计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2026,35(2):123−131 [doi: 10.15888/j.cnki.csa.010106] [CSTR: 32024.14.csa.010106]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

面向晶圆级芯片架构的系统仿真方法①

侯帅康[1], 王偲柠[2], 邵阳雪[2], 丁　博[2], 刘文斌[1], 宋　克[1], 王　雨[2]

[1](信息工程大学 信息技术研究所, 郑州 450002)
[2](嵩山实验室, 郑州 450003)
通信作者: 丁　博, E-mail: dingbossl@163.com

摘　要: 晶圆级芯片凭借更高的集成密度、更优的互连特性和更低的功耗, 已成为“后摩尔时代”集成电路领域未来的关键技术方向. 然而, 传统仿真方法在应对晶圆级芯片仿真时, 存在仿真效率低、跨芯粒通信建模缺失以及异构计算资源处理能力不足等问题. 针对晶圆级芯片架构的仿真需求, 本文提出了一种基于算子与芯粒协同的晶圆级芯片架构并行离散仿真方法, 通过算子与芯粒的协同并行离散仿真有效提高了系统的仿真效率. 首先, 构建基础的标准化芯粒库和算子库, 为架构仿真提供基础支持. 然后, 基于算子库将复杂应用计算任务拆分为多个算子, 协同多个芯粒实现并行离散仿真, 并结合通信模型确保了系统仿真结果的准确性. 仿真结果表明, 相对于常规的基于 SST 和 Gem5 仿真方法, 所提出的系统仿真方法不仅支持异构芯粒间通信的仿真建模, 而且在平均精度损失小于 1.3% 的情况下, 实现了 4.8 倍以上平均速度提升, 显著提升了晶圆级芯片系统的仿真效率.

关键词: 晶上系统; 芯粒; 模拟器; 系统离散仿真; 晶圆级芯片; 任务部署

引用格式: 侯帅康,王偲柠,邵阳雪,丁博,刘文斌,宋克,王雨.面向晶圆级芯片架构的系统仿真方法.计算机系统应用,2026,35(2):123–131. http://www.c-s-a.org.cn/1003-3254/10106.html

Method for Simulating Wafer-level Chip Architecture

HOU Shuai-Kang[1], WANG Cai-Ning[2], SHAO Yang-Xue[2], DING Bo[2], LIU Wen-Bin[1], SONG Ke[1], WANG Yu[2]

[1](Institute of Information Technology, Information Engineering University, Zhengzhou 450002, China)
[2](Songshan Laboratory, Zhengzhou 450003, China)

Abstract: Wafer-level chips, with their enhanced integration density, superior interconnect characteristics, and lower power consumption, represent a pivotal future technology in the integrated circuit field during the post-Moore era. However, conventional simulation methods suffer from low efficiency, a lack of cross-chiplet communication modeling, and inadequate handling of heterogeneous computing resources when applied to wafer-level chips. To address the simulation requirements for wafer-level chip architectures, this study proposes a parallel discrete simulation method based on the coordination of operators and chiplets. By leveraging the coordinated parallel discrete simulation of operators and chiplets, the method effectively enhances the simulation efficiency of the system. First, a foundational standardized chiplet library and an operator library are constructed to support the architecture simulation. Subsequently, complex computation tasks are decomposed into multiple operators using the operator library, and parallel discrete simulation is realized through the collaboration of multiple cores. Communication models are incorporated to ensure the accuracy of the system simulation results. Experimental results demonstrate that compared to conventional simulation methods based on SST and Gem5, the proposed approach not only supports simulation modeling of communication between heterogeneous chiplets but also achieves an average speedup of over 4.8 times with an average accuracy loss of less than

① 基金项目: 嵩山实验室自立项目 (221100211100)
侯帅康、王偲柠为共同第一作者
收稿时间: 2025-06-27; 修改时间: 2025-09-05; 采用时间: 2025-10-29; csa 在线出版时间: 2025-12-29
CNKI 网络首发时间: 2025-12-31

1.3%, significantly improving the simulation efficiency for wafer-level chip systems.

Key words: system on wafer; chiplet; simulator; system discrete simulation; wafer-level chip; task deployment

近年来, 以 DeepSeek、BERT、GPT-4 等为代表的大规模人工智能模型迅速发展, 对芯片的“转算存”能力提出更高的要求. 然而, 传统芯片的单芯粒设计受限于制造良率与成本, 在能力提升方面存在显著瓶颈[1,2]. 在此背景下, 基于先进封装技术的多芯粒芯片技术逐渐成为延续摩尔定律、提升芯片性能、降低成本和实现异构集成的重要途径[3,4]. 随着封装技术的发展, 封装多芯粒芯片的硅基板面积已经扩展到晶圆级, 并逐渐成为学术界和产业界所寻求的解决方案, 已有的产品如特斯拉的 Dojo 芯片[5]、Cerebras 的 WSE 引擎[6]等均展示了晶圆级芯片在大规模计算任务中的潜力. 此外, 软件定义晶上系统技术[7]通过变结构的方式进一步提升了晶上系统的灵活性、可扩展性和高性能, 为微电子领域的可持续发展提供一条可行之路.

晶圆级芯片可集成多种同构或异构芯粒, 其拓扑结构灵活多变, 不同架构对应着不同的系统性能表现, 设计空间复杂度较传统芯片显著增加. 晶圆级芯片的设计与优化涉及芯粒设计、片上网络、通信协议、任务调度等多个方面, 是复杂的系统工程问题[8]. 目前已有工作多聚焦于晶圆级芯片的架构设计、大模型的任务部署以及网络拓扑优化等方面, 部分工作中虽涉及特定架构的仿真方法, 但针对通用晶圆级芯片架构仿真的研究仍较为缺乏[9]. 高效的模拟器可以帮助研究者快速评估芯片性能, 指导架构优化, 从而缩短设计周期、降低开发成本[10].

目前, 晶圆级芯片仿真多沿用片上系统 (system on chip, SoC) 仿真器[11]. 例如, Gem5 模拟器[12]广泛用于 CPU 仿真, 支持多种 CPU 模型、内存子系统、片上网络以及外设建模, 兼容 ARM、x86、RISC-V、MIPS 等主流指令集架构. 针对 GPU 仿真, GPGPU-Sim[13]是一款高精度仿真框架, 具备 SIMT 核心建模能力, 支持多级存储层次、细粒度线程调度、片上互连网络及 CUDA 指令集[14]解析, 能够仿真从 Kepler 到 Ampere 架构的关键特性. 此外, SST (structural simulation toolkit) 是一种用于高性能并行系统的仿真框架[15], 具备良好的灵活性和扩展性. Zhi 等人[16]提出了一种基于开源模拟器的多芯粒系统仿真方法, 支持消息传递与共享内存两种内存模型, 其通过文件系统实现模拟器进程间的通信与同步, 允许多个独立模拟器协同工作.

综上所述, 晶圆级系统架构的仿真已经受到了学术界与业界的广泛关注. 然而, Gem5 和 GPGPU-Sim 模拟器的时钟建模较为复杂, 且多线程支持不足, 在处理大规模计算任务时仿真时间长达数日, 且仅适用于单一 SoC 架构. SST 仿真框架目前对多计算类芯粒之间的通信仿真支持仍不完善. 上述仿真工具均难以应对晶圆级系统异构多芯粒架构复杂通信仿真. 文献[16]所提方法虽支持同构芯粒间通信, 但对异构芯粒架构的适应性不足, 且无法有效应对复杂应用场景下仿真时间过长的问题. 总体而言, 现有晶圆级芯片仿真方法普遍存在仿真速度慢、灵活性差及对异构多芯粒通信支持不足等缺陷.

随着大模型算力需求的持续增长与晶圆级芯片异构资源的日益丰富, 研究高效、灵活的晶圆级芯片系统仿真方法具有重要的理论价值与应用前景. 因此, 本文以优化晶圆级系统仿真为目标, 提出了一种基于算子与芯粒协同的晶圆级芯片架构并行离散仿真方法, 在支持异构芯粒间通信仿真的同时, 显著缩短仿真时间. 本文主要研究工作如下.

1) 建立了面向晶圆级芯片仿真的标准化芯粒库, 集成了 CPU、GPU、DDR4 和 DSP 等多种典型芯粒的架构参数, 为大规模异构晶圆级芯片的仿真提供了基础元件支持.

2) 构建了基于深度学习特征的算子库, 涵盖卷积运算、全连接等核心计算算子, 并采用了基于大语言模型的算子自动化处理机制, 通过计算图数据依赖分析和并行子图聚类, 实现了算子与计算资源的高效映射优化.

3) 提出了一种晶圆级芯片系统仿真方法 (simulation method for wafer-level chip, SIM-WLC), 基于算子与芯粒协同的并行离散仿真机制, 包括任务划分调度、离散仿真模拟、通信消耗评估模型、结果分析及拟合等步骤, 实现复杂应用在晶圆级芯片架构上的高效仿真. 实验表明, 与现有仿真方法相比, SIM-WLC 在平均精度损失小于 1.3% 的条件下, 仿真速度平均提升 4.8 倍

以上, 并具备对异构芯粒间通信的精确建模能力.

1 系统仿真模型框架

1.1 系统仿真框架

为应对晶圆级芯片在仿真中面临的异构通信建模与仿真效率挑战, 本文提出了 SIM-WLC 仿真方法. 其核心思想是将复杂的应用通过算子库分解为细粒度的独立算子计算任务, 并利用离散事件仿真原理, 将这些算子任务协同映射到由标准化芯粒库构建的硬件架构上并行执行, 从而实现对系统性能的高效、精准评估. 如图 1 所示, SIM-WLC 框架包含以下 5 个核心技术与模块, 构成了其完整的技术内涵.

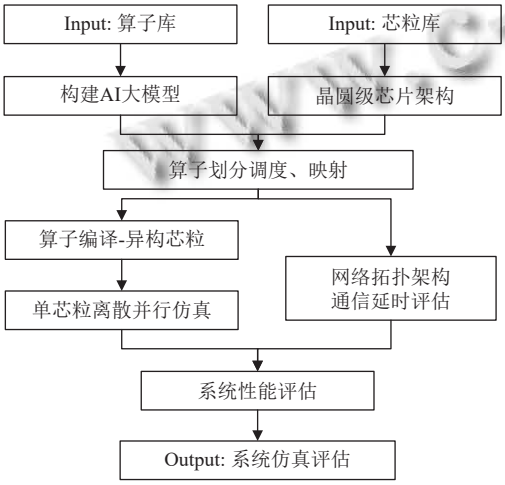


图 1 SIM-WLC 系统流程图

芯粒库构建: 构建了一个参数化的芯粒组件库C, 包含 CPU、GPU、CGRA、DSP 等多种典型芯粒模型. 每个芯粒$c_i \in C$均预定义了其参数规格、仿真器配置及功耗模型, 为异构晶圆级芯片的快速架构组合与性能评估提供基础.

算子库构建: 建立了一个标准化的算子库M, 涵盖卷积、全连接等核心计算操作. 每个算子$m_i \in M$被抽象为包含运算类型、计算复杂度、数据依赖关系等属性的模板, 支持基于模板的自动代码生成与优化, 是实现应用分解与硬件无关映射的关键.

任务图调度与映射: 将目标应用表征为一个有向无环任务图$G=(M,E)$, 其中顶点集合M代表算子, 边集合E代表算子间的数据流与通信依赖. 任务划分调度策略根据任务图G和硬件架构C, 将每个算子m_i分配到最优的芯粒c_i上执行, 并确定其调度序列.

并行离散事件仿真: 以算子任务的调度序列和通信事件为驱动, 通过离散事件推进仿真时间. 通过并行模拟各个芯粒上算子的执行过程与芯粒间的通信交互, 大幅提升仿真效率.

系统性能评估: 集成了通信时延评估与系统结果拟合模块. 通信模型对边集E中的通信事件进行参数化时延建模, 量化不同互连拓扑与协议的影响. 最终, 综合所有算子的执行时间线与通信时延, 拟合出整个系统的性能指标.

1.2 系统建模及符号定义

为了精确描述 SIM-WLC 的仿真过程, 本节给出系统的形式化模型与符号定义, 其关键要素定义如表 1 所示, 并用于后续的调度与仿真分析.

表 1 关键要素定义

变量名	定义
C	系统中所有芯粒集合, c_i代表一个具体芯粒 (如CPU、GPU)
M	应用中所有算子任务集合, m_i代表一个基础算子任务
E	算子间通信依赖的边集合
G	应用任务图, 定义为$G=(M,E)$
st_i	算子任务 m_i 的开始时间
et_i	算子任务 m_i 的结束时间

2 系统基础库构建

2.1 系统仿真框架

晶圆级芯片采用全晶圆集成技术, 实现多个同构/异构芯粒、同构/异构模组 (module) 的系统级封装. 如图 2 所示为典型异构芯粒、同构模组的晶圆级芯片系统模型, 该系统包含 9 个功能模组, 每个模组由 3 类异构芯粒 (chiplet 1、chiplet 2 和 chiplet 3) 构成. 为满足此类复杂系统的仿真需求, 本文构建了一个参数化的标准芯粒库, 通过多维参数对各类计算单元进行抽象建模, 以支持架构的快速组合与性能的精确评估.

芯粒模型的核心在于其参数体系, 涵盖类型标识、核心数量、运行频率、缓存容量、配套模拟器与编译器以及专用算子支持等关键属性. 这些参数共同定义了芯粒的计算能力、存储特性、功耗基线及任务映射可行性, 为系统级仿真提供可靠的量化输入. 基于该参数化方法, 所构建的芯粒库已集成包括 CPU、GPU、DSP、存储及路由单元在内的 34 种典型计算芯粒, 具备良好的模块化扩展能力, 可灵活适配不同工艺节点与功能配置的异构集成场景.

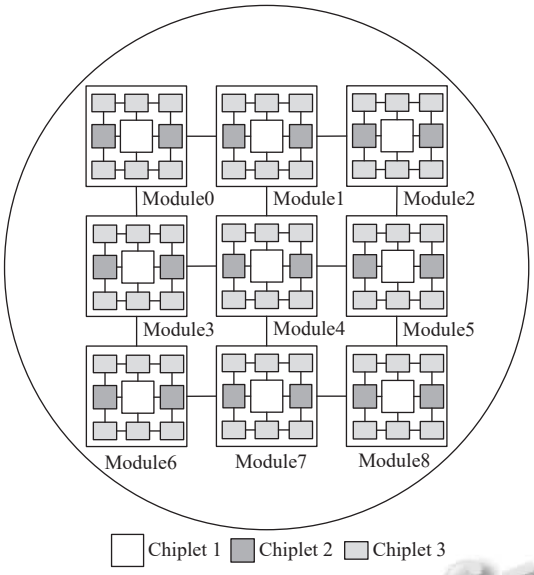


图 2 异构晶圆级芯片系统模型

表 2 与表 3 分别列出了部分 CPU 与 GPU 芯粒的详细配置信息. 通过调整各项参数, 可便捷地扩展芯粒类型与规模, 支持对 x86、RISC-V 和 ARMv7 等多种指令集架构的仿真和满足异构系统中图形处理与并行计算任务的建模需求, 增强了平台对不同计算范式的适应性.

表 2 CPU 芯粒基本信息

配置信息	C1	C2	C3
ISA	x86	RISC-V	ARMv7
Clock (GHz)	3.8	2.2	1.8
Cores	4	6	2
L1 Cache (KB)	256	128	64
L2 Cache (MB)	2	1	1
L3 Cache (MB)	6	—	4
Compiler	gcc	Riscv64-linux-gnu-gcc	aarch64-linux-gnu-gcc
Simulator	Gem5	SST	Gem5

表 3 GPU 芯粒基本信息

配置信息	GTX 480	RTX 2060	RTX 4090
Shaders	480	1920	16 384
SM	15	30	128
Clock (MHz)	700	1 365	2 235
L1 Cache (KB)	64	64	128
L2 Cache	786 KB	3 MB	72 MB
GDDR	1 536 MB	6 GB	24 GB
Compiler	NVCC	NVCC	NVCC
Simulator	GPGPU-Sim	GPGPU-Sim	GPGPU-Sim

为克服现有模拟器对全类型芯粒仿真支持不足的限制, 芯粒库中集成了对应的仿真模拟器信息. 这种设计有效解决了异构芯粒系统仿真中的工具兼容性问题, 确保各类计算单元均能获得准确的性能评估数据. 同时, 模块化架构为未来新型仿真工具的集成预留了扩展接口.

2.2 基础算子库构建

为支撑晶圆级芯片的系统仿真, 本文构建了结构化的基础算子库, 通过严格定义算子类型、数量配置、芯粒兼容性及源码实现方式等关键参数, 建立了高精度的计算仿真模型. 算子作为计算任务的基本单元, 其类型直接决定系统所支持的核心运算能力, 如矩阵乘法 Geem、卷积运算 Conv 等. 通过可配置的算子数量参数, 系统能够准确反映目标硬件的并行计算特性和资源利用率. 算子与芯粒的兼容性配置则确保运算任务能够匹配目标硬件的架构特性, 包括计算模式与访存带宽等, 从而有效保障仿真准确性. 此外, 算子实现方案的选择对仿真结果的可靠性至关重要, 这些参数共同决定了系统在计算吞吐量、缓存行为、功耗特性和时序特性等方面的仿真精度, 为系统级性能预测与优化提供可靠依据.

在仿真流程中, 算子是核心建模单元. 计算应用的执行流程可以分为 3 个关键步骤: 首先将应用算法分解为多个离散的算子任务, 随后将这些任务映射到目标芯粒的计算单元, 最终通过并行离散事件驱动机制实现高效仿真. 本文提出的 SIM-WLC 算子库采用了模块化架构, 集成包括 Gather、Add、Subtract、TypeCast、MatMul 和 Softmax 等在内的 22 种标准算子, 并支持通过扩展接口快速集成新型计算任务. 为验证算子库的实用性, 本文针对 GoogLeNet 与 GPT-2 两类典型模型构建完整任务模型, 涵盖算子基本信息、数据依赖与计算复杂度等核心参数, 具体配置如表 4 与表 5 所示, 为后续仿真实验奠定基础.

表 4 GoogLeNet 算子库

算子名称	算子类型	算子数量	支持芯粒类型	算子源码
Conv	卷积	57	CPU, GPU, CGRA	Python, C, CUDA
Concat	维度变换	9	CPU	Python, C
Flatten	维度变换	1	CPU	Python, C
Geem	矩阵乘	1	CPU, GPU	Python, C, CUDA
AvgPool	池化	1	CPU, GPU, CGRA	Python, C, CUDA
MaxPool	池化	13	CPU, GPU, CGRA	Python, C, CUDA
ReLU	激活函数	57	CPU, GPU	Python, C, CUDA

为进一步提升仿真构建效率, 系统集成算子源代码自动生成机制. 该机制基于预置的算子模板库, 通过解析算子类型与参数配置, 结合预定义代码生成规则, 动态输出适配不同芯粒架构的高效实现代码, 显著降低手动编写成本, 加速计算应用的仿真部署.

表 5 GPT-2 算子库

算子名称	算子类型	算子数量	支持芯粒类型	算子源码
Add	向量加	36	CPU, GPU, CGRA	Python, C, CUDA
Sub	向量减	1	CPU, GPU, CGRA	Python, C, CUDA
Cast	数据类型转换	3	CPU, GPU	Python, C, CUDA
Gather	数据收集	1	CPU, GPU	Python, C, CUDA
Where	条件选择	2	CPU, GPU	Python, C, CUDA
Expand	数据扩展	1	CPU	Python, C
Reshape	数据重塑	132	CPU	Python, C
Split	数据分割	12	CPU	Python, C
Transpose	数据转置	60	CPU	Python, C
Unqueeze	数据扩展	1	CPU	Python, C
MatMul	矩阵乘	72	CPU, GPU, CGRA	Python, C, CUDA
Norm	归一化	25	CPU, GPU	Python, C, CUDA
Softmax	归一化	12	CPU, GPU	Python, C, CUDA
FastGELU	激活函数	12	CPU, GPU	Python, C, CUDA

3 系统并行离散仿真与评估

3.1 多线程并行单芯粒加速仿真

SIM-WLC 主要面向晶圆级芯片的架构探索, 其核心目标为芯粒异构组合策略与跨芯粒互连拓扑等关键架构参数的性能评估. 基于该方法的功能定位, 仿真过程聚焦于计算任务的执行时间分析, 而无需验证计算结果的数值正确性. 在典型 AI 大模型计算中, 我们通过 Gem5 模拟器验证了在未实施稀疏计算优化的情况下, 数据依赖关系对总体计算时延的影响可以忽略, 这一结论为后续的并行化处理提供了理论基础.

为实现高效仿真, SIM-WLC 采用算子级并行仿真技术. 首先通过算子库将目标应用分解为多个独立计算单元, 随后采用符合张量维度约束的伪随机数生成器生成随机数据代替算子的真实输入, 从而有效消解算子间的数据依赖关系. 在任务划分与调度阶段, 各算子被映射至由芯粒库构建的晶圆级芯片架构的相应芯粒上, 形成多模拟器进程间的并行仿真执行模式, 具体流程如图 3 所示. 为确保仿真过程中的资源隔离性, 采用 UUID (universally unique identifier) 机制为每个仿真线程创建独立的存储空间, 所有线程特定的仿真输出数据均被隔离存储在相应路径目录中, 既保证了仿真效率, 又确保了结果的可信性.

通过将复杂应用划分为细粒度的独立算子单元, 该仿真方法充分发挥了宿主机多核处理器的并行计算能力. 实验结果表明, 在消除硬件瓶颈的理想条件下, 该方案相较于传统开源仿真器可实现数倍的性能加速, 为大规模芯片架构的快速评估提供了有效途径. 这一性能提升主要源于仿真框架对计算资源的充分利用以及对算子任务的高效并行调度, 显著提升了晶圆级芯片系统的仿真效率.

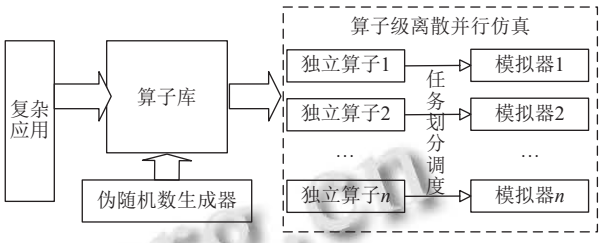


图 3 多线程并行单芯粒离散仿真流程

3.2 通信时延评估

在芯粒集成领域, 由于缺乏统一的互连标准, 不同协议的物理层规范导致其带宽和基础时延存在显著差异, 如表 6 所示, 给系统仿真带来挑战. 为实现高效的跨芯粒通信建模, 本文建立了精确的通信时延评估模型, 通过数学抽象替代传统仿真器中复杂的协议级建模, 从而大幅提升仿真效率.

表 6 通信协议带宽及时延

类型	Interface	Organization	Bandwidth (Gbps/lane)	Latency (ns)
Serial	IFOP[17]	AMD	6.4	<9
	Ultralink	Cadence	40	5.4
	USR/XSR[18]	Rambus/OIF	106.25	—
Parallel	OpenHBI3[19]	ODSA	6.4	<4
	AIB[20]	Intel	2	3.56
	LIPINCON	TSMC	8	—

不同协议物理层规范的差异化设计导致各互连协议在传输带宽阈值和基础时序参数方面存在本质区别, 因此已有仿真器常通过设置链路位宽与时延参数的配置实现协议建模. 基于现有片上网络仿真框架, 采用带宽与数据量的数学模型取代了仿真器中不同协议的仿真建模, 建立具有普遍适用性的通信时延评估模型如式 (1) 所示. 此外, 根据对计算应用任务分析, 构建算子的有向无环图[21], 量化提取跨芯粒通信的数据量, 进而计算通信时延.

$$T_C = \frac{Q_i}{BW_s} + l \tag{1}$$

其中, T_C 为通信时延, Q_i 为通信量, BW_s 为链路带宽, l 为链路传输时延.

两芯粒间的跳数可通过曼哈顿距离计算, 假定两

个芯粒分别为c_1和c_2, 其曼哈顿距离如式 (2) 所示:

$$Hops = |x_1 - x_2| + |y_1 - y_2| \tag{2}$$

其中, x_1和y_1为芯粒c_1的具体位置坐标, x_2和y_2为芯粒c_2的具体位置坐标. 再根据芯粒间的互连拓扑结构, 将曼哈顿距离与单跳数时延相乘, 即可估算该通信路径的时延长度, 如式 (3) 所示:

$$T_P = T_C \times Hops \tag{3}$$

在多任务并发场景下, 通信路径竞争将引入额外的拥塞时延. 若多条数据流共享同一段物理链路, 将产生串行排队等待现象, 即需要等待时延, 此时通信路径的总时延如式 (4) 所示:

$$T = T_P + T_W \tag{4}$$

其中, T_W为等待其他数据在阻塞路径传输完的时延.

如图 4 所示, 数据包 0 与数据包 1 分别沿路径 0→3→6→7→8 与 1→4→7→8 传输, 并在芯粒 7 处发生路径重叠. 假设数据包 0 率先到达, 芯粒 7 处形成传输队列[数据包 0, 数据包 1], 此时数据包 1 需等待前一数据包传输完成, 其总时延需计入排队等待时延. 此时, 数据包 0 的传输时延为$T_0 = t_0^0 + t_0^1 + t_0^2 + t_0^3$, 而数据包 1 的传输时延需要加上等待时延$t_0^3$, 即$T_1 = t_1^0 + t_1^1 + t_1^2 + t_0^3$. 该方法通过数学模型直接估算此类通信时延, 避免对每个网络数据包进行逐包仿真, 从而显著提升仿真效率.

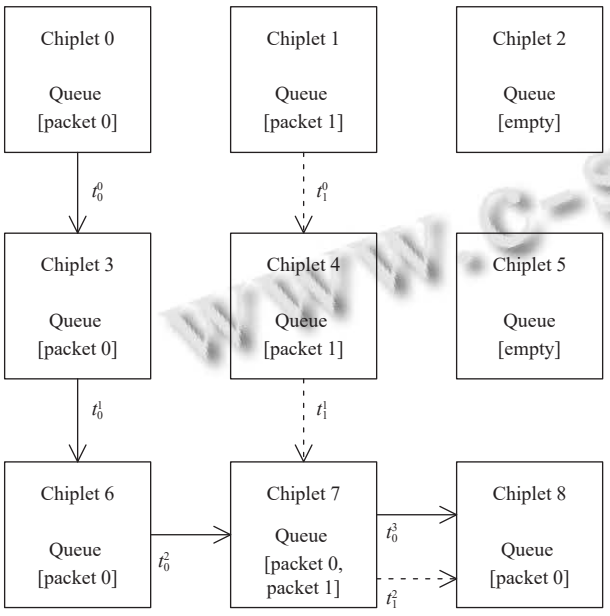


图 4 数据包传输通信路径

3.3 系统性能评估

在晶圆级芯片系统仿真中, 准确的性能评估建立在算子任务间通信时延的精确建模基础上. 通过引入各算子任务间的数据传输时延, 可获取准确的系统性能评估结果. 具体实现方法为: 首先依据数据依赖关系对算子任务进行拓扑排序, 然后基于排序结果计算每个算子任务的启动时间和完成时间, 从而确定各算子任务的数据传输时延.

算子任务在芯粒上运行存在两种依赖关系影响, 一种是两个算子之间存在数据前后向依赖关系, 当前算子需要等待前一个算子任务完成后才能开始执行. 假定边集合 $\boldsymbol{E}$ 中的边$e_a = (m_i, m_j)$, 则算子任务m_i的结束时间et_i与算子任务m_j的开始时间st_j之间的关系如式 (5) 所示:

$$st_j = et_i + T_P^a \tag{5}$$

其中, T_P^a为算子任务m_i与算子任务m_j之间的通信时延.

另一种是单个算子之前存在多个前驱算子, 但算子之间不存在数据依赖关系, 此时可以并行执行. 假如m_i与m_j不存在数据依赖关系, 但均与m_k有数据依赖关系, 即存在边$e_a = (m_i, m_k)$与$e_b = (m_j, m_k)$, 则m_k的开始时间与结束时间为$st_k = \max(et_i + T_P^a, et_j + T_P^b)$. 对于当前算子有多个前驱算子, 需要选择最大的结束时间作为当前算子的开始时间, 以保证数据依赖关系的正确性, 如式 (6) 所示:

$$st_k = \max(et_i + T_P^a), \ \text{if } e_a = (m_i, m_k) \in E \tag{6}$$

最终, 整体系统的运行总时间T_{total}即为最后一个算子任务的结束时间, 如式 (7) 所示. 通过系统性地整合任务依赖关系与通信时延特征, 本方法实现了对晶圆级芯片系统性能的准确评估, 为架构优化提供了可靠的量化依据.

$$T_{\text{total}} = \max(et_i), \ \text{for } m_i \in M \tag{7}$$

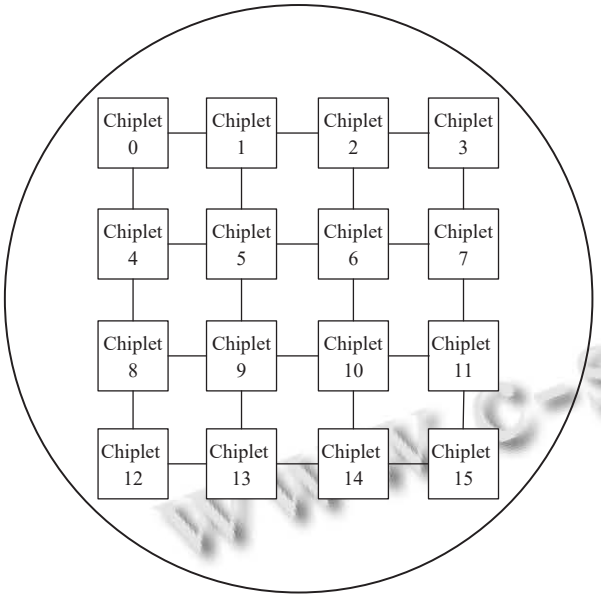
4 系统并行离散仿真及评估仿真结果与分析

4.1 系统仿真设置

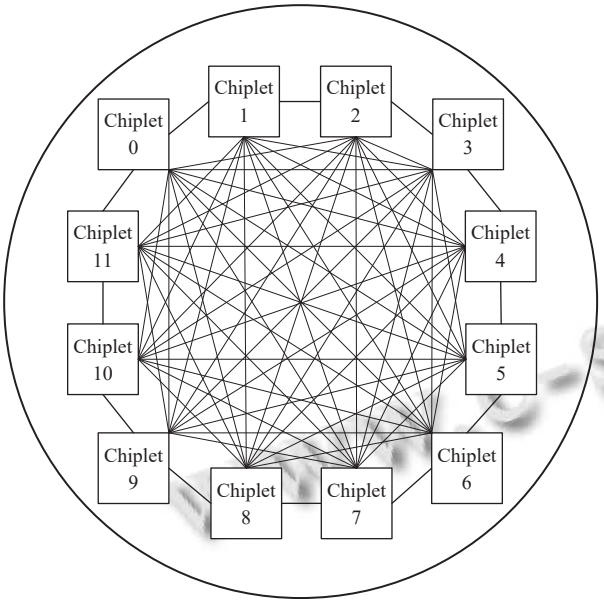
本文基于 Python 3.10 开发了 SIM-WLC 仿真框架, 实验环境为 Ubuntu 22.04 操作系统, 硬件环境为 2 X AMD EYPC 9654 CPU@2.2 GHz, 内存为 567 GB. 本文所提到的仿真实验均在以上环境下进行.

为全面评估仿真性能, 选取 2D Mesh 和 Crossbar 两种典型网络拓扑结构, 如图 5 所示, 并以 GoogLeNet 和 GPT-2 作为基准测试应用. GoogLeNet 是一种卷积神经网络, 主要用于图像识别, 在实验过程中抽象为

46 个算子集合的有向无环图. GPT-2 是一种大型语言模型, 主要用于自然语言处理, 在实验中抽象为 30 个算子集合的有向无环图. 通过与主流仿真器 SST 和 Gem5 进行对比实验, 验证所提方法的有效性与优越性.



(a) 晶圆级4×4 2D Mesh拓扑结构



(b) 晶圆级Crossbar拓扑结构

图 5 2D Mesh 拓扑与 Crossbar 拓扑结构晶圆级芯片示意图

4.2 仿真结果与分析

4.2.1 与 SST 对比

在 2D Mesh 拓扑结构下, 采用 GoogLeNet 应用进行对比实验, 抽象出 6 个计算子图, 结果如表 7 所示, 其中名称 15v1、15v2、16、21、25、46 为不同规模大小的 GoogLeNet 应用的算子数量.

由仿真结果可知, SIM-WLC 在保证平均仿真精度误差仅 0.02% 的同时, 实现了平均 7.2 倍的仿真速度提升. 值得注意的是, SST 仿真器在处理复杂计算任务时效率受限, 多个测试用例未能在合理时间内完成仿真, 这进一步凸显了 SIM-WLC 方法在大规模晶圆级芯片仿真中的实用价值. 该方法显著缩短了设计验证周期, 为快速架构探索提供了技术支撑.

表 7 SIM-WLC 与 SST 在 GoogLeNet 应用中的仿真结果对比

应用规模	仿真应用运行时间 (s)		仿真时间 (s)	
	SST	SIM-WLC	SST	SIM-WLC
15v1	35.36	35.36	612946	83818
15v2	30.01	30.01	527329	104127
16	NA	133.34	NA	450633
21	NA	142.65	NA	450633
25	56.1	56.06	974517	104127
46	NA	198.71	NA	450633
提升	0.02%		7.2 倍	

注: SST表示SST Vanadis RISC-V

4.2.2 与 Gem5 对比

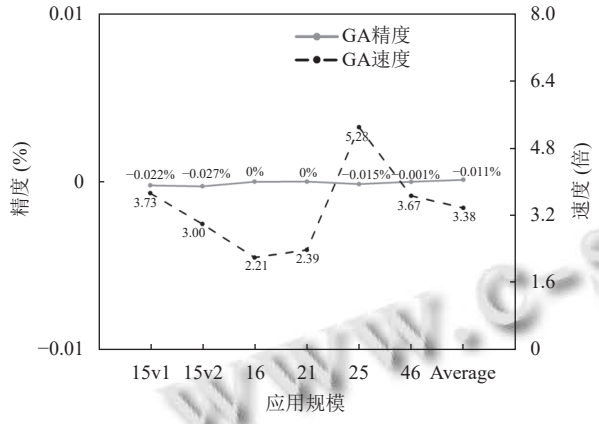
为进一步验证仿真性能, 我们构建了基于 RISC-V、ARM 和 x86 这 3 种指令集架构的晶圆级芯片架构模型, 分别采用了 2D Mesh 和 Crossbar 拓扑结构, 并在 GoogLeNet 和 GPT-2 这两个典型应用上进行测试.

在 GoogLeNet 测试中, 如图 6 所示, SIM-WLC 相较于 Gem5 不同架构实现均表现出显著优势. 其中, 与 Gem5 RISC-V 模拟器相比, 平均精度误差为 0.01%, 平均仿真速度提升 3.38 倍; 与 Gem5 ARM 模拟器相比, 平均精度误差为 6.24%, 平均仿真速度提升 4.17 倍; 与 Gem5 x86 模拟器相比, 平均精度误差为 2.15%, 平均仿真速度提升 3.47 倍.

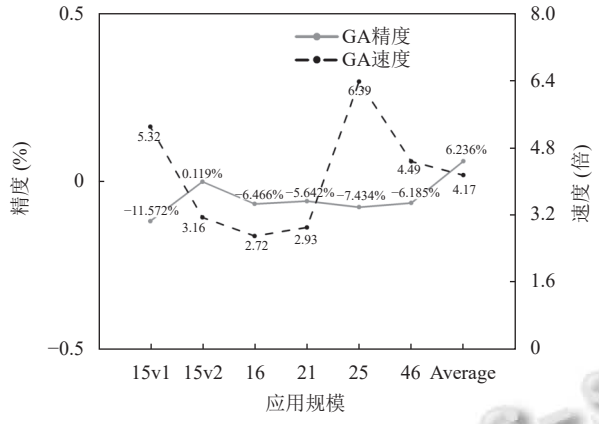
在 GPT-2 测试中, SIM-WLC 与 Gem5 中不同架构的对比结果如图 7 所示. 其中, 名称 10v1、10v2、10v3、15v1、15v2、30 为不同规模大小的 GPT-2 应用的算子数量. 仿真结果显示, 与 Gem5 RISC-V 模拟器相比, 平均精度误差为 0.36%, 平均仿真速度提升 5.08 倍; 与 Gem5 ARM 模拟器相比, 平均精度误差为 0.02%, 平均仿真速度提升 5.3 倍; 与 Gem5 x86 模拟器相比, 平均精度误差为 0.16%, 平均仿真速度提升 5.13 倍.

综上可知, SIM-WLC 仿真方法在晶圆级芯片系统仿真中具有显著优势. 相较于 SST 和 Gem5 等主流仿真器, 该方法在 GoogLeNet 和 GPT-2 等典型应用上实现了平均 4.8 倍以上的仿真速度提升, 同时将精度损失控制在 1.3% 以内. 该方法有效解决了传统仿真方法效

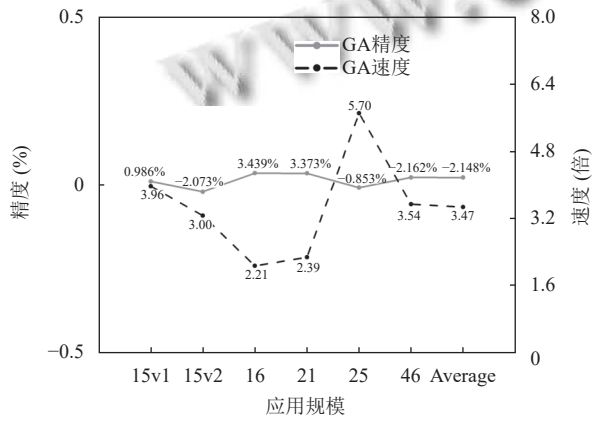
率低下的瓶颈, 大幅缩短了芯片设计周期. 同时, 该方法支持异构芯粒间通信的精确建模, 为不同指令集架构、网络拓扑和芯粒组合的性能评估提供了可靠手段, 显著降低了开发风险与成本. 为晶圆级芯片的架构探索和大规模人工智能计算系统的优化提供了关键支持, 对推动该技术从理论研究向产业化应用发展具有重要意义.



(a) 与Gem5 RISC-V构建亚晶圆级芯片对比

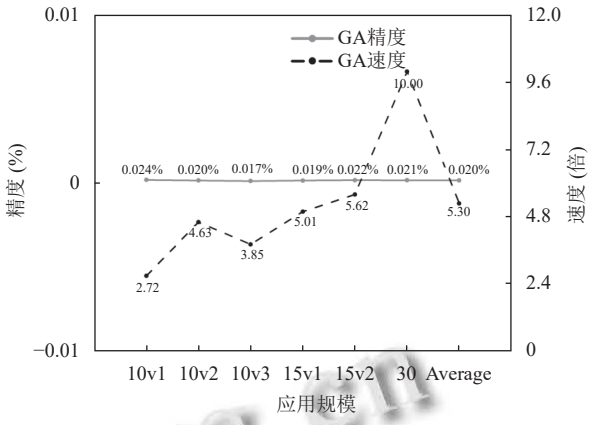


(b) 与Gem5 ARM构建亚晶圆级芯片对比

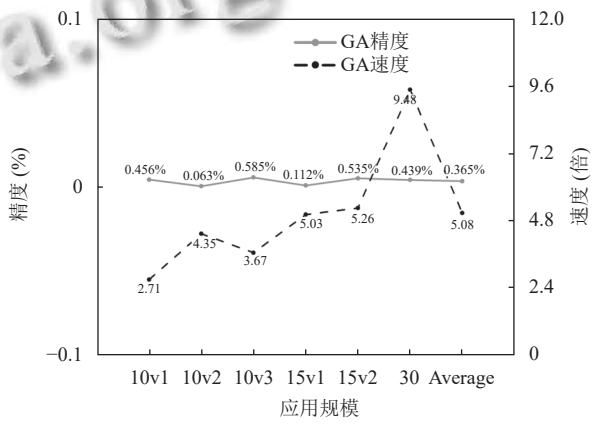


(c) 与Gem5 x86构建亚晶圆级芯片对比

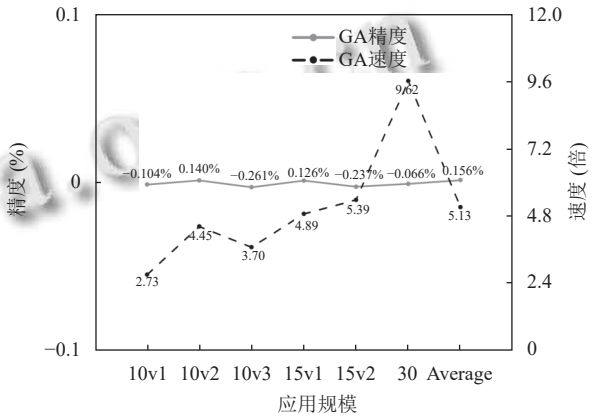
图 6 与 Gem5 仿真器在 GoogLeNet 应用中的对比



(a) 与Gem5 RISC-V构建亚晶圆级芯片对比



(b) 与Gem5 ARM构建亚晶圆级芯片对比



(c) 与Gem5 x86构建亚晶圆级芯片对比

图 7 与 Gem5 仿真器在 GPT-2 应用中的对比

5 结论与展望

本文针对现有晶圆级芯片架构仿真中存在仿真效率低和异构性支持不足等关键问题, 提出了一种基于算子与芯粒协同的并行离散仿真方法 SIM-WLC. 该方法通过构建标准化芯粒库与算子库, 实现了晶圆级芯片系统架构建模与面向复杂应用的系统级仿真. SIM-WLC

相较于其他仿真方案, 可支持异构系统芯粒间通信的仿真建模, 并在平均精度损失小于 1.3% 的情况下, 实现了 4.8 倍以上平均速度提升. 然而, 本文方法在仿真过程中对功耗和热效应等关键物理特性的建模尚不完善. 未来我们将继续探索基于机器学习的自适应任务调度与资源分配算法, 以适应动态变化的计算需求, 同时发展多物理场耦合仿真技术, 实现对晶圆级芯片系统更全面的综合仿真评估.

参考文献

1 Theis TN, Wong HSP. The end of Moore's law: A new beginning for information technology. Computing in Science & Engineering, 2017, 19(2): 41–50. [doi: 10.1109/MCSE.2017.29]

2 Feng YX, Ma KS. Chiplet actuary: A quantitative cost model and multi-chiplet architecture exploration. Proceedings of the 59th ACM/IEEE Design Automation Conference. San Francisco: ACM, 2022. 121–126. [doi: 10.1145/3489517.3530428]

3 Li T, Hou J, Yan JL, *et al*. Chiplet heterogeneous integration technology—Status and challenges. Electronics, 2020, 9(4): 670. [doi: 10.3390/electronics9040670]

4 Lau JH. Chiplet Design and Heterogeneous Integration Packaging. Singapore: Springer, 2023.

5 Talpes E, Sarma DD, Williams D, *et al*. The microarchitecture of DOJO, Tesla's exa-scale computer. IEEE Micro, 2023, 43(3): 31–39. [doi: 10.1109/MM.2023.3258906]

6 Lie S. Cerebras architecture deep dive: First look inside the HW/SW co-design for deep learning: Cerebras systems. Proceedings of the 2022 IEEE Hot Chips 34 Symposium (HCS). Cupertino: IEEE, 2022. 1–34. [doi: 10.1109/HCS55958.2022.9895479]

7 邬江兴, 刘勤让, 沈剑良, 等. 从 SoC 到 SDSoW: 微电子发展的新范式. 中国科学: 信息科学, 2024, 54(6): 1350–1368. [doi: 10.1360/SSI-2023-0219]

8 Hu Y, Lin XH, Wang HZ, *et al*. Wafer-scale computing: Advancements, challenges, and future perspectives [feature]. IEEE Circuits and Systems Magazine, 2024, 24(1): 52–81. [doi: 10.1109/MCAS.2024.3349669]

9 张聪武, 刘澳, 张科, 等. 面向通用处理器芯粒架构探索和评估的系统级模拟器. 电子与信息学报, 2024, 46(12): 4575–4588. [doi: 10.11999/JEIT240299]

10 Akram A, Sawalha L. A survey of computer architecture simulation techniques and tools. IEEE Access, 2019, 7: 78120–78145. [doi: 10.1109/ACCESS.2019.2917698]

11 陈伟健, 郭勇, 尹飞. 申威同时多线程功能模拟器实现与应用. 计算机工程, 2016, 42(6): 55–59, 67. [doi: 10.3969/j.issn.1000-3428.2016.06.010]

12 Binkert N, Beckmann B, Black G, *et al*. The Gem5 simulator. ACM SIGARCH Computer Architecture News, 2011, 39(2): 1–7. [doi: 10.1145/2024716.2024718]

13 Khairy M, Shen ZS, Aamodt TM, *et al*. Accel-Sim: An extensible simulation framework for validated GPU modeling. Proceedings of the 47th ACM/IEEE Annual International Symposium on Computer Architecture (ISCA). Valencia: IEEE, 2020. 473–486. [doi: 10.1109/ISCA45697.2020.00047]

14 Luebke D. CUDA: Scalable parallel programming for high-performance scientific computing. Proceedings of the 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro. Paris: IEEE, 2008. 836–838. [doi: 10.1109/ISBI.2008.4541126]

15 Rodrigues A, Hammond SD, Hemmert S, *et al*. A-SST initial specification. Albuquerque: Sandia National Laboratories, 2021. [doi: 10.2172/1854754]

16 Zhi HC, Xu XN, Han WJ, *et al*. A methodology for simulating multi-chiplet systems using open-source simulators. Proceedings of the 8th Annual ACM International Conference on Nanoscale Computing and Communication. New York: ACM, 2021. 18. [doi: 10.1145/3477206.3477459]

17 Naffziger S, Lepak K, Paraschou M, *et al*. 2.2 AMD chiplet architecture for high-performance server and desktop products. Proceedings of the 2020 IEEE International Solid-state Circuits Conference (ISSCC). San Francisco: IEEE, 2020. 44–45. [doi: 10.1109/ISSCC19947.2020.9063103]

18 Shivnaraine R, van Ierssel M, Farzan K, *et al*. 11.2 A 26.5625-to-106.25 Gb/s XSR SerDes with 1.55 pJ/b efficiency in 7 nm CMOS. Proceedings of the 2021 IEEE International Solid-State Circuits Conference (ISSCC). San Francisco: IEEE, 2021. 181–183. [doi: 10.1109/ISSCC42613.2021.9365975]

19 Lin MS, Huang TC, Tsai CC, *et al*. A 7-nm 4-GHz Arm[1]-core-based CoWoS[1] chiplet design for high-performance computing. IEEE Journal of Solid-state Circuits, 2020, 55(4): 956–966. [doi: 10.1109/JSSC.2019.2960207]

20 Greenhill D, Ho R, Lewis D, *et al*. 3.3 A 14nm 1GHz FPGA with 2.5D transceiver integration. Proceedings of the 2017 IEEE International Solid-state Circuits Conference (ISSCC). San Francisco: IEEE, 2017. 54–55. [doi: 10.1109/ISSCC.2017.7870257]

21 侯睿, 武继刚. 有向无环图的高效规约算法. 计算机科学, 2015, 42(7): 78–84.

(校对责编: 李慧鑫)