体系结构模拟器技术进展与分析

耿子端[1),2),3)] 张宇航[1),3)] 王啸峥[2),3)] 蒋金虎[2),3)] 张为华[1),3)] 陈左宁[4)]

[1)](复旦大学计算与智能创新学院 上海 200082)

[2)](复旦大学大数据研究院 上海 200082)

[3)](复旦大学并行处理研究所 上海 200082)

[4)](中国工程院 北京 100088)

摘 要 体系结构模拟器(computer architecture simulator)是以体系结构视角对处理器功能部件及部件间时序等关系进行建模,模拟处理器功能和性能(速度、功耗等)行为的软件。相较于传统的电路层(门级)模拟器,体系结构模拟器因其忽略门级电路实现细节,通常表现出更高的模拟速度。由于体系结构模拟器在处理器架构设计和优化空间探索等方面的重要作用,目前广泛应用于软硬件协同和体系结构创新中,已成为处理器设计工具链的一个关键环节。本文首先讨论了体系结构模拟器的构成、关键指标和面临的挑战。在此基础上调研了近年关于体系结构模拟器的国内外最新研究进展以及开源生态,分析了各个研究小组和公司机构对开发新的模拟技术和工具的贡献。最后对未来体系结构模拟器的发展方向进行了讨论并给出了可能的发展方向的建议。本文希望为领域研究者提供一些启发,推动体系结构模拟器在软硬件协同工具链的更广泛应用。

关键词 体系结构;模拟器;功能模拟;时序模拟;软硬件协同

中图法分类号 TP393 **DOI 号** 10.11897/SP.J.1016.2026.01170

A Survey of Computer Architecture Simulation

GENG Zi-Duan[1),2),3)] ZHANG Yu-Hang[1),3)] WANG Xiao-Zheng[2),3)] JIANG Jin-Hu[2),3)]
ZHANG Wei-Hua[1),3)] CHEN Zuo-Ning[4)]

[1)](College of Computer Science and Artificial Intelligence, Fudan University, Shanghai 200082)

[2)](Institute of Bigdata, Fudan University, Shanghai 200082)

[3)](Parallel Processing Institute, Fudan University, Shanghai 200082)

[4)](Chinese Academy of Engineering, Beijing 100088)

Abstract As semiconductor fabrication processes approach fundamental physical limits, the continued scaling predicted by Moore's Law confronts increasingly severe challenges. These challenges manifest as: (1) a deceleration in transistor count growth, which consequently hinders improvements in hardware computational capability; (2) the escalating prominence of power consumption and thermal dissipation, where the energy surge from increased transistor densities creates a critical bottleneck for performance gains; and (3) a significant increase in the complexity and associated costs of design and verification, which rise exponentially with process advancements. Against this backdrop, application-driven hardware-software co-design has emerged as a crucial pathway to achieving efficient computation, prompting leading IT organizations to acceler-

收稿日期:2025-05-26;在线发布日期:2025-11-11。本课题得到国家重点研发计划项目(No. 2022YFB4500404)资助。**耿子端**,博士研究生,中国计算机学会(CCF)会员,主要研究领域为计算机体系结构、模拟仿真。E-mail:zdgeng21@m.fudan.edu.cn。**张宇航**,硕士研究生,主要研究领域为计算机体系结构、模拟仿真。**王啸峥**,硕士研究生,主要研究领域为计算机体系结构、模拟仿真。**蒋金虎**,博士,高级工程师,中国计算机学会(CCF)高级会员,主要研究领域为计算机体系结构、操作系统、分布式存储等。**张为华(通信作者)**,博士,教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为编译优化、计算机体系结构、并行、系统软件等。E-mail:zhangweihua@fudan.edu.cn。**陈左宁**,博士,研究员,博士生导师,中国工程院院士,中国计算机学会(CCF)会士,主要研究领域为存储系统、操作系统、信息安全等。

ate their efforts in this domain. The architectural layer, serving as the critical interface between software and hardware, has become a primary driver for enhancing processor performance and improving transistor-level efficiency. A computer architecture simulator is a software tool that models the functional components of a processor and their temporal relationships from an architectural perspective, thereby simulating the processor's functional and performance behaviors (e.g., speed, power consumption). Leveraging architecture simulators, designers can rapidly model and evaluate the trade-offs of various processor design alternatives. Concurrently, these tools enable the collection and analysis of diverse internal processor data, fostering a deeper understanding of processor behavior and facilitating comprehensive design space exploration. Moreover, simulation platforms provide an environment for software development and testing, thereby accelerating the overall processor product development lifecycle. In contrast to traditional circuit-level (gate-level) simulators, computer architecture simulators abstract away low-level implementation details, enabling significantly higher simulation speeds. Given their vital role in architectural design and optimization, architecture simulators are now widely employed in hardware-software co-design and architectural innovation, establishing them as a critical component of the processor design toolchain. This paper first discusses the definition and components of architecture simulators, establishing four key metrics: Performance, Accuracy, Extensibility, and Usability. Based on these metrics, we then analyze the primary contemporary challenges facing simulation: low performance, poor accuracy, limited extensibility, and inadequate usability. Next, this paper presents an overview of typical simulators and their ecosystems. We formulate a three-dimensional selection criterion to ensure the chosen simulators are both representative and contemporary: (1) citation or use in top-tier computer architecture conferences within the last five years; (2) active repository maintenance (commits within the last five years); and (3) total citation count. Following this, the paper surveys recent domestic and international research advancements and open-source ecosystems in architecture simulation. We analyze the contributions of various research groups and corporate institutions toward developing novel simulation technologies and tools, and subsequently summarize the challenges and limitations inherent in these modern research methodologies. Finally, this paper explores future directions for architecture simulation and proposes recommendations for long-term technological investment and development, focusing on four key areas: (1) comprehensive and integrated frameworks, (2) breakthroughs in key enabling technologies, (3) intelligent analysis techniques, and (4) robust open-source ecosystem construction. We hope this paper provides valuable insights for researchers in the field and promotes the broader adoption and application of computer architecture simulators within the hardware-software co-design toolchain.

Keywords　architecture; simulator; functional simulation; timing simulation; hardware-software co-design

1　引　言

自处理器发明以来,摩尔定律作为推动处理器性能提升的核心规律,支撑了半个多世纪的飞速发展。然而,随着工艺制程接近物理极限,摩尔定律的推进面临日益严峻的挑战,具体表现在以下几个方面[1]:(1)晶体管数量增长速度减缓,导致硬件计算能力的提升愈加困难;(2)功耗与散热问题愈发突出,增加晶体管数量带来的能耗激增对性能提升的收益形成瓶颈;(3)设计和测试的复杂性随着制程进步显著增加,相关成本呈指数级增长。在此背景下,应用驱动的软硬件协同优化成为实现高效计算的重要路径[2]。通过深度挖掘应用特性,定制化设计和优化软硬件解决方案,不仅可以降低能耗与硬件成

本,还能更好地满足复杂应用的需求,尤其是当下人工智能(Artificial Intelligence,AI)大模型训练与推理等高算力的需求。

为了提升核心竞争力,IT 领域的头部企业都加快了软硬件协同的步伐,例如,Google 公司针对自己的典型应用设计的 TPU(Tensor Process Unit),苹果针对自己的手机和电脑场景设计的 A 系列和 M 系列处理器以及英伟达结合 AI 典型应用场景,进行的 GPU(Graphics Processing Unit)迭代设计优化等。这些实践充分表明,面向特定应用的体系结构创新已成为推动性能提升的关键手段。正如图灵奖获得者 David Patterson 所言,接下来的十年是体系结构发展的黄金十年[3],领域专用硬件优化空间巨大,体系结构创新将成为未来 10 年处理器硬件性能增长和提升晶体管效率的主要推动力。

体系结构模拟器是以体系结构视角对处理器功能部件及部件间时序等关系进行建模,模拟处理器功能和性能行为的软件。基于体系结构模拟器,可以快速对不同处理器设计进行建模,评估不同设计方案的优劣。同时,可以收集处理器内部不同维度的数据并进行分析,理解处理器的行为,进行设计空间探索。此外,模拟工具还能在硬件产品可用前,为软件提供开发测试平台,从而缩短处理器产品开发周期。如图 1 所示,处理器产品为了提高竞争力,产品从硬件设计到上市一般在 12 个月以内。然而传统开发流程中,硬件设计和生产以及软件开发分别需要约 9 个月时间。若等硬件可用再进行软件开发,真正留给软件平台迭代开发和测试的时间往往不足 3 个月。而如果在硬件设计初期就同步开发模拟器,在硬件设计初步完成时(如 3 个月)为软件提供模拟,则软件工程师可以基于模拟器进行相关分析、开发和测试,而硬件工程师则可以基于模拟器与

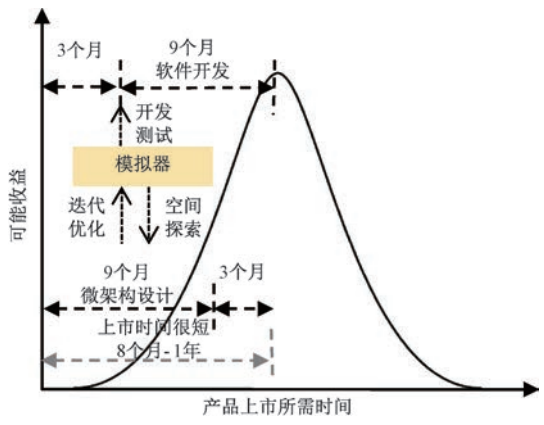

图 1 模拟工具可以缩短产品的发布周期[4]

软件的协同进行处理器进一步的空间探索并对模拟器进行迭代优化,从而实现了软硬件开发和设计的充分协同。

为了更好支撑体系结构创新与软硬件协同设计,理想的体系结构模拟器需要满足设计人员的多方面核心诉求:它必须运行得足够快,以提升评估效率;其模拟结果必须足够可信,才能指导做出正确的设计决策;它必须方便扩展,以方便适配不同新场景;同时它还必须足够易用,以降低使用门槛。因此,体系结构模拟器主要有四个关键指标:性能、准确性、可扩展性和易用性,四个方面直接影响着体系结构研究与产品开发的质量和效率。

在国际竞争加剧和技术自主需求的推动下,国内构建底层竞争力方面的投入不断加大,体系结构模拟在软硬件协同设计中的关键作用日益凸显。结合现有体系结构模拟的核心需求,本篇文章首先介绍了体系结构模拟器的基本概念、核心用途及其与电路层模拟的区别。随后,详细介绍了四个关键指标,并深入阐述当前模拟器面临的“性能差”“准确性差”“可扩展性差”“易用性差”的挑战。在此基础上,本文综述了为应对上述挑战,学术界与工业界所提出的前沿研究进展。最后结合当前的发展趋势和国内软硬件协同的诉求,给出了一些具体建议。

本文结构如下:第 2 节详细介绍了体系结构模拟器的定义、核心组成、用途以及与电路层模拟器的区别;第 3 节承接上文,深入剖析“性能”、“准确性”、“可扩展性”与“易用性”这四个关键指标,并详细阐述当前模拟器在这些方面所面临的具体挑战;第 4 节综述了体系结构模拟器的最新研究进展,并分析了工业界和学术界在该领域的贡献;第 5 节讨论了体系结构模拟器未来的发展方向,提出了综合框架、关键技术突破、智能分析技术和开源生态建设等建议;第 6 节对全文进行总结,希望为领域研究者提供一些启发,推动体系结构模拟器在软硬件协同工具链的更广泛应用。

2 体系结构模拟器

在本节中,我们主要介绍体系结构模拟器的定义、组成、用途以及与电路层模拟器的关系和区别。

2.1 体系结构模拟器的定义及组成

体系结构模拟器是从体系结构视角对处理器建模,完成处理器功能行为和时序行为模拟的软件。功能行为主要完成指令集及外设等模块的功能模

拟,为操作系统和应用提供执行环境,并在应用运行过程中收集时序模拟需要的指令流和数据流等信息。时序行为模拟粗粒度功能单元,如 ALU(Arithmetic and Logic Unit)、处理器核心流水线各个阶段、分支预测器、Cache 等的时序行为,以及这些功能单元之间的交互和 RTL(Register-Transfer Level)传输引发的时序行为,而忽略这些功能单元内部的细粒度门级(电路级)时序行为的模拟[5]。功能模拟和时序模拟两个部分协同工作,使体系结构模拟器在应用和系统软件的驱动下,高度准确地模拟目标体系结构的行为。

根据体系结构的定义,体系结构模拟器通常由功能模拟模块(Functional Model,FM)、时序模拟模块(Timing Model,TM)和交互耦合三部分组成,具体如图 2 所示。

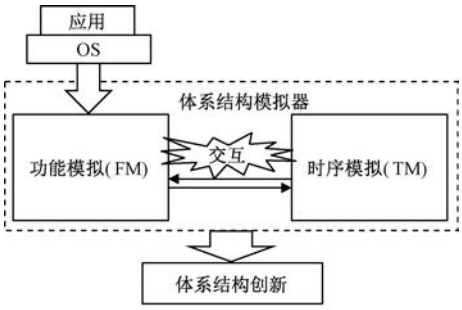

图 2　体系结构模拟器组成

2.1.1　功能模拟

功能模拟模块用来实现处理器的功能行为,为操作系统和应用提供与模拟的目标处理器(Guest)一致的运行环境。在功能模拟中,为了适应不同运行的硬件宿主(Host)平台,模拟器通常使用动态二进制翻译(Dynamic Binary Translation,DBT)技术将输入的 Guest 二进制文件翻译成 Host 机器的二进制可执行代码。同时,使用虚拟化技术提供各设备、IO(Input/Output)的功能以支持不同架构的程序在模拟环境下正确运行。因此,动态二进制翻译技术和虚拟化技术是功能模拟最关键的两种核心技术,对功能模拟的准确性和性能有至关重要的影响。

功能模拟器可以单独运行,在目标处理器可用前,为目标处理器的软件开发和测试提供基本运行环境;也可以与时序模拟器协同运行,在运行过程中收集软件行为产生的指令流和数据流,并将这些信息传递给时序模拟部分,从而驱动时序模拟运行。

2.1.2　时序模拟

时序模拟主要模拟处理器硬件的时序行为和性能特征,负责控制和管理每个时钟周期各个体系结构功能部件的时序行为和数据交互,确保模拟器按照与目标处理器硬件一致的时序完成操作。时序模型需要精确地反映各种微体系结构特性,例如流水线的深度、发射宽度、乱序执行窗口的大小、分支预测算法的细节、缓存的大小、关联度和替换策略等。这些参数直接影响模拟的性能结果。

时序模拟可以以不同的方式来进行驱动[6],具体可以分为时钟周期驱动(cycle-driven)、指令驱动(instruction-driven)和事件驱动(event-driven)。

时钟周期驱动模拟将时间划分为连续的时钟周期间隔,在每个时钟间隔内,基于前面的性能状态数据模拟各功能部件在对应时间间隔的时序行为并更新性能状态。时钟周期驱动模拟的方式非常直观,与硬件的实际运行方式相似,每个周期都会检查所有硬件部件的输入确认该部件是否需要进行时序模拟。但当系统中活动较少时,可能会进行不必要的检查,导致效率不高。

指令驱动模拟以处理器运行的指令流中的指令作为模拟的基本单位,每条指令根据指令执行过程所需的功能部件以及和前面指令的依赖关系确定和模拟当前指令使用功能部件的时序行为并更改相关功能单元的时序状态,供后继指令使用。由于每条指令的模拟过程中只触发该指令执行过程中相关功能单元的时序行为模拟,因此具有更高的模拟速度。

事件驱动模拟以功能单元的时序事件为粒度进行模拟,在时钟周期推进的过程中,检查相关功能单元对应的事件队列,处理事件,并生成新的事件及相关的时序信息,放入对应的事件队列中,供后继处理。事件驱动模拟的方式下,模拟器只在有事件发生时才进行处理,例如缓存命中/缺失、分支预测错误、资源冲突等。这使得模拟器在系统空闲或活动较少时效率较高,因为它可以跳过没有事件发生的时钟周期。

在模拟过程中,出于性能、准确性和软件复杂度等因素的考虑,通常会结合多种驱动方式,例如指令驱动的周期精确模拟(instruction-driven, cycle-accurate simulation),或事件驱动的周期精确模拟(event-driven, cycle-accurate simulation)。由于软件模拟面临巨大的性能挑战,因此,在时序模拟模型设计中,也会引入并行[7]或采样[4,8-10]等技术来加速时序模拟的速度,同时在建模过程中,也会在准确性和性能等方面进行平衡。

2.1.3 耦合方式

功能模拟与时序模拟可以通过不同的协同方式(耦合方式)组成整个体系结构模拟器。根据协同程度的不同,功能模拟与时序模拟器的耦合方式主要分紧耦合与松耦合两种。

在紧耦合的设计中,功能模拟器和时序模拟器通过共享的数据结构和控制逻辑协同工作,时序模拟器每个时钟周期与功能模拟器进行协同,获得指令流和数据流信息、更新时钟并进行时序模拟。紧耦合设计下功能模拟的每一步(或每条指令)的执行可能受到时序模型的直接影响和控制。例如,如果时序模型指示一个访存操作需要多个周期才能完成,功能模型必须等待时序完成相应的周期的时序模拟后才能继续。这种方式通常能提供更高的模拟精度,尤其是在处理复杂的依赖关系和资源冲突时。

在松耦合的设计中,功能模拟器和时序模拟器独立运行,功能模拟器收集正确执行路径的指令和访存行为并传递给时序模拟器。时序模拟器在进行时序模拟的过程中,也检测功能模拟器的时序相关功能行为,当发现时序关系违反(如分支预测错、多线程访存顺序错误或调度行为错误等)时,时序模拟器通知功能模拟器进行回滚和纠错。

在两种执行方式中,紧耦合方式与处理器实际执行方式更一致,易于理解。其优点在于能够更自然地模拟指令间的动态交互和资源竞争,但缺点是功能模拟和时序模拟之间的频繁交互可能成为性能瓶颈。同时,因为紧耦合的方式,当对功能模拟或时序模拟进行扩展时,涉及的代码逻辑也比较复杂。相比之下,松耦合方式可以获得更好的可扩展性和性能。因为功能模拟和时序模拟可以更独立地运行,甚至并行执行,从而充分利用多核处理器的能力。然而,确保两者之间状态的一致性和处理时序分歧(violation)的复杂性是其主要挑战。

2.2 体系结构模拟器的用途

随着摩尔定律的放缓和软硬件协同变得越来越重要,体系结构模拟器在体系结构创新中发挥的作用日益显著,主要体现在以下两个方面:

(1)处理器设计驱动的体系结构架构探索:在处理器迭代设计过程中,需要结合应用领域和工艺的变化,理解处理器的行为,明确新一代处理器的优化方向,进行体系结构设计和评估,并为系统软件团队提供开发验证平台。由于处理器设计团队掌握全部已有硬件的设计、实现细节以及工艺等方面的器件属性信息,因此在体系结构模拟器的构建过程中,通常会结合已有处理器的设计和时序信息,对模拟器的各个功能部件以及整个处理器的时序行为进行校准。在使用和评估的过程,希望模拟器的结果跟实际硬件的结果保持一致,即尽可能实现模拟结果的绝对精度。

(2)应用驱动的软硬件协同体系结构架构探索:随着软硬件协同对于应用厂商的重要性日益提高,应用厂商越来越关注硬件与应用场景的适配度,并希望构建与应用适配的硬件体系,从而提升竞争力。在软硬件协同创新过程中,需要基于体系结构模拟器收集和理解应用的体系结构特征,自顶向下分析当前使用的硬件与应用不适配的瓶颈,探索并评估各种改进优化的效果。

在使用体系结构模拟器进行处理器架构方向探索过程中,由于开发效率和模拟效率等方面的考虑,会在绝对精度、模拟速度与开发复杂度等方面进行平衡。同时,对于此使用场景,由于软件团队很多时候缺乏具体硬件细节参数,无法对硬件细节进行完整建模,因此在使用模拟器进行评估时,会更关注模拟结果的相对精度,即基于模拟器获得的两种不同处理器设计的性能差异,是否与真实硬件获得的两种不同处理器设计性能的差异保持一致,而非模拟器每次运行输出的时钟周期等性能结果是否与真实硬件完全一致。

2.3 与电路层模拟器的关系

虽然体系结构模拟器与电路层模拟器均属于处理器的软件模拟器,但由于两者用途不同,导致抽象粒度、实现方式、性能等方面存在差异,如图 3 所示。具体如下:

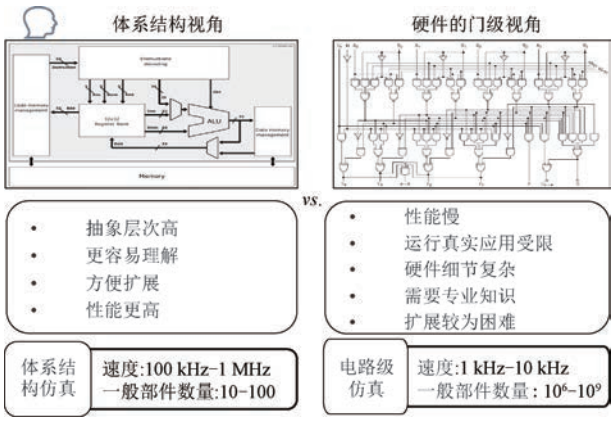

图 3 体系结构模拟器与电路层模拟器

(1)体系结构模拟器:体系结构模拟器的抽象粒度为功能单元级别,例如流水线阶段、分支预测器、

ALU、Cache 等,而忽略这些部件的门级实现。以 gem5 模拟器为例,共模拟了 82 个部件单元。其时序模拟通常以时钟周期为单位,而不涉及电路级的信号传播或时序约束。在软件实现过程中,一般采用高级语言,如 C++或 python 等。由于抽象层次较高,典型体系结构模拟器的模拟速度一般是真实硬件的 10^{-5} 级别,模拟速度可以达到 100 kHz～1 MHz。因此,在使用过程中,可以使用真实应用作为输入进行评估。

(2)电路级模拟器:电路级模拟器支持多种抽象层次的建模,包括系统行为级(如 RTL 级功能验证)、电路级(如门级时序)等。与体系结构模拟器不同,电路级模拟器通常需要包含各种硬件功能部件的详细实现,例如基于 RTL 的寄存器传输级建模或门级网表描述。其时序模拟关注电路级的信号传播延迟、时序约束以及逻辑门或晶体管级的物理时序特性,通常以亚纳秒级精度建模。对于一个复杂的集成电路,可能包含数十亿个晶体管和数以百万计的其他电子元件。实现语言一般采用 VHDL、Verilog 或 Chisel 等。由于抽象层次较低,包含太多硬件细节,相比体系结构模拟器,典型电路级模拟器速度通常会慢 2～3 个数量级,模拟速度只能达到 1 kHz～10 kHz。因此,在使用过程中,受限于较低的模拟速度,通常使用指令数较少的小型测试程序、特定工作负载片段或波形信号进行评估。

在现代处理器设计过程中,电路层模拟器和体系结构模拟器并非孤立运行,而是通常协同使用,充分发挥各自优势。

随着处理器复杂程度的增加,现代处理器设计中需要探索空间往往巨大。以在学术界广泛使用的开源超标量乱序执行 RISC-V 处理器 BOOM[11](berkeley out-of-order machine)为例,组合各个模块的参数,合法的微架构设计空间的大小约为 1.6×10^{8}[12]。因此,在这样的设计过程中,为了更好实现探索速度和精度的权衡(tradeoff),通常结合体系结构模拟器和电路层模拟器的优势,组合使用两种模拟器。具体如图 4 所示,在空间探索过程中,由于体系结构模拟器性能比电路级模拟器性能高至少 10^{3} 以上,且更易于扩展,因此一般先基于体系结构模拟器进行处理器体系结构的设计方向探索。设计方向明确后,再基于电路层模拟器对一些选定的设计方向进行细粒度的评估和对比。通过这种方式,可以更好地平衡评估速度和评估精确性。

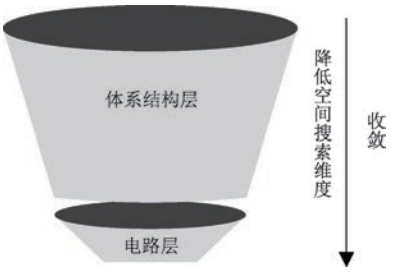

图 4　结合体系结构模拟器与电路层模拟器进行空间探索

3　体系结构模拟器关键指标与挑战

本节中,我们将主要讨论体系结构模拟器的关键指标,并基于这些指标讨论体系结构模拟器当前面临的主要挑战。

3.1　关键指标

体系结构模拟器的设计和使用中主要涉及性能、准确性、可扩展性和易用性四个关键指标[5,13],具体如下①:

(1)性能(Performance):模拟器的性能决定模拟器的运行效率,决定其处理大规模工作负载和复杂实验时的能力。目前典型模拟器的速度只有真实硬件的 10^{-5},甚至更低。模拟性能主要受建模抽象层次的影响,低抽象层次提供更高精度,但会降低性能。因此,提高模拟性能需在抽象层次与硬件细节间找到平衡。

(2)准确性(Accuracy):准确性衡量模拟器行为或性能结果与目标硬件实际行为或性能结果的吻合程度,是模拟器的核心要求。准确性受抽象层次和模拟模型等多维度因素的影响。它体现在多个方面:当由于商业机密等原因无法获取完整硬件细节时,模拟器需具备通过参数拟合来实现与硬件行为一致的相对精度;在面对 ISA 变更、新型加速器引入等情况时,需能便捷地对齐硬件参数。同时,面对 GPU、TPU 和非易失性存储器(non-volatile memory,NVM)等新型硬件技术的快速发展,模拟器也需设计适应新技术的建模方法,以准确模拟新兴计算和存储部件。

① Akram A 等人的工作中提出六个评估指标[13]:准确性、性能、细节呈现程度、开发难度、灵活性及用户友好性。然而,该划分方式的各个指标不够准确,如"细节呈现程度"并非一个独立的评估维度,它与性能、准确性及开发难度等指标都相关,是一种设计权衡。同时,如 2.3 节中讨论,不同细节呈现程度的模拟器在处理器设计过程中有不同的评估目标和用途,往往需要结合设计方向组合使用,比如体系结构模拟器会隐藏一些细节进行宏观设计和方向选择,而电路级模拟器则会在体系结构方向进一步明确后进行更多包含细节信息的评估。此外,按该论文的定义,"开发难度"与"灵活性"均与扩展模拟器功能的成本相关,可统一归纳为"可扩展性"。因此,本文将评估指标重新划分为性能、准确性、可扩展性和易用性四个方面。

(3)可扩展性(Extensibility):可扩展性指模拟器以较低人力和时间成本扩展功能模块或性能模型的能力。其内涵覆盖宏观和微观两个层面:宏观可扩展性聚焦于支持不同计算场景和多样化的上层应用场景(如安卓系统、车载 QNX 系统),关注模拟器与这些场景的软件与硬件环境的快速适配能力;微观可扩展性则聚焦于时序模型内部,要求能灵活地对具体硬件部件(如新指令、自定义缓存协议)进行扩展,并保证处理器设计变化时其时序影响被精确传递。

(4)易用性(Usability):易用性衡量模拟器的便捷程度,主要体现在其配套工具(如配置工具、可视化界面、检查点回滚工具)的丰富程度,直接影响用户的学习曲线和使用便捷度。易用性高的模拟器能吸引更多使用者,扩大其生态。例如,直观的图形化界面可简化参数设置和结果分析,相比命令行交互更适合非专业用户;丰富的配套工具(如便捷的检查点回滚、结果可视化工具)则可显著提升实验和调试效率。

3.2 挑 战

尽管理想的模拟器应兼顾所有四项关键指标,但实际上这些指标往往不是孤立的,而是彼此互相影响。因此,在设计中往往面临权衡和取舍。例如,高精度通常伴随低性能,而技术进步和需求变化也带来了新的挑战。目前,模拟器在以下方面存在不足:

(1)性能差:体系结构模拟器需通过软件在宿主机上精确模拟目标处理器。运行时,一方面需根据输入进行大量判断以确定执行路径及各功能部件的事件处理需求;另一方面需在时钟周期级别记录每条动态指令的运行结果及处理器状态变化(如寄存器状态、内存和缓存行为、分支预测器状态等)。一般性能较好的体系结构模拟器只能达到 MIPS(Million Instructions Per Second,MIPS)的模拟速度,而典型模拟器如 gem5 的速度仅约 100KIPS(Kilo Instructions Per Second,KIPS)。其运行 SPEC2006 测试集可能需要数月时间,难以支持复杂真实应用的输入和高效评估需求。

(2)准确性差:准确性方面的挑战主要体现在模拟模型与物理硬件的对齐困难。对于体系结构研究者,要精确模拟 GPU、TPU 等专用加速器或 NVM 等新型存储器件,需要深刻理解其内部工作机制,而这些细节往往属于商业机密,导致建模困难,难以保证模拟结果的可信度。其次,即便掌握所有设计细节,现代硬件的高度复杂性也给精确建模带来了巨大困难。比如动态频率调整、多核间的缓存一致性协议以及多核交互等行为,难以在模拟器中准确复现。

(3)可扩展性差:现有模拟器多采用紧耦合设计,导致功能模块和时序模块之间交互复杂,代码关联度过高。宏观上,在适配新型应用场景时,用户需深入理解模拟平台并具备丰富系统知识。例如,功能模拟器 QEMU 的场景适配性优于 gem5,但仍难以支持完整的 Android 图形栈或 QNX 实时操作系统,导致手机厂商或车厂的系统级验证和开发主要依赖真机或真车,而非模拟器。微观上,添加新型硬件单元(如一个自定义的预取器)需修改模拟器核心代码,涉及多个模块的适配;替换时序模型(如从简单缓存模型切换到复杂 MOESI 协议)则可能需要重构模块接口,显著增加开发成本。表 1 展示了典型模拟器的扩展耗时[8,14],反映了可扩展性不足的严重性:MARSS[15] 模拟器基于 PTLSim 与 QEMU 模拟器开发,4 名开发者耗时约 1.5 年才完成该工作。

表 1 模拟器扩展耗时

模拟器	结合工作	扩展工作
MARSS	PTLSim+QEMU	约 1.5 年,4 名开发者
gem5	GEMS+M5	数年,数名开发者

(4)易用性差:随着应用端用户对易用性的需求提升,主流模拟器仍存在不足。首先,模拟器依赖命令行交互,缺乏直观的图形化界面,用户需手动输入复杂命令,增加了操作难度。其次,模拟结果以文本日志为主,缺少直观的性能视图,用户需自行解析数据。最后,检查点(Checkpoint)创建和回滚等可以有效提升模拟效率和模拟过程中的容错程度,但很多场景对这些功能的支持有限。

4 体系结构模拟器最新研究进展

体系结构模拟器是涉及多技术维度的综合领域,作为处理器创新的重要环节,长期受到工业界和学术界的高度重视。本章将讨论典型模拟器及生态、工业案例以及最新研究进展。

4.1 典型模拟器及生态

由于功能模拟器是时序模拟的基础,而时序模拟涉及的维度较多,因此,我们将首先讨论典型功能模拟,而时序模拟则从通用处理器时序模拟、领域专用处理器时序模拟、内存时序模拟、磁盘时序模拟、片

上网络时序模拟、功耗模拟以及 RTL 级模拟与 FPGA 加速仿真等方面进行讨论。具体如表 2 所示。

表 2　代表模拟器统计

类别	关注重点		模拟器	特色	发表年份	最新版本	最新提交时间	引用数	主要维护者	重要会议涉及次数
功能模拟	纯功能		QEMU[16]	开源全系统功能模拟器	2005	10.1.1	2025.10	4523	QEMU 社区	10
			Simics[17]	商业级全系统模拟器	2002	7	2025.9	2903	Intel	3
时序模拟	通用处理器时序模拟	周期精确模拟	gem5[18]	典型体系结构模拟器	2011	25.0.0.1	2025.8	7016	gem5 社区	174
			Scarab[19]	为多核芯片设计的精确模拟器	2024	V2.0	2025.10	6	加利福尼亚大学圣克鲁斯分校	4
		相对精确模拟	Sniper[20]	高速多核处理器时序模拟器	2011	7.4	2024.11	1242	根特大学	28
			Zsim[21]	轻量级、快速用户级多核模拟器	2010	—	2023.11	821	斯坦福大学	23
	领域专用时序模拟	GPU	Accel-Sim[22]	业界领先的 GPGPU 模拟器	2020	1.3.0	2025.10	409	普渡大学	21
			Scale-sim[23]	AI 加速器设计的时序模拟器	2018	3.0.0	2025.8	393	佐治亚理工学院	17
		可重构计算	CGRA-ME[24]	针对粗粒度可重构阵列的建模框架	2017	2.0.1	2023.11	214	多伦多大学等	2
			OpenCGRA[25]	开源 CGRA 建模、编译和仿真框架	2020	—	2023.3	94	太平洋西北国家实验室	2
	内存时序模拟	传统内存	DRAMSim[26]	周期精确的 DRAM 内存模拟器	2005	3.0	2021.4	1949	马里兰大学	36
			Ramulator[27]	快速、模块化的 DRAM 模拟器	2015	2.0	2025.5	1031	卡内基梅隆大学	63
		存内计算	PIMSim[28]	集成于 gem5 的 PIM 模拟框架	2017	2.0	2022.11	72	中国科学院研究所	1
			PIMulator[29]	DRAM 存内计算模拟器	2020	—	2023.1	19	弗吉尼亚大学	2
	存储时序模拟		MQSim[30]	面向现代 SSD 的高性能模拟器	2018	—	2025.10	258	多个学术机构	16
			SimpleSSD[31]	用于全系统仿真的 SSD 模拟器	2017	2.0.13	2022.11	83	延世大学等	7
	片上网络时序模拟		Booksim[32]	周期级片上网络(NoC)模拟器	2013	2.0	2024.1	984	斯坦福大学	15
			Garnet[33]	集成在 gem5 中的片上网络模型	2009	3.0	2025.10	588	gem5 社区	3
	功耗模拟		McPAT[34]	集成的功耗、面积和时序分析框架	2009	1.3	2020.8	3409	惠普实验室	4
			AccelWattch[35]	现代 GPU 功率建模框架	2021	1.3.0	2025.10	158	西北大学等	2
	RTL 级模拟与 FPGA 加速仿真		FireSim[36]	基于云端 FPGA 的周期精确模拟器	2018	1.20.1	2025.5	388	FireSim 社区	8
			Verilator[37]	Verilog/SystemVerilog 模拟器	1994	5.040	2025.10	91	Verilator 社区	1

在选择每个类别中的代表性模拟器时,我们制定了包含三个维度的选择标准,确保所选模拟器的代表性与时效性。首先,我们考察其在近五年计算机体系结构顶级会议(ASPLOS, ISCA, MICRO, HPCA, DAC)中对应模拟器是否被引用或使用过。其次,我们评估其代码仓库是否仍在积极维护中(近五年仓库仍有提交)。通过这两个步骤筛选出满足学术时效性与社区活跃度标准的模拟器,构成候选池。最后,在候选池中,我们依据基础影响力(论文总被引用数)进行排序,选取排名最靠前两款模拟器作为该类别的代表。

(1)功能模拟器:功能模拟器旨在硬件不可用时提供与目标处理器架构一致的执行环境,支持目标代码在不同架构平台上运行。功能模拟关注模拟应用或系统的正确性而无需关注底层的时序行为,通常用于软件的开发和测试等场景。设计上通常采用直译或二进制翻译技术实现。流行的功能模拟器有 QEMU[16]、Simics[17]、Bochs[38]、Spike① 等,代表性的为 QEMU、Simics。QEMU 是一个开源、通用的机器模拟器和虚拟器。它通过动态二进制翻译技术,能够模拟多种 CPU 架构(如 x86,ARM,RISC-V),并支持全系统模拟,可以运行未经修改的客户机操作系统。因其强大的功能和灵活性,QEMU 不仅是独立的工具,也成为众多其他项目的基石。Simics 是由 Intel 公司维护的商业级全系统模拟器。其核心特色在于提供确定性、可重复的模拟环境,并支持反向执行(ReverseExecution),允许开发者像视频回放一样调试复杂的系统故障。Simics 广泛应用于航空航天、汽车电子等对可靠性要求极高的工业领域。

(2)周期精确的通用处理器时序模拟:周期精确时序模拟器精确建模处理器的微架构行为,支持体系结构优化和性能分析。它关注流水线执行、缓存

① Getting Started with RISC-V: Spike Simulator, https://www.yorku.ca/professor/drsmith/2024/07/28/getting-started-with-risc-v-spike-simulator。

行为、乱序执行、分支预测等架构细节，确保微架构级别的准确性。设计上基于功能模拟器，通过以时钟周期为最小单位，详细建模 CPU 流水线、访存层次结构和时钟同步机制。流行的周期精确时序模拟器有 gem5[18]、Scarab[19]、MARSS[15]、ChampSim[39]、SimpleScalar[6]等，代表性的为 gem5、Scarab。gem5 是一个开源流行的全系统周期精确模拟器，支持多种指令集体系结构（如 x86、ARM、RISC-V 等）。它通过详细的微架构模型实现周期精确模拟，广泛用于学术研究和工业验证。gem5 支持全系统模拟，能够运行完整的操作系统和真实工作负载。Scarab 是一个专为最先进、高性能的多核芯片设计的精确模拟器，目标是实现高精度模拟，同时还要保证运行速度快且易于使用。其速度可达 600KIPS。

（3）相对精确的通用处理器时序模拟：相对精确时序模拟器平衡模拟精度与执行效率，适用于早期架构设计和快速评估。它关注指令流整体性能趋势而非精确周期行为，精度略低于周期精确模拟器，但仿真速度更快。设计上采用更高抽象级别的性能模型。典型的相对精确时序模拟器包括 Sniper[20]、ZSim[21]、McSimA＋[40]等。代表性的为 Sniper、ZSim。Sniper 是一种高性能的多核模拟器，采用间隔模拟（interval simulation）方法，通过主要关注指令流引发的关键事件（长周期事件）估算性能。它在保持合理精度的同时，提高了模拟速度，适用于快速评估。ZSim 通过动态二进制翻译和事件驱动建模实现快速模拟。它支持多线程和共享内存行为的高效建模。

（4）领域专用处理器时序模拟：领域专用处理器时序模拟器关注 GPU、AI 加速器以及可重构计算架构等专用硬件的并行计算行为，支持异构架构设计和优化。设计上通常采用流处理器级别的建模方法，结合指令级或统计级分析，模拟 SIMD（Single Instruction Multiple Data，SIMD）/SIMT（Single Instruction Multiple Thread）执行模式或针对特定数据流、计算阵列进行建模。

典型的关注 GPU 计算行为的模拟器包括 Accel-Sim[22]、Scale-Sim[23]、GPGPU-Sim[41]、MGPU-Sim[42]等。最为代表性的为 Accel-Sim、Scale-Sim。Accel-Sim 是一个用于验证 GPU 模型的准确性的可扩展模拟框架，基于 GPGPU-Sim[41]开发升级，提高了其可配置性和准确性。Scale-Sim 专为神经网络加速器设计，通过矩阵分解和数据流建模，分析加速器的计算和内存行为。它支持快速评估不同加速器配置的性能和能效。

典型的关注可重构计算的模拟器有 CGRA-ME[24]、OpenCGRA[25]等。CGRA-ME 是一个开源的粗粒度可重构阵列（Coarse-Grained Reconfigurable Array，CGRA）建模与探索框架。它支持对 CGRA 的计算阵列、片上网络和存储层次进行周期精确的性能和功耗模拟。CGRA-ME 的核心特色在于其能够对应用的“时空映射”（即将计算图映射到可重构硬件的时序和空间位置）过程进行建模，帮助研究者探索不同映射算法对性能的影响，是可重构计算领域的重要研究工具。OpenCGRA 是一个集成了编译器和模拟器的开源 CGRA 研究框架。它提供了一套完整的工具链，能够将 C 语言编写的程序内核编译、映射并部署到可配置的 CGRA 模型上进行仿真。OpenCGRA 支持对数据流图的自动生成和调度，方便研究者探索编译器与硬件架构的协同设计。

（5）内存时序模拟：内存时序模拟器的目标是精确建模 DRAM、NVM 及混合存储架构的时序行为，研究内存控制器策略和访存优化。根据其是否在存储单元内执行计算，可分为传统内存时序模拟和存内计算时序模拟。

传统内存时序模拟主要关注存储层次结构、访问延迟、带宽利用率、缓存一致性协议等关键因素，以评估不同内存设计对系统性能的影响。设计上采用基于时序事件的模拟方法，构建详细的内存控制器模型、缓冲管理策略及预取机制。代表性的内存时序模拟器包括 DRAMSim[26]、Ramulator[27]、CACTI[43]、USIMM[44]等。最为代表性的为 DRAMSim、Ramulator。DRAMSim 是一个开源的 DRAM 模拟器，详细建模内存控制器的调度策略和 DRAM 芯片的时序行为。它支持多种 DRAM 标准，适用于内存系统设计研究。Ramulator 是一个快速、灵活的内存模拟器，支持模块化扩展和多种内存技术。它通过高效的事件驱动模型，广泛用于评估新型内存架构的性能。

随着存内计算（Processing-in-Memory，PIM）技术的发展，许多内存模拟器也扩展了其功能，用于对计算和存储一体化的新型架构进行建模，以研究内存控制器策略、访存优化以及 PIM 操作的性能。代表性的存算时序模拟器包括 PIMSim[28]、PIMulator[29]等。PIMSim 是一个集成于 gem5 中的 PIM 模拟框架，由中国科学院计算技术研究所等团队开发。它扩展了 gem5 的全系统模拟功能，并加入了

专门的 PIM 指令和硬件模型,适合评估 DDR-DRAM 上的 PIM 架构。它支持全系统(系统级)、周期精确模拟。同时,与 gem5 深度集成,可以方便测试 PIM 指令的影响。PIMulator 是一个 DRAM 存内计算模拟框架。它专注于模拟真实的、靠近内存的 PIM 操作(如 Ambit),能够评估 PIM 指令对整个系统端到端性能的影响,而不仅仅是内存子系统本身。

(6)存储时序模拟:存储模拟器的目标是分析存储设备,如 HDD(Hard Disk Drive,HDD)、SSD(Solid State Disk,SSD)在不同负载条件下的性能,以优化存储系统设计。它关注磁盘调度策略、队列管理、擦除写放大、数据读写延迟等关键因素,评估不同存储介质的吞吐能力和能耗特性。设计上通常采用请求队列建模、闪存转换层(Flash Translation Layer,FTL)仿真以及 I/O 访问模式分析,以提供存储子系统的详细行为建模。代表性的磁盘模拟器包括 MQSim[30]、SimpleSSD[31]、SSDSim[45]、NVSim[46]、NVMain[47]等。最为代表性的为 SSDSim、MQSim。MQSim 专为多队列 SSD 设计,通过建模多通道和多路访问,分析复杂 I/O 负载的性能。它支持动态调度策略的研究。SimpleSSD 是一种高保真度的 SSD 模拟器,能够模拟硬件和软件的详细特性,同时简化了存储内部结构的复杂性。与现有的 SSD 模拟器不同点在于,SimpleSSD 可以集成到公开可用的完整系统仿真器,如 gem5 中。

(7)片上网络时序模拟:片上网络(network-on-chip,NoC)模拟器的目标是建模多核和众核架构中的数据通信行为,研究片上数据传输的拓扑结构、路由算法、拥塞控制及吞吐率。它关注网络互连结构、流量模式、功耗与延迟等因素,以评估不同 NoC 设计的可扩展性和性能。代表性的 NoC 模拟器包括 BookSim[32]、Garnet[33]、Noxim[48]、Nocgen[49]等。最为代表性的为 BookSim、Garnet。BookSim 是一个灵活的 NoC 模拟器,支持多种拓扑结构和路由算法。BookSim 通过详细的流量建模,广泛用于研究 NoC 的性能和功耗优化。Garnet 是 gem5 的 NoC 模块,专注于周期精确的互联模拟。它支持多种一致性协议和流量模式,适合多核系统通信分析。

(8)功耗模拟:功耗模拟器通过能耗模型评估硬件能耗能效,它关注不同架构组件(CPU、内存、网络等)的功耗开销,帮助优化计算系统的能耗性能比。设计上结合指令流或硬件行为分析,以提供精细化的功耗估算。典型的功耗模拟器包括 McPAT[34]、AccelWattch[35]、Wattch[50]、GPUWattch[51]等。代表性的为 McPAT、Orion。McPAT 是一个多核功耗、面积和时序建模工具,支持多种处理器架构。它通过详细的硬件活动分析,提供高精度的能耗估算。AccelWattch 是专为现代 GPU 设计的功耗建模框架,可以通过仿真环境、基于跟踪的数据采集方式、硬件计数器,或这些方法的组合来进行功耗建模。

(9)RTL 级模拟与 FPGA 加速仿真:与前述使用基于 C++等高级语言对硬件行为进行抽象建模不同,RTL 级模拟直接作用于处理器的硬件描述语言(Hardware Description Language,HDL)源码(如 Verilog、VHDL)。这类工具的仿真精度最高,可达到与真实电路行为等价的"信号精确"级别,但其前提是必须拥有完整的、可综合的 RTL 设计代码。它们通常用于功能验证和性能评估的后期阶段。典型的包括基于 FPGA 的 FireSim[36]框架、Verilator[37]、ModelSim[52]、RAMP Gold[53]等。代表性的为 FireSim、Verilator。FireSim 是一个基于 FPGA 的周期精确模拟器,支持大规模系统模拟。它利用 AWS 云中的 FPGA 资源,实现快速的全系统模拟,特别适合 RISC-V 架构和数据中心研究。FireSim 支持高吞吐量模拟,能够运行完整操作系统和复杂工作负载。Verilator 是一个开源的高性能 Verilog/SystemVerilog 模拟器。它通过将 HDL 代码编译成优化的 C++或 SystemC 模型来执行。Verilator 广泛用于对 RTL 设计进行快速的功能验证和架构级的性能评估。

4.2　工业案例

随着软硬件协同的重要性,工业界一直十分重视体系结构模拟器的核心技术探索。主要以独立创新和学术工业联合的方式,尝试在体系结构模拟器的关键技术研究上获得突破。我们统计了近五年在计算机体系结构领域重要会议中由工业界主导或参与的论文数量,以探究体系结构研究领域的活跃度和影响力大的企业(见表 3)。统计结果显示,Intel、英伟达(NVIDIA)、华为、Google、ARM 等企业在这些会议上发表的论文数量较多,反映了它们在体系结构模拟器技术研发和学术合作方面的领先地位。统计表明,工业界在体系结构模拟器研究中更倾向于应用其进行处理器架构创新的研究,而学术界则主要在模拟器的关键技术上起主导作用。本节将结合具体案例,分析工业界是如何应用并推动体系结构模拟器技术发展的。

表 3 主要科技公司在体系结构重要会议发表的模拟仿真相关论文数量统计(2021～2025)

公司	论文数量	公司	论文数量
Intel	30	ARM	14
NVIDIA	24	Meta	13
Google	22	微软	11
华为	20	阿里	8
三星	19	AMD	6

(1)英伟达 NVIDIA:NVIDIA 是典型的通过大规模仿真驱动产品迭代的公司。其 GPU 架构复杂,设计空间巨大,基于体系结构模拟器进行早期性能评估和瓶颈分析[54]对其设计过程中的方向选择至关重要。例如,在经典的 Riva 系列图形处理器开发中,NVIDIA 就通过大规模投资仿真器集群,将硬件开发与验证周期从原先的 12 个月大幅缩短至 3 个月,实现了仿真驱动设计的敏捷开发模式。其内部基于模拟器的典型工作流程如图 5 所示,架构师结合应用需求,基于体系结构模拟工具进行处理器体系结构的设计探索和试错,当确定某个设计有效后,再基于 RTL 或门级模拟器进行进一步验证。通过这种方式,每年会有 100 项左右的优化方案被提交给公司的技术委员会,经过评估后,约有 20 项设计被选择加入到下一代处理器产品中。

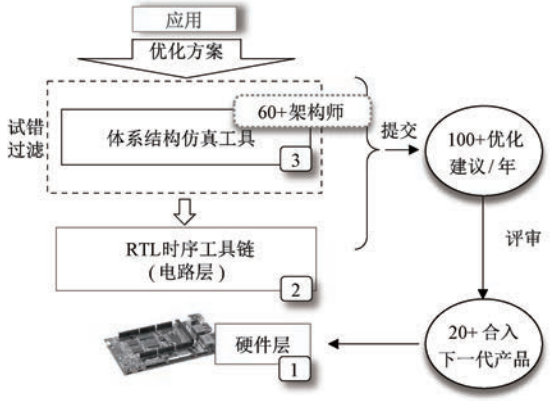

图 5 英伟达以体系结构模拟为核心进行处理器创新

除了内部应用,NVIDIA 也积极通过学术合作与开源贡献,推动模拟器技术的发展。例如,与 Google 合作开发的 NVArchSim[55]旨在构建一个快速且可信的系统级 GPU 模拟器;与华为、国防科大等合作开发的 HyFiSS[56]则专注于提升 GPGPU 模拟的效率与精度平衡;与 Intel 合作开发的 Dromajo[57]探索了利用逻辑 Fuzzer 增强的协同仿真技术,以提升处理器验证的完备性。

(2)Intel:Intel 也同样重视模拟器生态的工具开发与科研创新。在工具开发层面,Intel 推出的 Pin[58]是一个动态二进制插桩工具,已成为体系结构研究领域进行程序行为分析和构建自定义模拟工具的基准。其商业全系统模拟器 Simics 凭借强大的确定性模拟和反向执行调试能力,在工业界占据重要地位。

在技术研究上,Intel 同样积极与学术界合作。例如,与新加坡国立大学合作提出的 LoopPoint[10]是一种针对多线程应用的高效采样模拟技术,旨在解决传统模拟方法在面对长时间运行时效率低下的问题。这些工作表明,Intel 通过提供基础工具和探索前沿技术,对整个体系结构模拟领域的发展起到了基础性的支撑作用。

(3)Google:Google 使用体系结构模拟器完成了从底层硬件定义到上层软件生态的全栈式创新。在硬件层面,Google 推出的 TPU[59]在设计之初,通过了全面的体系结构模拟,对其在数据中心真实负载下的性能进行了精确分析和预测,证明了其专用设计的巨大优势,从而为耗资巨大的流片决策提供了坚实的数据支持。

在软件层面,Google 为安卓生态系统开发的 Cuttlefish① 平台,并在其中集成了 QEMU 和 gem5 等流行模拟器。它为开发者提供了一个功能完整、行为一致的安卓虚拟设备,使其可以在没有物理硬件的情况下进行应用开发、系统适配和兼容性测试。此外,Google 也通过与 NVIDIA 等伙伴合作开发 NVArchSim[55]等工具,共同致力于提升行业整体的模拟技术水平。

(4)其他产业界公司:除了上述公司,其他领域也在通过协同创新的方式,解决特定领域的模拟挑战。例如,Xilinx 联合康奈尔大学,探索了编译器驱动的模拟方法[60],以解决可重构硬件加速器这一新兴架构的仿真问题;存储公司美光科技(Micron Technology)则联合学术界,深入研究了现代内存子系统的基准测试、性能分析与模拟方法,以期建立更精准的内存系统模型[61]。

4.3 最新研究进展

本节中将首先调研近年体系结构模拟器最新研究进展及这些研究工作在工业界和学术界中的分布。在此基础上结合体系结构模拟器性能、准确性、可扩展性和易用性四个关键指标,分类讨论最新的研究工作。每类研究工作将先总体讨论该部分的总体研究方向分类,之后讨论一些具体的工作,最后讨论当前该部分工作面临的一些不足和挑战。

① Cuttlefish virtual Android devices, https://source.android.com/docs/devices/cuttlefish。

4.3.1　研究概况

海内外有许多著名的研究机构、高校和公司都在积极地进行体系结构模拟器的研究和开发。除探索关键技术外,模拟器也被广泛用于处理器架构创新研究。为了说明这个问题,我们调研了近五年在计算机体系结构领域重要会议的论文,主要包括ASPLOS、ISCA、MICRO、HPCA、DAC,并按模拟器关键技术研究(模拟器研究)和使用模拟器的体系结构创新研究(架构探索)进行了统计,具体如表 4 所示。

表 4　近五年模拟器相关论文统计

会议	模拟器研究	架构探索
ASPLOS	2	80
ISCA	8	106
MICRO	10	164
HPCA	8	184
DAC	3	108

统计的论文中模拟器研究相关的论文有 31 篇,使用模拟器的研究有 642 篇。我们也统计了模拟器研究的论文在工业界和学术界中的分布情况,具体如图 6 所示,其中高校独立的研究占 73%,公司独立的研究占 3%,而工业界和学术界联合的研究占 24%。工业界除了英伟达,Intel 等传统芯片公司外,Google 等云计算公司也开始针对核心应用硬件设计需求探索模拟器技术。此外,多篇模拟器关键技术相关论文获最佳论文奖,如 ATC19[9] 和 CGO24[62] 的全系统功能模拟优化的论文及 ICCAD2021[12] 的处理器空间探索优化的论文。

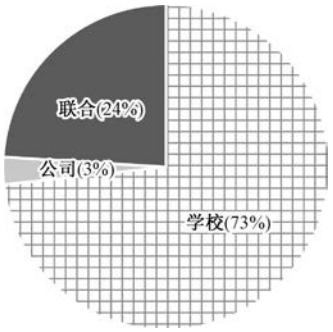

图 6　公司/高校相关论文分布情况

这些研究进展旨在推动体系结构模拟器在性能、准确性、可扩展性和易用性方面取得更大的突破。

4.3.2　针对性能的最新研究进展

概述:模拟性能直接影响评估效率,尤其希望使用真实应用作为模拟器输入进行处理器设计和优化时。性能受输入应用规模(指令数多则模拟时间长)、实现方式(软件优化影响性能)和建模层次(高层次性能高但精度低)等方面的影响。如图 7 所示,针对这三个方向,主要的加速技术路径可以归纳为三个层面:(1)通过采样模拟技术,选取程序中的代表性片段来减少需要模拟的指令总量;(2)通过加速模拟过程,利用现代多核硬件的计算资源或更好的算法来提升功能模拟或时序模拟的执行速度;(3)通过优化模拟模型本身,例如 ZSim、Interval Simulation、Prophet 等工作,提升建模的抽象层次,在保证关键事件精度的前提下,提高性能。

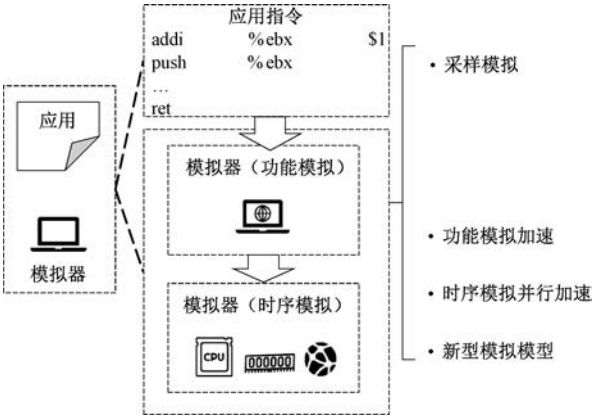

图 7　针对性能的最新研究进展

(1)采样模拟(Sampling Simulation):体系结构模拟器的模拟时间受驱动模拟的软件指令数影响。运行时间长的软件通常包含大量循环和递归,其不同迭代往往具有相似的行为和体系结构特征。采样技术通过分析访存或分支等特征,识别重复行为,并选取有代表性的代码片段作为整个重复行为的代表进行模拟,从而提升模拟性能。具有代表性的采样模拟有 UCSD 设计开发的 SimPoint[63]、CMU 设计开发的 Smarts[64]、采用采样加速的 LiveSim[65]、基于周期片段的模拟区域选择的 LoopPoint[10]、访存行为采样技术[66]、分层采样分析技术[67-68]、并行采样模拟技术[69],以及面向 GPU 模拟的细粒度采样方法 Photon[70]。

SimPoint 通过基本块向量(Basic Block Vector,BBV)分析,识别应用程序中的相似执行阶段,并仅模拟这些阶段的代表性部分。其自动化程度高,能够根据工作负载的动态行为选择采样点,无需手动干预。SimPoint 在模拟 SPEC 基准测试时,减少模拟指令数的同时保持了高精度,适用于各种类型的应用。

Smarts 是一种基于统计采样的模拟框架,通过分析指令流中的统计特征(如指令类型、分支行为)

选择采样区域。Smarts 结合静态分析和动态采样，能够在保持高精度的同时，减少模拟指令数。其在模拟多核工作负载时表现出色，在 8 路和 16 路乱序处理器的精确模拟上分别以 35 和 60 的平均加速比运行，特别适合快速评估处理器设计。

LiveSim 通过在线分析应用程序的执行行为，实时调整采样粒度，适应动态变化的工作负载。其动态采样策略能够在模拟过程中根据程序行为调整采样点，减少了冗余模拟。与完全模拟相比，在开始模拟后的 5 秒内，LiveSim 平均误差为 3.51%。

LoopPoint 通过基于循环的周期片段选择，优化了采样模拟的精度和效率。其核心思想是识别应用程序中的循环结构，并选择循环内部的代表性片段进行模拟。LoopPoint 实现多线程 SPEC CPU2017 基准测试的训练输入集高达 801 倍的加速，平均模拟误差仅为 2.33%。

访存行为采样技术工作中，Shen 等人提出了一种基于局部性相位预测的采样技术，通过分析程序的访存行为识别执行阶段。该方法利用缓存局部性特征，将程序划分为具有相似访存模式的相位，仅模拟代表性相位以减少指令数。其核心优势在于通过访存行为分析，优化采样点的选择，适用于数据库和大数据应用。

分层采样分析技术中，Zhang 等人提出了一种多级相位分析方法，将程序划分为粗粒度和细粒度相位，选择最具代表性的片段进行模拟。研究表明，与 SimPoint 相比，多级相位分析可以获得大约 8.3 倍的加速比，并且具有相似的精度；Li 等人进一步扩展了分层采样分析，设计了一种结合细粒度和粗粒度相位分析的多级采样模拟技术。这种方案使用细粒度模拟点来仅表示所选择的粗粒度模拟点，而不是整个程序执行，因此可以进一步减少功能和详细模拟时间。该方法减少了 90%功能模拟时间与 50%的精确模拟时间。与 SimPoint 相比，实现了 14.04 倍的几何平均加速比，且精度相似。

并行采样模拟技术工作中，Chen 等人提出了一种动态调整和部分采样的多线程模拟方法。通过将采样任务分配到多核宿主机的多个线程，并动态调整采样粒度，该方法能够提升模拟效率。该技术特别适合多线程应用或多核系统的快速性能分析。

Photon 是一种面向 GPU 模拟的细粒度采样方法，考虑了不同层次的 GPU 执行，如内核、线程束和基本块。Photon 不需要预先对 GPU 工作负载进行性能分析，而是采用基于识别高度重复软件行为的轻量级在线分析方法。结果显示，Photon 将 ResNet-152 在批量大小为 1 的情况下进行一次推理的模拟时间从 7.05 天缩短到仅 1.7 小时，且采样误差仅为 10.7%。

(2)模拟加速技术：模拟器运行速度直接影响评估效率。然而，由于模拟器通过软件模拟目标硬件，性能面临严峻挑战，这也是体系结构模拟器亟须解决的核心问题。加速技术可分为功能模拟加速、时序模拟并行加速和新型模拟模型三类，分别从不同维度优化模拟性能。

①功能模拟加速：功能仿真作为体系结构模拟器的前端，对整体性能至关重要，同时也常独立用作目标架构的软件开发环境，显著影响软件开发和验证效率。功能模拟器在模拟过程中主要涉及目标 CPU 模拟和外设模拟，因此加速的研究也主要关注在这两个维度。

对于目标 CPU 的模拟加速，主要涉及目标指令翻译性能、内存模拟性能及硬件利用能力。动态二进制翻译 DBT 是关键技术，通过在运行时将目标架构的指令翻译为宿主机架构的指令，减少解释执行的开销。代表性的工作如基于学习的动态二进制翻译技术[71-73]、基于 LLVM 的高效动态二进制翻译技术[74]、可重定向的系统级动态二进制翻译的技术[9]，以及面向众核硬件的功能仿真并行化[75-76]。

基于学习的动态二进制翻译技术工作中，Wang 等人提出了一种基于学习的 DBT 框架，通过机器学习分析目标指令和宿主指令之间的映射关系，自动生成高效的翻译规则，减少了手动优化的工作量。该方法通过训练神经网络模型，预测指令翻译的最佳路径，提高了翻译速度，尤其适用于跨 ISA（如 ARM 到 x86）的模拟。Song 等人进一步提出了一种增强的学习方法，结合静态分析和动态反馈，优化翻译规则的生成过程，使翻译性能提升了约 20%，其创新点在于引入强化学习模型，能够根据模拟器的运行时状态优化翻译路径，减少冗余计算。Jiang 等人通过参数化方法，减少了训练数据需求，同时保持翻译规则的泛化能力，该方法通过分析指令的语义和执行模式，生成轻量级翻译规则，使 DBT 框架更适合资源受限的场景。研究表明，Jiang 的方法在模拟复杂指令集时，翻译速度比基于强化学习的方法快约 24%，且代码覆盖率从 69.7%扩展到 95.5%。

基于 LLVM 的高效动态二进制翻译技术工作中，Engelke 等人利用 LLVM 的中间表示（IR）和优化器，开发了一种高效的 DBT 框架。LLVM 的优

化器能够对翻译后的代码进行高级优化(如循环展开、常量传播),减少了执行开销。研究表明,该框架在模拟复杂指令集(如 x86)时,翻译速度比传统 DBT 方法快约 140%,同时支持多种目标架构。其模块化设计便于扩展,适用于通用功能模拟器。

可重定向的系统级动态二进制翻译的技术工作中,Spink 等提出了一种可重定目标的系统级动态二进制翻译管理器,支持从高层次的来宾 ISA 规范自动生成模块,从而降低了支持新架构的开发成本。该系统在虚拟机提供的裸机环境中结合离线与在线优化技术,实现了高效的 JIT(Just-In-Time,JIT)编译,并在 SPEC CPU2006 基准测试中相比 QEMU 平均获得 2.21 倍(整数)至 6.49 倍(浮点)的性能提升。

面向众核硬件的功能仿真并行化工作中,Wang 等提出 COREMU 系统,其通过复用多个顺序仿真器实例并利用轻量通信层解耦多核并行带来的复杂性,实现了最多 255 核的高效仿真。研究表明,COREMU 在模拟多核系统时,性能比传统顺序仿真器提升约 50%;Hong 等提出 HQEMU 系统,利用多线程方法实现可重定目标的动态二进制翻译器,将翻译与优化任务分离至独立线程,以提升多核系统上的执行效率与支持灵活性。HQEMU 基于 QEMU 和 LLVM 构建,在 x86 到 x86-64 的翻译中,其整数与浮点性能相较 QEMU 提升分别达到 2.4 倍与 4 倍,接近本地执行速度。其关键创新在于通过多线程并行化翻译和优化过程,充分利用多核宿主机的计算资源。

对于外设的模拟,则主要涉及利用宿主机外设高效模拟目标外设,如基于 QEMU 的安卓模拟器 Trinity[77](加速 GPU 外设模拟)和 DAOW[78](加速 Windows 上的 Android 应用)。Trinity 是一种基于 QEMU 的安卓模拟器,通过将宿主机的 GPU 直接映射到模拟环境中,加速了 GPU 外设的模拟。Trinity 利用硬件虚拟化技术,实现了高效的设备直通,使安卓应用在模拟器上的图形渲染速度接近真实设备。研究表明,Trinity 在运行图形密集型应用时,展示了平均 97%的本地硬件性能和 99.3%的可靠应用支持,改善了移动应用开发和测试的效率。

而 DAOW 是一种在 Windows 上模拟 Android 应用的框架,通过优化 Android 运行时和系统调用,加速了跨平台模拟。DAOW 的关键创新在于减少系统调用的开销和内存管理的复杂性,通过定制化的运行时环境,实现接近本地的硬件性能。该框架特别适合移动游戏和应用的开发验证,降低了真机测试的依赖。

②时序模拟并行加速(Parallel Simulation):当前多核乃至众核硬件已成为宿主机的典型架构,为并行加速时序模拟提供了可能。并行模拟通过拆解串行模拟过程,利用并行计算资源加速模拟。由于模拟过程中存在共享资源(如共享 Cache,片上网络等)的访问,且应用执行时这些访问具有不确定性,因此降低同步开销对并行化效率至关重要。代表性的并行模拟器有 MIT 设计开发基于通用多核平台的 Graphite[79],以及基于 FPGA 加速的 RAMP gold[53]和 HAsim[80]。

Graphite 是一种基于通用多核平台的并行模拟器,通过将模拟任务分配到多核宿主机上的多个线程,实现了高效的并行化。其松耦合设计架构是关键,每个模拟线程负责一部分处理器核心或组件的模拟,并通过消息传递机制同步关键事件。Graphite 支持模拟 1024 核系统,使用八个 8 核主机时,其模拟速度约为本机原生执行的 1/41,同时保持高精度。

RAMP Gold 是一种基于 FPGA 的时序模拟加速框架,通过将模拟任务映射到 FPGA,利用其高并行性和低延迟特性加速模拟。RAMP Gold 在 FPGA 上实现完整的系统级模拟,包括处理器核心、内存层次和外设,模拟速度接近真实硬件,同时保持可编程性。研究表明,RAMP Gold 在模拟多处理器系统时,性能比软件模拟器高出两个数量级,特别适合探索大规模多核架构设计。

HAsim 是另一种基于 FPGA 的模拟框架,采用复用技术,将多个模拟任务在 FPGA 上并行执行,进一步提高了资源利用率。HAsim 通过将模拟任务分解为细粒度的子任务,并在 FPGA 上动态调度,实现了高吞吐量的模拟。

③新型模拟模型:通过修改抽象层次或驱动方式提升速度。传统的模拟模型采用时钟驱动与事件驱动相结合的方式,即在每个时钟周期检查处理器流水线各阶段输入缓冲区的事件,处理后将结果和时间信息传递至后续功能部件的输入缓冲区。这种方式能够精确捕捉处理器的时序行为,但存在性能瓶颈。首先,该方式引入分支开销,事件检查涉及大量条件判断,引入频繁的分支预测错误,降低指令流水线效率。其次,传统模拟模型的访存局部性差,缓冲区访问模式不规则,导致缓存未命中率升高,增加内存访问延迟。最后,规律性流水线行为的重复模拟也会消耗大量计算资源。由于规律性流水线行为(如指令解码、执行)具有较强的可预测性,而引发流

水线停顿(Stall)的长延时事件难以预测,且这些长延时事件对模拟器最终性能影响较大。

因此,一种优化方向是通过针对长延时事件建模并快速计算规律性行为的执行时间,从而大幅提升模拟速度。代表性的工作如 ZSim[21]、Interval simulation[81]和 HyFiSS[56]。另一种优化方式针对事件驱动带来的分支判断和局部性不佳问题,将驱动方式改为指令驱动。对于进入流水线的每一条指令,根据指令流经流水线需要的资源以及前序指令对功能资源的占用情况,确定当前指令对各个功能部件的占用情况;对于一些无法确定的动态行为,则通过一定的补偿算法提升模型的精度,代表性的模拟器如 Prophet[82]。

ZSim 是一种针对多核系统的快速体系结构模拟器,专注于长延时事件的精确建模,而对规律性流水线行为进行快速近似计算。这种方法显著降低了模拟开销,使 ZSim 能够以简单模型 1500MIPS 的模拟速度、详细模型 300MIPS 的模拟速度运行复杂多核工作负载,同时保持较高的时序准确性。ZSim 性能比传统并行模拟器快 2-3 个数量级。它采用动态二进制翻译技术,通过轻量级用户级虚拟化支持复杂工作负载,无需全系统模拟,高效设计使其广泛用于探索多线程应用和共享内存系统的性能瓶颈。

Interval Simulation 提出了一种高层次的抽象方法,通过将模拟过程划分为时间间隔(interval)来简化时序建模。每个间隔代表一组指令的执行,模拟器仅跟踪长延时事件对性能的影响,而非逐周期模拟所有指令的流水线行为。Interval Simulation 通过分析指令间的依赖关系和资源竞争,估算每个间隔的执行时间,从而大幅减少计算量。研究表明,该方法与周期精确模拟相比,可实现一个数量级的模拟加速,同时保持较高的时序准确性。Interval Simulation 已被集成到 Sniper 等模拟器中,广泛用于早期架构设计和快速性能评估。其核心优势在于通过牺牲部分微架构细节,显著提升模拟速度,适合快速迭代设计空间。

Lu 等人提出了 HyFiSS(Hybrid Fidelity Stall-aware Simulator),针对通用 GPU 架构 GPGPU 中的仿真速度与精度平衡问题。HyFiSS 结合了高精度与快速模拟的混合策略,通过在停顿敏感阶段切换到高保真模式,在其他阶段采用简化模型,从而提高整体仿真效率。该方法在多个 GPGPU 基准程序上验证了其在延迟分析、吞吐量建模等方面的准确性,同时模拟速度优于现有保真模拟器数倍。

Prophet 是一种面向并行指令的模拟器,专为多核体系结构设计。它采用指令驱动的模拟框架,通过静态分析指令流和动态调度算法,快速计算指令在流水线中的执行路径和资源占用情况。Prophet 支持模拟 4096 核,对于全系统和用户级模拟,Prophet 分别实现了约 98 和 235MIPS 的平均模拟速度,每周期指令数 IPC(Instructions Per Cycle,IPC)错误率仅为 3%。其高效的指令驱动设计使其特别适合模拟大规模多核系统,以及用于研究多核架构的性能优化和资源调度策略。Prophet 的模块化设计还允许用户轻松扩展以支持新型指令集或微架构。

总结与讨论:尽管上述加速技术在不同场景下都取得了显著效果,但每条技术路径也都存在一些不足。采样模拟对于行为模式不规则、动态变化剧烈的真实应用(如交互式应用、复杂的服务器负载),很难找到具有代表性的采样点,可能导致较大的模拟误差,遗漏一些低频事件,从而导致准确性偏差大。该方法也很难应用到系统软件,如全系统模拟场景。对于并行模拟的方法,随着宿主机核心数的增加,通信和同步的开销可能超过并行带来的收益,导致加速比无法线性扩展,出现性能瓶颈。新型模拟模型的方法会牺牲部分微架构细节的精确性,可能不适合用于研究非常细粒度的硬件行为。同时,体系结构模拟器是一个复杂系统,很多时候需要综合维度的加速,而在组合使用这些技术时,需要综合考虑哪些因素,目前也缺乏深入讨论和研究。

4.3.3 针对准确性的最新研究进展

概述:体系结构模拟器的准确性是其核心性能指标,直接影响其在处理器设计验证和性能评估中结果的可信度。一方面,在已知硬件参数的情况下,为提升模拟速度,传统建模方法常忽略部分硬件细节或采用宏观抽象视角,导致模拟结果与真实硬件的结果存在偏差。此外,随着新型硬件不断涌现,传统建模方法难以体现新硬件特性,因此需针对新硬件改进或重新设计模拟模型,以确保准确性。另一方面,由于商业机密等原因,建模过程中可能无法获取处理器的完整设计细节,只能基于可用信息完成建模,再通过数据分析结合硬件测试结果对参数进行拟合,以实现与硬件一致的相对精度。如图 8 所示,提升准确性的研究可分为两大场景:一是当硬件参数已知时,如何通过更准确的模型来刻画真实硬件的行为,包括新型模拟模型、领域专用硬件模拟模型;二是当硬件参数未知(如商业闭源处理器)时,如

何通过自动化方法拟合出等效的参数配置—准确拟合硬件参数。

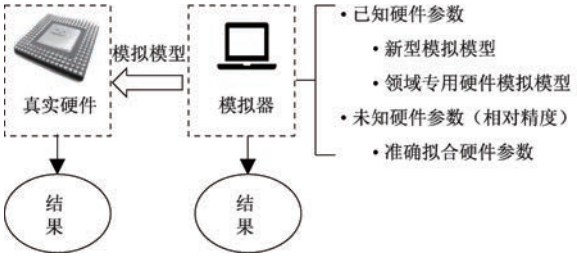

图 8　针对准确性的最新研究进展

(1)新型模拟模型:新型模拟模型的设计旨在解决传统模拟器为追求模拟速度而忽略硬件细节的问题,同时保持或提高模拟的准确性。为简化计算和提高速度,传统模拟器常忽略或简化缓存行为、分支预测复杂性及内存访问模式等细节,在设计模拟模型时采用更宏观的抽象视角进行建模,导致模拟结果与真实硬件性能存在显著差异。因此,新型模拟模型综合平衡模拟器性能、准确性、可扩展性,在保证模拟速度的同时提升准确性。代表性的模拟器如ZSim,Zsim 通过动态二进制翻译技术来提高模拟速度,但同时为确保准确性,ZSim 详细建模内存层次结构,包括多级缓存和目录式一致性协议,精确捕捉多核环境下的共享内存访问行为。此外,ZSim 支持异构系统模拟,能够处理不同类型的核心和加速器,确保其准确反映现代复杂计算平台的动态行为;Interval simulation:对规律的模拟通过快速计算的方式提升模拟速度,同时采用事件队列的方式,对缓存行为、分支预测、内存访问等硬件细节进行准确模拟,保证了模拟准确性;指令驱动模拟器 Prophet 为了保证准确性,在分支预测错误或缓存访问顺序错误等场景采用补偿算法来提升精度,从而提供可靠的模拟指标。

(2)领域专用硬件模拟模型:随着半导体技术快速发展,领域专用硬件不断涌现,传统建模方法无法体现新硬件特性,需针对其改进或重新设计模拟模型以确保准确性。本节调研了最新研究进展,将其按主要应用领域分为以下四类:存储与内存模拟、加速器与专用硬件模拟、GPU 与图形模拟以及数据流与新兴架构模拟。每类模型针对特定硬件需求提供了创新解决方案。

①存储与内存模拟:此类模型专注于模拟现代存储与内存系统的复杂行为。新的固态驱动器(Solid State Disk, SSD)模拟框架 Amber[83] 是 SimpleSSD 的第二代版本,它是一种专为 SSD 设计的模拟框架,旨在提供精确的全系统模拟能力。Amber 支持对 SSD 内部资源(如嵌入式 CPU 核心、DRAM 和多种闪存技术)的详细建模,并能够在全系统模拟环境 gem5 中运行。Amber 包括完整的固件栈,如 DRAM 缓存逻辑、闪存转换层和主机接口层,并通过修改主机 DMA(Direct Memory Access, DMA)引擎和系统总线支持多种标准协议,实现与主机系统的无缝集成。

Øsim[84] 是一个大内存模拟器,能够模拟比宿主机内存大 20-30 倍的环境。它通过高效的内存管理技术,准确模拟大数据和 AI 应用中的内存行为,满足现代高内存需求系统的评估需求。

Mocktails[85] 专注于捕捉具有移动架构的内存行为。它通过分析移动设备的独特内存访问模式,改进模拟器在移动系统中的准确性,特别适用于智能手机和嵌入式设备的性能评估。

Pouya Esmaili-Dokht 等人设计了 Mess[61] 框架(Memory System Simulator Framework),致力于解决现有内存子系统建模缺乏一致性与可移植性的问题。Mess 基于带宽-延迟曲线对内存层次结构进行抽象建模,支持包括 DDR4、HBM2、CXL 在内的多种主流和前沿存储技术。该框架可无缝集成至 ZSim、gem5 等模拟器中,能够对复杂应用中的内存访问行为进行准确分析。

②加速器与专用硬件模拟:此类模型针对加速器和专用硬件的模拟需求,广泛应用于机器学习和内存密集型计算。NeuroMeter[86] 是一个针对机器学习加速器的综合建模框架。它首创架构级功率、面积和时序建模,提供了高精度的性能预测能力。NeuroMeter 通过集成这些模型,帮助研究者和设计者探索机器学习加速器的设计权衡,确保模拟结果与真实硬件行为一致。

gem5-SALAM[87] 是 gem5 模拟器的扩展,支持基于 LLVM 的加速器建模。它通过集成 LLVM 框架,准确模拟异构系统中加速器的行为,提升了复杂系统模拟的保真度。

Li 等人提出编译驱动模拟方法[60],该方法利用编译器技术模拟可重构硬件加速器。通过编译器优化和动态调度,它准确捕捉加速器的自适应行为,确保模拟结果反映可重构系统的真实性能。

Chen 等人提出 LLMCompass[88],一种面向大语言模型推理任务的高效硬件设计模拟框架。该工作聚焦于推理阶段的系统级建模与评估,集成了自动映射机制和基于面积、功耗与性能的多维代价模

型,能够在通用 GPU 上快速完成复杂模型的模拟,并提供低于 5%误差的性能估计。该框架降低了从模型到硬件系统设计之间的验证门槛,为 LLM 类加速器设计提供了可行的工程路径。

Hyun 等人提出了 uPIMulator[89],一个专为 UPMEM 通用型 PIM(Processing-In-Memory)架构设计的周期级仿真平台。UPMEM 是首个商用、可编程的通用 PIM 技术,其硬件结构与编程模型高度异构,传统模拟工具难以精准建模。uPIMulator 结合 LLVM 编译链与自定义链接器与汇编器,能够从 UPMEM-PIM 源代码生成 ISA 兼容指令并进行精确的周期级模拟。其硬件模型完整覆盖 DPU 微架构、Revolver 管线、scratchpad/DRAM 分级内存模型及 CPU/DPU 通信通道。uPIMulator 的提出为 PIM 系统的深入研究提供了高保真、可扩展的模拟平台,也为未来通用型 PIM 设计指明了发展方向。

UniNDP[90]是一个针对近内存处理(Near DRAM Processing,NDP)架构的统一编译与仿真工具,专注于内存密集型机器学习工作负载。UniNDP 的周期精确、指令驱动仿真器通过跟踪处理单元、内存单元、缓冲区和内部总线的工作状态,精确评估硬件性能,并支持指令重排序和乱序执行以探索性能上界。实验表明,UniNDP 在多种 NDP 架构上实现 1.05-3.43 倍的加速,为 NDP 架构设计和部署提供了重要指导。

③GPU 与图形模拟:此类模型专注于 GPU 架构的模拟,涵盖图形渲染和通用计算。

Emerald[91]是一个为图形化和通用 GPU 应用程序统一设计的模拟器。Emerald 支持 OpenGL (v4.5)和 OpenGL ES (v3.2)着色器在 GPGPU-Sim 的计时模型上运行,并与 gem5 和 Android 集成,因此支持系统级 SoC 模拟。

MGPUSim[42]是一个周期精确的多 GPU 模拟器,MGPUSim 基于 AMD 图形核心 GCN3 指令集架构,内置支持多线程执行,以实现快速、并行化和准确的模拟。在性能准确性方面,MGPUSim 与实际 GPU 硬件相比,平均误差仅为 5.5%,同时在 4 核 CPU 上实现了 3.5 倍(功能仿真)和 2.5 倍(精确仿真)的平均加速,同时保持与串行仿真相同的准确性。

GCoM[92]提供了一个详细的 GPU 核心模型,该模型能够捕捉现代 GPU 中关键的停顿事件,从而为计算机架构师提供更准确的早期设计空间探索工具。使用 NVIDIA RTX 2060 配置进行的实验表明,GCoM 在建模准确性方面优于现有的 GPU 分析模型。GCoM 的平均绝对误差为 10.0%,而现有的 GPU 分析模型的平均绝对误差为 44.9%。

SeyyedAghaei 等人提出了一种面向 GPU 架构的尺度建模仿真方法[93],旨在以低成本高效率地预测未来大规模 GPU 系统的性能。该方法通过仿真两个规模较小的 GPU 系统并结合 LLC 失效率曲线,在无需构建目标系统详细仿真模型的前提下,准确预测包括 128-SMGPU 系统甚至多芯片 GPU 系统的性能。该工作通过基于失效率曲线分区的预测模型分别建模三类典型扩展趋势,并针对强扩展与弱扩展工作负载提供不同的预测路径。实验结果显示,在预测 128-SM 系统性能时,平均误差仅为 4%(强扩展)和 1.7%(弱扩展),优于传统比例缩放与回归预测方法,同时在弱扩展下可获得最高 9.3 倍的仿真加速。该方法也首次应用于多芯片 GPU 性能预测,平均误差仅为 2.5%。

④新兴架构模拟:此类模型针对新兴计算范式(如数据流架构)的模拟需求。Yang 等人构建了一个支持数据流执行语义的抽象模拟器框架[94],旨在支撑未来数据流架构的研究需求。该框架通过对计算单元、通信路径及调度逻辑进行模块化建模,为模拟动态可变拓扑、任务映射和负载调度等提供了灵活支持。该方法可有效捕捉数据流处理器中的关键性能瓶颈,为异构并行架构的设计优化提供了重要参考。

这些方法应对传统模型在精确性、组件交互及新型处理器技术建模中的局限性,通过改进建模方法、优化参数拟合和开发新型硬件模型,提升了体系结构模拟器的准确性。

(3)模拟器参数拟合:由于商业机密或技术限制,体系结构模拟器建模过程中常无法获取处理器的完整设计细节或某些功能单元的具体信息,需基于公开或部分信息完成初步建模,再通过数据分析和硬件测试结果拟合参数,以实现与硬件接近的相对精度。现代处理器架构复杂,参数和配置繁多,搜索空间庞大,传统拟合方法效率低下,尤其在处理大规模参数空间时。最新研究采用先进搜索算法和自动化工具,提升参数拟合的准确性和效率。

自动化拟合技术通常遵循以下迭代优化工作流程:将模拟器本身视为“黑盒”函数,并将模拟器的可调参数(如缓存大小、流水线宽度、端口数量等)定义为一个巨大的搜索空间。自动化拟合工具自动地执行以下循环:(1)根据其内部的搜索策略,在参数空

间中选择一组候选参数;(2)使用这组参数配置并运行体系结构模拟器,得到一个性能结果(如每周期指令数);(3)将该模拟结果与从真实硬件上测得的基准值进行比较,计算出误差;(4)将此误差作为反馈,指导其搜索策略在下一轮迭代中寻找更适配的参数组合。上述过程不断重复,直至误差收敛到足够小,从而找到一组能使模拟器行为最逼近真实硬件的"最优"参数。

PMEvo[95]是一种通过进化优化推断乱序处理器端口映射的参数拟合工具。它利用线性映射方法,从物理机的性能测量数据中提取关键参数,优化模拟器的配置以提高保真度。PMEvo 特别适合处理复杂处理器架构,其进化算法能够高效探索参数空间,找到接近真实硬件行为的配置。

OpenTuner[96]是一个通用的程序自动调优框架,广泛应用于优化模拟器参数配置。它通过集成多种搜索技术(如遗传算法、模拟退火),自动调整模拟器的参数组合,以最小化模拟结果与硬件性能的偏差。

DiffTune[97]由 MIT 开发,基于可微分编程技术精细化模拟器参数。它将参数调优问题建模为可优化的目标函数,通过梯度下降方法从硬件测量数据中学习最佳参数配置。DiffTune 能够使模拟器的时钟周期与真实机器高度一致,提高准确性,尤其适用于复杂处理器架构的模拟。

VAESA[98]由加州大学伯克利分校(UCB)设计,利用变分自动编码器(Variational Auto Encoders, VAEs)加速参数空间搜索。VAEs 通过学习参数空间的连续表示,高效探索庞大设计空间,快速找到高精度的模拟器配置。VAESA 特别适合硬件加速器设计,其方法可扩展至其他复杂模拟场景。

EXAMINER[99]由浙江大学和香港理工大学联合开发,是一种自动定位模拟器与真实设备指令不一致的工具。它通过比较模拟器输出与真实硬件的指令执行行为,识别不一致点并提供修正建议,从而改进模拟准确性。EXAMINER 特别适用于 ARM 架构的模拟器验证。

总结与讨论:上述最新研究方法在实践中也各自面临挑战与局限性。领域专用硬件模拟模型通常与特定的硬件或应用领域深度绑定,缺乏通用性,将其设计思想和实现迁移到其他类型的硬件上非常困难。而自动化参数拟合方法虽然通过优化使宏观性能指标对齐,但目前方法的拟合只针对特定场景和输入,当输入场景变化时,拟合结果是否需要重新计算面临挑战。因为随着硬件多样性、差异的变大以及场景的丰富,如何设计通用拟合工具,是一个重要而有挑战的问题。

4.3.4　针对可扩展性的最新研究进展

概述:模拟器是复杂的软件系统,一个模拟器包含的代码规模常超过百万行。例如 QEMU 拥有 1821644 行代码①,gem5 拥有 1098295 行代码②。对于如此复杂的软件系统,可扩展性至关重要。如图 9 所示,模拟器可扩展性主要分为宏观和微观两方面:宏观可扩展性聚焦对于不同宏观模拟模块,可以根据需求进行扩展和替换;微观可扩展性则聚焦于时序模型内部具体硬件部件的灵活插拔、交互及其时序影响的精确传递。

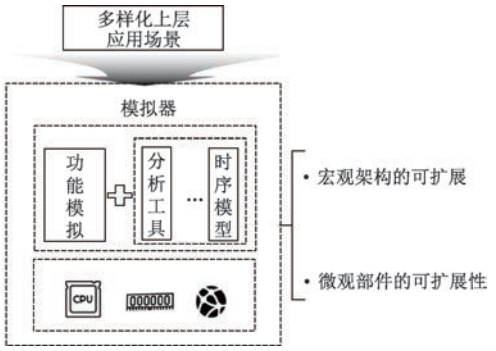

图 9　针对可扩展性的最新研究进展

(1)松耦合模拟器架构(Loosely Coupled Design Architecture):主要设计并实现的功能模拟模块和时序模拟模块的松耦合交互关系。随着处理器技术发展,研究人员需扩展已有模拟器。传统紧耦合架构模块间高度依赖,新增或替换模块困难。同时,紧耦合架构需要频繁进行模块间的数据传递和同步,导致系统开销较大。松耦合架构中,功能模拟模块单向传递正确执行路径指令至时序模块。只有在时序模拟发现错误分支预测或共享资源访问顺序错误等问题时,时序模拟模块才与功能模拟模块进行同步,要求功能模拟模块按时序模拟提供的信息进行回滚。这种松耦合的方式降低耦合度和信息传递量,提升性能的同时减少开发工作量。代表性的松耦合架构模拟器有:使用定时反馈机制的 COTSon[100]、多核架构下的 Transformer[8,14]以及在其基础上扩展的 RPSim[4]。

COTSon 是一种全系统模拟器,功能仿真层基于 AMD 的 SimNow 技术,支持 x86 和 x86_64 平台,提供快速的指令执行模拟。其时序模拟模块通

① QEMU, https://openhub.net/p/qemu。
② gem5, https://openhub.net/p/gem5。

过定时反馈机制与功能模拟模块交互，仅在关键事件时进行同步，大幅减少了同步开销。COTSon 的松耦合架构提供了高度的可扩展性，通过其可插拔设计，允许用户开发自定义模块以满足特定建模需求。

Transformer 是一种专为多核架构设计的松耦合功能驱动全系统模拟器，由 Zhang 等人于 2015 年开发。Transformer 通过架构无关的接口将 FM 和 TM 分离，使两者能够独立运行，同时通过轻量级机制检测并恢复 FM 和 TM 之间的执行偏差，确保周期精确性。例如，Transformer 的 FM 负责指令执行和软件行为，而 TM 模拟微架构细节和时序特性，仅在共享资源竞争或核心间通信时进行同步。这种设计降低了同步开销，并简化了扩展过程。研究表明，扩展 Transformer 以支持 x86 功能模型（基于 QEMU）仅需约 180 行代码，耗时约两个月，展示了其卓越的可扩展性。

RPSim 则是在 Transformer 的松耦合架构基础上，针对 SoC 软件开发的快速原型需求进行了创新扩展。它利用松耦合特性解决了两个核心痛点：一是 IP 核的快速集成，它将 IP 核的网表部署在 FPGA 上运行，并将 FPGA 视为一个特殊的、通过通用接口与 FM 通信的功能单元，FM 从 FPGA 获取时序信息后再传递给 TM；二是新指令的快速扩展，它通过让用户在应用层调用特殊库函数的方式来模拟新指令，由 FM 识别这些库函数并生成正确的时序信息给 TM，从而避免了耗时的编译器修改。RPSim 的工作展示了松耦合架构在支持软硬件协同仿真和快速迭代方面的巨大潜力。

(2)部件模型可扩展性：指允许用户通过对外接口编写新部件扩展模拟器功能，便于交互、配置和扩展。为满足现代计算系统中多样化的应用需求和灵活定制，模拟器需向模块化、易扩展及用户友好的方向发展。传统模拟器通常具有固定的架构设计，扩展新功能或部件需大幅修改内部代码，限制了用户添加新部件或功能模块的能力，使扩展复杂耗时，且缺乏友好的配置接口，用户需直接修改源代码，增加了技术门槛和成本。最新的研究技术对部件模型可扩展性做了大量研究工作，如下：

SSD 模拟框架 Amber 的可扩展性体现在其模块化设计上，允许用户轻松添加或修改 SSD 组件，例如不同的闪存技术、DRAM 缓存大小或自定义 FTL 算法。Amber 能够捕捉 SSD 组件在各种操作系统和硬件平台下的动态性能和功耗特征，使其成为研究 SSD 行为和优化 SSD 设计的重要工具。

Accel-Sim[22] 是一种专为 GPU 建模设计的可扩展模拟框架，支持 NVIDIA 的机器指令集体系结构进行基于踪迹(trace-based)的模拟。它能够模拟任何 CUDA 二进制文件，包括使用 cuBLAS、cuDNN 和 pyTorch 等库的复杂应用程序，无需功能实现。Accel-Sim 的可扩展性体现在模块化设计上，允许研究人员轻松添加或修改组件以建模新 GPU 架构或特性。其提供了一套全面的微基准测试和自动相关绘图工具，简化了模型验证和调整过程。

Vulkan-Sim[101] 是一种基于 GPGPU-Sim 和 Mesa 的 GPU 架构模拟器，专为 Vulkan 光线追踪工作负载设计。它通过集成 Mesa 和添加专用的光线遍历和交叉单位，实现了对 Vulkan 光线追踪管道的详细模拟。Vulkan-Sim 的可扩展性体现在其模块化设计上，允许用户添加新的光线追踪单元或修改现有模型以支持新兴 GPU 技术。

gem5[18,102] 同样也是一个可扩展的全系统模拟器。gem5 使用 Python 脚本进行配置和扩展，允许用户通过 Python 定义新组件或修改现有组件，并通过良好的 API 集成到模拟中。gem5 还支持 SystemC 接口，允许直接将硬件描述语言（Hardware Description Language，HDL）模型集成到模拟中，这对于模拟自定义硬件或与其他 SystemC 工具协同工作尤为有用。

总结与讨论：从上述最新研究来看，虽然针对各种维度的松耦合技术进行了很多研究，可以极大扩展已有工具的生态适配性，但这些技术并没有支持到主流的具有代表性的系统中，如 gem5 仍然使用紧耦合设计，导致适配受限，系统可扩展性差等问题。因此，未来推动这些技术在典型工具中落地，将极大提高模拟效率，降低开发的人工成本。

4.3.5 针对易用性的最新研究进展

概述：近年来，体系结构模拟器的易用性成为研究者关注的重要方向。由于模拟器通常具有复杂的配置接口、庞大的参数体系和难以追踪的运行过程，非专家用户在使用过程中面临较高门槛。如图 10 所示，提升易用性的研究主要通过以下三个方面展开：(1)通过实时监控与交互式控制，将模拟过程可视化；(2)通过图形化界面，简化模型构建与配置的复杂度；(3)通过检查点等技术，实现实验流程的自动化。

(1)实时监控与交互式控制：指模拟器能够在模拟过程中动态展示关键性能指标，并允许用户对仿真的状态进行交互式控制。最新的研究技术 Akit-

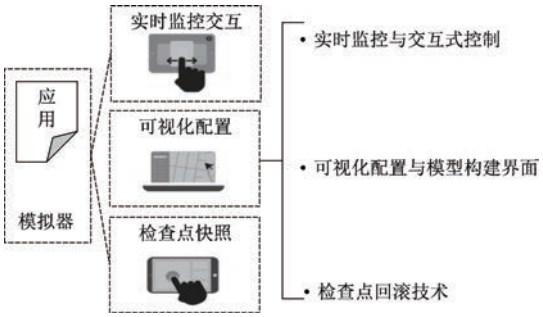

图 10　针对易用性的最新研究进展

aRTM[103]提供了一个基于 Web 的交互式工具,能够在模拟器运行过程中实时展示执行状态,并允许用户动态暂停、快进或终止仿真,从而实现“及早失败、快速反馈”的设计理念。该工具通过仪表盘式的界面揭示关键性能指标(如 IPC、缓存命中率和分支预测正确率),并在案例研究中帮助用户定位仿真错误,甚至促成模拟器本身的修补。用户研究表明,AkitaRTM 不仅解决了常见的工作流低效问题,还通过可视化教学的方式,增强了仿真器对新手研究者的教育价值。

(2)可视化配置与模型构建界面:指模拟器支持用户通过拖拽组件的方式搭建系统模型,并自动生成底层配置脚本,减少了用户对底层语言和复杂数据结构的依赖。相关研究技术有 gem5 模拟器。为了简化模拟器配置和体系结构建模流程,gem5 社区开发了基于 Qt/PySide2 的图形用户界面,用户可以在左侧目录中检索模拟对象(SimObject),将其拖拽至画布上形成组件层次关系,并在属性表中直接调整参数。该 GUI(Graphical User Interface,GUI)支持通过可视化连线完成端口连接,自动生成对应的 Python 配置脚本,极大降低了手写配置脚本的复杂度,并让体系结构设计更加“所见即所得”。

(3)检查点快照:检查点回滚技术(Checkpoint)是提升模拟效率和可复现性的关键易用性特性,它指将某一时刻的完整系统状态(包括 CPU 寄存器、内存、设备状态等)保存为快照,并能在后续任意时刻快速恢复的能力。主流模拟器普遍支持该技术。例如,gem5 提供了成熟的检查点机制,其典型用法是先使用 AtomicSimpleCPU 等快速的功能模型执行程序的引导和初始化部分,然后在目标区域前创建检查点,随后加载该检查点并切换到 O3CPU 等详细的时序模型进行高精度分析。商业模拟器 Simics 同样支持该技术,在此基础上其支持的“反向执行”功能允许用户不仅能恢复到某个检查点,还能在模拟过程中像“倒带”一样回溯,极大地便利了对瞬态错误和复杂系统行为的调试。

总结与讨论:近年来关于“易用性”研究的主流路径改善了用户体验,但它们也引入了新的问题和局限性。例如可视化配置界面虽然直观,但其功能更新往往滞后于模拟器。一些高级、冷门或最新添加的参数可能无法通过可视化页面进行配置,导致高级用户仍需依赖复杂的命令行脚本。而检查点回滚技术往往不支持在线回滚,需要应用或系统在新的进程中重新加载回滚,这种方式属于离线回滚,需要与录制过程相同的运行环境,这将大大限制它们的使用,因为一些用户出于隐私或商业原因,可能无法与程序员共享第三方库或敏感输入。未来的发展趋势将是智能化与集成化,即利用 AI 技术自动推荐配置参数、自动解析结果、自动定位性能瓶颈等,并将这些智能分析、可视化配置、回滚技术、交互式运行的功能集成于一个统一的、对用户友好的集成开发环境中。

5　讨论与建议

近年来,软硬件协同受到越来越多国内企业的重视,而国际形势也促使国内在持续投入突破各种底层根技术,如处理器、操作系统和编程语言等。而体系结构模拟器作为处理器等领域的核心工具之一,发展水平直接关系到我国在处理器领域的创新能力,需要在技术等方面做长期的投入和积累,主要包括综合框架、关键技术突破、智能分析技术、开源生态建设等四个方面,具体如图 11 所示。

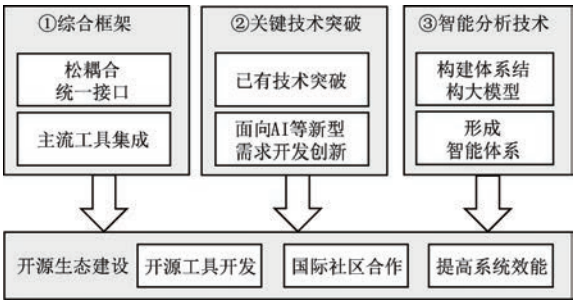

图 11　体系结构模拟的思考

5.1　综合框架

在体系结构模拟和分析过程中往往涉及多种需求和多种工具,如功能模拟、时序模拟以及不同系统软件和硬件体系的适配。若各个模块形成统一架构,则更容易实现满足多样化需求的系统。然而,目前尚未形成统一的综合框架,导致工具之间协同困难,限制了模拟器的适配能力和应用范围。这种适

配性不足主要体现在两个方面:一方面,功能模拟与后端工具协同不足。例如,QEMU 作为当前最流行的开源功能模拟器,与 gem5 以及 Valgrind、DynamoRIO 等分析工具之间缺乏有效的互通与协同机制。这种不兼容性导致开发过程中功能重复实现,资源浪费严重。同时,Valgrind 和 DynamoRIO 在各自框架内独立开发了功能模拟能力,但无法与 QEMU 实现互操作,不仅限制了全系统分析的能力,也削弱了 gem5 对不同软件环境的适配能力。工具割裂化问题增加了开发负担,降低了整体模拟效率。另一方面,功能模拟器与应用场景的适配性较差,如图 12 所示,QEMU 对安卓系统的支持不完善,表现为对安卓运行环境所需的移动平台硬件支持不足。QEMU 在车载系统及其他嵌入式平台的模拟上也存在类似的问题,影响了其在这些领域的适用性。

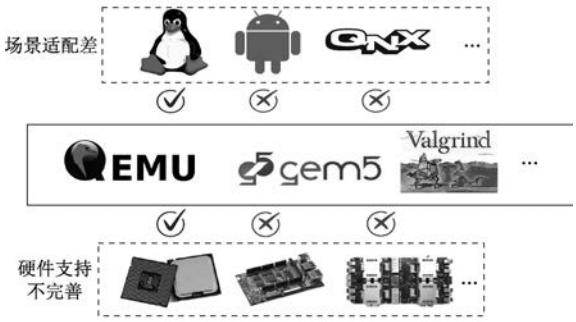

图 12 开源模拟器适配性差

为解决此问题,需要构建一个以功能模拟为核心的统一综合框架。该统一框架需要以当前生态最完善的体系结构模拟相关工具为基础,并在架构层面提供一种标准的指令集无关的数据传递方式和统一的工具驱动运行方式,使得便于扩展新工具和集成主流的模拟与分析工具。同时需要提升智能化能力,提升评估过程中的数据分析和决策能力。

要实现这一目标,框架的设计需围绕几个关键机制展开,具体如图 13 所示。(1)框架需要具有较好的模块化,不同模块采用松耦合设计。模块的划分方法需要与主流工具或行业内典型的划分方法一致,减少扩展过程中彼此的干扰。模块间的松耦合设计则在扩展新功能或新模块时可以具有更好的开发效率,也有利于模拟器的并行加速。(2)不同模块间需要有统一的指令集无关的数据交互机制和接口,以及需要统一的不同工具的驱动协同机制和接口。这些接口作为前端(功能模拟)与后端(各种分析工具)之间的交互标准,实现了两者的协同。前端功能模块(如 QEMU)负责正确执行程序并产生运行时动态指令流和内存访问流等事件;后端工具集则独立地消费这些事件并进行相关分析。统一的数据格式可以降低后端工具对前端模块或模拟环境(如指令集变化)的依赖,从而提升前后端模块的独立性。同时,必要的协同机制,可以满足不同工具的需求。如后端分析工具需要进行路径探索(覆盖性测试)或时序关系敏感的分析(分支错误路径或线程调度),在这些场景下需要前端有回滚机制,而后端工具基于传递数据检测时序或路径的不一致并根据需求驱动前端模块进行回滚。(3)该框架需要集成现有不同主流工具,兼容这些主流工具的生态。通过标准化的中间接口,研究者可以便捷地将业界主流工具作为插件接入框架。例如,可以利用 QEMU 强大的全系统功能模拟能力作为前端,同时接入 gem5 时序模型、Valgrind 工具集进行时序模拟与性能分析。框架可以避免在功能模拟等基础模块上的重复开发,并方便地与已有工具进行横向对比和协同分析。(4)随着智能技术的提升以及各种设计的累积,框架需要支持大模型等智能化技术,辅助研究者进行更高效的设计决策。在体系结构模拟和分析过程中,可以累计海量数据,相关数据可以用于训练不同目标的体系结构大模型,将原始的性能数据转化为有价值的洞察。例如,用于因果分析的大模型或者用于支持决策的大模型。通过集成丰富的智能工具集,能够提升在体系结构模拟过程中产生数据的处理和分析效率。

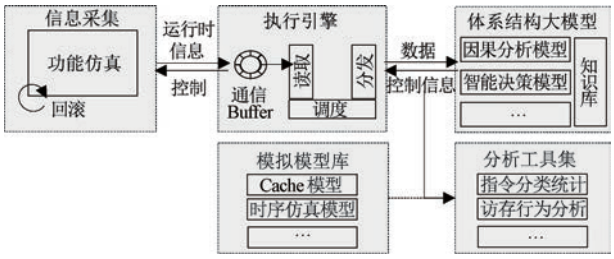

图 13 体系结构模拟综合框架

基于该框架思路,研究团队已完成以 QEMU 为功能模拟前端的框架设计与实现。QEMU 完成指令流和访存行为的收集,并转换成体系结构架构无关的中间表示形式传递给后端分析工具。执行引擎定义多种数据传递和工具驱动方式,满足不同工具需求。QEMU 端支持多种粒度(指令粒度、基本块粒度等)回滚机制,执行引擎及工具端支持一些通用时序违反判断机制(如共享内存访问顺序不一致等)以及驱动 QEMU 进行回滚机制。基于该框架,

已将 Valgrind 和 gem5 接入该框架。通过接入该框架,原有 Valgrind 的工具集可以进行包含操作系统行为的各种分析,而不只局限于应用层次的分析。通过接入该框架,gem5 可以在功能段继承 QEMU 的生态,克服原来支持板卡种类有限的问题。也可以克服原来 16 核及以上全系统仿真易于崩溃的问题,可以扩展到 64 核模拟。目前基于该框架的部分功能已在国内某些头部公司的软硬件协同设计中得到应用。

5.2　关键技术突破

随着计算需求的快速变化,尤其在 AI 驱动的计算场景下,如大模型训练与推理对新型计算架构的迫切需求,处理器架构探索与软硬件协同设计日益复杂,对体系结构模拟器的性能、准确性和易用性提出了更高要求。为此,体系结构模拟器需要在现有技术维度上实现突破,同时面向 AI 等新型需求开发创新技术,以提升模拟效率、精度和适配性。

在已有技术维度上,体系结构模拟器可通过混合模型、采样技术和并行加速实现精度与速度的平衡,混合模型结合周期精确模拟和事件驱动模拟,根据模拟场景动态调整粒度采样技术,通过分析代表性指令片段减少计算量,提升速度。同时,结合现有典型可用硬件并行特点,通过并行模拟技术利用多核主机或分布式云计算平台,优化模拟效率。此外,模块化架构的改进是提升可扩展性的关键,通过标准化接口和松耦合设计,模拟器可快速集成新型功能单元,降低开发成本。

同时,随着 AI 技术,乃至大模型的飞速发展,AI 技术将赋能各行各业,并成为硬件处理的典型应用。因此,需要结合 AI 领域应用进行体系结构探索的需求及 AI 应用的特点,有针对性地对体系结构模拟相关技术进行有针对性的优化和设计。AI 应用如大模型训练和推理通常涉及高度并行的计算、大量内存访问以及 Tensor 计算等专用操作,同时性能涉及计算、存储和网络的多个维度的特性。因此,需要深入分析 AI 应用的特点和进行体系结构性能评估的需求,开展有针对性的体系结构模拟器架构、加速和优化技术相关研究,并基于这些技术突破开发实现各种软件工具,从而有效推动各种 AI 应用体系结构模拟评估的需求。

5.3　智能分析技术

近年来,人工智能大模型已在多个领域取得突破性进展,成为提升生产效率的重要工具。例如,GitHub Copilot 利用大模型辅助编程,提高了开发效率;谷歌在 TPU[59] 设计过程中使用 AI 进行设计空间搜索,大幅减少了探索时间,提高了芯片优化效率;国内中国科学院计算技术研究所等单位提出的 QiMeng[93] 系统,展示了利用大模型实现处理器芯片和相关基础软件全自动设计的可行性,为该方向提供了前沿的实践案例。然而,在体系结构仿真领域,尽管仿真过程涉及大量的数据和复杂的参数配置,但目前仍缺乏专门针对该领域需求的大模型工具,导致体系结构分析在很大程度上仍依赖人工经验。

人工智能大模型是辅助体系结构分析的重要工具。因此要在体系结构仿真领域发挥智能分析的优势,需要做好数据积累,并根据体系结构的实际需求构建专门的体系结构大模型,形成一个智能体系。如图 13 所示,该模型是包含多个协同工作子模块的智能体系,以实现从“数据”到“洞察”的转化。如因果分析模型可以学习指令依赖、资源竞争和性能指标间的复杂关系,自动推断出性能瓶颈的根源;智能决策与生成模型能够根据性能、功耗、面积(PPA)的综合目标,自动生成满足要求的微架构规格,甚至直接输出参数化的 RTL 代码框架或可配置的模拟器脚本;知识库构建将大量仿真实验的结论、已知的架构设计原则以及顶级会议论文中的设计思想进行结构化存储,形成一个体系结构领域的专业知识库。

5.4　开源生态建设

开源生态是体系结构模拟器技术演进的核心驱动力。通过开源生态的建设,可以促进工具的普及和应用,降低开发门槛,并加速体系结构创新的发展。以国产开源框架 DeepSeek 为例,其通过开放架构设计、模块化接口与社区驱动模式,吸引了全球学术界和工业界的开发者参与贡献。DeepSeek 的实践表明,开源能够打破技术壁垒,并依托真实场景反馈驱动工具链的实用化演进。然而,国内尚未形成适配国产体系结构的相关工具生态。首先,现有的国际主流开源工具(如 gem5、QEMU、Dynamo-RIO 和 Valgrind)在性能优化和功能覆盖上仍存在较大差距,难以满足高精度、大规模的模拟需求。其次,开源工具的扩展性和适配性较差,大多数工具的架构通常针对特定应用场景设计,难以直接适配国产体系结构的特殊性和复杂性,用户在定制和扩展时需要投入大量精力。最后,当前的开源生态存在碎片化问题,大多数分析工具的研究集中于单点优化,缺乏系统性和整合性,不同工具各自为政,缺乏统一的设计框架和协作机制,影响了开源工具的可

持续发展。

针对这些问题，需要在开源生态建设上采取多维度、系统性的措施，以逐步建立适配国产芯片发展和体系结构探索的相关开源工具生态。应结合产业界的需求有组织地加大对开源工具开发的技术投入和研发力度，引导并围绕几个主流工具开展相关研发工作，以提升工具的性能和功能水平，满足更广泛的需求，提高工具的竞争力和影响力并避免生态的分散。同时，可以鼓励国内开发人员积极参与开源体系结构设计社区，贡献代码、开发工具，推动开源体系结构设计的发展。通过与国际社区的合作和交流，学习先进的技术和经验，不断改进和完善国内开源工具的功能和性能，促进开源工具研究朝着系统性和整合性发展，避免碎片化现象，提高整体生态系统的效能。

6 结 论

文章结合处理器软硬件协同设计创新的需求，对体系结构模拟器的定义、用途、关键指标与挑战、当前主流技术和生态进行了调研和综述。虽然体系结构模拟器作为重要工具，是处理器设计过程中的一个重要环节，但目前在性能、准确性、扩展性和易用性等方面还存在不足。同时，由于国内应用端驱动的体系结构设计和分析仍处于早期阶段，相关体系结构模拟器的生态和人才培养也还有所欠缺。因此需要结合国内领域专用处理器发展的需求和体系结构模拟器面临的挑战，加强关键技术的突破以及生态和人才的建设。

参 考 文 献

[1] Leiserson C E, Thompson N C, Emer J S, et al. There's plenty of room at the top: What will drive computer performance after Moore's law? Science, 2020, 368(6495): eaam9744

[2] Rothmann M, Porrmann M. A survey of domain-specific architectures for reinforcement learning. IEEE Access, 2022, 10: 13753-13767

[3] Hennessy J L, Patterson D A. A new golden age for computer architecture. Communications of the ACM, 2019, 62(2): 48-60

[4] Wang H, Min Q, Li Y, et al. RPSim: A rapid prototyping full-system simulator for SoC software development//Proceedings of the 9th IEEE International Conference on Networking, Architecture, and Storage. Tianjin, China, 2014: 259-267

[5] Hennessy J L, Patterson D A. Computer architecture: A Quantitative Approach. 5th Edition. San Francisco, USA: Morgan Kaufmann, 2011

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

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

[8] Zhang W, Wang H, Lu Y, et al. A loosely-coupled full-system multicore simulation framework. IEEE Transactions on Parallel and Distributed Systems, 2015, 27(6): 1566-1578

[9] Spink T, Wagstaff H, Franke B. A retargetable system-level DBT hypervisor//Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference. Renton, USA, 2019: 505-520

[10] Sabu A, Patil H, Heirman W, et al. LoopPoint: Checkpoint-driven sampled simulation for multi-threaded applications//Proceedings of the IEEE International Symposium on High-Performance Computer Architecture. Seoul, Republic of Korea, 2022: 604-618

[11] Celio C, Patterson D A, Asanovic K. The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized risc-v processor. UCB/EECS-2015-167, 2015, 1(1):1-23

[12] Bai C, Sun Q, Zhai J, et al. BOOM-Explorer: RISC-V BOOM microarchitecture design space exploration framework//Proceedings of the IEEE/ACM International Conference On Computer Aided Design. Munich, Germany, 2021: 1-9

[13] Akram A, Sawalha L. A survey of computer architecture simulation techniques and tools. IEEE Access, 2019, 7: 78120-78145

[14] Fang Z, Min Q, Zhou K, et al. Transformer: A functional-driven cycle-accurate multicore simulator//Proceedings of the 49th Annual Design Automation Conference. San Francisco, USA, 2012: 106-114

[15] Patel A, Afram F, Chen S, et al. MARSS: A full system simulator for multicore x86 CPUs//Proceedings of the 48th Design Automation Conference. San Diego, USA, 2011: 1050-1055

[16] Bellard F. QEMU, a fast and portable dynamic translator//Proceedings of the FREENIX Track: 2005 USENIX Annual Technical Conference. Anaheim, USA, 2005, 41(46): 10-5555

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

[18] Lowe-Power J, Ahmad A M, Akram A, et al. The gem5 simulator: Version 20.0 +. arXiv preprint arXiv, 2020, 2007.03152

[19] Oh S, Xu M, Khan T A, et al. Udp: Utility-driven fetch directed instruction prefetching//Proceedings of the 51st ACM/IEEE Annual International Symposium on Computer Architecture. Buenos Aires, Argentina, 2024: 1188-1201

[20] Carlson T E, Heirman W, Eeckhout L. Sniper: Exploring the level of abstraction for scalable and accurate parallel

multi-core simulation//Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. Seattle, USA, 2011: 1-12

[21] Sanchez D, Kozyrakis C. ZSim: Fast and accurate microarchitectural simulation of thousand-core systems. ACM SIGARCH Computer Architecture News, 2013, 41(3): 475-486

[22] Khairy M, Shen Z, Aamodt T M, et al. Accel-Sim: An extensible simulation framework for validated GPU modeling//Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture. Valencia, Spain, 2020: 473-486

[23] Samajdar A, Zhu Y, Whatmough P, et al. Scale-sim: Systolic cnn accelerator simulator. arXiv preprint arXiv, 2018, 1811.02883

[24] Chin S A, Sakamoto N, Rui A, et al. CGRA-ME: A unified framework for CGRA modelling and exploration//Proceedings of the IEEE 28th International Conference on Application-specific Systems, Architectures and Processors. Seattle, USA, 2017: 184-189

[25] Tan C, Xie C, Li A, et al. OpenCGRA: An open-source unified framework for modeling, testing, and evaluating CGRAs//Proceedings of the IEEE 38th International Conference on Computer Design. Hartford, USA, 2020: 381-388

[26] Wang D, Ganesh B, Tuaycharoen N, et al. Dramsim: a memory system simulator. ACM SIGARCH Computer Architecture News, 2005, 33(4): 100-107

[27] Kim Y, Yang W, Mutlu O. Ramulator: A fast and extensible DRAM simulator. IEEE Computer architecture letters, 2015, 15(1): 45-49

[28] S. Xu, X. Chen, Y. Wang, Y. Han, X. Qian and X. Li. PIMSim: A Flexible and Detailed Processing-in-Memory Simulator. IEEE Computer Architecture Letters, 2019, 18(1): 6-9

[29] Mosanu S, Sakib M N, Tracy T, et al. PiMulator: A fast and flexible processing-in-memory emulation platform//Proceedings of the 2022 Design, Automation & Test in Europe Conference & Exhibition. Antwerp, Belgium, 2022: 1473-1478

[30] Tavakkol A, Gómez-Luna J, Sadrosadati M, et al. MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices//Proceedings of the 16th USENIX Conference on File and Storage Technologies. Oakland, USA, 2018: 49-66

[31] Jung M, Zhang J, Abulila A, et al. SimpleSSD: Modeling solid state drives for holistic system simulation. IEEE Computer Architecture Letters, 2017, 17(1): 37-41

[32] Jiang N, Becker D U, Michelogiannakis G, et al. A detailed and flexible cycle-accurate network-on-chip simulator//IEEE International Symposium on Performance Analysis of Systems and Software. Austin, USA, 2013: 86-96

[33] Agarwal N, Krishna T, Peh L S, et al. GARNET: A detailed on-chip network model inside a full-system simulator//Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software. Boston, USA, 2009: 33-42

[34] Li S, Ahn J H, Strong R D, et al. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures//Proceedings of the 42st Annual IEEE/ACM International Symposium on Microarchitecture. New York, USA, 2009: 469-480

[35] Kandiah V, Peverelle S, Khairy M, et al. AccelWattch: A power modeling framework for modern GPUs//Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture. Greece, 2021: 738-753

[36] Karandikar S, Mao H, Kim D, et al. FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud//Proceedings of the 45th ACM/IEEE Annual International Symposium on Computer Architecture. Los Angeles, USA, 2018: 29-42

[37] Snyder W. Verilator and systemperl//Proceedings of the 41th Design Automation Conference. San Diego, USA, 2004, 79: 122-148

[38] Lawton K P. Bochs: A portable pc emulator for unix/x. Linux Journal, 1996, 1996(29es): 7-es

[39] Gober N, Chacon G, Wang L, et al. The championship simulator: Architectural simulation for education and competition. arXiv preprint arXiv, 2022, 2210.14324

[40] Ahn J H, Li S, Seongil O, et al. McSimA+: A manycore simulator with application-level+ simulation and detailed microarchitecture modeling//Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software. Austin, USA, 2013: 74-85

[41] Fung W W L, Sham I, Yuan G, et al. Dynamic warp formation and scheduling for efficient GPU control flow//Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture. Chicago, USA, 2007: 407-420

[42] Sun Y, Baruah T, Mojumder S A, et al. MGPUSim: Enabling multi-GPU performance modeling and optimization//Proceedings of the 46th International Symposium on Computer Architecture. Phoenix, USA, 2019: 197-209

[43] Balasubramonian R, Kahng A B, Muralimanohar N, et al. CACTI 7: New tools for interconnect exploration in innovative off-chip memories. ACM Transactions on Architecture and Code Optimization, 2017, 14(2): 1-25

[44] Chatterjee N, Balasubramonian R, Shevgoor M, et al. Usimm: the utah simulated memory module. University of Utah, Tech. Rep, 2012: 1-24

[45] Hu Y, Jiang H, Feng D, et al. Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity//Proceedings of the International Conference on Supercomputing. Tucson, USA, 2011: 96-107

[46] Dong X, Xu C, Xie Y, et al. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012, 31(7): 994-1007

[47] Poremba M, Xie Y. Nvmain: An architectural-level main memory simulator for emerging non-volatile memories//

IEEE Computer Society Annual Symposium on VLSI. Amherst, USA, 2012: 392-397

[48] Catania V, Mineo A, Monteleone S, et al. Noxim: An open, extensible and cycle-accurate network on chip simulator//Proceedings of the IEEE 26th International Conference on Application-Specific Systems, Architectures and Processors. Toronto, Canada, 2015: 162-163

[49] Chan J, Parameswaran S. NoCGEN: A template based reuse methodology for networks on chip architecture//Proceedings of the 17th International Conference on VLSI Design. Mumbai, India, 2004: 717-720

[50] Brooks D, Tiwari V, Martonosi M. Wattch: A framework for architectural-level power analysis and optimizations. ACM SIGARCH Computer Architecture News, 2000, 28(2): 83-94

[51] Leng J, Hetherington T, ElTantawy A, et al. GPUWattch: Enabling energy optimizations in GPGPUs. ACM SIGARCH computer architecture news, 2013, 41(3): 487-498

[52] Hatnik U, Altmann S. Using ModelSim, Matlab/Simulink and NS for simulation of distributed systems//Proceedings of the 2004 International Conference on Parallel Computing in Electrical Engineering. Dresden, Germany, 2004: 114-119

[53] Tan Z, Waterman A, Avizienis R, et al. RAMP gold: an FPGA-based architecture simulator for multiprocessors//Proceedings of the 47th Design Automation Conference. Anaheim, California, USA, 2010: 463-468

[54] Lin D L, Ren H, Zhang Y, et al. From rtl to cuda: A gpu acceleration flow for rtl simulation with batch stimulus//Proceedings of the 51st International Conference on Parallel Processing. Bordeaux, France, 2022: 1-12

[55] Villa O, Lustig D, Yan Z, et al. Need for speed: Experiences building a trustworthy system-level gpu simulator//IEEE International Symposium on High-Performance Computer Architecture. Seoul, Republic of Korea, 2021: 868-880

[56] Yang J, Wen M, Chen D, et al. HyFiSS: A Hybrid Fidelity Stall-Aware Simulator for GPGPUs//Proceedings of the 57th IEEE/ACM International Symposium on Microarchitecture. Austin, USA, 2024: 168-185

[57] Kabylkas N, Thorn T, Srinath S, et al. Effective processor verification with logic fuzzer enhanced co-simulation//Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture. Greece, 2021: 667-678

[58] Reddi V J, Settle A, Connors D A, et al. Pin: a binary instrumentation tool for computer architecture research and education//Proceedings of the 2004 Workshop on Computer Architecture Education: Held in Conjunction with the 31st International Symposium on Computer Architecture. Munich, Germany, 2004: 22-es

[59] Jouppi N P, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit//Proceedings of the 44th Annual International Symposium on Computer Architecture. Toronto, Canada, 2017: 1-12

[60] Li Z, Ye Y, Neuendorffer S, et al. Compiler-driven simulation of reconfigurable hardware accelerators//Proceedings of the IEEE International Symposium on High-Performance Computer Architecture. Las Vegas, USA, 2022: 619-632

[61] Esmaili-Dokht P, Sgherzi F, Girelli V S, et al. A mess of memory system benchmarking, simulation and application profiling//Proceedings of the 57th IEEE/ACM International Symposium on Microarchitecture. Austin, USA, 2024: 136-152

[62] Jiang J, Liang C, Dong R, et al. A System-Level Dynamic Binary Translator using Automatically-Learned Translation Rules//Proceedings of the IEEE/ACM International Symposium on Code Generation and Optimization. Edinburgh, UK, 2024: 423-434

[63] Nair A A, John L K. Simulation points for SPEC CPU 2006//26th International Conference on Computer Design. Lake Tahoe, CA, USA, 2008: 397-403

[64] Wunderlich R E, Wenisch T F, Falsafi B, et al. SMARTS: Accelerating microarchitecture simulation via rigorous statistical sampling//Proceedings of the 30th Annual International Symposium on Computer Architecture. San Diego, USA, 2003: 84-97

[65] Hassani S, Southern G, Renau J. LiveSim: Going live with microarchitecture simulation//IEEE International Symposium on High Performance Computer Architecture. Barcelona, Spain, 2016: 606-617

[66] Shen X, Zhong Y, Ding C. Locality phase prediction. ACM SIGPLAN Notices, 2004, 39(11): 165-176

[67] Zhang W, Li J, Li Y, et al. Multilevel phase analysis. ACM Transactions on Embedded Computing Systems, 2015, 14(2): 1-29

[68] Li J, Zhang W, Chen H, et al. Multi-level phase analysis for sampling simulation//Proceedings of the Design, Automation & Test in Europe Conference & Exhibition. Grenoble, France, 2013: 649-654

[69] Chen C C, Peng Y C, Chen C F, et al. DAPs: Dynamic adjustment and partial sampling for multithreaded/multicore simulation//Proceedings of the 51st Annual Design Automation Conference. San Francisco, USA, 2014: 1-6

[70] Liu C, Sun Y, Carlson T E. Photon: A Fine-grained Sampled Simulation Methodology for GPU Workloads//Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture. Toronto, Canada, 2023: 1227-1241

[71] Wang W, McCamant S, Zhai A, et al. Enhancing cross-isa dbt through automatically learned translation rules. ACM SIGPLAN Notices, 2018, 53(2): 84-97

[72] Song C, Wang W, Yew P C, et al. Unleashing the Power of Learning: An Enhanced Learning-Based Approach for Dynamic Binary Translation //Proceedings of the 2019 USENIX Annual Technical Conference. Renton, USA, 2019: 77-90

[73] Jiang J, Dong R, Zhou Z, et al. More with less-deriving more translation rules with less training data for dbts using parameterization//Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture. Athens, Greece, 2020: 415-426

[74] Engelke A, Okwieka D, Schulz M. Efficient LLVM-based dynamic binary translation//Proceedings of the 17th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. Virtual, USA, 2021: 165-171

[75] Wang Z, Liu R, Chen Y, et al. COREMU: a scalable and portable parallel full-system emulator//Proceedings of the 16th ACM Symposium on Principles and Practice of Parallel Programming. San Antonio, USA, 2011: 213-222

[76] Hong D Y, Hsu C C, Yew P C, et al. HQEMU: a multi-threaded and retargetable dynamic binary translator on multicores//Proceedings of the 10th Annual IEEE/ACM International Symposium on Code Generation and Optimization. San Jose, USA, 2012: 104-113

[77] Gao D, Lin H, Li Z, et al. Trinity: High-performance mobile emulation through graphics projection//Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation. Carlsbad, USA, 2022: 285-301

[78] Yang Q, Li Z, Liu Y, et al. Mobile gaming on personal computers with direct android emulation//Proceedings of the 25th Annual International Conference on Mobile Computing and Networking. Los Cabos, Mexico, 2019: 1-15

[79] Miller J E, Kasture H, Kurian G, et al. Graphite: A distributed parallel simulator for multicores//Proceedings of the 16th International Conference on High-Performance Computer Architecture. Bangalore, India, 2010: 1-12

[80] Pellauer M, Adler M, Kinsy M, et al. HAsim: FPGA-based high-detail multicore simulation using time-division multiplexing//Proceedings of the 17th International Conference on High-Performance Computer Architecture. San Antonio, USA, 2011: 406-417

[81] Genbrugge D, Eyerman S, Eeckhout L. Interval simulation: Raising the level of abstraction in architectural simulation//Proceedings of the 16th International Conference on High-Performance Computer Architecture. Bangalore, India, 2010: 1-12

[82] Zhang W, Ji X, Lu Y, et al. Prophet: A parallel instruction-oriented many-core simulator. IEEE Transactions on Parallel and Distributed Systems, 2017, 28(10): 2939-2952

[83] Gouk D, Kwon M, Zhang J, et al. Amber: Enabling precise full-system simulation with detailed modeling of all SSD resources//Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture. Fukuoka, Japan, 2018: 469-481

[84] Mansi M, Swift M M. Øsim: Preparing system software for a world with terabyte-scale memories//Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems. Lausanne, Switzerland, 2020: 267-282

[85] Badr M, Delconte C, Edo I, et al. Mocktails: Capturing the memory behaviour of proprietary mobile architectures//Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture. Valencia, Spain, 2020: 460-472

[86] Tang T, Li S, Nai L, et al. Neurometer: An integrated power, area, and timing modeling framework for machine learning accelerators industry track paper//Proceedings of the IEEE International Symposium on High-Performance Computer Architecture. Seoul, Republic of Korea, 2021: 841-853

[87] Rogers S, Slycord J, Baharani M, et al. gem5-salam: A system architecture for llvm-based accelerator modeling//Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture. Athens, Greece, 2020: 471-482

[88] H. Zhang, A. Ning, R. B. Prabhakar and D. Wentzlaff. LLMCompass: Enabling Efficient Hardware Design for Large Language Model Inference//Proceedings of the ACM/IEEE 51st Annual International Symposium on Computer Architecture. Buenos Aires, Argentina, 2024: 1080-1096

[89] B. Hyun, T. Kim, D. Lee and M. Rhu. Pathfinding Future PIM Architectures by Demystifying a Commercial PIM Technology//Proceedings of the IEEE International Symposium on High-Performance Computer Architecture, Edinburgh, UK, 2024: 263-279

[90] T. Xie et al. UniNDP: A Unified Compilation and Simulation Tool for Near DRAM Processing Architectures//Proceedings of the IEEE International Symposium on High Performance Computer Architecture. Las Vegas, USA, 2025: 624-640

[91] Gubran A A, Aamodt T M. Emerald: Graphics modeling for SoC systems//Proceedings of the 46th International Symposium on Computer Architecture. Phoenix, USA, 2019: 169-182

[92] Lee J, Ha Y, Lee S, et al. GCoM: a detailed GPU core model for accurate analytical modeling of modern GPUs//Proceedings of the 49th Annual International Symposium on Computer Architecture. New York, USA, 2022: 424-436

[93] H. SeyyedAghaei, M. Naderan-Tahan and L. Eeckhout. GPU Scale-Model Simulation//Proceedings of the IEEE International Symposium on High-Performance Computer Architecture. Edinburgh, UK, 2024, 1125-1140

[94] N. Zhang et al. The dataflow abstract machine simulator framework//Proceedings of the ACM/IEEE 51st Annual International Symposium on Computer Architecture. Buenos Aires, Argentina, 2024, 532-547

[95] Ritter F, Hack S. Pmevo: portable inference of port mappings for out-of-order processors by evolutionary optimization//Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation. London, UK, 2020: 608-622

[96] Ansel J, Kamil S, Veeramachaneni K, et al. Opentuner: An extensible framework for program autotuning//Proceedings of the 23rd International Conference on Parallel Architectures and Compilation. Edmonton, Canada, 2014: 303-316

[97] Renda A, Chen Y, Mendis C, et al. Difftune: Optimizing cpu simulator parameters with learned differentiable surrogates//Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture. Athens, Greece, 2020: 442-455

[98] Huang Q, Hong C, Wawrzynek J, et al. Learning a continu-

ous and reconstructible latent space for hardware accelerator design//Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software. Singapore, 2022: 277-287

[99] Jiang M, Xu T, Zhou Y, et al. EXAMINER: Automatically locating inconsistent instructions between real devices and CPU emulators for ARM//Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. Lausanne, Switzerland, 2022: 846-858

[100] Argollo E, Falcón A, Faraboschi P, et al. COTSon: infrastructure for full system simulation. ACM SIGOPS Operating Systems Review, 2009, 43(1): 52-61

[101] Saed M, Chou Y H, Liu L, et al. Vulkan-Sim: A GPU architecture simulator for ray tracing//Proceedings of the 55th IEEE/ACM International Symposium on Microarchitecture. Chicago, USA, 2022: 263-281

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

[103] A. Mosallaei, K. E. Isaacs and Y. Sun. Looking into the Black Box: Monitoring Computer Architecture Simulations in Real-Time with AkitaRTM//Proceedings of the 57th IEEE/ACM International Symposium on Microarchitecture. Austin, USA, 2024, 780-794

GENG Zi-Duan, Ph. D. His main research interests include computer architecture, computer architecture simulation.

ZHANG Yu-Hang, master. His main research interests include computer architecture, computer architecture simulation.

WANG Xiao-Zheng, master. His main research interests include computer architecture, computer architecture simulation.

JIANG Jin-Hu, Ph. D., senior engineer. His main research interests include operating system and parallel storage.

ZHANG Wei-Hua, Ph. D., professor. His main research interests include compilers, computer architecture, parallelization, and systems software.

CHEN Zuo-Ning, Ph. D., professor. Ph. D. supervisor, Academician of Chinese Academy of Engineering. Her research in interests include storage system, operating system and information security.

Background

Computer architecture simulators are vital tools in the field of computer science, specifically within the subfield of processor architecture design and optimization. These software systems model the functional and timing behaviors of processors from an architectural perspective, allowing researchers to evaluate and refine designs before hardware implementation. This capability is essential for exploring design optimization spaces, analyzing processor performance across diverse workloads, and accelerating the development of innovative processor products. As such, simulators are a cornerstone of hardware-software co-design and architectural innovation, enabling efficient exploration of complex processor architectures and supporting the development of software for emerging hardware platforms.

Globally, research on computer architecture simulators has made significant progress over decades, resulting in widely adopted tools such as gem5, ZSim, Sniper, and SimpleScalar. These simulators offer unique strengths, balancing accuracy, simulation speed, and extensibility to meet various research and industrial needs. However, the increasing complexity of processor architectures—driven by trends such as heterogeneous computing, AI-driven workloads, and stringent energy efficiency requirements—has exposed limitations in existing simulators. Key challenges include slow simulation speeds for large-scale workloads, insufficient accuracy for modeling emerging hardware like AI accelerators, and limited adaptability to new architectural paradigms. International efforts have advanced through techniques like parallel simulation, sampling methods, and modular designs, which improve performance and flexibility. Despite these advancements, current solutions still fall short of fully addressing the demands of modern computing environments, particularly in supporting high-performance, application-specific hardware designs. This paper significantly advances the field by providing a comprehensive survey of recent developments in computer architecture simulator technology. It thoroughly examines the composition, key performance metrics, and persistent challenges of simulators, offering insights into cutting-edge techniques from both academia and industry. By identifying critical trends—such as enhanced simulation speed, improved accuracy for novel hardware, increased extensibility, and greater usability—the paper aims to guide future research directions and promote the broader adoption of simulators in hardware-software co-design toolchains. Its primary contribution lies in synthesizing global progress, analyzing the state-of-the-art, and proposing actionable strategies to address unresolved challenges, thereby fostering innovation in processor architecture design and optimization.

This work was supported by the National Key Research and Development Program of China (Grant No. 2022YFB4500404).