TeraPool：一种面向物理设计的、具有1024个 RISC-V 内核的共享L1内存扩展型集群设计，配备高带宽主存链路

Yichao Zhang、Marco Bertuletti、Chi Zhang、Samuel Riedel、Diyou Shen、Bowen Wang、Alessandro Vanelli-Coralli 和 Luca Benini

摘要—由精简指令处理器（处理单元 - PE）构成的共享L1内存集群，常被用作现代大规模并行计算架构（例如通用图形处理器）中的构建模块。扩展这些架构，增加集群数量，会带来计算和功耗开销；这是因为必须将大型数据结构拆分并合并为多个数据块，再通过高延迟的全局互连，跨存储层次结构移动这些数据块。扩展集群规模可减少缓冲、复制和同步开销。然而，全连接的核心到L1内存交叉开关的复杂度会随处理单元数量呈平方级增长，因而构成重大的物理实现挑战。本文提出 TeraPool，这是一种可实现物理设计的、具备 >1000 浮点运算能力的 RISC-V 处理单元扩展型集群设计，通过低延迟分层互连共享多兆字节的 >4000存储体 L1 存储器（延迟为1/7/9/11个周期，具体取决于目标频率）。TeraPool采用12纳米 FinFET 技术实现，在0.80 V/25 °C条件下，典型工作频率接近1 GHz（910 MHz）。能效高的处理单元到L1内存分层互连在存储体访问时仅消耗9-13.5 pJ，相当于一次 FP32 浮点乘加运算成本的0.74-1.1 × 。我们设计了一条高带宽主存链路，用于管理共享L1中的数据传入和传出，支持以 HBM2E 主存的全带宽进行传输。在910 MHz下，该集群在基准测试内核上可实现最高1.89 TFLOP/s单精度峰值性能，以及最高200 GFLOP/s/W能效（平均 IPC/PE 为0.8），证明了将共享L1集群扩展至一千个处理单元的可行性，是文献中已报道最大集群处理单元数量的四倍。

索引词—众核，RISC-V，可扩展性，物理设计感知

arXiv:2603.01629v1 [cs.DC] 2 Mar 2026

1 引言

当今生成式人工智能（AI）、数字孪生、6G无线网络、自动驾驶汽车和机器人的计算需求每 6 至 9 个月就会翻一番 [1]。这种指数级增长使计算需求攀升至千万亿次浮点运算，并需要大量存储容量来处理海量高速数据流，以及拥有数万亿参数的机器学习（ML）模型 [2], [3]。这些以人工智能为中心的数据密集型工作负载要实现快速适应、纵向扩展和持续演进，离不开高度可编程的高性能硬件，以及精简的编程模型和工具链。在严格的功耗限制下，性能需求不断增长，同时还需保持高度的编程灵活性和易用性，这推动计算架构走向灵活、可扩展且能效高的解决方案 [4], [5]。

在这一充满挑战的背景下，采用大量精简型可编程处理单元（PE）、以中等频率（1-2GHz）运行的众核架构，其能效远高于基于少数大型处理器、以极高频率（5+GHz）运行的架构 [6], [7]。现代众核设计中最常见的架构构建模块是处理单元簇通过低延迟、高带宽互连与共享L1存储器紧密耦合。然而，集群的规模通常仅限于几十到一百多个处理单元（PE）[8], [9]，[10]。超过这一数量后，通常通过扩展规模（见图1b），增加许多松散耦合的集群来提升性能；这些集群通过高延迟全局互连访问主存和其他集群的一级缓存。不过，松散耦合的集群会带来软硬件开销，包括同步、簇间通信、数据分配与拆分以及工作负载分配 [11]。为了控制这些性能损失并提高能效，直接扩展共享L1集群（见图1a）十分理想 [12]。例如，英伟达的图形处理器（GPU）架构遵循共享L1架构模型，其集群称为流式多处理器（SM），其中多线程张量核心共享L1数据存储器。从*A100* [13] 到*H100* [14]，，单个流式多处理器（SM）的峰值浮点（FP）计算能力通过将浮点处理单元（FP-PE）数量翻倍并提高处理速率而提升了 4× 。此外，共享内存容量也从192 kB增至256 kB。

从架构和物理设计的角度来看，处理单元至L1存储器互连设计已成为集群扩展的主要挑战：（**i**）处理单元与存储体之间的基准交叉开关互连网络，其复杂度会随着集群中处理单元和存储体数量的增加而呈二次方增长 [15]，，从而成为物理实现的瓶颈。（**ii**）要在大规模、统一地址且物理

• *Yichao Zhang*、*Marco Bertuletti*、*Chi Zhang*、*Samuel Riedel*、*Diyou Shen* 和 *Bowen Wang* 均任职于瑞士苏黎世 *8092* 的苏黎世联邦理工学院集成系统实验室（*IIS*）。电子邮箱：{*yichzhang, mbertuletti, chizhang, sriedel, dishen, bowwang*}*@iis.ee.ethz.ch*

• *Luca Benini* 和 *Alessandro Vanelli-Coralli* 任职于苏黎世联邦理工学院集成系统实验室（*IIS*），同时也任职于意大利博洛尼亚 *40126* 的博洛尼亚大学电气、电子与信息工程系（*DEI*）。电子邮箱：{*lbenini, avanelli*}*@iis.ee.ethz.ch*。

©本文已提交给 IEEE，以供考虑发表。版权可能在不另行通知的情况下转让，届时此版本可能不再可访问。

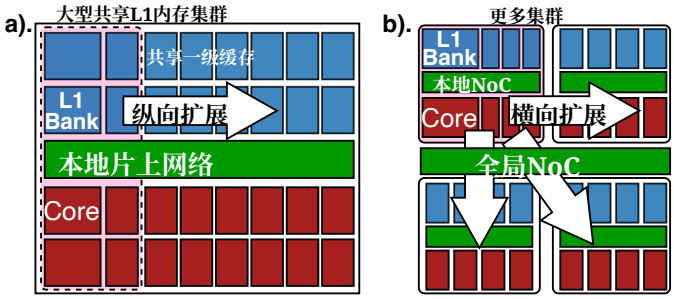


图1. 集群设计中纵向扩展与横向扩展的示意图。

分布式存储体中保持低延迟的存储器访问，对于尽量减少处理单元停顿并确保较高的计算利用率至关重要。（iii）此外，这项要求还需要在集群内的L1存储器与其余存储层次结构之间建立连接，进一步加剧物理布线资源的压力。因此，在扩展规模的同时维持集群的功耗、性能与面积（PPA），需要仔细权衡物理可行性与互连性能。迄今为止，这些互连设计挑战将集群扩展限制在数十个或至多数百个处理单元的规模[16]。

我们提出了 TeraPool，这是一种在物理上可行、能效高、通用且易于编程的众核集群设计。我们突破了当前纵向扩展的瓶颈，实现了包含 1000 个处理单元的共享一级缓存集群，同时维持接近 GHz 的频率并提供 TFLOP/s 级性能。在早期工作 [17], 的基础上，我们提出以下贡献：

- 一种基于分层交叉开关互连的大规模共享一级缓存集群设计，其规模是文献中已报道最大设计的 4 倍 [16], , 包含 1024 个处理单元（RV32IMAF 核心），共享 4MiB（4096 个存储体）的 L1 暂存存储器（SPM）；
- 针对处理单元与存储体之间的分层交叉开关互连网络建立了平均存储器访问时间（平均内存访问时间）模型。该模型的洞见指导了三级层级处理单元-一级缓存互连网络（*Tile*、子组和组）的设计与优化，实现了非统一内存访问延迟：组内为 1–5 个周期，远程组访问最高为 7–11 个周期，具体取决于目标工作频率；
- 一种高带宽存储器链路（HBML），支持在集群的一级缓存存储体之间传入和传出数据，且开销可忽略不计。通过与一种开源、周期精确的 *DRAMsys5.0*存储器模型 [18]协同仿真，并采用混合地址映射方案，我们的 HBML 实现了高达 97% 的带宽利用率，达到业界标准高带宽存储器（HBM2E）的水平；
- 对各类关键数据并行内核的性能评估表明，TeraPool 能够处理大型数据块，同时将数据传输开销降至最低。
- 采用 GlobalFoundries 的 12纳米 FinFET 技术，详细介绍物理设计方法、布局规划及功耗、性能和面积分析。

本文其余部分安排如下：第2节详细介绍扩展共享一级缓存集群的动机。第 3 节介绍分层 PE 到 L1 互连的平均内存访问时间建模，并通过延迟、吞吐量和物理布线复杂度评估设计选择。第4节介绍核心复合体设计和分层集群架构。接下来，第5节讨论 HBML 设计、*DRAMsys5.0* 协同仿真以及混合内存地址映射方案。第6节详细介绍物理实现和功耗、性能和面积分析。第7节介绍 TeraPool 精简的编程模型和软件性能。最后，第8节和第10节分别讨论相关工作和结论。我们的设计（包括对开源 *DRAMsys5.0* 模型的支持和完整的软件基础设施）采用宽松许可证开源发布 [1]。

2 共享内存集群纵向扩展与横向扩展

在本文中，集群 指处理单元通过低延迟互连共享 L1内存的计算块。其主要设计参数汇总于表1。本节讨论扩展策略，重点说明以下方式的优势与权衡：（i）在单个集群中进行纵向扩展，以及（ii）扩展至多个集群。

表1 集群设计参数说明

符号	说明
L	主存延迟（周期）
W	L1 存储器中的问题分块大小（字）
BW	集群与主存之间的带宽（字/周期）
AI	算术强度（操作/字）
N_{PEs}	集群中的处理单元数量
U	处理单元的计算利用率
S	集群中处理单元和存储体的扩展因子

2.1 单集群扩展

由于问题规模通常超出集群的 L1 存储器容量，因此通常采用分块：将大型数据结构拆分成多个数据块并移入 L1。纵向扩展集群可增加处理单元数量和共享一级缓存容量，从而支持更大规模的问题分块；更大的芯片面积则可容纳更多输入/输出端口，进而提升 L1 与主存之间的 BW。我们分析两种工作负载情形，以展示纵向扩展的优势：

○ 1 对于**没有**数据复用的工作负载，$AI \leq 1$ 通常会导致性能受主存限制。小型集群可能需要额外的集群间数据传输。此外，较大的数据块可改善处理单元之间的工作负载均衡，减少同步开销，并增强系统应对主存延迟和带宽抖动的能力。

○ 2 对于**有**数据复用的工作负载，较大的 L1 容量可减少集群对主存 BW 的需求。以矩阵乘法（MatMul）为例，每个矩阵块的维度为 m、 $W = 3m^2$ 和 $AI = m^3/3m^2 = \sqrt{W}/3\sqrt{3}$。当集群按 S 的因子扩展时， W 呈线性扩展，从而使 AI 增加，体现出更高的数据复用率，具体如下：

$$W' = S \times W, AI' = \frac{\sqrt{SW}}{3\sqrt{3}} = \sqrt{S} \times AI \quad (1)$$

根据 *Kung* 原理 [19], , 当公式 (2) 成立时，集群不会受到主存带宽和延迟的瓶颈限制：考虑集群扩展

1. https://github.com/pulp-platform/mempool

倍数为 S，由于设计元素相同，我们假设 L 和 U 保持不变。左侧表示数据传输成本，其中 W 和 BW 都随 S 扩展，其比值保持不变。右侧反映计算需求；由于 N_{PEs} 和 W 都随 S扩展，它们的比值保持不变，而 AI 随 $\sqrt{S}$增加，从而改善计算与存储平衡。

$$L + \frac{W}{BW} < \frac{\sqrt{S}AI \times W}{N_{PEs} \times U} \tag{2}$$

这表明，随着扩展因子 S 增大，不等式对更大的 L 和更小的 BW 仍然成立，说明大规模共享L1存储器集群更能在容忍主存传输延迟较高、带宽有限的情况下保持较高的计算利用率。

2.2 横向扩展至多个集群

作为纵向扩展策略的替代方案，可以实例化多个较小的集群，每个集群都通过全局互连访问主存和其他集群，从而形成一种松耦合的扩展架构（图1b）。与纵向扩展集群中等效的计算和存储资源相比，扩展至多集群/众多集群系统会带来计算和功耗开销，导致硬件组件利用不足、能效降低 [20]。我们总结了三种随扩展因子增大而增加的常见开销：

- **同步开销：**由于指令流彼此独立、L1存储器重填顺序各异以及主存访问冲突，难以使各集群的处理时间完全平衡。集群间利用率不均会导致同步开销，并且由于长尾效应 [21]，集群数量越多，同步开销越大。

- **分块开销：** 在松散耦合的集群之间进行问题分块，会在数据结构拆分以及跨存储层次结构合并部分结果时引入额外的计算开销。例如，深度神经网络（DNN）并行化中采用的 模型并行 [22]需要对部分结果进行归约求和，其执行时间会随着集群数量的增加而增加。

- **数据传输开销：**问题分块通常会导致数据重复，因为共享数据必须复制到每个集群。此外，每个集群产生的部分结果必须通过传输回主存后再进行合并，然后通过高延迟的全局互连重新分发，以便后续处理。这些因素会降低集群的存储器利用率，并引入随集群数量增加而增长的数据传输开销 [17]。

此外，将多个集群连接到主存需要使用高延迟的全局片上网络（NoC），从而产生额外的硬件开销。大量芯片面积被用于集群间片上网络路由，进而降低了面积利用率和能效 [23]。

这些考量促使我们探索高效的集群扩展方案，以期在物理上可行的互连设计下实现较高的计算利用率。

3 互连设计

带宽和延迟是处理单元至L1存储器互连设计中的关键性能指标。保持低延迟可确保处理单元通过发出少量非阻塞的L1内存访问来隐藏延迟，从而避免加载存储单元（LSU）停顿并保持较高的利用率。尽管具有高直径和低基数交换机的片上网络拓扑能够实现较高的时钟频率，但并不适合处理单元至L1内存的互连，因为每经过一跳，延迟都会增加几个周期。这会导致处理单元吞吐量下降，因为要隐藏片上网络的延迟，就必须支持过多尚未完成的存储器事务 [24]。理想情况下，采用全组合、全连接（FC）的对数级交叉开关 [2] [15]，可在处理单元与SPM存储体之间实现单周期数据传输，从而达到最低延迟。握手协议包含正向请求（包括地址、写入数据和控制信号）以及反向响应（包括请求ID、读取数据和确认信号）。同一仲裁交换机上的并发请求采用轮询策略处理。

遗憾的是，已有研究表明 [25]，尽管交叉开关的延迟最优，但其呈二次增长的布线复杂度给实现高频率带来了无法克服的障碍。最终，过大的交叉开关在物理设计中将变得无法布线。因此，我们需要在延迟、工作频率和物理可布线性之间寻求折中。我们采用层次化设计。该层次结构基于称为 *Tile* 的基本构建模块（见图 2），其中全连接交叉开关连接一组物理位置相邻的处理单元与 L1 SPM 存储体。在更高层级中， *Tile* 会组成组，处理单元通过远程访问端口访问其他 *Tile* 的 L1 存储体。这些端口连接到

下一层级中的 Tile 间交叉开关构成一个完全共享内存的集群。在层级边界处引入流水线寄存器，以缩短较长的物理路径并提高工作频率，但代价是延迟增加。这种模式形成了一种 NUMA 架构，其中请求路径和响应路径都采用了完整仲裁。

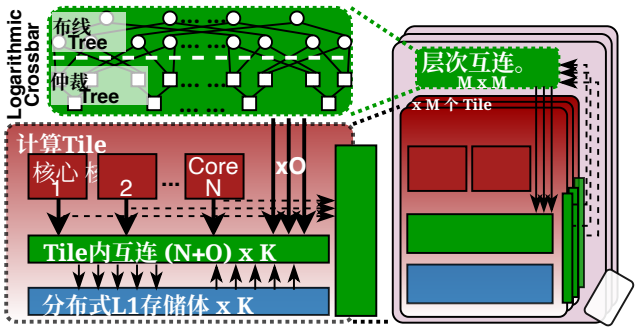


图 2. 采用对数级分级交叉开关互连网络的 *Tile* （构建模块）设计。

本节介绍分层 PE 到 L1 互连的平均内存访问时间模型。此外，我们利用该模型提供的洞见，探讨各种分层设计方案的物理设计可行性，并展示了构建超大规模扩展集群的最优配置：该集群包含 1024 个核心，共享 4096 个 L1-SPM 存储体。

2. 对数分级树形拓扑；通过解复用器/多路复用器实现布线/仲裁，并采用组合逻辑控制，可实现细粒度地址交错和单周期延迟。

3.1 分层互连平均内存访问时间模型

平均内存访问时间建模分析从接收内存请求的*Tile*视角进行，相关参数汇总于表2。内存请求来自本地***Tile***（由同一 Tile 内的处理单元发送的请求）或远程***Tile***（由其他 Tile 中的处理单元发送的请求）。假设随机地址请求均匀分布，请求由其本地Tile内处理单元发送的概率记为 P_{Ltile}，其值为 $1/N_{Tiles}$。集群的平均内存访问时间由交叉开关争用和流水线寄存器共同影响，计算方式为对各来源的延迟进行概率加权求和，如下式所示：

$$\begin{aligned} T_{cluster} &= P_{Ltile} \cdot L_{Ltile} + P_{Rtile} \cdot L_{Rtile} \\ P_{Rtile} &= 1 - P_{Ltile} = 1 - \frac{1}{N_{Tiles}} \\ L_{L/Rtile} &= E_{L:\ n\times k} + L_{pipeline} \end{aligned} \tag{3}$$

当多个请求经过同一仲裁树节点并指向同一分层端口时，对数级交叉开关中便会产生争用，从而因仲裁而为每个请求额外引入一个周期的延迟。该争用延迟 $E_{L:\ n\times k}$ 可根据多个请求在单个周期内经过同一仲裁器节点的概率进行建模。

表2 互连建模参数说明

符号	说明
N_{Tiles}	集群中构建模块（***Tile***）的数量
$T_{cluster}$	当所有处理单元执行随机时集群的平均内存访问时间
	地址访问
$P_{Ltile/Rtile}$	内存请求源自以下位置的概率
	本地/远程 Tile
$L_{Ltile/Rtile}$	从以下位置发出的请求的内存访问延迟
	本地/远程 Tile
$L_{pipeline}$	流水线寄存器引入的延迟
n, k	交叉开关输入端和输出端的数量
p	交叉开关输入端的请求注入率
$P_{req}(x)$	将 x 请求发送到仲裁器的概率
	单个周期
$E_{L:\ n\times k}$	通过以下结构进行随机目标访问的期望延迟
	一个 $n \times k$ 交叉开关

我们的分析从简单的仲裁器开始，并扩展到多级分层交叉开关设计。首先，基本的*N-to-1* 仲裁器模型构成竞争分析的基础。该仲裁器模型基于 Binomial(*n*, *p*) 分布，其中每个输入端都被视为一次独立随机试验， n 表示输入端数量， p 表示请求概率。二项分布的概率质量函数（PMF） $P_{req}(x)$表示在一个周期内向仲裁器发送 x 个请求的概率。当 x 个发起端发送请求时，延迟为 $x-1$。没有请求时（$x=0$）的延迟为零，因为仲裁器中没有访问请求。为计算期望延迟，我们对每个周期内出现 $x = 1 \to n$ 个请求的概率求和：

$$E_{L:\ n\times 1} = \sum_{x=1}^{n} L(x)P_{req}(x) = \sum_{x=1}^{n}(x-1)P_{req}(x) \tag{4}$$

其次，我们将此分析扩展至*n-to-k* 仲裁器模型。当收到一个完全随机的地址请求时，该请求被转发至指向被监测Tile的监测点输出端的概率为 $\frac{1}{k}$。到达同一输出端的请求数量仍服从参数为 $(n, p/k)$ 的二项分布，注入率为 p；其余输出端口则构成一个*n-to-(k-1)* 仲裁器。因此，必须使用递归方法计算期望延迟。如果没有请求到达监测点输出端（$x=0$），则该请求会被重定向至其他输出端，从而导致平均延迟增加，延迟为 $E_{L:\ n\times(k-1)}$。平均竞争开销延迟的概率质量函数（PMF）可表示为：

$$E_{L:\ n\times k} = \begin{cases} E_{L:\ n\times 1} + P_{req}(0)E_{L:\ n\times(k-1)}, & k > 1 \\ \sum_{x=1}^{n}(x-1)P_{req}(x), & k = 1 \end{cases} \tag{5}$$

最后，为了分析多级分层互连交叉开关，需要确定请求到达每一级输入端的概率，该概率取决于请求从上一级输出端转发过来的概率：

$$p^{stage(N)} = P_{req}^{stage(N-1)} = 1 - P_{req}^{stage(N-1)}(0) \tag{6}$$

为提高在单个周期内多项争用仍未解决的条件下平均内存访问时间计算的准确性，待处理请求会动态调整交叉开关的注入率。我们在每个层级交叉开关级[3] 引入输入队列，以模拟请求排队，从而实现注入率的动态调整。

3.2 TeraPool互连分析

在本小节中，我们分析了多种层级多级交叉开关互连设计，以确定最优的TeraPool架构；该架构规模扩展至1024个处理单元，并配备4096个共享一级缓存组。我们基于以下指标评估互连性能：

- **零负载延迟：**在无仲裁争用或输入排队的情况下，单个随机地址内存请求的平均延迟周期。其计算方法是将各层级访问延迟按其访问概率加权求和，从而反映互连的理想延迟。

- **平均内存访问时间：**所有处理单元在同一周期内发送随机地址内存请求所需的平均延迟周期，其中计入仲裁争用和流水线周期；这些内容已在上一小节中讨论。

- **吞吐量：**互连中请求的最大注入率，以 req/pe/cycle 表示。在连续注入请求时，可隐藏 $L_{pipeline}$，但仲裁争用会导致吞吐量下降，如$1/E_{L:\ n\times k}$。

- **互连复杂度：**内部路由和仲裁各级中的交叉开关叶节点数量反映了路由复杂度，近似为 $n \times k$。

- **组合延迟：**路由层级的数量。请求经过路由树的 $\log_2 n$ 级，到达 $n \times k$ 个叶节点之一，随后经过 $\log_2 k$ 级仲裁开关，到达目标存储体。此延迟近似为 $\log_2 n + \log_2 k$。

此外，为了评估层级交叉开关中最复杂级的物理可行性（在表*4*中称为关键复杂级。），我们评估了互连复杂度（$n \times k$）从 256 到 4096 时的布线拥塞，

3. 我们使用 Python 3。7 脚本对带输入队列的模型进行建模。

表3 不同复杂度下对数分级交叉开关互连的布线质量（GF12纳米，13M）

互连。复杂度	拥塞* H	拥塞* V	拥塞* 总体	Area (kGE)	关键路径 (ns)
256	0.13%	0.07%	0.10%	109	0.59
512	0.26%	0.11%	0.19%	196	0.73
1024	0.56%	0.12%	0.34%	361	0.91
1280	1.72%	0.47%	1.09%	503	1.06
1536	3.25%	0.82%	2.04%	669	1.08
2048	34.46%	15.09%	24.77%	923	1.13
3072	172.30%	294.31%	233.31%	1274	1.27
4096	247.10%	368.90%	308.00%	1485	1.47

* 水平层、垂直层以及总体设计的平均布线轨道溢出率。

采用 GlobalFoundries 的 12纳米 FinFET 技术和 13 层金属层。如图3和表3所示，当复杂度<2048时，互连仍可布线。复杂度每翻一倍，面积和导线数量就增加 1.8×，关键路径延迟则增加 <1.3×。超过 2048 后，布线将变得不可行，导致后端制程（BEOL）资源溢出 25-308%（12.4×），同时关键路径延迟也随之增加，进而限制集群性能。

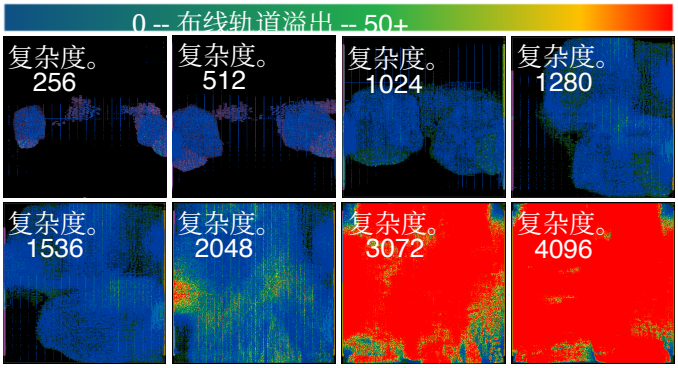


图3. 不同复杂度下基于对数交叉开关的互连的布线拥塞（GF12纳米，13M）。

表4给出了通过多级全连接交叉开关将1024个处理单元连接到4096个L1内存体的不同层级结构选择的比较，范围从扁平化（非层次化）实现到三级层次结构实现，突出了物理可行性与互连性能之间的权衡。

首先，扁平化设计（*1024C*）会产生一个极大的布局布线（PnR）实现模块，其中1024个处理单元仅通过单个交叉开关全连接到4096个存储体。尽管它提供高吞吐量和1周期（最低）的内存访问延迟，但由于互连复杂度不切实际，在物理上不可行。引入*Tile*层次结构（αC–βT）有助于降低复杂度，但*Tiles* 间交叉开关的高路由复杂度仍使物理布线不可行。平均内存访问时间增加，因为大多数内存请求被导向远程 *Tiles* ，并需要在*Tile* 边界进行仲裁。仲裁率随每个*Tile*中处理单元数量的增加而上升，从而进一步降低互连吞吐量。

其次，通过引入组 这一层次结构，我们采用了两级交叉开关互连网络（αC–βT-δG），通过进一步拆分来降低关键互连节点的复杂度，并提高物理布线的可行性。这种选择是一种权衡：它降低了吞吐量，同时增加了平均内存访问时间和零负载延迟。在物理实现中，将***Tile*** 扁平化为组，有助于充分利用可用的后端互连资源，并简化***Tile***之间的互连布线，无需手动放置端口。然而，实现4个大规模组仍然不可行；每个组包含256个处理单元和1024个存储体，时序和布线优化所需的设计工作量巨大，详见第6.1节。此外，物理延长的路径长度限制了可实现的设计频率。设计一个包含4个以上组 的集群会带来巨大的楼层规划挑战，难以安排组 的布局，使对角访问路径保持较短，同时在保持流水线周期数相同的情况下，平衡不同物理距离上的组间组连接。

最后，我们采用三级层次化设计，将每个*Group* 划分为 *SubGroups* (αC–βT–γSG–δG)，同时在每个实现层级保留 4 个层次化模块 ($\gamma = \delta = 4$)，以确保对角访问路径均衡。为实现物理上可行的设计，同时保持低延迟和高吞吐量，TeraPool 集群的最优配置由 4 个*Groups*组成，每个 Group 包含 4 个 *SubGroups* ，每个 SubGroup 包含 8 个*Tiles*，每个*Tile*包含 8 个处理单元，如图 6 所示。

4 TeraPool 集群架构

本节详细介绍 TeraPool 的架构设计要素，重点阐述构建层次化实现时的物理设计感知方法，即自底向上进行设计。

4.1 处理单元

TeraPool 的处理单元采用基于*RV32IMA*指令集架构（ISA）的单发射、单级 RISC-V*Snitch*核心，其中指令在核心内以单个周期完成译码和执行 [17]，。Snitch 具有一个私有的、全相联的、基于标准单元存储器（SCM）的 32 条指令 L0 指令缓存（I$），旨在快速重复执行计算热点循环。我们构建了 *Snitch* 核心复合体（CC），并加入 ISA 扩展，包括在整数处理单元（IPU）中实现的*Xpulpimg*[4]，以及在浮点子系统（FP-SS）中实现的*zfinx*、*zhinx*和*smallfloat* [26]。为满足众核计算集群通常面临的严格面积约束，我们通过直接使用整数寄存器文件执行浮点和半精度操作数，免去了成本高昂的专用浮点寄存器文件和加载存储单元（LSU），同时采用单指令多数据（SIMD）方式支持对两种半精度类型进行并行操作。

在存储器访问方面，*Snitch* 支持非阻塞指令执行，以容忍多周期 L1内存访问延迟。加载和存储指令的依赖关系由记分板记录，并通过事务表跟踪其退役情况，如图 4 所示。事务表会跟踪未完成的内存请求，使核心无需阻塞等待响应即可连续发出一系列加载和存储指令，并最终转而执行其他不存在数据依赖的指令。鉴于 TeraPool 的非统一内存访问互连设计，事务表会按顺序退役加载指令

4. Xpulpimg 扩展包含领域专用指令 [16]，，例如乘累加（MAC）和加载后递增。

表4 1024个处理单元全连接4096个L1存储体的分层互连设计分析

层次结构 互连.*	互连质量						设计挑战			
	零负载 (周期)	AMAT (周期)	吞吐量 (请求/处理单元/周期)	总 复杂度。	关键 复杂度。	关键 组合逻辑 延迟	物理 布线	Path 平衡	Base 模块 规模	互连 性能
1024C	1.000	1.130	0.885	4194304	4194304	22				
4C-256T	2.992	6.081	0.245	87040	65536	16				
8C-128T	2.984	10.075	0.124	54272	16384	14				
16C-64T	2.969	18.077	0.062	74752	4096	12				
4C-16T-16G	4.867	5.318	0.431	163840	320	8.3				
4C-32T-8G	4.742	5.443	0.409	122880	1024	10				
8C-16T-8G	4.734	5.794	0.358	90112	512	9				
8C-32T-4G	4.484	6.676	0.272	69632	1024	10				
16C-8T-8G	4.719	6.669	0.273	110592	1536	10.6				
16C-16T-4G	4.469	8.612	0.178	90112	1280	10.3				
4C-16T-4SG-4G	6.367	8.457	0.270	121856	4096	12				
8C-8T-4SG-4G	6.359	9.198	0.230	89088	1024	10				
16C-4T-4SG-4G	6.344	11.049	0.159	109568	1536	10.6				

*4096 1KiB片上存储器组按存储体因子4，与处理单元一起划分到各个层次结构中。*层次结构记为 αC–βT–γSG–δG，其中 δ 为Group的数量，每个Group包含 γ 个子组，每个子组包含 β 个Tile，每个Tile包含 α 个处理单元。

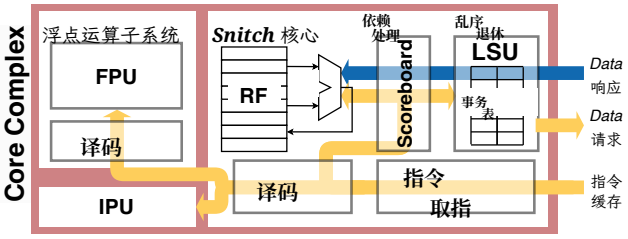


图 4. Snitch的框图，突出显示加载存储单元，以及加载存储单元用于标记未完成加载与存储的事务表。

而记分板则确保指令按顺序送至执行单元。通过循环展开调度多个加载指令，我们可以在集群的 L1暂存存储器中隐藏存储器访问延迟。支持的未完成事务数量可通过硬件配置；该数量选在盈亏平衡点，此时条目数量既能避免过高的面积开销，进一步的性能提升也已受可用 ISA寄存器数量限制。我们发现，对于大多数内核而言，8 个未完成事务已足够。例如，在分块矩阵乘法算法中，[27],a 个输出矩阵块——32 个ISA寄存器所支持的最大数量——最多需要 8 个输入事务。

Snitch的硬件占用面积小、延迟容忍度高且可扩展，使其成为实现可调大规模共享一级存储器众核设计的理想候选。

4.2 集群设计

图5中的*Tile* 是基准层次级别，通过全连接交叉开关将8个核心复合体连接到共享软件管理存储器的32个存储体，提供单周期、零负载访问延迟。共享

4KiB 二路组相联一级指令缓存由*Snitch* 核心共享，并为*Tile* 内的所有核心提供服务，随后连接至零级指令缓存。为支持浮点除法和平方根指令（这类指令对矩阵求逆等操作至关重要，但使用频率低于基本浮点运算符），我们按照轮询策略，由四个核心复合体共享一个浮点除法与平方根（DIVSQRT）单元。每个*Tile* 都有 7 个端口连接至一级互连网络：一个通过 8×8 全连接交叉开关连接到同一子组内的其他*Tile*；三个通过三个8×8 全连接交叉开关连接到同一组内其他 3 个子组中的*Tile*；其余 3 个端口连接到 3 个

远程*Group*中的Tile，通过三个 32×32 全连接交叉开关实现。我们的互连在各层次级数上提供 4KiB/周期的峰值带宽和 1.875KiB/周期的二分带宽。

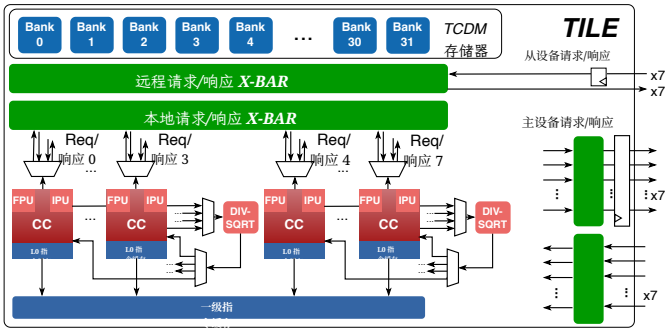


图 5. Tile 框图：包含 8 个核心复合体和 2 个共享的浮点 DIVSQRT 单元；本地交叉开关可实现对紧耦合数据存储器（TCDM）的 1 周期访问，7 个远程请求/响应端口则连接至更高层级。

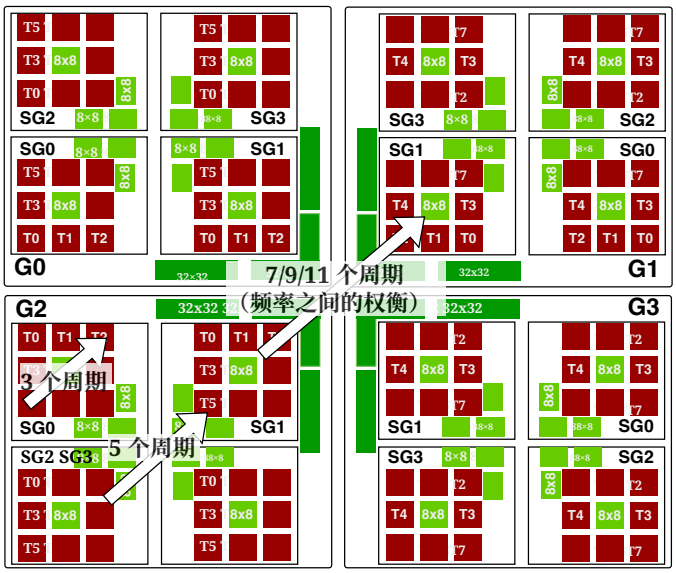


图 6. 集群级框图，展示 Tile 的布局视图和互连交叉开关，并突出显示各层级交叉开关的复杂度以及一级存储器访问延迟。

层级互连设计提供了灵活性，

可通过添加溢出寄存器来切断较长的远程访问路径。溢出寄存器置于发起端和目标 Tile 边界处的主请求端口与响应端口上，以及交叉开关之后的外发主端口上。因此，在同一子组内的往返访问延迟为 3 个周期，在同一组内访问其他子组的往返访问延迟为 5 个周期。远程组访问的延迟可通过硬件参数配置，并取决于目标工作频率。我们用下标标识不同配置，下标表示核心访问各层级的延迟。在 TeraPool$_{1\text{-}3\text{-}5\text{-}7}$中，在集群层级每个从端口的请求路径和响应路径上都添加一个寄存器，使远程组访问的往返延迟增加 2 个周期。TeraPool$_{1\text{-}3\text{-}5\text{-}9}$ 配置在组层级的从端口上添加溢出寄存器，而 TeraPool $_{1\text{-}3\text{-}5\text{-}11}$ 配置还在主路径上添加了一个寄存器。图 6 展示了包含*Tile*布局的完整集群层级概览。这些配置体现了可实现频率与 L1最坏情况延迟之间的权衡，我们将在第 6 节进行分析。

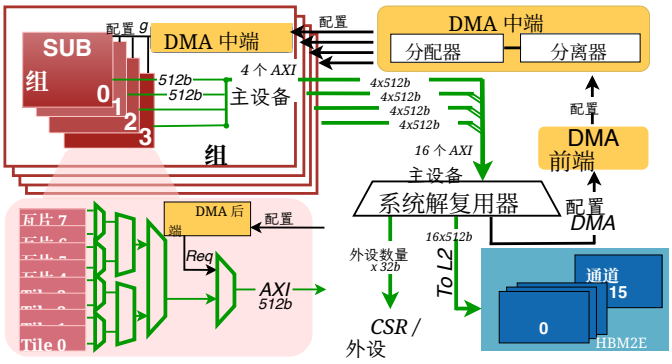


图 7. 系统级高带宽存储器链路设计，采用分层 AXI4 互连和模块化 DMA 引擎。

5 高带宽存储器链路

我们介绍了HBML，即主存访问接口，它由直接内存访问（DMA）引擎和系统级全局互连组成。为评估TeraPool的HBML能力，我们使用开源、周期精确的*DRAMsys5.0* 模拟器 [18] 进行主存协同仿真。

5.1 分层AXI互连

HBML互连采用开放标准高级可扩展接口（AXI4）协议，将TeraPool集群连接到主存，并作为外设、主机控制和控制与状态寄存器（CSRs）的系统级接口。然而，由于物理布线资源有限，尤其是在已受PE到L1布线复杂度制约的设计中，要为每个由软件管理、采用物理寻址的L1 SPM存储体提供高带宽连接，是一项重大挑战。

为尽量减小面积和布线开销，我们采用图7所示的树形AXI4 HBML互连。每个*Tile* 由其*Snitch* 核心共享一个512位AXI4主设备端口。核心请求通过由目标地址控制的解复用器进行路由，并被引导至L1 SPM互连或系统级组件。AXI4互连还

用于填充 L0 指令缓存。在每个 子组内，来自 8 个 *Tile* 的 AXI4 端口通过树形结构进行仲裁，并连接到单个 512 位 AXI4 主设备端口。最后，每个子组的 16 个 AXI4 主设备经过基于地址的系统解复用器，被导向 DMA 前端（见第 5.2 节）、L2 主存或内存控制器，以及控制与状态寄存器/外设接口。

5.2 模块化 DMA 引擎

尽管 TeraPool 集群将其 L1 SPM 存储体分布在大规模层次化设计中，但仍保持统一物理地址空间。因此，只需一个 DMA 引擎即可管理 L1 与 L2 主存之间由软件控制的数据传输。然而，采用单个传统 DMA 引擎会带来模块布局方面的挑战，还需要建立与全部 4096 个 SPM 存储体的连接，从而导致路由复杂度显著增加。另一方面，采用多个 DMA 引擎会使编程变得复杂，开发人员需要管理不同 DMA 单元的传输地址范围。

为简化编程接口并降低路由复杂度，我们采用模块化 iDMA [28]，由三个组件组成：前端、中端和后端。前端通过指定源地址、目标地址和传输大小来启动数据传输。随后，传输请求被转发到中端，并根据子组的 L1 地址边界划分为子任务。这些子任务被分配给 DMA后端，由其管理 L2 主存与系统 AXI4 互连之间的数据传输。此外，后端 负责连接系统级互连与 L1 SPM 交叉开关，从而将数据高效地传送到各个存储体。

5.3 主存

为展示 TeraPool 的 HBML 管理高带宽数据流的能力，我们采用开源的*DRAMSys5.0* 模拟器 [18]，，这是一个面向动态随机存取存储器（DRAM）子系统、基于*SystemC TLM*-2.0 的周期精确框架。我们将主存配置为包含两个 DRAM 堆栈，每个堆栈有 8 个*Micron MT54A16G808A00AC-36*HBM2E 通道，支持以 2.8/3.2/3.6Gbit/s/pin 双倍数据速率（DDR）传输。

与*DRAMSys5.0* HBM2E 子系统进行协同仿真，需要编译一个基于*C++/SystemC*的寄存器传输级（RTL）集群模型。大规模集群的编译过程极其耗时，尤其是在硬件开发迭代期间，每次架构调优都需要重新编译整个模型。为解决这一低效问题，我们将配置好的*DRAMSys5.0*模型编译为动态链接库，使其兼容*QuestaSim*和*VCS*等 RTL 仿真器。这种方法显著加快了设计开发，使架构调优能够快速进行。我们将 *DRAMSys5.0* 动态链接库仿真环境 [5]开源。

5.4 存储器映射方案

TeraPool采用混合寻址方案，如图8左侧所示。4MiB一级软件管理片上存储器被划分为一个顺序区域

5. https://github.com/pulp-platform/dram rtl_sim

以及一个交错区域（默认分别为512KiB和3.5MiB，但可在设计时配置）。在顺序区域中，内存请求被限制在一个*Tile*内，以尽量降低延迟和功耗，使程序员能够将栈等私有数据本地存储在处理单元的*Tile*中。在交错区域中，对所有存储体应用字级交错内存映射。数据在所有处理单元之间均匀分布，从而尽量降低平均访问延迟并减少存储体冲突。核心访问交错地址空间的零负载延迟，以及在随机访问模式下的平均延迟，如图8右侧所示。该方法以极低的硬件成本缓解远程访问带来的性能与功耗开销：布线交叉，以及一个用于打乱相应请求地址位的多路复用器。此外，交错使访问模式独立于SPM存储体大小，而SPM存储体大小可在设计时灵活配置。

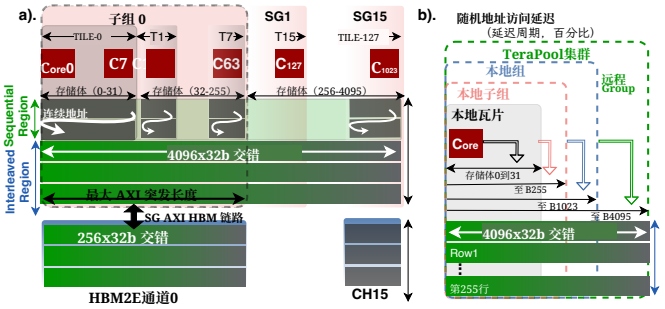


图8。a). 混合内存地址映射方案。b). 核心在不同层级随机访问一级缓存存储体时的访问延迟。

在每个子组中，交错区域的数据分布在256个存储体中，每个存储体行存储一个32位字。因此，通过HBML在L2 主存与其他位置之间传输数据时，经由单次AXI4突发传输可传输的连续地址数据最大为256个32位字，由子组的AXI4主设备负责处理。为保持这一最大突发请求长度并提高传输效率，我们为每个子组配置1个直接内存访问后端，以避免进一步拆分请求。我们使L2 主存的数据交错粒度与AXI4突发传输长度一致，将每个HBM2E通道的交错粒度配置为256个32位字。此配置可在单次突发传输所请求的数据跨越多个HBM2E通道时，减少突发传输拆分，并降低不同AXI4主设备的请求跨入不同HBM2E通道所造成的系统互连冲突。

为了充分利用主存带宽（详见第5.3节），该带宽支持可配置的 DDR 速率 716.8-921.6GB/s。为匹配 HBM2E 峰值带宽，TeraPool 的目标实现频率设为 700-900 MHz，并利用 HBML 的 16×512bit AXI4接口。为验证这一分析，我们研究了 TeraPool 在 500 MHz 下的表现

并在 900MHz 下，在不同的 HBM2E DDR 速率配置下，对整个 4MiB 集群内存进行密集的数据传输（输入和输出）。结果如图 9 所示。在较低的目标频率（500MHz）下，带宽受 TeraPool 工作频率限制，仅达到 HBM2E 峰值带宽的 61.8%-49.4%。相比之下，在 700-900MHz 下，三种相应的 HBM2E DDR 配置均实现了接近峰值的利用率（ 97% ），在 900MHz 时带宽最高可达 896GB/s。带宽的轻微损失

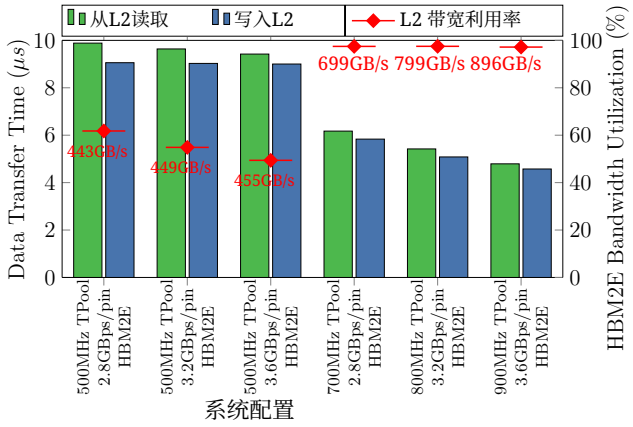


图 9。TeraPool 的一级缓存读写 4MiB 数据时的数据传输性能；这些读写由 HBML 管理，并通过作为 L2 主存的 16 个 HBM2E 通道完成。

归因于直接内存访问前端配置周期以及不可避免的 DRAM 刷新周期。这项分析进一步说明了将 TeraPool 的设计目标设定为接近 GHz 频率的合理性，并展示了 TeraPool 的 HBML 能力，即使以业界标准的 HBM2E 作为主存、其延迟达到数百个周期，也能高效管理高带宽、长延迟的数据传输。

6 物理实现

本节介绍 TeraPool 物理设计的详细信息。TeraPool集群在 81.8 mm^2 的芯片面积内实现，采用*GlobalFoundries* 的 12 nm LP+ 鳍式场效应晶体管工艺和 13 层金属堆叠。我们使用*Synopsys* 的 *Fusion Compiler 2022.03* 进行合成、布局和布线。我们的多级 FC 互连设计非常适合自底向上实现流程：该流程为每个层次化模块定义了清晰的边界，并允许添加用于时序控制的溢出寄存器。我们使用 Synopsys 的 PrimeTime 2022.03，并参考布局后门级仿真获得的开关活动，在典型工作条件（TT/0.80 V/25 °C）下确定功耗。

6.1 设计可行性与布局规划

为验证我们在第 3.2 节中对大规模基础模块实现所需的大量电子设计自动化（EDA）工作量的分析——这些工作量会导致运行时间难以管理、优化性能下降——我们实现了 16*C*-8*T*-8*G* 层次化配置（参见表4）。图 11 展示了实现一个 TeraPool 组时 EDA 工具运行时间的相对分布。对于无法实现的 16*C*-8*T*-8*G* 层次化配置，EDA 工具的总相对运行时间几乎是 $\text{TeraPool}_{1\text{-}3\text{-}5\text{-}9}$ 的 $3.5\times$ 倍，其中时序优化占工作量的 80% 以上。具体而言，其布线阶段比其他配置 $5.5\times$慢。尽管进行了大量设计优化工作，组设计中的八个 16×16 互连仍导致严重的布线拥塞，产生大量金属短路。此外，绕行布线显著增加了时序路径的长度，使得设计无法在 500MHz 下完成时序收敛

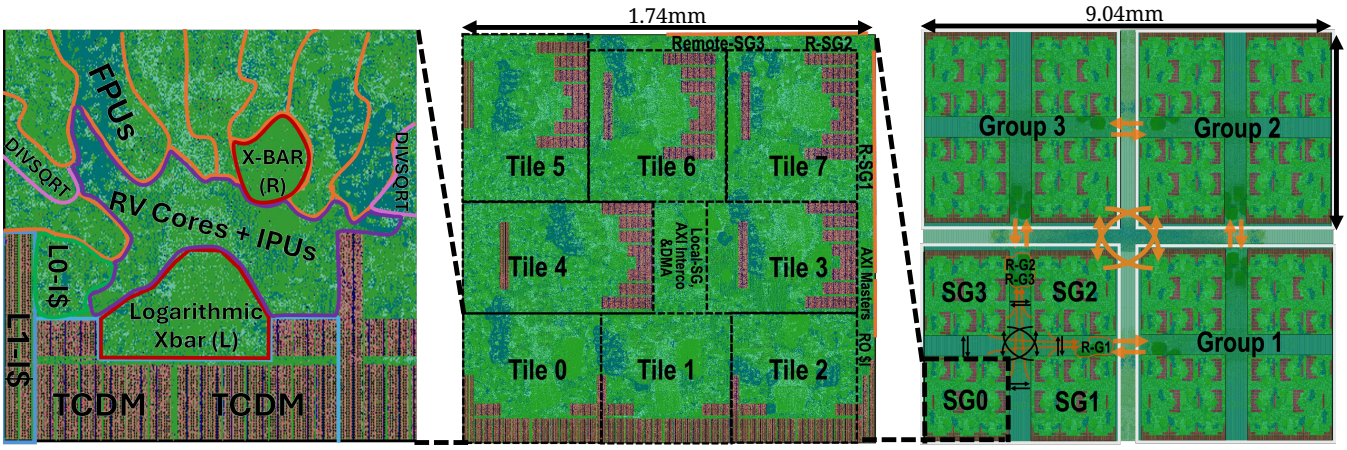


图10. 每个TeraPool层次化实例的布局布线及标注视图。

目标条件（TT/0.80V/25°C）。这些发现充分支持我们在第3.2 节中关于 TeraPool 物理感知的层次化互连分析的见解。

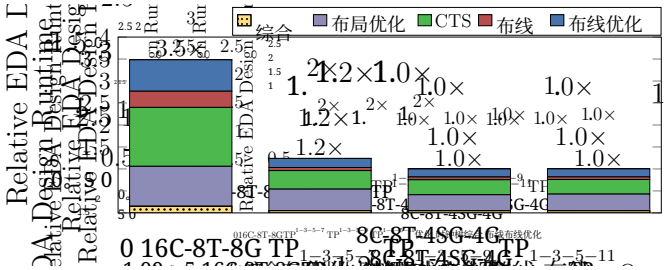


图11. 不同设计配置下TeraPool 的 Group实现时间相对占比的细分。

图10展示了带有模块布局标注的物理布局快照。Subgroup的实现尺寸为3.03 mm^2，，从而实现了每个核心0.047 mm^2/的面积以及令人满意的58%面积利用率。每个Subgroup包含8个Tile，每个Tile的SPM宏均经过分组，并采用U形布局以包围本地交叉开关，从而最小化总体距离并避免宏过度堆叠。Subgroup的中心和边缘留空，以便放置互连单元；底部一侧的宏U形布局进行了旋转，以更好地利用设计面积。远程子组和远程组的接口端口放置在东北侧边界。Subgroup和Group模块采用中心对称网格排列，其间留出通道用于对互连进行布局布线。为进一步提高面积利用率，我们将接口端口放置在Subgroup模块后方，并逐渐缩小通道宽度，直至后端互连资源受限。在集群顶层，分配了宽度为0.68 mm的通道，用于连接中心区域各Group的交叉开关，以及将AXI到HBM2E通道连接至边界。这些布线通道使每个核心的总体面积增加至0.079 mm^2/。

6.2 性能与逻辑面积

我们为远程组访问配置并实现了硬件可参数化延迟，分别设置为 7、9 和 11 个周期，对应的工作频率分别为 730、850 和 910 MHz（TT/0.80 V/25 °C）。这些结果清楚地展示了延迟与工作频率之间的权衡。当配置的延迟超过 11 个周期后，设计速度不再受远程组访问路径的限制，而是受 Snitch 核心的限制：该路径从指令缓存之后的一个寄存器开始，经过 Snitch 和请求互连，最终到达 SPM 存储体的时钟门控处。

等效门（GE）定义为一个二输入*NAND* 门的面积，是衡量逻辑面积的技术无关指标。如图12所示，我们的层级互连设计仅占

8.5%，HBML仅占集群面积中可忽略不计的9.2%。面积占比最大的部分是SPM存储体，其次是*Snitch* 计算簇和指令缓存。核心复合体还可进一步划分为核心本身（7.3%）、整数处理单元（9.1%）以及面积效率极高的FP-SS（22%）。这一细分结果表明，设计中的大部分逻辑面积用于存储和计算资源，而处理单元到一级存储器互连及HBML所占面积极小。

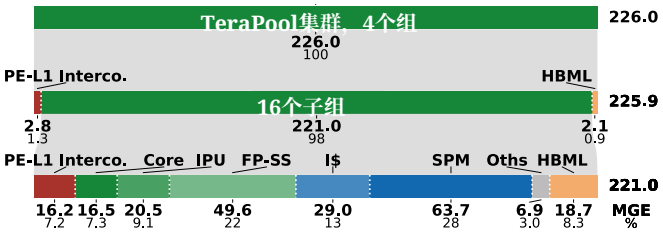


图12. 层级面积细分图，注释标示各组件相对于其直接父组件的百分比。

6.3 能耗分析

图13展示了 TeraPool 在远程组访问延迟分别为 7、9 和 11 个周期时，指令执行的能耗分解（pJ/指令/核心）；测量时对应的工作频率分别为 730、850 和 910MHz（TT/0.80V/25°C）。为展示不同存储器访问距离下的能耗，我们给出了 32位加载指令（*ld*）访问以下存储体时的能耗分解：其本地*Tile*内的存储体、同一子组内另一个*Tile*中的存储体、同一组内另一个子组 中的存储体，以及远程组中的存储体。随着存储器访问距离增加，平均能耗分别比本地*Tile*高出10%、20% 和 58%。

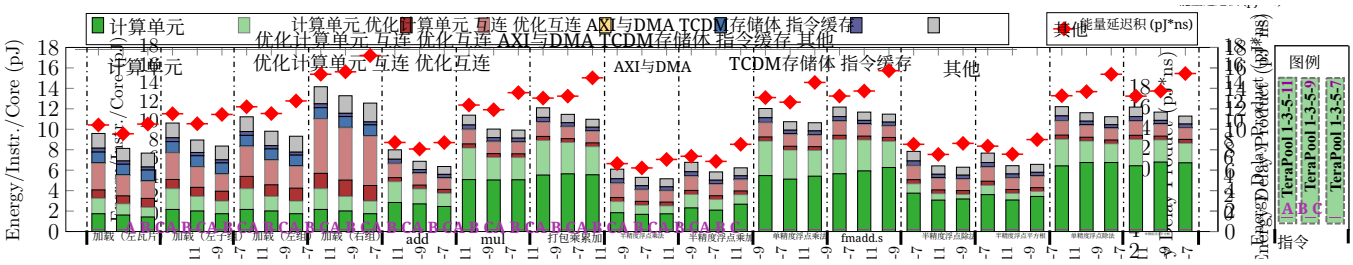


图13. 在典型的0.80 V/25 °C条件下，不同TeraPool配置（远程Group访问延迟为7/9/11个周期）的指令能耗分解。访存指令（ld）的目标包括本地Tile、子组、组和远程Group；浮点指令分别以单精度（.s）和半精度（.h）呈现；除法/平方根能耗按每个DIVSQRT单元量化。

互连（2.5 −6.8pJ）和 L1 SPM存储体（1.06 pJ）是能耗的主要组成部分，最高占总能耗的 51%。对于算术指令，TeraPool 的整数运算能耗为 6.4–13.5 pJ，半精度（16 位）浮点运算能耗为 5.2–7.9 pJ，单精度（32 位）浮点运算能耗为 11.3–12.2 pJ。计算单元（整数处理单元、FP-SS 和 DIVSQRT）是能耗的主要来源，在整数、半精度浮点和单精度浮点计算中的平均占比分别为 69.9%、60% 和 72.3%。当互连交叉开关未被访问时，例如执行单精度浮点乘加（浮点乘加，fmadd.s）运算期间，该运算消耗 12.19 pJ，其中仅 14.5% 的总功耗主要来自时钟传播和漏电导致的内部单元功耗。得益于存储器时钟门控，未被访问时，SPM存储体的能耗低于 0.1 pJ，能耗降低 98%，可忽略不计。

图中浅色部分紧邻计算单元 和互连 部分，表示物理设计为达到目标频率而引入的优化单元的能耗。随着频率升高，这些部分略有增加，这是由于采用了低阈值优化单元；这类单元旨在通过牺牲较高功耗来降低延迟，以达到接近GHz的工作频率目标。例如，在典型条件（TT/0.80 V/25 °C）下，工作频率从730 MHz提高到910MHz时，远程*Group* 存储器访问的能耗仅增加1.6 pJ，单精度浮点乘法（*fmul*）仅增加 1.4 pJ，平均增幅为16%。

能量延迟积（EDP）是一项关键指标，可对能效和延迟进行归一化 [29]，从而深入了解能耗与性能之间的权衡。为确定 TeraPool 的最佳能效配置，我们分析了每条指令的 EDP（单位为 pJ × ns），并在图13中用红色标记突出显示。结果清楚表明，对于大多数操作，采用 9 周期远程组延迟、运行频率为 850 MHz 的 TeraPool 配置，在能效与性能之间实现了最佳平衡。总而言之，TeraPool 实现了高能效设计；即使对于高复杂度的 32 位浮点运算，每 5 ∼15pJ/操作/核心也仅消耗少量能量，其中大部分能量用于计算组件。

7 性能分析

与管理多个集群之间的数据传输相比，共享内存集群的优势之一是编程简单。TeraPool 支持一种精简的分叉-汇合编程模型，并提供 C 运行时函数，可在单程序多数据（SPMD）执行中，高效地切换顺序代码段与并行代码段 [30]。启动后，所有处理单元都会并发执行 C 二进制程序；程序员可以访问各处理单元的唯一 ID，并在并行计算中为每个核心静态分配独立任务（分叉）。处理单元完成任务后，会通过对共享内存位置执行原子取值并加一来同步（汇合）。

为评估互连性能和集群总体利用率，基准测试涵盖了顺序和非顺序存储器访问模式下的数据并行执行。对于顺序存储器访问执行，采用 A 乘 X 加 Y（AXPY）和点积（DOTP）作为本地访问内核；处理单元主要以低延迟从共享内存的邻近区域取数，且每个输入仅由一个核心取用，因此非常适合评估低层级互连网络。通用矩阵乘法（GEMM）则作为全局访问内核，其数据结构分布在所有共享 SPM 存储体中。采用分块实现，以充分利用寄存器堆并最大限度地提高计算强度。

针对非顺序存储器访问，我们采用基 4 抽取频率的 Cooley-Tukey 快速傅里叶变换（FFT），以减轻本地存储体的内存访问负担。在 N 点 FFT 的 k^{th}级中，每个核心计算 4 个蝶形运算，输入端之间的距离为 $N/(4 \times 4k)$。处理不同 FFT 的处理单元彼此独立同步。我们还纳入采用压缩稀疏行格式的稀疏矩阵加法（SpMMadd），该运算对两个图逐元素相加，是一种关键的 *GraphBLAS*运算 [31]，用于评估互连在不规则分布式内存访问下的表现。

在图 14 a) 中，我们展示了采用能效配置时的内核性能：远程组延迟为 9 个周期，工作频率为 850MHz（TT/0.80V/25°C）。我们将每周期指令数（IPC）和架构停顿占比按总周期数的百分比进行分解。由于所有内存访问均为本地访问，AXPY 和 DOTP 的平均内存访问时间（AMAT）都较低，IPC 分别高达 0.85 和 0.83，这得益于本地*Tile* 互连的高带宽。性能损失仅归因于计算结束时的同步屏障（等待中断（WFI））。由于归约操作，DOTP 的 AMAT 和同步开销更高。对于 GEMM 这一全局访问主导型内核，核心需要从整个集群中取数，其测得的 AMAT 与第 3 节中给出的随机访问解析模型高度吻合，验证了该模型可为设计指导提供准确依据。IPC 仍保持在较高水平，为 0.7；而 LSU 停顿占比上升，原因是全局访问需要处理单元通过共享互连端口从所有存储体取数。

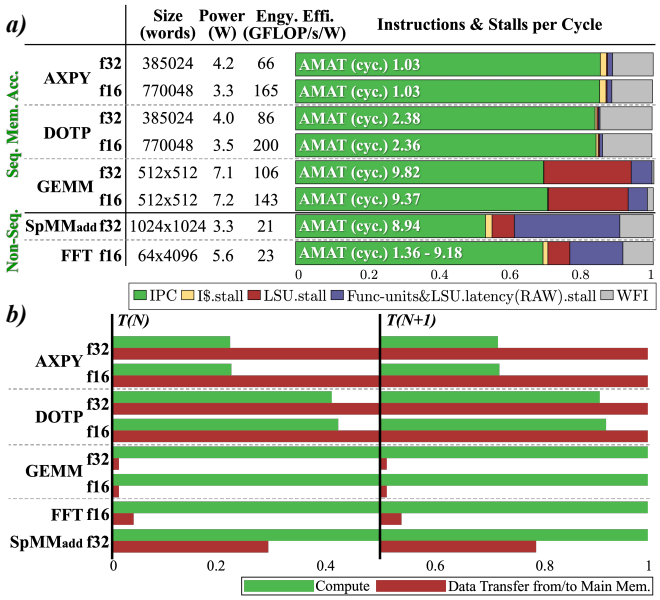


图14。a）TeraPool中内核执行期间，指令和停顿占总周期的比例；b）采用双缓冲的内核与HBM2E之间进行数据传输的时序分解。

较小比例的读后写（RAW）停顿则源于互连冲突导致的延迟增加。得益于较大的共享内存，同步开销可以忽略不计，使集群能够高效处理更大规模的问题分块。

对于非顺序存储器访问，我们并行运行64个独立的4096点快速傅里叶变换，并在同步屏障之间计算各级，以尽量减少开销。其IPC达到0.7，由于计算过程中还需要数据打包和数据混洗等数据控制指令，因此性能为23GFLOP/s/W。不同级的平均内存访问时间介于1.36至9.18个周期之间。通过子组内部及子组之间的互连进行数据访问，仅造成6.15%的加载存储单元停顿。尽管处理单元并未针对稀疏工作负载进行专用扩展优化，SpMMadd仍实现了较高的IPC，达到0.53。频繁的条件分支给非专用处理单元带来压力；ISA定义寄存器数量有限，妨碍了循环展开，并导致读后写停顿。然而，在分析内存访问时，我们发现，非顺序窄访问仅造成6%的互连争用，这表明我们的互连能够以非阻塞方式支持多个不规则并行访问。

为评估 TeraPool 的 HBML 的高带宽，我们使用通过业界标准 HBM2E 访问数据的基准测试内核，并采用双缓冲来隐藏传输延迟。我们使用两个 L1缓冲区空间：一个用于执行当前内核（*T(N)*），另一个用于传输下一轮所需的数据（*T(N+1)*）。如图 14 b) 所示，即使对于访存受限程度很高的内核，DOTP 的计算阶段仍占 82%，而 AXPY 则占 44%，这是因为快速计算无法隐藏结果传输和新数据加载的延迟。对于计算受限内核，TeraPool 能实现高性能：其大型共享一级缓存完全隐藏了 HBM2E 的延迟。

这些分析凸显了我们的PE至L1互连和HBML即使面对PE至L1流量模式高度全局化、争用不可忽视且……的内核，也能提供相当大比例峰值工作负载的能力，使用 HBM2E 主存进行的长延迟数据传输。

8 相关工作

本节将我们的工作与采用各种纵向扩展/横向扩展拓扑以实现高核心数的、基于集群的可编程多核和众核架构进行比较。我们优先考虑算法灵活性，重点关注通用处理单元集群，而非由高度专用、面向单一领域的处理单元（例如神经处理单元（NPUs））构成的密集阵列。表5概述了这些架构。我们将当前最先进的架构分为两大类：核心数为< 100 的集群多核设计，以及具有 > 100的众核设计。

在多核设计类别中，我们发现了横向扩展和规模扩展两种拓扑：（**i**）通过二维网格片上网络进行横向扩展的基于交叉开关的集群，其路由器链路例如 *Kalray MPPA3-80* [32]，；以及（**ii**）采用基于交叉开关的互连的规模扩展集群，例如 *RamonRC64* [10], [33]。在后一种设计中，所有核心均连接到一个共享的 4MB 片上存储器，并采用多级对数交叉开关。为降低布线复杂度，核心数量受到限制。凭借大得多的核心数量，我们的工作明显优于这些设计，实现了 32-128× 性能。

在众核设计类别中，在横向扩展架构里，*TensTorrent* 的 *Wormhole* [34] 将 400 个处理单元集成在一个 10×8 *Tensix* 矩阵 中，并通过二维网格片上网络互连。在每个*Tensix* 中，仅 5 个处理单元通过具有 4 个流水线级的交叉开关共享 1.5 MB L1 缓存。在相同的往返流水线下，TeraPool 的组包含 256 个处理单元，它们与 1MiB 共享内存中的组本地部分互连，实现了 76.8× 的处理单元与本地共享内存之比，这得益于我们的单级处理单元设计采用全组合逻辑交叉开关，仅在分层边界处加入溢出寄存器，以打断较长的远程访问路径。在*Esperanto ET-SoC-1* [9], 中，交叉开关将 32 个*Minion* 核心连接到 4MB L1 缓存，并通过二维网格片上网络横向扩展至 1088 个核心，同时利用带有大型向量寄存器文件的 RISC-V 向量扩展（RVV）处理单元来隐藏事务延迟。TeraPool 仅通过纵向扩展便达到相近的核心数量，延迟由一个尺寸可忽略的事务表隐藏，而不是在我们的轻量级标量核心设计中依赖功耗高昂的大型寄存器堆。我们的设计实现了 16× 更高的 L1 带宽和 32× 更高的 L2 带宽。*HammerBlade* [35] 从一个带有 4kB 暂存存储器的单核心*Tile*横向扩展，采用混合二维网格拓扑：*Tile* 在水平方向通过 Half-Ruche 通道连接，在垂直方向通过直接网格链路连接，并在缓存 *Tile*与 HBM2 之间通过虫洞跳跃链路进行缓存回填。共享片上存储器区域（*Tile* 组）在设计时定义于一个 128 核心 *Cell* 内（最高 0.5 MiB），比我们的共享内存规模 8× 更小。尽管采用了这种复杂的拓扑结构，访问共享内存仍可能需要多达 52 个周期，并要求每个*Tile*支持多达 63 个未完成请求。相比之下，TeraPool 在共享内存大小与最大访问延迟之比上取得了 46×优势。因此，*HammerBlade*需要数据放置感知编程来缓解延迟惩罚，从而增加了编程复杂度。

“*Occamy* [23]”从一个集群扩展而来，其中 8 个 RISC-V 处理单元通过基于交叉开关的互连共享低延迟（1 个周期）的 128KiB 一级缓存。为支持大型数据集上的并行处理，24 个扩展集群彼此互连；数据由各集群内的直接存储器访问控制器管理，而集群间

互连占芯片面积的 14.8%。相比之下，TeraPool 去除了集群间互连，采用由单个直接内存访问控制器管理的 $32\times$ 更大一级缓存，进一步简化了编程。最后，在现代图形处理器的流式多处理器中，处理单元通过基于交叉开关的互连共享存储器。例如，英伟达 *H100* [14], [36] 由流式多处理器构成，其中 128 个单精度浮点单元共享 256kB 存储器，并通过数据驱动型片上网络网格扩展，处理单元总数达到 18 432 个。在 TeraPool 中，共享存储器 $16\times$ 更大，每个处理单元的 $4\times$ 一级缓存带宽也更高，同时仍具备物理可行性。此外，为支持单指令多线程（SIMT）执行，每个流式多处理器都包含多个调度器、分派单元和大型寄存器文件，显著增加了硬件复杂度。相比之下，TeraPool 的低延迟共享存储器支持未完成事务，无需依赖线程切换来隐藏延迟，从而降低硬件成本。此外，其单程序多数据执行采用统一的多兆字节一级缓存地址空间，无需在不同存储区域间复制共享数据。

据我们所知，文献中规模最大的规模扩展共享内存集群是 MemPool [16]。它采用了与我们工作相似的设计理念，包含 256 个核心，共享一个 1MiB 的暂存存储器（SPM）。借助我们提出的分层互连和物理设计感知方法，我们将集群规模扩展了 $4\times$，并集成了功能更强、支持多精度浮点的处理单元。我们同时确保了物理可行性，使所有处理单元都能访问任意一个共 4096 个的共享内存存储体，延迟最多仅增加 2–6 个周期（具体取决于目标频率）；非阻塞加载/存储单元与恰当的指令调度可有效隐藏这部分延迟。

由于 TeraPool、Occamy 和 MemPool 都是开源架构，分别代表小型（数十个处理单元）、大型（数百个处理单元）和超大型（数千个处理单元）集群，因此可以对它们进行定量、公平的比较。表6比较了维持单位计算量工作负载所需的主存访问量（字节/FLOP）和利用率（IPC），涵盖 AXPY 和 GEMM 这两个关键内核。显然，我们希望前者尽可能小，后者尽可能大。对于像 AXPY 这样数据复用率较低的内核，不同设计的字节/FLOP 保持不变。作为规模最大的集群设计，TeraPool 实现了同样高的利用率（0.85 IPC），因为数据可以存储在处理单元附近，从而利用低延迟本地互连的高带宽。对于 GEMM，权衡则更为复杂：由于远程 组 互连的延迟增加，以及同一 *Tile* 中的核心同时访问同一 组时在层级边界处产生的冲突，TeraPool 的 IPC 有所下降。尽管如此，IPC 仍然保持在较高的 0.7。另一方面，TeraPool 的存储器容量较大，可支持大规模分块和更多数据复用。IPC 下降 19%，但维持工作负载所需的系统级每核带宽降低了 44% 和 85%，这一收益足以弥补 IPC 的下降；尤其考虑到众所周知的“存储墙”问题，这一点更为重要。

9 集群纵向扩展的权衡与未来工作

虽然我们展示了更大规模集群在硬件和软件方面的优势，但扩大规模会在PE到L1存储器互连方面引入两项主要权衡。首先是布图面积。尽管我们的互连确保了物理可行性，但在实现模块之间需要布线通道（占芯片面积的40%），以便布局交叉开关并进行布线。相比之下，遵循相同设计原则的较小集群（如 MemPool [16],）由于层次级数较少，面积开销也更低。第二项权衡涉及层间二分带宽的扩展性。峰值带宽与二分带宽之比较高（$peakBW/bisecBW$），意味着内存访问未局部化时会出现更多争用。与 MemPool（峰值带宽为1KiB/周期，二分带宽为0.75KiB/周期）相比，TeraPool 的规模扩大了 $4\times$，峰值带宽线性提升至4KiB/周期。然而，受布线资源限制，二分带宽仅达到1.875KiB/周期，使峰值带宽与二分带宽之比增加了60%。

为应对这些权衡，未来的工作将重点探索面积效率更高的互连设计。一个有前景的方向是采用带有宏单元上布线的二维网格片上网络 [16]，用于PE到L1存储器通信。网格规则的布线拓扑结构能够利用更高金属层上的受控布线，并减少通道开销；可配置的路由器数量则可以充分利用布线资源，从而提升二分带宽。然而，由于端到端延迟会随着跳数增加而上升，片上网络不太适合对延迟敏感的核到L1存储器访问。将二维网格网络与交叉开关结合起来，或许能提供一种兼顾可扩展性与低延迟的平衡方案。

另一个演进方向是面向应用的架构定制。尽管本文介绍的TeraPool采用通用处理单元，但未来的部署可能受益于针对特定工作负载和数据格式定制的专用ISA扩展或协处理器（例如针对稀疏性和量化数据格式的扩展或协处理器）。此外，本地Tile折叠式顺序暂存存储器地址区域非常适合用于集成专用引擎，例如张量阵列或神经网络处理器，以加速以人工智能为中心的计算。

10 结论

TeraPool 是一种在物理上可行的众核集群设计，包含 1024 个可独立编程、具备浮点运算能力的 RISC-V 内核，通过非统一内存访问层级交叉开关互连网络共享 4 MiB 的 L1暂存存储器，带宽最高可达 4096 B/周期。该 932 GB/s 高带宽存储器链路旨在高效管理大型共享一级缓存中的数据传入和传出。与目前最大规模的先进集群相比，其处理单元数量提升了 $\times 4$。

通过对核心与一级缓存之间互连的全面分析，我们成功使用*GlobalFoundries*的 12LPPLUS 鳍式场效应晶体管技术完成了该集群的布局与布线。TeraPool 可提供 0.27–0.74 TFLOP/s 的单精度性能，以及最高 1 TFLOP/s 的半精度性能；在实际应用内核上，其能效达到 23 GFLOP/s/W 至 200 GFLOP/s/W。这项工作表明，共享一级缓存存储器范式能够扩展至千核规模，同时保持物理可行性，提供强大的并行计算能力。统一地址的共享一级缓存存储器可容纳数兆字节的数据块，同时消除集群间全局互连的成本，最大限度地减少数据传输、拆分与合并以及同步开销。

表5 最先进水平（SoA）基于集群的众核设计比较

		处理单元（PE）	数据类型	执行模型	每个集群的处理单元数量	处理单元总数	共享一级缓存（MiB）	一级/二级缓存互连带宽（字节/周期）	一级缓存延迟（周期）	峰值浮点运算/周期/集群[a]	扩展性拓扑结构	Open来源
	本工作	32位 RISC-V	Int32/16-SIMD; FP32/16-SIMD; 复数32位 (16位实部与虚部)	**SPMD**	**1024**	1024	**4**	**4096/1024**	**1-11** （非统一内存访问）	**2048**	**纵向扩展 交叉开关**	**Yes**
Multi-core (No.PEs <100)	**Kalray MPPA3-80 [32]**	64位超长指令字	Int16/8; FP64/32/16	单程序多数据; LWI	16	80	3.8	64/1024	23	32	横向扩展 二维网格片上网络	No
	Ramon RC64 [10], [33]	32位超长指令字	Int32/16/8	MIMD	64	64	3.8	1024/N.A.	N.A.	128	纵向扩展 交叉开关	No
Many-core (No.PEs >100)	**TensTorrent [34] Wormhole**	32位 RISC-V	FP32/16/8	SIMD	5	400	1.43	20/不适用	$>4^b$	10	横向扩展 二维网格片上网络	No
	Esperanto ET-SoC-1 [9]	64位向量扩展	8位整数/不适用; 32位浮点数/16	SIMD	32	1088	3.8	256/32	N.A.	64	横向扩展 二维网格片上网络	No
	英伟达 H100 [14]	64/32位PTX	8位整数；64位浮点数/16/8; 脑浮点16；张量浮点32	SIMT	128 (SM)	18432	0.244 (SM)	128/不适用	16.57[c] （平均）	1736	横向扩展 数据驱动	No
	锤子[d] 刀片 [35]	32位 RISC-V	Int32/-; 32位浮点数/-	SPMD	128 (Cell)	2048	0.5 (Cell)	512/不适用	4 + 2 × 跳数	256	横向扩展 二维蜂窝片上网络	**Yes**
	Occamy [23]	64位 RISC-V 整数	64/32/16-单指令多数据; FP64/32/16/8-单指令多数据	SPMD	8	432	0.125	32/256	1	32	横向扩展 交叉开关	**Yes**
	MemPool [16]	32位 RISC-V	Int32/16-SIMD	SPMD	256	256	1	1024/256	1-5 （非统一内存访问）	512	纵向扩展 交叉开关	**Yes**

a 在32位运算中，一次乘加运算计为两次运算。b 一条4周期互连流水线，核心延迟未知。c 基准测试参考自 https://chipsandcheese.com/p/nvidias-h100-funny-l2-and-tons-of-bandwidth，最后访问日期为2025年2月28日。d 我们假设每个HB *Cell* 对应一个*Tile* 组 作为共享内存集群，其中组 暂存存储器地址空间在所有*Tile*之间共享。

表6 数据传输成本与计算IPC对比

	集群的最大分块（MiB）	本地数据分布 内核（单精度AXPY） 主存传输 字节/FLOP	本地数据分布 内核（单精度AXPY） 计算IPC	全局数据分布** 内核（单精度矩阵乘法） 主存传输 字节/FLOP	全局数据分布** 内核（单精度矩阵乘法） 计算IPC
规模扩展 超大型集群 (**本研究***)	**4.00**	**6.00**	**0.85**	**0.009**	0.70
规模扩展 大型集群 (**MemPool [16]***)	1.00	6.00	0.85	0.016	0.88
扩展规模 多个集群 (**Occamy [23]***)	0.125	6.00	0.85	0.062	0.89

* 处理单元、事务表和指令缓存的配置相同。** 主存中的问题矩阵为 8 MiB；输出矩阵基于分块。

致谢

本研究获得瑞士联邦教育、研究与创新国务秘书处（SERI）通过 SwissChips 计划提供的资助。

参考文献

[1] J. Sevilla、L. Heim、A. Ho、T. Besiroglu、M. Hobbhahn 和 P. Villalobos，《机器学习三个时代的计算趋势》，载于 国际联合神经网络会议，意大利帕多瓦：IEEE，2022年7月，第 1–8 页。[2] X. Zhu、J. Li、Y. Liu、C. Ma 和 W. Wang，《大语言模型模型压缩综述》，计算语言学协会学报，第 12 卷，第 1 期，第 1556–1577 页，2024年11月。[3] B. Peccerillo、M. Mannino、A. Mondelli 和 S. Bartolini，《硬件加速器综述：分类体系、趋势、挑战与展望》，系统架构杂志，第 129 卷，第 1–51 页，2022年8月，文章编号 102561。[4] K. Rupp，《50 年微处理器趋势数据》，GitHub 仓库，2022, [在线]。网址：https://github.com/karlrupp/ microprocessor-trend-data。[5] G. Singh 等，《近存计算：过去、现在与未来》，微处理器与微系统，第 71 卷，第 1–15 页，2019年8月，文章编号 102868。[6] R. Muralidhar、R. Borovica-Gajic 和 R. Buyya，《高能效计算系统：从架构、抽象与建模到技术与标准》，*ACM* 计算调查，第 54 卷，第 11 期，第 1–37 页，2022年9月。

[7] M. Horowitz，《计算的能源问题（以及我们能为此做些什么）》，载于*IEEE*国际固态电路会议技术论文集，美国加利福尼亚州旧金山：IEEE，2014年2月，第10–14页。[8] M. Ishii、J. Detrey、P. Gaudry、A. Inomata 和 K. Fujikawa，《在 Kalray MPPA-256处理器上执行快速模运算以实现节能的 ECM》，*IEEE*《计算机汇刊》，第66卷第12期，第2019–2030页，2017年5月。[9] D. R. Ditzel等，《在 Esperanto 的 ET-SoC-1芯片上使用逾1,000个 RISC-V/张量处理器加速机器学习推荐》，*IEEE*《微型》，第42卷第3期，第31–38页，2022年1月。[10] R. Ginosar等，《基于 Ramon Space RC64 的人工智能/机器学习推理引擎》，载于欧洲星载数据处理研讨会。线上活动：ESA-ESTEC，2021年6月，第1–33页。[11] S. Lie，《数百万核、多晶圆人工智能集群》，载于*IEEE Hot Chips* 研讨会，美国加利福尼亚州帕洛阿尔托：IEEE，2021年8月，第1–41页。[12] W. Lang、S. Harizopoulos、J. M. Patel、M. A. Shah 和 D. Tsirogiannis，《迈向节能数据库集群设计》，*ACM VLDB* 会议录，第5卷第11期，第1684–1695页，2012年7月。[13] NVIDIA公司，《NVIDIA A100张量核心GPU架构》，NVIDIA公司，技术报告，2020, [在线]。获取地址：https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf。[14] ——，《NVIDIA H100张量核心GPU架构》，NVIDIA公司，技术报告，2023, [在线]。获取地址：https://resources.nvidia.com/ en-us-tensor-core。[15] H. Luan 和 A. Gatherer，《多端口分布式共享内存架构的组合学与几何学》，载于*IEEE/ACM*国际片上网络研讨会。德国汉堡：IEEE，2020年11月，第1–6页。[16] S. Riedel、M. Cavalcante、R. Andri 和 L. Benini，《MemPool：一种具有低延迟共享一级缓存的可扩展众核架构》，*IEEE*《计算机汇刊》，第72卷第12期，第3561–3575页，2023年8月。[17] Y. Zhang、S. Riedel、M. Bertuletti、A. Vanelli-Coralli 和 L. Benini，《TeraPool-SDR：面向下一代开源软件定义无线电、具备1.89 TOPS、1024个RV核心及4 MiB共享一级缓存的集群》，载于大湖区*VLSI*研讨会。美国佛罗里达州克利尔沃特：ACM，2024年6月，第86–91页。[18] L. Steiner、M. Jung、F. S. Prado、K. Bykov 和 N. Wehn，《DRAMSys4.0：用于深入分析 DRAM 的开源仿真框架》，《国际并行编程杂志》，第50卷第2期，第217–242页，2022年4月。[19] H. T. Kung，《均衡计算机架构的存储器需求》，《*SIGARCH*计算机体系结构新闻》，第14卷第2期，第49–54页，1986年5月。[20] J. Ma、G. Yan、Y. Han 和 X. Li，《用于估算异构众核处理器横向扩展与纵向扩展功耗效率的分析框架》，*IEEE*《计算机汇刊》，第65卷第2期，第367–381页，2016年4月。[21] J. Dean 和 L. A. Barroso，《规模化中的长尾》，《*ACM*通讯》，第56卷第2期，第74–80页，2013年2月。

[22] R. M. Radway、A. Bartolo、P. C. Jolly等，《利用网络化计算芯片实现大型片上存储器以进行神经网络推理》，《自然·电子学》，第4卷第1期，第71–80页，2021年1月。[23] P. Scheffler等，《Occamy：用于12纳米鳍式场效应晶体管工艺下8至64位稠密与稀疏计算的432核、双芯粒、双HBM2E、768 DP-GFLOP/s RISC-V系统》，*IEEE*《固态电路杂志》，第1–15页，2025年1月，提前发表。[24] X. Li、Z. Li、Y. Ju、X. Zhang、R. Wang 和 W. Zhou，《COP：暗硅中众核系统的组合优化功耗预算方法》，*IEEE*《计算机汇刊》，第72卷第5期，第1356–1370页，2023年10月。[25] X. Zhao、S. Ma、Z. Wang、N. E. Jerger 和 L. Eeckhout，《CD-Xbar：面向高性能图形处理器的汇聚-发散交叉开关网络》，*IEEE*《计算机汇刊》，第68卷第9期，第1283–1296页，2019年3月。[26] L. Bertaccini、G. Paulin、M. Cavalcante、T. Fischer、S. Mach 和 L. Benini，《RISC-V核心上的迷你浮点数：混合精度短点积的ISA扩展》，*IEEE*《新兴计算主题汇刊》，第12卷第4期，第1040–1055页，2024年2月。[27] S. Ristov、M. Gusev 和 G. Velkoski，《采用分块技术进行矩阵乘法的最佳模块大小》，载于国际信息与通信技术、电子与微电子大会。克罗地亚奥帕蒂亚：IEEE，2014年7月，第295–300页。[28] T. Benz等，《高性能、能效高的模块化DMA引擎架构》，*IEEE*《计算机汇刊》，第73卷第1期，第263–277页，2024年1月。[29] W. Choi等，《面向异构众核系统上深度卷积网络高效训练的片上通信网络》，*IEEE*《计算机汇刊》，第67卷第5期，第672–686页，2018年11月。[30] J. Diaz、C. Mu˜noz-Caro 和 A. Ni˜no，《多核与众核时代的并行编程模型与工具综述》，*IEEE*《并行与分布式系统汇刊》，第23卷第8期，第1369–1386页，2012年1月。[31] J. Kepner等，《GraphBLAS的数学基础》，载于*HPEC*。美国马萨诸塞州沃尔瑟姆：IEEE，2016年12月，第1–9页。[32] B. D. de Dinechin，众核架构的定性方法。美国新泽西州霍博肯：Wiley，2021年4月，第2章，第27–51页。[33] R. Ginosar、P. Aviely、T. Israeli 和 H. Meirov，《RC64：高性能抗辐射加固众核处理器》，载于 *IEEE* 航空航天会议。美国蒙大拿州大天空：IEEE，2016年3月，第1–9页。[34] J. Vasiljevic、L. Bajic、D. Capalija、S. Sokorac、D. Ignjatovic 和 D. 等，Thompson，《软件2.0的计算基底》，*IEEE*《微型》，第41卷第2期，第50–55页，2021年3月。[35] D. C. Jung、M. Ruttenberg、P. Gao、S. Davidson、D. Petrisko 和 M. B. 等，Taylor，《可扩展、可编程且稠密：HammerBlade开源RISC-V众核处理器》，载于*ISCA*。阿根廷布宜诺斯艾利斯：ACM/IEEE，2024年8月，第770–784页。[36] Z. Jin等，《揭示真实图形处理器片上网络特性：对互连架构的启示》，载于*IEEE/ACM*国际微体系结构研讨会。美国得克萨斯州奥斯汀：IEEE，2024年11月，第885–898页。



Yichao Zhang 于2015年获得中国黑龙江大学工学学士学位，并于2017年获得新加坡南洋理工大学理学硕士学位。他曾在楷登电子和新加坡联发科技从事物理超大规模集成电路设计工作，直至2021年。他目前正在贝尼尼教授的数字电路与系统组攻读博士学位。他的研究兴趣包括物理上可行的众核并行计算架构以及单指令多数据处理。



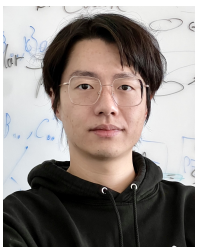
MarcoBertuletti在意大利米兰的米兰理工大学获得电气工程理学学士和理学硕士学位。目前，他正在贝尼尼教授的数字电路与系统组攻读博士学位。他的研究兴趣包括面向下一代电信的RISC-V 处理器多核与众核集群设计。



ChiZhang 于2019年在中国华中科技大学获得理学学士学位，并于2022年在瑞典皇家理工学院获得理学硕士学位。目前，他正在贝尼尼教授的数字电路与系统组攻读博士学位。他的研究兴趣包括高性能计算、存储系统和近存计算。



SamuelRiedel 分别于2017年和2019年在苏黎世联邦理工学院获得电气工程与信息技术理学学士和理学硕士学位。他目前正在贝尼尼教授的数字电路与系统组攻读博士学位。他的研究兴趣包括计算机体系结构，重点关注众核系统及其编程模型。



DiyouShen 于2021年获得科罗拉多大学博尔德分校理学学士学位，并于2024年获得苏黎世联邦理工学院电气工程与信息技术理学硕士学位。他目前正在贝尼尼教授的数字电路与系统组攻读博士学位。他的研究兴趣包括高性能计算和容错互连。



Bowen Wang 于2021年获得哈尔滨工业大学理学学士学位，并于2024年获得苏黎世联邦理工学院电气工程与信息技术理学硕士学位。目前，他正在贝尼尼教授的数字电路与系统组攻读博士学位。他的研究兴趣包括计算机体系结构、灵活的众核系统和神经网络部署。



Alessandro Vanelli-Coralli 是博洛尼亚大学正教授，也是苏黎世联邦理工学院的高级科学家。他的研究工作侧重于无线通信，特别是卫星通信。他参与了卫星移动通信系统方面的国家及国际研究项目，并担任多个由欧洲航天局和欧盟委员会资助项目的主承包商。自2010年起，Vanelli-Coralli博士一直担任IEEE ASMS会议的联合主席。



Luca Benini 担任苏黎世联邦理工学院数字电路与系统讲席教授，同时也是博洛尼亚大学正教授。他获得了斯坦福大学博士学位。他的研究兴趣包括高能效并行计算系统、智能传感微系统和机器学习硬件。他是ACM会士、欧洲科学院成员，也是意大利工程与技术科学院成员。