XDMA: 一种用于异构多加速器 SoC 中布局灵活数据传输的分布式可扩展 DMA 架构

Fanchen Kong*, Yunhao Deng*, Xiaoling Yi, Ryan Antonio, Marian Verhelst
MICAS-ESAT, KU Leuven, Belgium
{fanchen.kong, yunhao.deng, xiaoling.yi, ryan.antonio, marian.verhelst}@esat.kuleuven.be

摘要—随着现代人工智能工作负载越来越多地依赖异构加速器，确保加速器内存之间具有高带宽和布局灵活的数据传输已成为一个紧迫的挑战。直接内存访问（DMA）引擎承诺实现数据移动时的高带宽利用，但通常仅对连续内存访问最优化，因此需要额外的软件循环来转换数据布局。这反过来又会导致过高的控制开销和片上互连的利用率低下。为了克服这种低效性，我们提出了 *XDMA* ，这是一种分布式且可扩展的 DMA 架构，能够实现具有高链路利用率的布局灵活的数据传输。我们引入了三个关键创新点：(1) 数据流引擎作为 *XDMA* 前端，用硬件地址生成器替代软件地址生成器；(2) 一种分布式的 DMA 架构，最大化链路利用并分离配置与数据传输；(3) 用于 *XDMA* 的灵活插件，允许在数据传输过程中即时处理数据。*XDMA* 在合成工作负载中展示了比软件实现高出 151.2×/8.2× 的链路利用率，并在实际应用中实现了平均 2.3× 倍的速度提升。我们的设计相比于最先进 DMA 解决方案增加了 <2%的面积开销，同时消耗了系统功率的 17%。*XDMA* 证明了共同优化内存访问、布局转换和互连协议是解锁异构多加速器 SoC 性能的关键。

***Index Terms*—DMA，多核 SoC，异构系统**

I. 介绍

计算性能需求的增长和硅技术的进步推动了将多个异构加速器集成到单一系统级芯片（SoC）[1], [2] 中，以在计算密集型任务中实现更高的性能和能效。这些加速器包括广义矩阵乘法（GeMM）加速器 [3]、内存计算（IMC）[4]、稀疏数据加速器 [5] 以及安全协处理器 [6]。为了实现高能效并避免停滞，这些加速器通常使用专用的存储子系统。然而，在实践中，尽管在存储子系统和加速器之间的数据访问已经得到了高度优化，但不同加速器之间数据交换被忽视了，这限制了异构 SoC 的整体性能。跨异构加速器的数据复制提出了两个相互关联的挑战：（1）现代工作负载由于缺乏数据重用而越来越受到内存容量的限制；（2）内存中的数据布局必须与加速器的各种访问模式对齐，如

该项目部分资金来自欧洲研究理事会（ERC）的资助协议编号 101088865，欧盟的地平线 2020 计划（CONVOLVE）的资助协议编号 101070374，弗拉芒人工智能研究项目以及鲁汶大学。

*两位作者对这项研究做出了同等贡献。

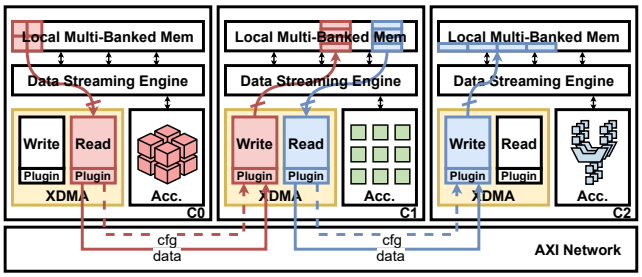


图 1: *XDMA* 在多加速器 SoC 中移动数据

GeMM 需要分块布局、SIMD 需要行优先布局等。次优的布局会导致推理延迟比针对特定加速器优化的格式 [7] 增加高达 100× 倍，因为显式数据布局转换在能源和延迟方面是昂贵的。

直接内存访问（DMA）引擎是实现高带宽数据在内存之间传输的关键组件。然而，传统的 DMA 只能复制连续的数据序列。因此，布局转换只能通过软件控制循环来实现，这会带来显著的控制开销。一种可能的缓解方法是将布局变换卸载到独立的加速器上，使 DMA 能够保持突发传输。然而，这种方法为中间数据引入了额外的延迟和能量成本，削弱了加速器专业化的益处。为了克服这种低效性，我们提出了 ***XDMA***[1]（其操作机制如图 1 所示），这是一种可扩展的 DMA 架构，统一了高利用率内存传输和高效的数据布局转换。

本工作的主要**贡献**是：

- 我们设计了一种具有分离读/写端口的分布式 DMA 架构，通过两阶段电路交换协议进行通信，绕过 AXI 限制以维持高链路利用率。
- 我们用硬件解决方案替换了软件管理的循环，实现了带最小带宽惩罚的 N 维仿射地址生成。
- 我们设计了标准化和灵活的 *XDMA* 插件以支持实时计算和布局转换。

[1]***XDMA 前端*** 是作为 **零食** 的一个组件开源的，而 ***XDMA 后端*** 则单独开源。

	Architecture	Technology	Address Gen	Data Access	Comp-while-transfer	Open-Sourced
HyperDMA [8]	Distributed	RTL	ND	Direct (Coarse-grained)	None	No
ESP DMA [1]	Monolithic	FPGA	1D	Through interconnect	None	Yes
Gemmini DMA [9]	Monolithic	FPGA, Silicon	2D	Through interconnect	Transpose, Scaling	Yes
IDMA [10]	Monolithic	FPGA, Silicon	Optional ND	Through interconnect	In-stream Acc. port	Yes
AMD DMA v7.1 [11]	Monolithic	FPGA	Optional 2D	Through interconnect	None	No
TI EDMA3 [12]	Monolithic	Silicon	3D	Through interconnect	None	No
XDMA	**Distributed**	**FPGA, Silicon**	**ND**	**Direct (Fine-grained)**	**Flexible Plugins**	**Yes**

表 I: *XDMA* 与 SoTA DMA 架构的比较

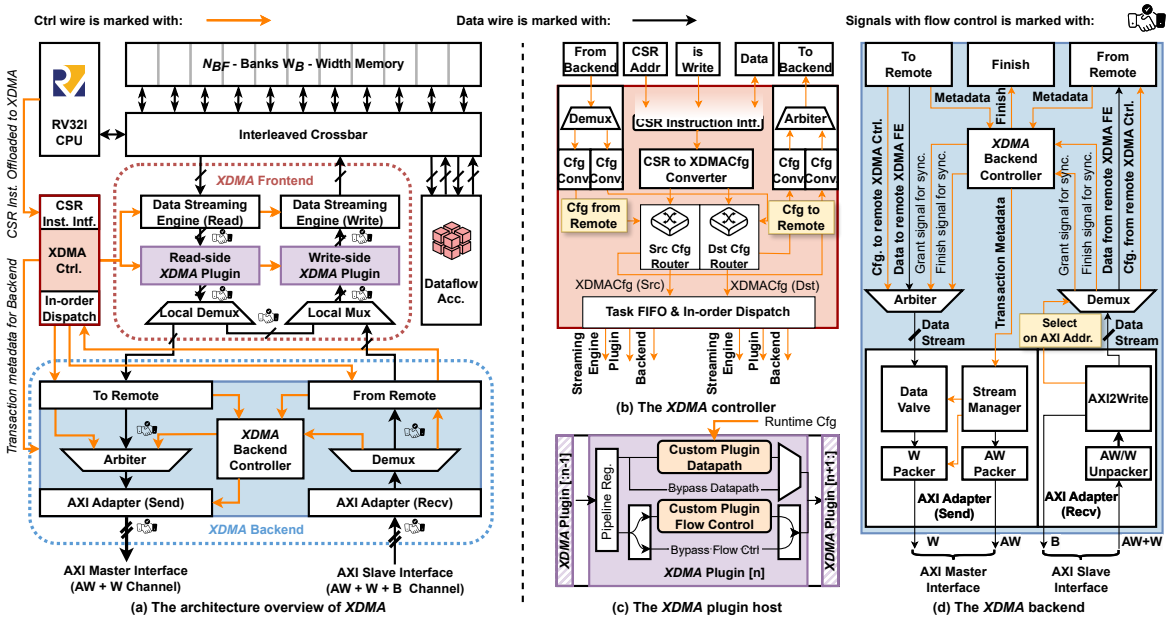


图 2: *XDMA* 的架构（a）和三个选定的子模块（b, c, d）

Parameters	Symbol	Parameters	Symbol
Mem. Base Addr.	$Addr_{Mem}$	Mem. Width	W_B
Mem. Size	$Size_{Mem}$	AXI Width	W_{AXI}
Src./Dst. Buf. Depth	$D_{Buf,src/dst}$	Src./Dst. Dim.	$Dim_{src/dst}$
Src./Dst. #Channel	$N_{C,src/dst}$	Src./Dst. Ext. List	$Ext_{src/dst}$

表 II: 设计时参数为 *XDMA*

- 我们使用合成和真实工作负载验证了 *XDMA* ，相较于软件循环基线实现了 8.2× 到 151.2× 的改进，并且平均链路利用率比加速器+DMA 设计高出 2.7×。*XDMA* 的面积开销比最先进的 DMA 少于 2%，系统能耗占 17%。

表 I 比较了 *XDMA* 架构与学术界和工业界的最新 DMA。

II. *XDMA* 架构

XDMA 提出了一种新颖的分布式 DMA 架构。图 2(a) 显示了 *XDMA* 的硬件层次结构。*XDMA* 控制器（§II-B）接收指令并将配置转发到本地或远程的 *XDMA* 单元。*XDMA* 数据路径（§II-C）包括三个构建模块：(1) 前端与内存接

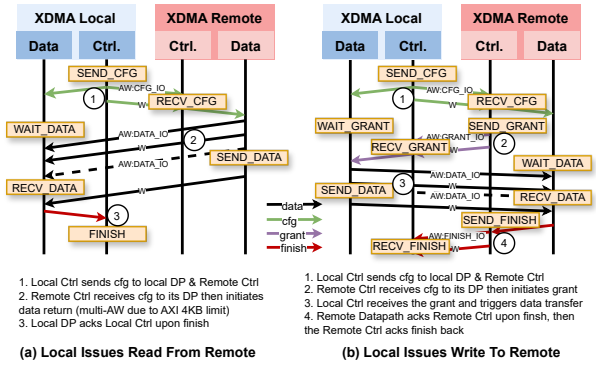


图 3: 读取和写入请求的 *XDMA* 编排

口，提供灵活的内存访问；(2) 插件提供了一个标准接口，用于集成在传输过程中处理数据的自定义模块；(3) 后端将数据封装成 AXI 帧，并管理两个半 *XDMA*s 单元之间的隧道。*XDMA* 的参数化设计使其能够轻松集成到各种 SoC 中。表 II 详细列出了 *XDMA* 的设计时参数。

A. *XDMA* 编排

两个半的 *XDMA*s 之间的协调遵循电路交换原则，包括一个两相流，如图 3 所示：首先是一个 **CFG 传输阶段**，交易 CFG 被转发到远程对等方，然后是一个**数据传输阶段**，其中链路完全被数据占用。*XDMA* 采用分布式架构，每个单元都具有主端口和从端口，这与仅使用主端口的传统 DMAs 区别开来。这使得 *XDMA* 能够将所有传输封装到 AXI 写请求中，简化设计的同时保持全双工传输能力。虽然 CFG 和数据事务共享 AW/W 通道，但由于所有事务都是一对一的，并且不同 *XDMA* 事务之间没有依赖关系，因此不会发生死锁。此外，如果出现争用情况，*XDMA* 可以根据 FIFO 原则自主仲裁任务。

B. *XDMA* 控制器

XDMA 控制器（图 2(b)）将卸载的 CSR 指令转换为 `XDMA 配置结构`来描述一个 *XDMA* 任务。然后，两个配置路由器将 `XDMA 配置路由`到连接到正确内存区域的 *XDMA* 。如果这项任务需要两个 *XDMA* 的合作，两个路由器会通过 AXI 互连将 CFG 转发到远程侧。最后，源（Src.）和目标（Dst.）配置到达任务 FIFO 和顺序调度单元，该单元监视前端的状态，并在前一个任务完成后调度新任务。

C. *XDMA* 数据路径

XDMA 前端（如图 2(b) 所示）以 N-D 线性模式访问本地内存，并为 *XDMA* 后端预取数据。我们利用了专为数据流加速器 [13] 设计的数据流引擎，该引擎包括一个 Dim 维的地址生成器和一个深度为 D_{buf} 的数据缓冲区。地址生成器有效地将地址计算任务从处理器中卸载出来，而数据缓冲区则减轻了在转换不同数据布局过程中可能发生的银行冲突。

XDMA 后端（图 2(d)）作为前端和 AXI4 互连之间的兼容层。它通过将信号（`配置`，`数据`，`资助`，`完成`）映射到独立的 MMIO 地址并发起写请求到对方的 MMIO，在两个 *XDMA*s 单元上基于 AXI 协议建立虚拟隧道。每个后端既是 AXI 主设备也是从设备，因此每两个配对可以独立协作。

XDMA 通过自定义插件在本地和远程数据传输过程中实现即时数据操作，这些插件可以插入到 *XDMA* 前端中。两个插件主机，一个后读取器和一个前写入器，共享统一的架构，如图 2（c）所示。一个或多个插件可以级联，并且每个插件都可以有自己的控制位向量。

III. *XDMA* 评估

我们评估了在多加速器系统芯片中 *XDMA* 的性能。我们首先分析了 *XDMA* 高效转换数据布局（§III-B）的能力。为了验证 *XDMA* 对实际工作负载的适应性，我们随后在 AMD Versal™ VPK180 FPGA 上原型化了这款 *XDMA* 连接的 SoC，并展示了其在 DeepSeek-V3 [14] 大语言模型（§III-C）KV 缓存预填/加载工作负载中的有效性。最后，我们将设计综合为 ASIC 实现以获得面积和功耗结果（§III-D）。我们改变关键的设计时参数 $D_{buf,src/dst}$，该参数影响性能-面积权衡，从 3 到 9，以展示此参数如何影响 *XDMA* 设计。

A. 系统环境设置

我们将一个 4MB、32 银行、每银行 64 位的内存，两个 RV32I 核心 [15]，一个加速器和一个 *XDMA* 组成一个加速器集群。由于本文不专注于评估加速器的性能，我们附加了一个基本的 8×8×8 GeMM 主要用于面积估计。我们设置了一个源自 Occamy [16] 的双簇 SoC 来评估 *XDMA*s ，AXI 互联的宽度配置为 512 位。`数据大师` [13] 被选作 *XDMA* 的数据流引擎。我们的基线是 iDMA [10] 和 Gemmini 内置的 DMA [9]，它们分别代表最先进的通用 DMA 和工作负载最优的 DMA。

所有 RTL 仿真均使用 Verilator 进行。硅综合使用 Synopsys Design Compiler® 与 GF 22nm FDX™ 技术在 1GHz 0.8V 下执行。功耗分析通过合成网表和门级切换活动使用 Synopsys PrimeTime® 进行。

B. 矩阵布局变换

我们将各种 4D 数据布局转换和数据复制工作负载的平均链路利用率在不同的硬件/软件配置下进行了比较。我们选择了四种数据布局：MN、MNM8N8、MNM8N16 和 MNM8N32，这些是 2D/3D GeMM 数组的最佳数据布局。矩阵大小从 32×32 到 512×512。如图 4 所示，评估了六种不同的硬件/软件配置，共产生了 768 个测试点。

每个测试的有效带宽是通过传输的总数据量除以测量的执行时间计算得出。然后通过将有效带宽除以理论带宽来计算链路利用率。对于软件管理的 DMA 配置（①，②），地址计算发生在数据移动之前。

结果显示 *XDMA*9 (⑥) 达到了最高且最稳定的链路利用率。所有硬件加速的解决方案（③-⑥）远远优于软件循环方法（①，②），证实了我们在 §I 中的论点，即软件控制可以造成性能瓶颈。在软件管理的方法中，Gemmini [9]DMA 因为 Rocket 内核的更高 I/O 性能而优于 iDMA [10]，减少了控制开销。解决方案 ③ 改进了前两种方法，但由于中间结果导致了额外的内存开销。一般来说，*XDMA*9 (⑥) 分别平均超过 SoTA 解决方案（①②③）151.2×/8.2×/2.4×。

当比较 *XDMA* 与不同的 D_{buf}（④-⑥）时，*XDMA*9（⑥）在平均值上分别比 *XDMA*3 和 *XDMA*5 高出 1.7× 和 1.1×。

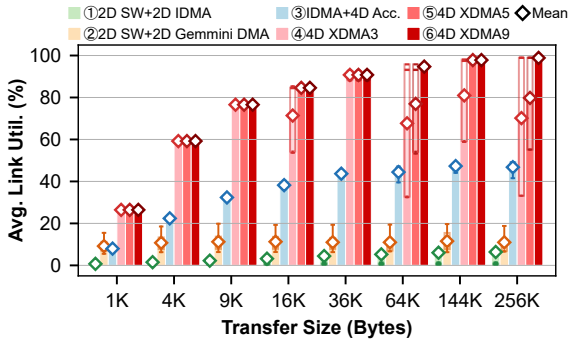


图 4: 4D 矩阵重塑的平均链路利用率：①2D 软件控制循环 + 2D iDMA 复制，②2D 软件控制循环 + 2D Gemmini DMA 复制，③iDMA 复制 + 专用 4D 布局转换加速器，以及 ④⑤⑥ 4D*XDMA* 使用 $D_{buf,src/dst} = 3, 5, 9$(*XDMA*3/5/9)

实验	形状	输入/输出布局	操作	#分类准确率比率
Prefill 1	2048×512	MNM8N8/MN	Reshape	38542 / 2.34×
Prefill 2	2048×512	MN/MNM8N8	Reshape	42934 / 2.60×
Load 1	2048×512	MNM8N8	Transpose	37509 / 2.28×
Load 2	4096×512	MNM8N8	Transpose	74884 / 2.28×
Load 3	8192×512	MNM8N8	Transpose	149639 / 2.28×

表 III: KV 缓存预填/加载评估的 *XDMA*

我们观察到具有较小 D_{buf} 的设置变化更大，因为较小的缓冲区无法始终防止由银行冲突引起的停顿。所有剩余的测试都在 *XDMA*9 上进行以实现最大性能。

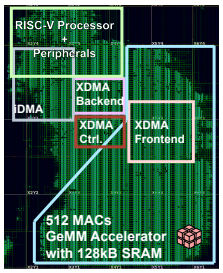
C. 实际工作负载评估在 FPGA 上

我们在 VPK180 FPGA 上实现了评估集群。图 5 详细描述了带有注释的 FPGA 布局、运行频率和资源利用率，针对一个具有 *XDMA* 特征的 8×8×8 GeMM 加速器集群，显示 *XDMA* 引入了大约 8%的面积开销。

接下来，我们使用 Deepseek-v3 的 KV 缓存矩阵形状 8192×512（Batch=1），代表个人使用场景来基准测试由 *XDMA* 提供的跨集群数据复制性能。评估的工作负载包括：(1) 预填充阶段：集群 1 中的一个 GeMM 加速器（最佳布局：MNM8N8）计算 KV 缓存，随后在集群 2 的一个 SIMD 加速器上进行 RMSNorm 处理（最佳布局：MN）。最后，将处理后的 RMSNorm 数据以 MNM8N8 布局存储到另一个集群；(2) 负载阶段：KV 缓存数据在集群 1 中经过第一个 GeMM 后，传输并同时转置到集群 2 进行转换器操作。表 III 显示，在 *XDMA* 上执行的工作负载与基线设置（iDMA+加速器）相比，延迟改善了 2.3×。

D. 面积和功率评估

最后，我们将该簇与 8 个 ×、8 个 ×、8 个 GeMM 和一个 *XDMA*（具有 128kB 的减小内存尺寸）综合到 ASIC



平台	Versal™ VPK180
频率	100MHz
查找表总数	268k
总注册数	67k
查找表 通用矩阵乘法	150k (56.0%)
Regs GeMM	18k (26.9%)
查找表 输入直接内存访问	8k (2.9%)
寄存器 iDMA	8.8k (13.1%)
查找表 *XDMA*9	20.5k (7.6%)
规章 *XDMA*9	5.9k (8.8%)

图 5: 加速器集群的 FPGA 结果带有 *XDMA*

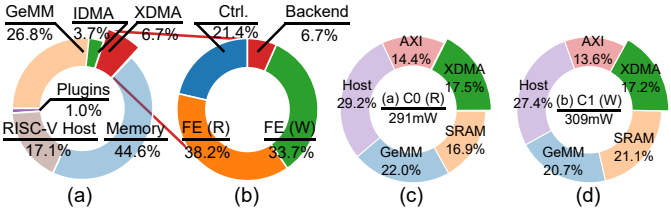


图 6: 评估集群 *XDMA* 的面积和功率分解：(1)(2) 面积细分；(3)(4) 数据从集群 0 复制到集群 1 时的功率细分。

实现中。*XDMA* 占据加速器集群区域的 6.7%，并在执行 1D 内存复制任务时消耗总集群功率的 17%（图 ~ 6）。

IV. 结论

在本文中，我们提出了 *XDMA* ，这是一种分布式且可扩展的 DMA 架构，旨在高效地进行异构加速器之间的内存布局转换。*XDMA* 在非连续数据复制任务中比 SW 管理的解决方案提供了 8.2×-151.2× 的改进。我们还展示了在 FPGA 和硅技术上的实现结果。与 iDMA 相比，*XDMA* 面积开销小于 2%，并且消耗了加速器集群总功耗的 17%。

参考文献

[1] M. C. Dos Santos *et al.*, ‘‘14.5 a 12nm linux-smp-capable risc-v soc with 14 accelerator types, distributed hardware power management and flexible noc-based data orchestration,’’ in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67. IEEE, 2024, pp. 262–264.

[2] V. Schmulbach *et al.*, ‘‘Nectar and rasoc: Tale of two class socs for language model interference and robotics in intel 16,’’ in *2024 IEEE Hot Chips 36 Symposium (HCS)*. IEEE Computer Society, 2024, pp. 1–1.

[3] H. Liao *et al.*, ‘‘Davinci: A scalable architecture for neural network computing,’’ in *2019 IEEE Hot Chips 31 Symposium (HCS)*. IEEE Computer Society, 2019, pp. 1–44.

[4] P. A. Hager *et al.*, ‘‘11.3 metis aipu: A 12nm 15tops/w 209.6 tops soc for cost-and energy-efficient inference at the edge,’’ in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67. IEEE, 2024, pp. 212–214.

[5] M. Shi *et al.*, ‘‘Bitwave: Exploiting column-based bit-level sparsity for deep learning acceleration,’’ in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2024, pp. 732–746.

[6] A. Ghosh *et al.*, ''A 334 μw 0.158 mm 2 asic for post-quantum key-encapsulation mechanism saber with low-latency striding toom–cook multiplication,'' *IEEE Journal of Solid-State Circuits*, vol. 58, no. 8, pp. 2383–2398, 2023.

[7] J. Tong *et al.*, ''Feather: A reconfigurable accelerator with data reordering support for low-cost on-chip dataflow switching,'' in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2024, pp. 198–214.

[8] M. Peng *et al.*, ''Hyperdma: Enhancing high-performance computing and ai workflows with advanced data transfer capabilities,'' in *2024 9th International Conference on Integrated Circuits and Microsystems (ICICM)*. IEEE, 2024, pp. 636–644.

[9] H. Genc *et al.*, ''Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration,'' in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 769–774.

[10] T. Benz *et al.*, ''A high-performance, energy-efficient modular dma engine architecture,'' *IEEE Transactions on Computers*, vol. 73, no. 1, pp. 263–277, 2023.

[11] A. Xilinx, ''Axi dma logicore ip product guide,'' 2022.

[12] T. I, ''Enhanced direct memory access (edma3) controller,'' 2015.

[13] X. Yi *et al.*, ''Datamaestro: A versatile and efficient data streaming engine bringing decoupled memory access to dataflow accelerators,'' *arXiv preprint arXiv:2504.14091*, 2025.

[14] A. Liu *et al.*, ''Deepseek-v3 technical report,'' *arXiv preprint arXiv:2412.19437*, 2024.

[15] F. Zaruba *et al.*, ''Snitch: A tiny pseudo dual-issue processor for area and energy efficient execution of floating-point intensive workloads,'' *IEEE Transactions on Computers*, vol. 70, no. 11, pp. 1845–1860, 2020.

[16] G. Paulin *et al.*, ''Occamy: A 432-core 28.1 dp-gflop/s/w 83% fpu utilization dual-chiplet, dual-hbm2e risc-v-based accelerator for stencil and sparse linear algebra computations with 8-to-64-bit floating-point support in 12nm finfet,'' in *2024 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, 2024, pp. 1–2.