

理器厂商使用模拟器辅助进行处理器开发的经验有4方面:

1) 处理器厂商一般有多个功能不同的模拟器用于辅助进行处理器开发.其中性能模拟器一般要同时支持踪迹和执行驱动2种模式,以方便在这2种模式之间切换,以此加快模拟速度.性能模拟器一般要求时钟精准,且与RTL校准后误差尽量小[23],其运行速度一般要比RTL仿真速度快1 000倍以上才有实际使用价值[23].性能模拟器还要支持多种调试手段,例如GDB、快速前进(fast forward)、检查点等,还要支持数据可视化功能,方便与RTL进行校准.为了加快模拟器仿真速度,要善于使用类似于SimPoint[7]的取样方法对负载进行处理.对于多核互联模拟器考虑到仿真速度,可以使用时钟近似模式和抽象的处理器核,但要可以精确反映性能变化趋势.

2) 一般都有硬件支持从硬件直接抓取真实程序执行踪迹或者事务踪迹[23-24],例如AMD可以直

接关闭 L2 抓取 bus 上的信息,IBM 也可以获取真实处理器上的执行踪迹.此外模拟器要可以和多种常用软件工具协同工作,以方便使用,例如踪迹获取工具、代码自动注释工具、文档产生工具等.

3) 模拟器开发投入很大,要持续与 RTL 进行校准[23].模拟器的开发首先是一项软件工程,因此好的软件架构模拟器成功的首要条件,相比而言计算机体系结构的知识也非常重要但是次要的[16].AMD 模拟器的微结构代码约有 10 万行,其他结构包含共享库约有 40 万行代码[16],因此模块化的设计、良好的代码接口、使用源代码管理工具等必不可少.在资源不足的情况下,基于开源模拟器开发模拟器也是不错的选择.

4) 模拟器校准时,需要通过微程序进行充分验证[8],此时可以跟验证团队紧密合作,共享验证微程序.性能模拟器编写语言一般选择兼顾开发和执行效率的 C++,其次是 C 语言.为了加速模拟器仿真速度,模拟部件尽量并行执行,且要尽可能支持可移植性.

3 性能模拟器校准

用于进行微结构探索的性能模拟器的准确度会直接影响微结构设计和改进的决策,因此需要对其进行校准.校准方法可以分为 2 种:1)在进行处理器设计时,与 RTL 进行校准;2)开源模拟器为了提高研究可信度,在目标硬件微结构数据未知的情况下,与已有的硬件平台进行校准,本节对上述校准方法进行总结.

3.1 与 RTL 进行校准

龙芯中科技术有限公司对 Sim-Godson 模拟器和 ICT-Godson 模拟器进行校准,后者与 RTL 几乎一样准确,其校准方法学与模拟器直接同 RTL 校准类似,具有一定参考价值,本节以此为例阐述模拟器与 RTL 的校准方法[8].

如图 8 所示,在处理器开发的逻辑设计阶段开始模拟器设计和校准工作,在其他阶段如果需要修改逻辑设计,则同时对模拟器进行修正.龙芯中科技术有限公司使用 2 种类型的负载[8]对模拟器进行校准,即微程序[35]和完整应用程序(主要是 SPEC CPU2000).

在校准初期,针对特定处理器模块编写特定微程序进行各模块校准[8].例如通过 if-else 循环测试分支预测器,以时钟周期为单位对比模拟器和 RTL 运行结果,发现不同后定位原因并修正模拟器.重复

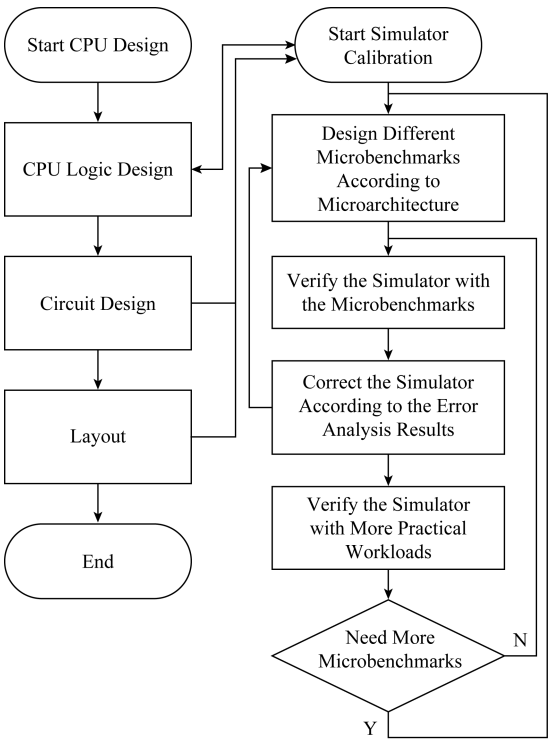


Fig. 8 Simulator calibration process

图 8 模拟器校准流程

上述步骤直到微程序验证结果的误差可控后,继续使用完整负载进行校准,此时如果结果显示误差原因指向某个模块,则增加对该模块进行校准的微程序并重复上述校准步骤.若一时无法定位误差原因,则重复使用微程序对误差进行分析.

龙芯中科技术有限公司在模拟器校准过程中还提出:1)分析程序运行结果并寻找误差原因是模拟器校准最重要的工作;2)很多由 SPEC CPU2000 发现的模拟器误差,都可以由一些简单微程序发现,而分析 SPEC CPU2000 的误差要比分析微基准程序困难得多;3)仅仅比较程序运行时间或 IPC(instructions per cycle)等宏观参数得出的结论有可能不正确,要综合考虑分支预测正确率等相关部件.例如在 Sim-Godson 校准过程中,浮点部件还没有校准之前,SPEC CPU2000 中多个程序 IPC 误差已经小于 10%,只有通过查看浮点部件延迟和发射策略等参数,才会发现浮点部件误差较大.

3.2 与真实硬件进行校准

在真实硬件微结构数据未知的情况下进行模拟器校准,需要首先通过程序测试获取处理器微结构详细信息.文献[36]认为模拟异构系统的模拟器如果不准确,会直接导致异构系统性能和功耗的严重误判,因此对模拟器 MARSSx86[37]与 Intel Core i7-

920 CPU 进行了校准,其校准分为 2 步:1)通过简单的微程序(general matrix multiplication, GEMM)[38]校准处理器核流水线;2)基于校准后的流水线继续进行访存系统校准.

本文以校准计算类指令的执行周期为例介绍其校准方法.模拟器与真实处理器校准过程中,不同类型指令执行延迟是最重要的校准参数,但是由于 Intel 未公布该微结构参数,因此文献[36]通过设计微程序对指令执行延迟进行测算.具体地:

1) 查询 Intel 编程手册获取 i7-920 处理器每个时钟周期可以执行 4 个双精度浮点运算(double precision FLOPs/cycle,FPC),每个双精度浮点运算的寄存器宽度是 64 b.指令 mulpd 和 addpd 使用的寄存器是 128 b 宽,上述每条指令每个周期需要 2 个双精度运算.

2) 设计微程序,使得每个 mulpd 和 addpd 的指令组合恰好在 i7-920 处理器上一个周期执行完成.

3) 如图 9 所示,定义寄存器复用距离(register reuse distance,RRD),即相同指令的目的寄存器号相同情况下2 条指令之间的延迟周期数,例如图 9 中的指令 addpd 复用距离是 3 周期.

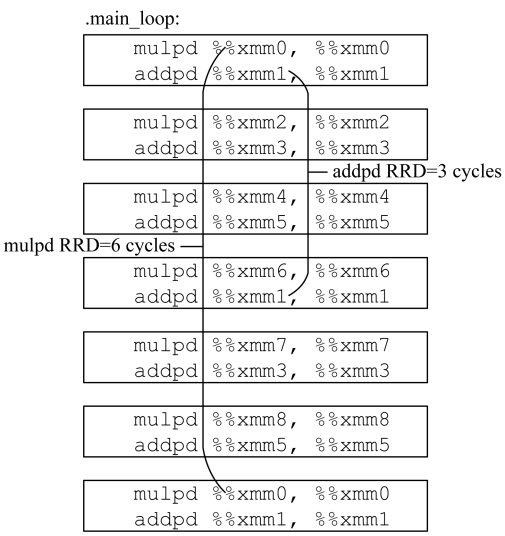


Fig. 9 Microbenchmark for testing instruction latencies

图 9 测试指令延迟的微程序

4) 如果 addpd 指令和 mulpd 指令的执行延迟分别小于 3 和 6 周期,则该微程序的执行峰值速度应该能达到 4FPC,反之指令 addpd 和 mulpd 必须要等待指令之间寄存器相关导致的依赖解决后才能继续执行,因此会导致浮点计算部件无法达到峰值性能.经过测试,图 9 中的微程序可以达到 3.94 FPC,可以断定 addpd 和指令 mulpd 的指令执行周期不大于 3 和 6 周期.

5) 通过反复调整 addpd 和 mulpd 的复用距离,可以得到指令 addpd 和 mulpd 执行周期分别为 3 和 5.通过上述程序获取的数据调整模拟器中指令执行延迟.

类似地,对其他流水线微结构信息例如发射宽度、缓存系统架构等,都需要通过微程序测试获取相关数据后进行校准[36],基于此再进行访存子系统的校准,本文不再详述.

4 模拟器优化方法

随着处理器核性能越来越高,核数越来越多,模拟器的规模也越来越大,模拟速度及模拟器可扩展性越来越成为模拟器发展的制约因素,因此模拟器需要进行优化.本文把模拟器的优化分为纵向优化(scale up)和横向优化(scale out).纵向优化是指通过技术手段,对串行执行或者小规模模拟器进行加速;横向优化是指对于模拟多核或者多处理器的模拟器,在付出较小代价的同时,获取模拟器多核扩展性的大幅提升.本节对新兴的优化方法进行着重介绍,对常规的优化方法进行简单总结.

4.1 模拟器纵向优化

4.1.1 基于 FPGA 加速的模拟器

FPGA 加速模拟技术[34](FPGA-accelerated simulation technologies, FAST)可以通过 FPGA 对时钟精准模拟器进行加速.基于该技术实现的模拟器可以分为 2 部分,模拟器前端运行一个功能模拟器,可以运行完整的全系统.模拟器后端通过 FPGA 硬件实现处理器微结构,因此可以对分支预测、流水线、缓存等详细微结构进行时钟精准模拟.模拟器运行时,前端功能模拟部分把指令执行结果等信息通过缓冲区输入给运行在 FPGA 上的模拟器后端,后端基于缓冲区中的信息校正微结构推测执行(例如分支预测等)过程中的执行结果.FPGA 加速模拟技术同时具有软件的灵活性,并兼顾了运行速度,其运行速度一般要比基于软件的时钟精准模拟器快 1 个数量级[34].类似模拟器还有 RAMP[39],STIMUL[40]等.

4.1.2 二进制翻译

模拟器二进制翻译技术是把模拟器所模拟的目标架构的指令,翻译成模拟器所运行的宿主机架构的指令然后进行执行.具体地,当模拟器执行目标指令时,模拟器把目标指令替换为宿主机上事先定义好的函数或者指令集,使得所模拟的目标架构指令

执行速度接近于宿主机执行速度.

二进制翻译虽然加速了模拟器执行速度,但其缺点是目标架构的指令被宿主机指令替换,因此执行踪迹无法获取,同时导致模拟器与宿主机架构绑定,可移植性变差.基于二进制翻译进行加速的模拟器有 SimOS[13],QEMU[32]等.

4.1.3 直接执行(基于 KVM 虚拟化和取样技术)

直接执行是指模拟器的目标架构和模拟器的宿主架构相同的情况下,模拟器在运行时直接在宿主机上执行目标架构指令的加速方法,直接执行可以认为是二进制翻译中的特例.文献[41]把直接执行和取样方法结合起来对模拟器进行加速,基于 KVM 虚拟化在多个程序取样之间进行快速推进,其模拟速度可以达到宿主机速度的 63%,约2.0GIPS(giga instructions per second).通过 IPC 参数对模拟器误差进行评估,平均误差约 2.2%.基于 KVM 的模拟器加速方法是基于 SMARTS[42]取样方法基础上发展而来,如图 10(a)所示.SMARTS 取样方法为了平衡模拟精度和模拟器仿真速度,把运行时程序分为 3 个模式:1)功能预热模式.为了对需要长时间训练的微结构状态进行保存,模拟了缓存和分支预测器,除此之外该模式把每条指令当作静态流水线原子指令进行模拟,并且没有时序信息.2)详细预热模式.使用乱序执行处理器对整个系统进行详细模拟,并

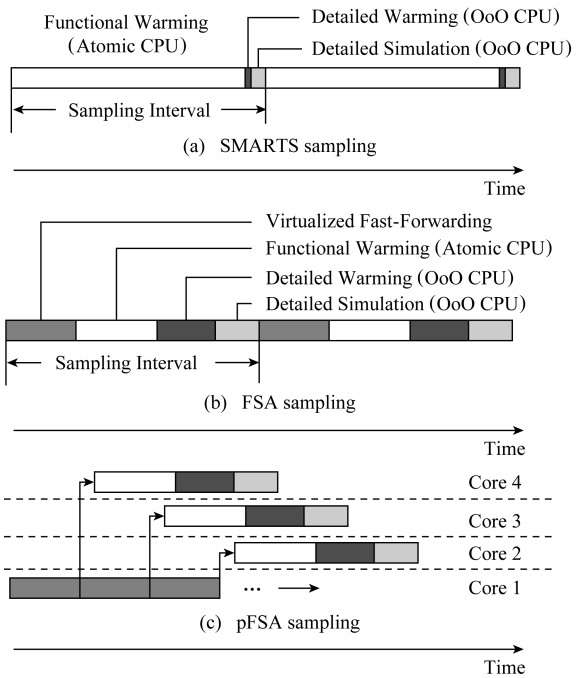


Fig. 10 Comparison of different simulation sampling strategies

图 10 关于多种模拟取样策略之间的对比

对 CPU 内部的某些微结构(例如读写缓存、重排序缓存等)进行预热.3)详细模拟模式.详细模拟处理器所有微架构,并对需要的微结构数据进行测量.上述方法需要功能预热阶段执行大量程序指令,导致花费大量时间,为了对取样进行加速,文献[41]提出了 FSA(full speed ahead)取样方法.如图 10(b)所示,基于 SMARTS 取样基础上添加了一个新的模式:虚拟化快速推进模式.上述新增模式可以使用基于 KVM 的虚拟化技术把约 95%指令直接运行在宿主机上,可以大大加快原来功能预热模式的执行速度,但为了保证对缓存和分支预测器的预热,保留了 SMARTS 中功能预热的方式.

FSA 模式虽然加快了功能预热模式下指令执行速度,但是还有大约 75%~95%的时间用于详细预热和详细模拟阶段.为了对上述阶段进行并行化加速,文献[41]继续提出了 pFSA(parallel full speed ahead),如图 10(c)所示,为了使得每一个取样点独立并行执行,需要复制取样点之前模拟器状态.在完成一个取样点详细模拟之前就使得模拟器运行到下一个取样点继续并行执行.具体地,模拟器完全以虚拟化快速推进模式运行在处理器核 1 上,其运行速度可以接近宿主机.当需要对模拟器进行取样分析时,复制模拟器状态到处理器核 2 上,开始以 SMARTS 方式进行取样处理,以此类推.如果复制模拟器状态的开销足够低,则模拟器可以接近宿主机的速度运行在宿主机器上.

上述方法被集成到了 gem5[2]模拟器中,类似的模拟器还有基于 Xen 半虚拟化环境的 PTLsim[43],因此运行 PTLsim 模拟器的操作系统要进行定制化修改,导致其可移植性相对不如 KVM 好(KVM 目前是 Linux 标准组件),并缺乏对一些重要的底层硬件组件(例如中断时钟、IO 设备等)的模拟.文献[41]的方法虽然可以使得模拟器运行速度接近宿主机,但其缺点是:1)目标指令集要和宿主机相同;2)需要 KVM 支持;3)对于特权指令仍需进行模拟.

4.2 模拟器横向优化

4.2.1 区间模拟

区间模拟技术[44]是一种新型多核模拟器加速方法,通过提高模拟器的抽象层级并且对处理器核使用数学分析模型进行抽象,而省略了通过流水线对指令执行时间进行详细追踪的步骤,因此可以一定程度上通过数学模型替换处理器核的时钟精准模拟方式,大幅提高模拟速度并降低模拟器开发难度.

其核心思想如图 11 所示,指令流水线的正常执行流会被分支预测错误和 TLB(translation lookaside buffer)、cache 缺失等事件打断并分成不同的区间长度,上述缺失事件及频率由模拟器决定,而区间的长度由数学分析模型计算.因此综合使用模拟各个部件的模拟器并结合数学分析模型,就可以对在多核处理器上同时执行的多线程程序性能进行评估,Sniper[45]和 COTSon[46]就是以这种模式实现的模拟器.文献[44]对 Sniper 模拟器使用 SPEC CPU2000 和多线程 PARSEC 进行测试,与 M5[47]全系统时钟精确模拟器进行精度和仿真速度的对比,运行 SPEC CPU2006 单线程的误差是 5.9%,运行 PARSEC 多线程测试的误差是 4.6%,但其速度要比时钟精确方式快一个数量级.文献[44]的区间模拟器分析模型的代码量只有 1 000 行 C 代码,而 M5[47]中乱序执行代码量接近 2.8 万行.开发区间模拟器目的不是替代时钟精确模拟器,而是在牺牲小部分精确性的同时,大幅提高模拟器开发和仿真速度,因此可以用于快速探索多核处理器架构的设计和对粗粒度微架构和系统级架构进行快速取舍,而时钟精确模拟可以用于对某些微架构进行详细探索.

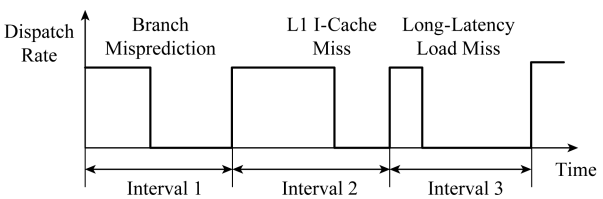


Fig. 11 Analysis performance on an interval basis determined by disruptive miss events

图 11 基于缺失导致的区间进行性能分析

4.2.2 SST 模拟框架

SST(structural simulation toolkit)[48-49]是为了对大规模高性能计算系统进行模拟而诞生,可以模拟多达 512 个处理器节点,其目标是成为设计和评估未来高性能计算系统的标准框架.基于 SST 模拟器可以把多个现有的模拟器按需集成起来,组成一个模块化并行模拟系统.SST 已经内建多种处理器、内存系统、网络系统等模型,并提供简易用户接口,方便快速扩展新的模型.为了提高灵活度,SST 提供一系列组件可以在精度和模拟时间之间权衡.SST 的软件架构由模拟器核心部分和可动态插拔的组件组成,基于 MPI 和离散事件模拟技术构建,如图 12 所示,其中模拟器核心部分提供模拟服务,包括模拟器配置和启动、功耗面积评估、检查点的支持、模拟器数据收集等.SST 模拟核心还提供通用接口用于连接供应商提供的组件或者开源的模拟器组件.目前基于 SST 开发的模拟器有 MacSim[50],ExaSAT[51]等,自从发布以来 SST 被引用了 208 次.

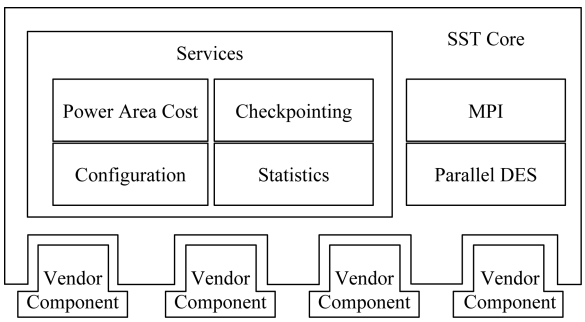


Fig. 12 SST structure

图 12 SST 框架

4.2.3 并行模拟加速

为了对大规模系统进行模拟,需要可以并行运行的模拟环境,这催生了并行离散事件仿真(parallel discrete event simulation, PDES)技术[52-53].PDES 可以把离散事件分解到多处理器上并行执行,可以大幅提高模拟速度,因此被广泛应用到工程、计算机科学、经济、军事[54]等领域.

并行模拟器虽然能加快基于离散事件仿真的模拟器运行速度,但是需要维护各个处理单元之间的同步关系,随着处理单元的规模越来越大,其同步开销占整个模拟器运行时间比例也越来越大,因此其可扩展性在规模巨大时成为急需解决的问题.IBM 使用的 BGLSim[21]模拟器就是基于 PDES 技术构建的.

5 相关模拟器介绍

随着各种体系结构和处理器设计方法学的发展,体系结构模拟器也相应地出现了一些新的形态,本节对近期出现的新型模拟器进行总结介绍.

5.1 异构模拟器

新型的专用加速器、GPU 已成为计算机体系结构研究和处理器设计的重点.相应地,研究人员开发了多款开源的模拟器对新型计算机体系结构进行模拟.为方便研究人员选择合适的模拟器平台和工具进行实验,本文总结专用加速器和 GPU 模拟器及相关工具如表 2 所示,其中被引用次数高且持续维护的模拟器相对更适合进行异构原型系统的构建和异构体系结构的研究.

Table 2 Comparison of Heterogeneous Simulators

表 2 异构模拟器对比

Simulator	Published Year	Developer and Link	Forward Mode	Citation Count	Language	Target Architecture	Operating System	Last Update Date
GPGPU-Sim[57]	2009	University of British Columbia https://github.com/gpgpu-sim/gpgpu-sim_simulations	Functional/ Cycle Accurate	1232	C++	GPGPU	×	2018-11
Qsilver[58]	2004	The University of Virginia Link not found	Cycle Accurate	96		GPU	×	
gem5-APU[59]	2015	AMD https://gem5.googlesource.com/amd/gem5	Cycle Accurate		C++ Python	APU	√	2018-11
GPU Ocelot [60]	2009	Georgia Institute of Technology http://code.google.com/p/gpuocelot/	Functional	25	C++	GPU	×	2013-04
FusionSim[61]	2013	University of Toronto https://sites.google.com/site/fusionsimulator/	Cycle Accurate	16	C++	CPU-GPU	×	2012-06
MV5[62]	2011	Argonne National Laboratory https://sites.google.com/site/mv5sim	Cycle Accurate	11	C++ Python	GPU-like SIMD/SIMT	×	2011-02
GpuTejas[63]	2014	Indian Institute of Technology http://www.cse.iitd.ac.in/tejas/gputejas/index.html	Cycle Accurate	5	Java	GPU	×	2019-05
gem5-gpu[64]	2014	University of Wisconsin-Madison https://gem5-gpu.cs.wisc.edu/wiki/	Functional/ Cycle Accurate	126	C++ Python	CPU-GPU	√	2017-01
HSAemu[65]	2014	National Tsing Hua University https://github.com/SSLAB-HSA/HSAemu	Functional/ Cycle Accurate	10	C	Heterogeneous System	√	2013-11
FATSEA[66]	2010	University of Murcia (Spain) Link not found	Cycle Accurate	1	C++	GPU	×	
Barra[67]	2009	Univ.de Perpignan https://gforge.inria.fr/scm/?group_id=5827	Functional	100	C++	GPGPU	×	2015-08
ATTILA[68]	2006	Norwegian University of Science and Technology https://github.com/attila-gpu/attila-sim	Cycle Accurate	112	C++	GPU	×	2015-04
Multi2Sim [69]	2012	Northeastern University https://github.com/Multi2Sim/multi2sim	Cycle Accurate	351	C++	CPU-GPU	×	2018-04
MacSim [50]	2012	Georgia Tech https://github.com/gthparch/macsim	Cycle Accurate	39	C++	CPU-GPU	×	2018-11
MGSim[70]	2013	Northeastern University https://github.com/svp-dev/mgsim	Cycle Accurate	19	C++	CPU-GPU	×	2017-04

Note: √ means support operating system; × means don't support operating system.

5.2 硬件构建语言 Chisel

加州大学伯克利分校设计的开放指令集 RISC-V 已成为处理器设计和研究领域的热门,为了实现处理器的敏捷开发,伯克利分校还开发了高度参数化的硬件构建语言 Chisel[55].使用 Chisel 语言设计处理器,可以直接使用面向对象的设计方法学描述处理器功能,这与传统意义上开发周期精确模拟器的方式很像,但特别之处在于:通过编写一次硬件代码可以生成包含 C++时钟精准模拟器、FPGA Verilog 和 ASIC Verilog 这 3 个目标[55].具体地在生成模拟器方面,基于最新 Chisel3 编写的硬件代码可以产生 Firrtl 中间描述语言(intermediate representation, IR),从 Firrtl 可以直接翻译或者转换成 Verilog,进而通过 Verilator 工具可以生成时钟精准的 C++模拟器和测试框架.但是,通过这种方式生成的模拟器代码可读性和可修改性都比较差.

Chisel 语言使逻辑设计和模拟器开发得到了统一,使得处理器设计效率提高一个数量级[56],因此可以大幅加速硬件设计,这是新的硬件敏捷开发方

法学,同时也是未来处理器设计和模拟器发展的一个重要方向.

5.3 自研(In-house)模拟器

一些体系结构文献在进行实验评估时,使用的是自研模拟器,表3对体系结构会议ISCA(International Symposium on Computer Architecture),MICRO(International Symposium on Microarchitecture),ASPLOS(International Conference on Architectural Support for Programming Languages and Operating Systems),HPCA(International Symposium on High Performance Computer Architecture)中近几年出现的In-house模拟器进行了总结介绍,以方便科研人员在开发自研模拟器时进行参考.从表3可以看出,有多个自研模拟器是基于trace(一般可基于Intel Pin工具获取)作为模拟器的输入,此类模拟器适合于对访存系统进行建模和分析,其优点是运行速度快、构建模拟器相对容易.但是由于Pin工具运行于用户态,因此其缺点是只能对用户态应用程序进行性能分析,无法对操作系统参与的行为(例如调度、内存管理等)进行分析.

Table 3 Introduction of Some In-house Simulators of Architecture Conference

表3 部分体系结构会议中使用的In-house模拟器介绍

Conference	Simulator Introduction in the Paper	Target Architecture
ASPLOS 2017[82]	Intel Pin trace tool based CPU performance simulator	X86
ISCA 2016[83]	Trace based cache performance simulator	X86
HPCA 2016[84]	Intel Pin trace tool based CPU performance simulator which is unable to simulate the effects of wrong-path instructions and the operating system activity is not simulated	X86
HPCA 2016[85]	Intel Pin trace tool based CPU performance simulator which is unable to simulate the effects of wrong-path instructions and the operating system activity is not simulated. The virtual to physical address translation is performed on the simulator via the Linux's pagemap interface	X86
HPCA 2016[86]	Trace based CPU memory subsystem performance simulator	—
HPCA 2017[29]	QEMU based cycle-approximate performance simulator which can run unmodified operating system	ARMv7/ARMv8
HPCA 2019[71]	Cycle accurate cache simulator which can be used for estimating the time and energy for cache	—
HPCA 2019[30]	QEMU based cycle-approximate performance simulator which can run unmodified operating system	ARMv7/ARMv8
ISCA 2016[87]	Cycle accurate simulator and runs samples taken by SimPoint	X86
ISCA 2016[88]	Trace based performance simulator	—
HPCA 2019[89]	Trace based performance simulator	—
MICRO 2016[90]	Cycle accurate simulator	—
ISCA 2018[91]	Cycle accurate simulator	X86
ISCA 2010[92]	Trace based cache performance simulator	—
MICRO 2017[31]	QEMU based cycle-approximate performance simulator which can run unmodified operating system.	ARMv7/ARMv8

Note: "—" means unpublished information

5.4 专用模拟器

本文主要围绕处理器性能模拟器进行介绍,但是处理器的设计是一个不断取舍的过程,其他诸如功耗、面积等方面也是处理器设计过程中必须考虑的因素,这方面也有对应的专用模拟器模拟和分析.本文所述专用模拟器是从体系结构层面,针对处理器某个部件或者某些方面进行编写的模拟器,例如cache模拟器、功耗模拟器、面积模拟器等.这类模拟器一般用于在处理器设计之初对其功耗和面积进行初步评估和取舍或者用于对科研论文中提出的创新思想所带来的开销进行评估.本节对主流会议中所使用的开源专用模拟器进行总结介绍.

2.3节所述的TagSim[23]和文献[71]中所用的模拟器就是专用的cache模拟器,此类模拟器一般只对数据cache进行模拟,主要关注访存指令,因此普遍基于二进制插桩工具(Intel Pin工具和DynamoRIO[72]等)编写.

CACTI[73],Wattch[74],SimplePower[75],PowerTimer[76]可以对处理器能耗进行评估,文献[76]从微结构级对功耗模拟器进行了详细总结,本文不再

赘述.Orion[77]模拟器可以针对片上网络(network on chip, NoC)的功耗、面积进行评估.CACTI 模拟器除了可以评估处理其功耗外,还可以以时序为限制的情况下对处理器面积进行详细评估,基于此还衍生出了用于评估 3D 封装 DRAM 的模拟器 CACTI-3DD[78]、用于评估片外 IO 的模拟器 CACTI-IO[79]和面向 SRAM 的支持降低漏电功耗技术的模拟器 CACTI-P[80]等.McPAT[81]集成了 CACTI-P[80]模拟器,是首款集成了对功耗、面积和时序进行模拟的多核和众核模拟器,目前已经被集成到 gem5[2],Sniper[45]等多款性能模拟器中.McPAT[81]和 CACTI[73]普遍被用于主流会议中进行处理器功耗、面积的评估.

6 总结及展望

体系结构模拟器对于处理器设计非常重要,微结构的开发、处理器的迭代和性能提升离不开模拟器的辅助.基于模拟器进行处理器设计,未来还有很多挑战.本文从处理器设计的角度出发,首先介绍了开源模拟器在处理器设计中的地位;其次,对处理器厂商使用模拟器辅助进行处理器设计的经验进行了介绍和总结,并对性能模拟器的校准方法进行了介绍;最后对模拟器优化方法和新型模拟器进行了介绍.

随着计算机体系结构的发展,异构处理器、众核处理器、虚拟化技术成为高性能计算未来发展的重要方向.首先,随着处理器核数越来越多,未来多核互联模拟器会变得越来越复杂.此外,随着虚拟化的发展,系统整体性能评估需要在模拟器上运行虚拟化环境、完整的操作系统及负载.在处理器功耗越来越重要的情况下,通过模拟器对功耗进行评估也势在必行,目前这方面还有很大提升空间.上述挑战都是未来体系结构模拟器研发要面临的问题.

参 考 文 献

[1] Mukherjee S S, Adve S V, Austin T, et al. Performance simulation tools [J]. Computer, 2002, 35(2): 38-39

[2] Binkert N, Beckmann B, Black G, et al. The gem5 simulator [J]. ACM SIGARCH Computer Architecture News, 2011, 39(2): 1-7

[3] Nowatzki T, Menon J, Ho C, et al. gem5, GPGPUSim, McPAT, GPUWattch, "Your favorite simulator here" considered harmful [EB/OL]. 2014 [2019-01-23]. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.699.3967&rep=rep1&type=pdf

[4] Butko A, Garibotti R, Ost L, et al. Accuracy evaluation of gem5 simulator system [C] //Proc of the 7th Int Workshop on Reconfigurable and Communication-Centric Systems-on-Chip. Piscataway, NJ: IEEE, 2012: 1-7

[5] Gutierrez A, Pusdesris J, Dreslinski R G, et al. Sources of error in full-system simulation [C] //Proc of the 14th Int Symp on Performance Analysis of Systems and Software (ISPASS). Piscataway, NJ: IEEE, 2014: 13-22

[6] Brooks D, Tiwari V, Martonosi M. Wattch: A framework for architectural-level power analysis and optimizations [C] //Proc of the 27th Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2000: 83-94

[7] Sherwood T, Perelman E, Hamerly G, et al. Automatically characterizing large scale program behavior [J]. ACM SIGARCH Computer Architecture News, 2002, 30(5): 45-57

[8] Zhang Fuxin, Zhang Longbing, Hu Weiwu. Sim-Godson: A Godson processor simulator based on SimpleScalar [J]. Chinese Journal of Computers, 2007, 30(1): 68-73 (in Chinese)
(张福新, 章隆兵, 胡伟武. 基于 SimpleScalar 的龙芯 CPU 模拟器 Sim-Godson [J]. 计算机学报, 2007, 30(1): 68-73)

[9] Gao Xiang, Zhang Fuxin, Tang Yan, et al. SimOS-Goodson: A Goodson-processor based multi-core full-system simulator [J]. Journal of Software, 2007, 18(4): 1047-1055 (in Chinese)
(高翔, 张福新, 汤彦. 基于龙芯 CPU 的多核全系统模拟器 SimOS-Goodson [J]. 软件学报, 2007, 18(4): 1047-1055)

[10] Gao Xiang. Research on memory simulation and optimizations in CMPs [D]. Hefei: University of Science and Technology of China, 2007 (in Chinese)
(高翔. 多核处理器的访存模拟与优化技术研究[D]. 合肥: 中国科学技术大学, 2007)

[11] Austin T, Larson E, Ernst D. SimpleScalar: An infrastructure for computer system modeling [J]. Computer, 2002, 35(2): 59-67

[12] Desikan R, Burger D, Keckler S W, et al. Sim-alpha: A validated, execution-driven Alpha 21264 simulator [EB/OL]. [2019-01-23]. https://www.cs.utexas.edu/ftp/dburger/papers/TR-01-23.pdf

[13] Rosenblum M, Herrod S A, Witchel E, et al. Complete computer system simulation: The SimOS approach [J]. IEEE Parallel Distributed Technology: Systems Applications, 1995, 3(4): 34-43

[14] Barroso L A, Gharachorloo K, Bugnion E. Memory system characterization of commercial workloads [C] //Proc of the 25th Annual Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 1998: 3-14

[15] Keller T. SimOS-PPC: Full system simulation of PowerPC architecture [EB/OL]. (1999-03-10) [2019-01-23]. https://www.cs.utexas.edu/users/cart/arch/spring99/slides/Keller990412.pdf

[16] Michael K. Experiences in simulation at IBM [EB/OL]. 2012 [2019-01-23]. http://csa.cs.pitt.edu/presentations/csa2012_kistler-michael.pdf

[17] Bohrer P, Peterson J, Elnozahy M, et al. Mambo: A full system simulator for the PowerPC architecture [J]. ACM SIGMETRICS Performance Evaluation Review, 2004, 31 (4): 8-12

[18] Asaad S, Bellofatto R, Brezzo B, et al. A cycle-accurate, cycle-reproducible multi-FPGA system for accelerating multi-core processor simulation [C] //Proc of the ACM/SIGDA Int Symp on Field Programmable Gate Arrays. New York: ACM, 2012: 153-162

[19] Cadence. Palladium Z1 enterprise emulation platform datasheet [EB/OL]. [2019-01-23]. https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/system-design-verification/palladium-z1-ds.pdf

[20] Magnusson P S, Christensson M, Eskilson J, et al. Simics: A full system simulation platform [J]. Computer, 2002, 35 (2): 50-58

[21] Ceze L, Strauss K, Almasi G, et al. Full circle: Simulating Linux clusters on Linux clusters [C] //Proc of the 4th LCI Int Conf on Linux Clusters: The HPC Revolution. New York: ACM, 2003

[22] OpenSim Ltd. OMNeT++ [EB/OL]. [2019-01-23]. http://omnetpp.org/

[23] Barnes L. Performance modeling and analysis at AMD a guided tour [EB/OL]. (2007-04-27) [2019-01-23]. http://ispass.org/ispass2007/keynote2.pdf

[24] Lahiri K. Performance analysis for modern server CPUS [EB/OL]. [2019-01-23]. https://www.cse.iitk.ac.in/users/biswap/CASS18/performance-AMD.pdf

[25] Bedichek R. SimNow™: Fast platform simulation purely in software, 8 [R]. New York: ACM, 2004

[26] Harbor S. Disrupting the datacenter: Qualcomm Centriq™ 2400 processor [EB/OL]. (2017-11-08) [2019-01-23]. https://www.qualcomm.com/media/documents/files/qualcomm-centriq-2400-media-deck.pdf

[27] Wolford B, Speier T, Bhandarkar D. Qualcomm Centriq™ 2400 processor [EB/OL]. [2019-01-23]. https://www.qualcomm.com/media/documents/files/qualcomm-centriq-2400-processor.pdf

[28] Pham B, Hower D, Bhattacharjee A, et al. TLB shootdown mitigation for low-power many-core servers with L1 virtual caches [J]. IEEE Computer Architecture Letters, 2018, 17 (1): 17-20

[29] Hower D R, Cain H W, Waldspurger C A. PABST: Proportionally allocated bandwidth at the source and target [C] //Proc of the 23rd Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2017: 505-516

[30] Perais A, Sheikh R, Yen L, et al. Elastic instruction fetching [C] //Proc of the 25th Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2019: 478-490

[31] Sheikh R, Cain H W, Damodaran R. Load value prediction via path-based address prediction: Avoiding mispredictions due to conflicting stores [C] //Proc of the 50th Int Symp on Microarchitecture. New York: ACM, 2017: 423-435

[32] Bellard F. QEMU, a fast and portable dynamic translator [C] //Proc of the USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2005: 41-46

[33] Chiou D, Sunwoo D, Angepat H, et al. Parallelizing computer system simulators [C] //Proc of the 22nd Int Symp on Parallel and Distributed Processing. Piscataway, NJ: IEEE, 2008: 1-5

[34] Chiou D, Sunwoo D, Kim J, et al. FPGA-Accelerated simulation technologies (FAST): Fast, full-system, cycle-accurate simulators [C] //Proc of the 40th Int Symp on Microarchitecture. New York: ACM, 2007: 249-261

[35] Desikan R, Burger D, Keckler S W. Measuring experimental error in microprocessor simulation [C] //Proc of the 28th Annual Int Symp on Computer Architecture. New York: ACM, 2001: 266-277

[36] Asri M, Pedram A, John L K, et al. Simulator calibration for accelerator-rich architecture studies [C] //Proc of the 11th Int Conf on Embedded Computer Systems: Architectures, Modeling and Simulation. Piscataway, NJ: IEEE, 2016: 88-95

[37] Patel A, Afram F, Chen Shunfei, et al. MARSS: A full system simulator for multicore x86 CPUs [C] //Proc of the 48th ACM/EDAC/IEEE Design Automation Conf. Piscataway, NJ: IEEE, 2011: 1050-1055

[38] Van Zee F G, Van de Geijn R A. BLIS: A framework for rapidly instantiating BLAS functionality [J]. ACM Transactions on Mathematical Software, 2015, 41(3): 14:1-14:33

[39] Patterson D A. RAMP: Research accelerator for multiple processors—A community vision for a shared experimental parallel HW/SW platform [C] //Proc of the 6th Int Symp on Performance Analysis of Systems and Software. Piscataway, NJ: IEEE, 2006: 19-21

[40] Chung M K, Shim H, Kyung C M. Performance improvement of multiprocessor simulation by optimizing synchronization and communication [C] //Proc of the 16th Int Workshop on Rapid System Prototyping. Piscataway, NJ: IEEE, 2005: 158-164

[41] Sandberg A, Nikoleris N, Carlson T E, et al. Full speed ahead: Detailed architectural simulation at near-native speed [C] //Proc of the 15th Int Symp on Workload Characterization. Piscataway, NJ: IEEE, 2015: 183-192

[42] Wunderlich R E, Wenisch T F, Falsafi B, et al. SMARTS: Accelerating microarchitecture simulation via rigorous statistical sampling [C] //Proc of the 30th Annual Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2003: 84-95

[43] Yourst M T. PTLsim: A cycle accurate full system x86-64 microarchitectural simulator [C] //Proc of the 8th Int Symp on Performance Analysis of Systems Software. New York: ACM, 2007: 23-34

[44] Genbrugge D, Eyerman S, Eeckhout L. Interval simulation: Raising the level of abstraction in architectural simulation [C] //Proc of the 16th Int Symp on High-Performance Computer Architecture. Piscataway, NJ: IEEE, 2010: 1-12

[45] Carlson T E, Heirmant W, Eeckhout L. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation [C] //Proc of the 25th Int Conf for High Performance Computing, Networking, Storage and Analysis. Piscataway, NJ: IEEE, 2011: 1-12

[46] Ryckbosch F, Polfliet S, Eeckhout L. Fast, accurate, and validated full-system software simulation of x86 hardware [J]. IEEE Micro, 2010, 30(6): 46-56

[47] Binkert N L, Dreslinski R G, Hsu L R, et al. The M5 simulator: Modeling networked systems [J]. IEEE Micro, 2006, 26(4): 52-60

[48] SST Development Team. Structural simulation parallel discrete event framework and architectural simulation components [EB/OL]. [2019-01-23]. https://github.com/sstsimulator

[49] Rodrigues A F, Hemmert K S, Barrett B W, et al. The structural simulation toolkit [J]. SIGMETRICS Performance Evaluation Review, 2011, 38(4): 37-42

[50] Kim H, Lee J, Lakshminarayana N B, et al. Macsim: A CPU-GPU heterogeneous simulation framework user guide [EB/OL]. Georgia Institute of Technology, 2012. [2019-01-23]. http://comparch.gatech.edu/hparch/macsim/macsim.pdf

[51] Unat D, Chan C, Zhang Weiqun, et al. ExaSAT: An exascale co-design tool for performance modeling [J]. International Journal of High Performance Computing Applications, 2015, 29(2): 209-232

[52] Perumalla K S. Parallel and distributed simulation: Traditional techniques and recent advances [C] //Proc of the 18th Winter Simulation Conf. Piscataway, NJ: IEEE, 2006: 84-95

[53] Fujimoto R M. Parallel discrete event simulation [J]. Communications of the ACM, 1990, 33(10): 30-53

[54] Steinman J S. The WarpIV simulation kernel [C] //Proc of the 19th Workshop on Principles of Advanced and Distributed Simulation. Piscataway, NJ: IEEE, 2005: 161-170

[55] 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. New York: ACM, 2012: 1216-1225

[56] 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)

[57] Bakhoda A, Yuan G L, Fung W W L, et al. Analyzing CUDA workloads using a detailed GPU simulator [C] //Proc of the 9th Int Symp on Performance Analysis of Systems and Software. Piscataway, NJ: IEEE, 2009: 163-174

[58] Sheaffer J W, Luebke D, Skadron K. A flexible simulation framework for graphics architectures [C] //Proc of the 11th ACM SIGGRAPH/EUROGRAPHICS Conf on Graphics Hardware. New York: ACM, 2004: 85-94

[59] Bradford B. AMD's gem5 APU simulator [EB/OL]. (2015-06-14) [2019-01-23]. http://www.gem5.org/wiki/images/7/7a/2015_ws_03_amd-apu-model.pdf

[60] Gregory D, Andrew K, Sudhakar Y. Gpuocelot: A binary translation framework for PTX [EB/OL]. [2019-01-23]. http://code.google.com/p/gpuocelot/

[61] Zakharenko V, Aamodt T, Moshovos A. Characterizing the performance benefits of fused CPU/GPU systems using FusionSim [C] //Proc of the 16th Design, Automation Test in Europe Conf Exhibition. Piscataway, NJ: IEEE, 2013: 685-688

[62] Meng Jiayuan, Skadron K. A reconfigurable simulator for large-scale heterogeneous multicore architectures [C] //Proc of the 11th Int Symp on Performance Analysis of Systems and Software. Piscataway, NJ: IEEE, 2011: 119-120

[63] Malhotra G, Goel S, Sarangi S R. GpuTejas: A parallel simulator for GPU architectures [C] //Proc of the 21st Int Conf on High Performance Computing (HiPC). Piscataway, NJ: IEEE, 2014: 1-10

[64] Power J, Hestness J, Orr M S, et al. Gem5-GPU: A heterogeneous CPU-GPU simulator [J]. IEEE Computer Architecture Letters, 2015, 14(1): 34-36

[65] Ding JiunHung, Hsu W, Jeng B, et al. HSAemu—A full system emulator for HSA platforms [C] //Proc of the 9th Int Conf on Hardware/Software Codesign and System Synthesis. Piscataway, NJ: IEEE, 2014: 1-10

[66] Østby K, Aragón J, Garcia J, et al. FATSEA—An architectural simulator for general purpose computing on GPUs [C] //Proc of the 2nd Workshop on Rapid Simulation & Performance Evaluation: Methods and Tools. New York: ACM, 2010

[67] Collange S, Daumas M, Defour D, et al. Barra: A parallel functional simulator for GPGPU [C] //Proc of the 18th Int Symp on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. Piscataway, NJ: IEEE, 2010: 351-360

[68] del Amor M Á M, Gonzalez C, Roca J, et al. ATTILA: A cycle-level execution-driven simulator for modern GPU architectures [C] //Proc of the Int Symp on Performance Analysis of Systems and Software. Piscataway, NJ: IEEE, 2006: 231-241

[69] Ubal R, Jang B, Mistry P, et al. Multi2Sim: A simulation framework for CPU-GPU computing [C] //Proc of the 21st Int Conf on Parallel Architectures and Compilation Techniques. Piscataway, NJ: IEEE, 2012: 335-344

[70] Poss R, Lankamp M, Yang Qiang, et al. MGSim—A simulation environment for multi-core research and education [C] //Proc of the Int Conf on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS). Piscataway, NJ: IEEE, 2013: 80-87

[71] Wang Xiaowei, Yu Jiecao, Augustine C, et al. Bit prudent in-cache acceleration of deep convolutional neural networks [C] //Proc of the 25th Int Symp on High Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2019: 81-93

[72] Bruening D, Zhao Qin, Amarasinghe S. Transparent dynamic instrumentation [C] //Proc of the 8th ACM SIGPLAN/SIGOPS Conf on Virtual Execution Environments. New York: ACM, 2012: 133-144

[73] Naveen M, Rajeev B, Norman P J. CACTI 6.0: A tool to model large caches, HPL-2009-85 [R]. Chicago: HP Laboratories, 2009

[74] Brooks D, Tiwari V, Martonosi M. Wattch: A framework for architectural-level power analysis and optimizations [C] // Proc of the 27th Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2000: 83-94

[75] Wu Ye, Vijaykrishnan N, Kandemir M, et al. The design and use of SimplePower: A cycle-accurate energy estimation tool [C] //Proc of the 37th Design Automation Conf. Piscataway, NJ: IEEE, 2000: 340-345

[76] Brooks D, Bose P, Srinivasan V, et al. New methodology for early-stage, microarchitecture-level power-performance analysis of microprocessors [J]. IBM Journal of Research and Development, 2003, 47(5): 653-670

[77] Kahng A B, Li Bin, Peh L, et al. ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration [C] //Proc of the Conf on Design, Automation and Test in Europe. Piscataway, NJ: IEEE, 2009: 423-428

[78] Chen Ke, Li Sheng, Muralimanohar N, et al. CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory [C] //Proc of the 15th Design, Automation & Test in Europe Conf & Exhibition. Piscataway, NJ: IEEE, 2012: 33-38

[79] Jouppi N P, Kahng A B, Muralimanohar N, et al. CACTI-IO: CACTI with off-chip power-area-timing models [C] // Proc of the Int Conf on Computer-Aided Design (ICCAD). Piscataway, NJ: IEEE, 2012: 294-301

[80] Li Sheng, Chen Ke, Ahn J H, et al. CACTI-P: Architecture-level modeling for SRAM-based structures with advanced leakage reduction techniques [C] //Proc of the Int Conf on Computer-Aided Design (ICCAD). Piscataway, NJ: IEEE, 2011: 694-701

[81] Li Sheng, Ahn J H, Strong R D, et al. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures [C] //Proc of the 42nd Int Symp on Microarchitecture. New York: ACM, 2009: 469-480

[82] Bhattacharjee A. Translation-triggered prefetching [C] // Proc of the 22nd Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2017: 63-76

[83] Jain A, Lin C. Back to the Future: Leveraging Belady's algorithm for improved cache replacement [C] //Proc of the Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2016: 78-89

[84] Michaud P. Best-offset hardware prefetching [C] //Proc of the 22nd Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2016: 469-480

[85] Lastras-Monta M A, Ghofrani A, Cheng K T, et al. A low-power hybrid reconfigurable architecture for resistive random-access memories [C] //Proc of the 22nd Int Symp on High Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2016: 102-113

[86] Palangappa P M, Mohanram K. CompEx: Compression-expansion coding for energy, latency, and lifetime improvements in MLC/TLC NVM [C] //Proc of the 22nd Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2016: 90-101

[87] Hashemi M, Khubaib, Ebrahimi E, et al. Accelerating dependent cache misses with an enhanced memory controller [C] //Proc of the 43rd Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2016: 444-455

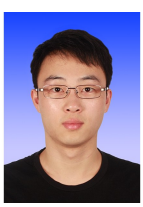
[88] Chi Ping, Li Shuangchen, Xu Cong, et al. PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory [C] //Proc of the 43rd Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2016: 27-39

[89] Ajdari M, Park P, Kim J, et al. CIDR: A cost-effective in-line data reduction system for terabit-per-second scale SSD arrays [C] //Proc of the 25th Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2019: 28-41

[90] Teran E, Wang Zhe, Jimenez D A. Perceptron learning for reuse prediction [C] //Proc of the 49th Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2016: 13-24

[91] Nori A V, Gaur J, Rai S, et al. Criticality aware tiered cache hierarchy: A fundamental relook at multi-level cache hierarchies [C] //Proc of the 45th Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2018: 96-109

[92] Stuecheli J, Kaseridis D, Daly D, et al. The virtual write queue: Coordinating DRAM and last-level cache policies [C] //Proc of the 37th Int Symp on Computer Architecture. New York: ACM, 2010, 38: 72-82



Zhang Qianlong, born in 1988. PhD candidate in the Institute of Computing Technology, Chinese Academy of Sciences. His main research interests include computer architecture and operating system.

Hou Rui, born in 1978. PhD, professor in the Institute of Information Engineering, Chinese Academy of Sciences. His main research interests include CPU architecture and security, as well as data center server design.

Zhao Boyan, born in 1990. PhD, assistant professor in the Institute of Information Engineering, Chinese Academy of Sciences. His main interests include data center server architecture, high-speed interconnect and CPU microarchitecture.

Yang Sibo, born in 1980. PhD candidate in Peking University. His main research interests include computer architecture and CPU microarchitecture design.

Zhang Lixin, born in 1971. PhD, professor. His main research interests include computer architecture, processor design, high performance computing, data center systems, edge computing, and industrial Internet.