Duet：实现处理器与嵌入式 FPGA 的协同

Ang Li 普林斯顿大学 新泽西州普林斯顿 08544，美国 angl@princeton.edu

August Ning 普林斯顿大学 新泽西州普林斯顿 08544，美国 aning@princeton.edu

David Wentzlaff 普林斯顿大学 新泽西州普林斯顿 08544，美国 wentzlaf@princeton.edu

arXiv:2301.02785v1 [cs.AR] 7 Jan 2023

摘要——摩尔定律的消亡推动了硬件加速的发展。然而，专注于对稳定算法进行整体加速，忽视了更广泛领域中大量细粒度加速机会，也浪费了主机处理器的计算能力。

本文介绍 Duet，这是一种可扩展的众核-FPGA 架构，通过非侵入式、双向缓存一致性集成，使嵌入式 FPGA（嵌入式 FPGA）成为与处理器平等的协作伙伴。与现有的 CPU-FPGA 混合系统中处理器仅发挥辅助作用不同，Duet 通过两类制造后增强，充分释放处理器和嵌入式 FPGA 的潜力：细粒度加速，**将应用程序划分为小型任务，并把频繁调用且计算密集的任务卸载到各种小型加速器上，同时利用处理器处理动态控制流和较难加速的任务；**硬件增强，**利用嵌入式 FPGA 仿真的硬件组件提升处理器效率，或减少特定执行模型中的软件开销。**

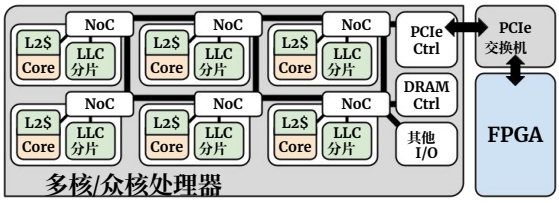
我们开发了 Duet 的寄存器传输级实现，以高保真度评估该架构。使用合成基准测试进行的实验表明，Duet 可将处理器与加速器通信延迟降低高达 82%，并将带宽提升至原来的 9.5 倍。我们还使用七个应用基准测试对该 RTL 实现进行了进一步评估，取得了 1.5 至 24.9 倍的加速比。

I. 引言

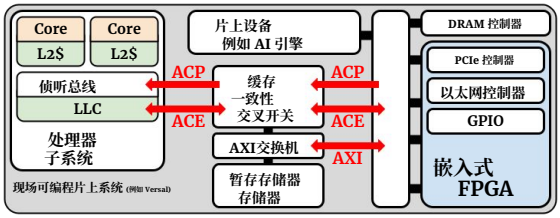
摩尔定律的终结 [33]以及处理器性能增长的停滞 [44]催生了硬件加速 [12], [22], [34]。由于基于 ASIC 的加速器一次性工程成本高、通用性低，现场可编程门阵列（FPGA）凭借制造完成后的门级可重配置能力和接近 ASIC 的性能，正日益受到青睐 [16], [20], [8]。FPGA 既可以集成为独立器件（图1a），也可以集成到现场可编程片上系统（FPSoC）中（图1b）。传统的基于 FPGA 的加速范式是

粗粒度加速（图 2b），将整个算法卸载到 FPGA 上。尽管粗粒度加速能显著提升性能和能效，但它在实际中仅适用于有限的算法：这些算法既要足够稳定（以证明加速器设计成本合理），又要足够庞大（以抵消控制和数据传输开销）。

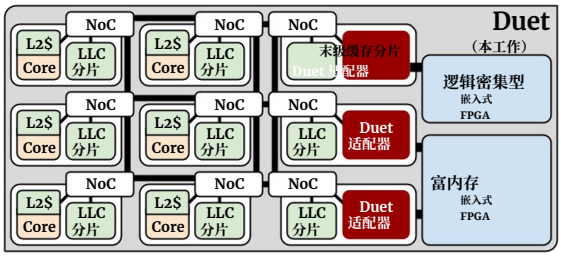
本文提出了 **Duet**（图 1c，第 II节），这是一种新颖的缓存一致性众核 FPGA 架构，旨在支持两种新颖的硬件加速范式。



（a）多核/众核处理器与独立 FPGA二**级缓存**=私有缓存；**片上网络**=片上网络；**末级缓存**=末级缓存



（b）现场可编程片上系统（FPSoC），例如 Versal [19]高级可扩展接口 [3]：非一致性互连；加速器一致性端口 [3]：单向一致性互连；ACE [3]：完全一致性互连。



(c) Duet（本工作）

每个 Duet 适配器都可以独立配置，以支持单向或双向一致性，以及内存映射的非一致性接口。Duet 支持可扩展集成：1）任意数量的处理器；2）多个独立的嵌入式 FPGA（eFPGA）；3）每个 eFPGA 的多个片上网络接入点。

图 1：CPU-FPGA 系统

细粒度加速（图2c，第III-A节）将算法划分为较小的任务，并仅将频繁调用、计算密集型的任务卸载到各种小型加速器上。处理器仍发挥着关键作用，负责处理动态控制流、内存/IO 密集型任务，或其他不易加速的计算。例如，细粒度加速器可用于实现 Tangent 等特殊指令、排序等基本算法，或图遍历过程中每个节点/边对应的计算负载等内层循环体。**硬件**

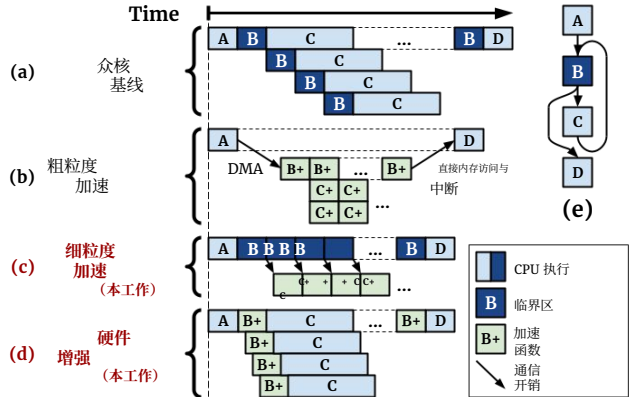


图 2：加速一个假想程序 (a-c) 众核基线与不同加速范式的执行时间；(d) 硬件增强示例，其中嵌入式 FPGA 模拟无锁任务调度器；(e) 程序的控制流图。

增强（图2d，第III-B节）采用与应用程序无关的方法，利用 FPGA 仿真硬件组件来减少处理器空闲时间，或降低特定执行模型中的软件开销。例如，在硬件中实现的无锁数据结构可以降低共享内存多线程程序中的同步开销；硬件任务调度器则能以低于软件调度器的开销实现任务级并行。

现有 CPU-FPGA 系统不适用于细粒度加速和硬件增强，原因有二：首先，当大量核心和加速器通过短而频繁的消息进行通信时，集中式、针对带宽优化的 CPU-FPGA 互连表现不佳；其次，即使在支持处理器与 FPGA 之间双向缓存一致性的最先进 FPGA 片上系统上，以缓存行为粒度的存储器共享也会产生不可忽视的开销。由于计算与内存比极高，这种开销对于粗粒度加速尚可接受，但对于小型且频繁调用的细粒度加速器而言则至关重要。

Duet 专为细粒度加速和硬件增强所提出的独特架构需求而设计。本质上，**Duet 通过新颖、轻量级的 Duet 适配器，使嵌入式 FPGA 与处理器成为平等的参与者，并将其作为一等公民集成到片上网络（NoC）中**。具体而言，Duet 包含以下创新之处：

- **可扩展性**：Duet 支持将一个或多个具有不同资源配置的嵌入式 FPGA 紧密集成到可扩展的众核系统中。每个嵌入式 FPGA 均可连接多个 Duet 适配器，以提高总内存带宽。
- **混合缓存一致性**：Duet 采用混合方案，以一致性方式集成嵌入式 FPGA。每个 Duet 适配器都包含一个或多个私有本地硬件**代理缓存**（第 II-C 节），用于将依赖于平台的缓存一致性协议转换为供嵌入式 FPGA 使用的简单内存接口。此外，每个代理缓存都可以在嵌入式 FPGA 编程时配置，以支持可选的双向一致性软缓存，由嵌入式 FPGA 资源构建而成。软缓存通过利用数据局部性提升加速器性能，同时与处理器一致性地共享数据。
- **非侵入式集成**：对于细粒度加速和硬件增强，仍有相当多的计算任务运行在处理器上，并且对整体性能至关重要。然而，由嵌入式 FPGA 模拟实现的软缓存直接参与缓存一致性，可能会拖慢缓存系统，因为嵌入式 FPGA 的运行时钟频率较低，且会产生跨时钟域开销。为应对这一挑战，Duet 适配器运行在处理器的（高速）时钟域中，并将 FPGA 侧的一致性维护移出关键路径。因此，缓存系统的运行速度与每个 Duet 适配器都作为额外的处理器专属私有缓存时一样快。
- **即插即用集成**：由于 Duet 适配器负责在嵌入式 FPGA 与片上网络之间进行转换，因此 Duet 几乎不需要对现有众核系统进行硬件改动。这种解耦至关重要，因为修改成熟的处理器设计往往会导致性能下降，甚至引发硬件错误；尤其是在先进工艺节点上，设计复杂度和验证成本会急剧攀升 [23]。

为了高保真地评估 Duet，我们构建了 Dolly——一个 RTL 级原型（Verilog 和 SystemVerilog）。此外，我们利用一系列开源项目搭建了完整的软件开发与加速器设计工具链，其中包括 OpenPiton [6],BYOC [5]，PRGA [31]，Yosys [54]，和 VTR [38]。**为了推动对紧密集成的硬件缓存一致性 CPU 与嵌入式 FPGA 系统开展进一步研究，我们已将 Dolly 及其工具链开源，可通过以下网址获取：https://github.com/PrincetonUniversity/Duet**。

实验表明，Duet 适配器为 CPU-eFPGA 集成带来的硬件开销可以忽略不计（第V-B节）。与让 eFPGA 直接参与一致性协议相比，代理缓存可将处理器与加速器之间的通信延迟最多降低 82%，并将带宽最多提升至 9.5 倍（第V-C节）。在不同的 eFPGA 时钟频率下，延迟降低和带宽提升均保持稳定。在所选基准测试中（第V-D节），Dolly 相较于对应的仅处理器基线，整体性能最高提升 24.9 倍；相较于其他 CPU-FPGA 系统，最高提升 4 倍。

本研究的主要贡献如下：

- 提出 Duet，这是一种众核 FPGA 架构，通过采用新颖的 Duet 适配器，以可扩展、非侵入且保持缓存一致性的方式集成嵌入式 FPGA。
- 通过示例指出并展示了 Duet 所支持的两种新型硬件加速范式，即细粒度加速和硬件增强。
- 介绍 Dolly：Duet 的寄存器传输级原型，以及用于软件开发和加速器设计的完整工具链。
- 通过合成基准测试和应用基准测试，评估 Dolly 的芯片面积消耗和性能。
- 发布 Dolly 及其工具链，供开源访问。

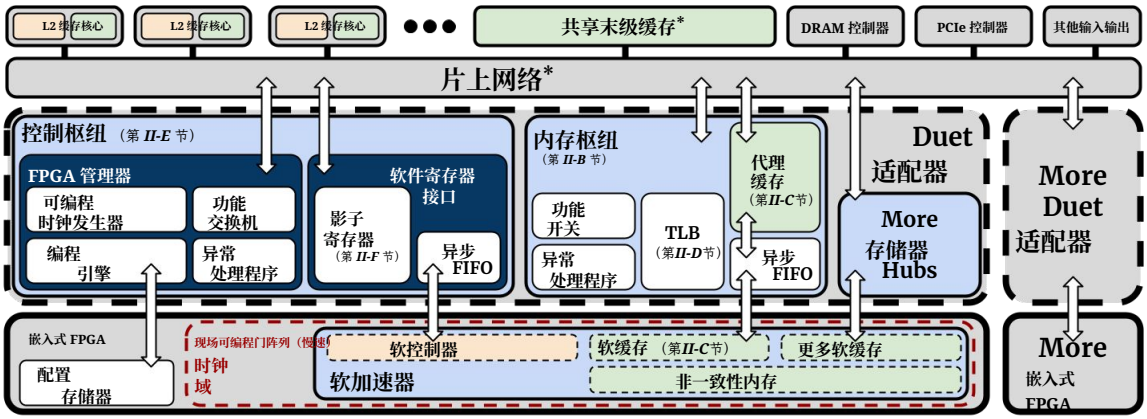


图3：Duet 架构与仿真软加速器 * 为简化起见，图中展示的是基于总线的片上网络和集中式末级缓存。Duet 可适配其他片上网络拓扑和分布式末级缓存。例如，Dolly（第 IV 节）采用二维网格和分布式末级缓存。

II. 架构

在深入介绍 Duet 的架构之前，我们先澄清本文中使用的几个术语：**软件** 指运行在处理器上的程序；**硬件**、**硬** 缓存等，指制造时固定的组件，与利用嵌入式 FPGA 中的可重配置资源模拟的**软** 组件相对。此外，我们使用 **软加速器** 一词，同时指代细粒度加速器和硬件增强组件。

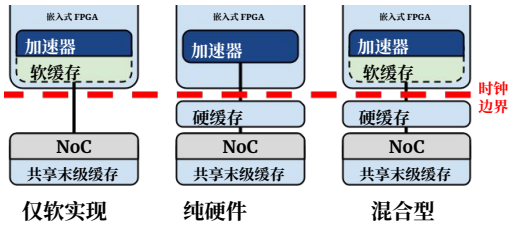


图 4：FPGA 侧缓存组织选项

A. 概述

图 3展示了 Duet 架构的概览。Duet 通过新颖的硬件 **Duet 适配器**，将嵌入式 FPGA 作为片上网络中的一等公民进行集成。每个 Duet 适配器包含一个或多个 **内存枢纽** 以及一个 **控制枢纽**。内存枢纽使软加速器能够进行双向一致性内存访问，并在嵌入式 FPGA 的内存接口与片上网络之间进行转换。控制枢纽将嵌入式 FPGA 呈现为片上设备，可通过内存映射输入/输出（MMIO）访问。

嵌入式 FPGA 仿真的软加速器通常以远低于系统其余部分的时钟频率运行，这给最大限度地减少处理器-加速器通信开销带来了挑战。尤其是，任何进入或离开慢速时钟域的流量都必须承担跨时钟域（CDC）开销，而每个慢速时钟周期都会显著增加总通信时间（图 5a & 图 6a）。几十个周期看似微不足道，但由于加速任务执行时间短且调用频繁，累积的开销可能会大幅侵蚀实际加速比。为应对这一挑战，Duet 适配器采用多种策略，将慢速时钟域中的逻辑移出关键路径。

B. 内存枢纽

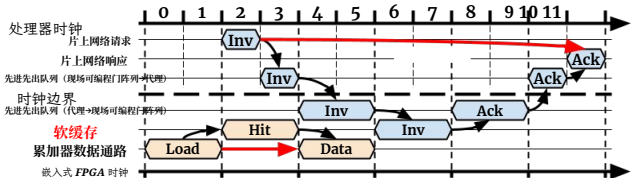
每个 Duet 适配器可以包含多个内存枢纽，每个内存枢纽都通过独立连接接入片上网络。软加速器可以使用全部或任意子集来访问存储器。每个内存枢纽由以下部分组成：异常处理程序、一组功能开关，以及**代理缓存**（第II-C节），均由硬件实现。除了硬件代理缓存外，每个内存枢纽还可以支持一个可选的双向一致性**软缓存**，由嵌入式 FPGA 资源构建。处理器可以通过片上存储器映射输入/输出配置异常处理程序和所有功能开关。

异常处理程序通过超时机制和奇偶校验检查来监控嵌入式 FPGA 的输出。当检测到异常时（例如由寄存器传输级或软件缺陷导致），它会发出错误代码，并停用同一 Duet 适配器中的所有内存枢纽。停用后，内存枢纽将不再接受来自嵌入式 FPGA 的任何内存请求，但代理缓存仍会正常工作，从而确保妥善处理所有正在传输的一致性消息。此机制可防止加速器缺陷在微架构层面导致系统停止运行。

功能开关允许处理器根据嵌入式 FPGA 的状态和软加速器的规格配置内存枢纽。例如，在嵌入式 FPGA 重配置期间，应停用内存枢纽；如果使用软缓存，则应将内存枢纽配置为将失效请求转发至嵌入式 FPGA。

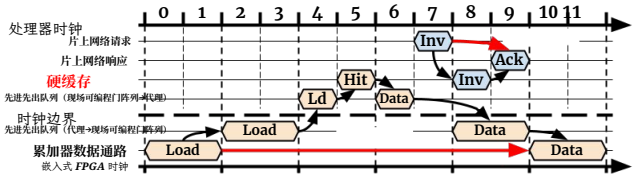
C. 代理缓存

构建一致性集成的 CPU-FPGA 系统时，一个关键挑战是如何组织 FPGA 侧缓存。图4 展示了各种方案：

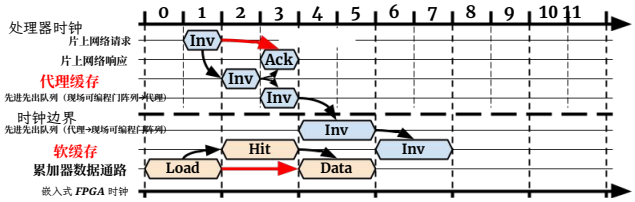


(a) 仅使用**软**缓存时的缓存操作

失效=远程缓存发出的失效；**确认**=确认



(b) 仅使用**硬**缓存时的缓存操作



(c) 使用 Duet 时的缓存操作（本文工作）

图 5：不同缓存组织方式下的缓存操作。为简便起见，本图展示的是单级异步FIFO，且 FPGA 时钟的频率为系统时钟频率的一半。在实际硬件中，异步FIFO通常分为两到四级，FPGA 时钟的频率可能低至系统时钟频率的 1/10。

仅软实现 嵌入式 FPGA 直接连接到片上网络。若使用软缓存，则必须实现系统级缓存一致性协议。**纯硬件** 在嵌入式 FPGA 与片上网络之间加入硬件私有缓存。缓存一致性对嵌入式 FPGA 不可见，因此不支持软缓存。**混合型** 硬件私有缓存代表嵌入式 FPGA 参与系统级缓存一致性，并支持使用本地缓存一致性协议的软缓存。

方案 1，仅软实现存在三个主要缺点。首先，设计符合平台相关的复杂缓存一致性协议的缓存，成了加速器开发人员的负担。复杂的控制逻辑会消耗更多逻辑资源，导致更高的访问延迟，而且很少能在不同设备间复用——为 ACE 设计的软缓存 [3] 几乎无法用于 CHI [4]。其次，由于软缓存能够访问微架构状态（例如，可以无限期阻塞片上网络消息），系统无法完全遏制软缓存中的故障或恶意行为。第三，如图 5a所示，由于时钟域跨越开销以及 FPGA 时钟域中较慢的周期，软缓存可能会拖慢硬件缓存系统。尽管如此，仅软实现仍是大多数商用 FPSoC 的实际默认方案，因为这些以FPGA为中心的架构是为粗粒度加速而设计的。具体而言，一致性内存共享并不常见，且主要用于简化编程；软缓存 IP 可从供应商处获得许可；而且加速器故障通常会导致整个系统失效。

方案 2，纯硬件，解决了仅软实现方案的所有缺点，但也存在其他局限。首先，CDC 开销现在由加速器数据通路承担（5b）。其次，缓存实现在制造时便已固定，对于某些加速器而言可能并非最优。

考虑到仅软实现和纯硬件方案的缺点，Duet 采用了混合型方案。一个私有、本地的硬件**代理缓存**实现了依赖于平台的全系统缓存一致性协议，并向嵌入式 FPGA 提供简单的内存接口。每个代理缓存都可通过功能开关进行配置，以支持由嵌入式 FPGA 模拟的软缓存；该软缓存可以与加速器数据通路紧密集成。此外，与混合方案的朴素实现相比，代理缓存还包含两项关键创新：

首先，**代理缓存既不要求软缓存提供任何确认消息，也不接受此类消息**。因此，代理缓存始终能及时响应一致性消息（图 5c），使缓存系统不受缓慢的嵌入式 FPGA 时钟影响。为维持一致性，代理缓存要求软缓存采用直写方式，但允许使用写缓冲（代理缓存本身则可采用回写或直写方式，具体取决于末级缓存的一致性协议）。此外，由于异步 FIFO 会按顺序传递消息，软缓存接收失效消息、缓存行填充和写确认的顺序，始终与代理缓存发送这些消息的顺序一致。严格证明此类协议的正确性不在本文讨论范围之内，但 OpenSPARC T1 [41] 中的 PCX 协议及其扩展——BYOC 中的 TRI 协议 [5], ——就是满足上述要求的两个示例。

其次，该协议简单而灵活。在常见情况下，软缓存只需支持两种请求类型（**加载** 和**存储**）以及三种响应类型（**加载确认**、**存储确认** 和**失效**化）。是否使用写缓冲区、写缓冲区包含多少个条目，以及写后读转发是否符合应用程序的一致性假设，均由加速器设计者决定。代理缓存可配置为与写分配或不写分配的软缓存配合使用，也可配置为启用原子操作；启用原子操作需要软缓存逐步支持更多消息类型。

D. 内存保护与虚拟化

CPU-FPGA 系统面临的另一项关键挑战是内存保护与虚拟化。代理缓存采用物理索引、物理标记，因为它是在硬件中实现的，且更靠近末级缓存。不过，嵌入式 FPGA 应使用虚拟地址还是物理地址，取决于软加速器的类型。硬件增强组件可能是可信固件，因此可以获准访问物理内存。相反，面向特定应用的细粒度加速器类似于用户程序，可能存在故障或恶意行为，因此最好限制其使用虚拟地址。

为了让软加速器能够进行虚拟内存访问，Duet 为每个

内存枢纽都添加了一个地址转换后备缓冲器（TLB）。如果软加速器获准访问物理内存空间，则可以禁用 TLB；否则，加速器发起的内存访问必须经由 TLB 转换，同时由代理缓存进行推测处理。发生页故障时，TLB 会向处理器发送中断，然后内核级中断处理程序通过存储器映射输入/输出更新 TLB；如果页面访问被判定为无效，则终止加速器。

一种特殊情况是使用软缓存，但加速器只能访问虚拟地址，即软缓存采用虚拟索引、虚拟标记。为使代理缓存向软缓存转发失效操作时，能够从物理地址反向映射到虚拟地址，代理缓存会在每个缓存行的物理标记旁存储虚拟页号。这也排除了软缓存中存在同义别名（不同虚拟地址映射到同一物理地址）的可能性。具体来说，当代理缓存通过不同的虚拟地址收到针对同一物理缓存行的加载请求时，可以先使现有虚拟地址失效，再作出响应。

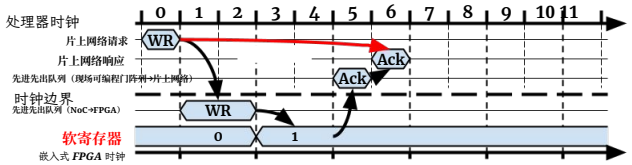
E. 控制枢纽

控制枢纽由两个子模块组成：**现场可编程门阵列管理器** 和**软寄存器接口**。现场可编程门阵列管理器为嵌入式 FPGA 的编程和监控提供必要的硬件支持。编程引擎将位流加载到配置存储器中，并执行完整性检查以检测数据损坏。可编程时钟发生器既可以对系统时钟进行分频，也可以集成一个单独的锁相环，以便更精细地控制 FPGA 时钟的生成。异常处理程序和功能开关与内存枢纽中的类似。具体来说，功能开关可用于设置超时限制、重置软加速器，或清除先前记录的错误代码。

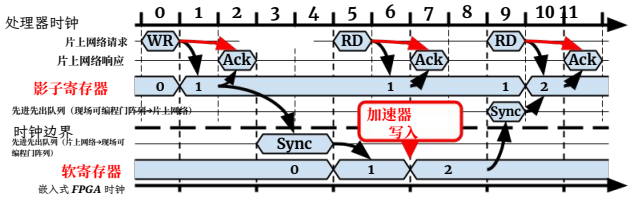
软寄存器接口使软加速器能够实现类似于片外外设上常见的软“设备控制器”。嵌入式 FPGA 仿真的软寄存器可通过片上存储器映射输入/输出访问，而且通常还具有其他作用，而不只是存储数值。例如，读取软寄存器可能会从加速器内部的先进先出队列中取出数据，因此重复读取会返回不同的值。控制枢纽停用时，软寄存器接口会向所有处理器访问返回无效数据，以避免系统停止运行。

F. 影子寄存器

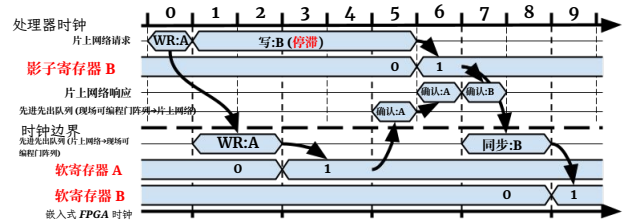
为避免不必要的副作用，存储器映射输入/输出通常遵循严格的内存排序模型，例如输入/输出排序。不幸的是，正如第 II-A 节所强调的，这些严格排序的访问可能会因嵌入式 FPGA 运行时钟较慢而使处理器流水线停顿（图 6a）。为解决这一低效问题，软寄存器接口增设了多种 **影子寄存器**，它们位于快速时钟域（图6b）。当处理器向影子软寄存器写入时，影子寄存器会先确认请求，然后再将写入转发至嵌入式 FPGA。反过来，软加速器会主动



(a) 对普通软寄存器的访问



(b) 对影子软寄存器的访问（本工作）



(c) 两次软件寄存器写入的严格顺序（本工作）图 6：访问软寄存器和影子寄存器

通过异步 FIFO 同步影子软寄存器，因此影子寄存器无需通知嵌入式 FPGA，即可立即响应处理器的读请求。

Duet 支持四种影子寄存器：普通型、面向 *FPGA* 的先进先出队列、面向 *CPU* 的 *FIFO*和令牌 *FIFO*。普通型影子寄存器只保留多次写入中的最后一个值，适合传递常量参数。顾名思义，面向 FPGA 和面向 CPU 的 FIFO 会记录一侧的所有写入，并允许另一侧按顺序读取。面向 CPU 的 FIFO 是阻塞式的，即处理器读取会一直等待，直到软加速器将数据推入 FIFO 或请求超时。相比之下，面向 CPU 的令牌 FIFO 不含数据，是一种非阻塞式 FIFO；响应处理器读取时，它会消耗一个令牌，或者返回“空”。令牌 FIFO 专门用于模拟并行编程模型中的非阻塞 try_join 语义。

请注意，如果软件需要进行必须一直发送到端点的不可缓冲访问，普通软寄存器仍然可用。例如，可以将一个软寄存器专门用作屏障，以同步处理器和嵌入式 FPGA。处理器通过读取软寄存器来表示其已到达屏障，而嵌入式 FPGA 则通过确认此次读取来表示其已到达屏障。为保持输入/输出排序，影子寄存器访问会相对于其他影子寄存器或普通寄存器访问按顺序处理并响应，如图 6c所示。

III. 应用

Duet 支持两种新颖的硬件加速范式，即细粒度加速和硬件增强。本节将通过示例介绍这两种范式，并探讨 Duet 为何是实现它们的理想架构。

清单 1：Barnes-Hut 模拟算法的关键计算循环 [7]。高亮显示的函数是静态的、计算密集型的，非常适合进行细粒度加速。

```
// 计算粒子所受的合力                                    "p"
void计算合力 (BHTreeNode 节点, 粒子 p): if (距离 (节点, p)  >  节点.半径 * THRESHOLD)

      // “p”与子树中的粒子相距足够远，
      // “node”，因此我们用低阶近似来估算合力
      // 多极展开并停止遍历子树
      p.force  +=                    (节点, p);
   否则如果(IsLeafNode (node))
      对于(Particle q : node.children)
         p.force  +=                 (q, p);
   else
      // 通过递归调用遍历子树
      对于(BHTreeNode child : node.children)
         CalculateNetForce（child, p）;
// 并行计算所有粒子所受的合力
voidCalculateNetForceOnAll粒子 (
      BHTreeNode root, vector<粒子>allParticles):
   parallel_for (粒子 p : allParticles)
      CalculateNetForce (root, p);
```

A. 细粒度加速

1) 概述： 从本质上说，细粒度加速与粗粒度加速相似，因为二者都会将计算卸载到专用加速器上。事实上，这两种范式之间并没有明确的边界，而是存在一个连续谱——加速函数越短，或者加速器与处理器交互越频繁，就越接近连续谱中的细粒度一端。

细粒度加速具有以下优势：首先，它通过让处理器执行算法中较难加速的部分，充分利用处理器的计算能力，例如动态控制流、内存/IO 密集型任务、复杂数据结构的管理等；其次，通过减少移至现场可编程门阵列上的逻辑，降低软件和加速器设计成本；第三，与单体式加速器相比，由多个小型加速器组成的集合更易于组合和复用，也更能适应软件更新。

2) 示例：Barnes–Hut（BH）算法 [7] 是一种近似算法，用于模拟通过引力等特定物理力相互作用的粒子动力系统。在每个时间步，模拟区域会被划分为存储在八叉树中的分层立方单元（针对三维空间），以便计算远处单元中的粒子所施加的总力
通过质心抽象或低阶多极展开进行近似。清单1 展示了计算所有粒子合力的关键函数。

先前的工作 [17] 提出了一种基于 FPGA 的粗粒度 BH 加速器，其中包含复制的流水线，用于并行计算多个粒子的力。然而，尽管采用了特殊的遍历缓存来促进流水线之间的数据重用，控制流发散仍常常导致流水线利用率较低。此外，该加速器依赖主机处理器对 BH 树进行序列化，并将节点指针预加载到遍历缓存中，因此当数据集超出 FPGA 的块随机存取存储器容量时便无法使用。总之，若无法取得令人满意的结果，构建粗粒度 BH 加速器会是一项艰巨的任务。

图 7 展示了 BH 在双

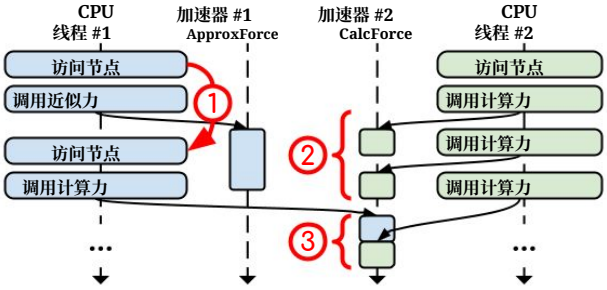


图 7：采用细粒度加速的多线程 BH①循环携带依赖和动态控制流由处理器处理；②处理器和加速器通过软件流水线并行运行；③多个 CPU 线程对加速器进行分时复用，以提高利用率。

核心系统上运行并搭载两个细粒度加速器的时间线。除了易于编程之外，这种软硬件协同设计还具有可扩展性和灵活性：可以实例化每种类型的多个加速器来提高并行度；如果 THRESHOLD 较高，即 ApproxForce 的调用次数较少，系统便可以分配更多嵌入式 FPGA 资源来实现更多和/或更快的 CalcForce 加速器；反之，如果 THRESHOLD 较低，则可以减少分配。

如果没有 Duet 的架构支持，细粒度的 BH 加速器将难以高效运行。首先，加速器在每次执行时只会访问随机内存位置上的少数几个缓存行，因此无法充分利用现有大多数 CPU-FPGA 架构中面向块操作、针对带宽优化的存储器系统。相比之下，Duet 的混合缓存组织可最大限度地降低加速器的内存访问延迟，尤其是以缓存行为粒度进行随机访问时。其次，多个 CPU 线程会频繁调用这些加速器，并以分时复用的方式使用它们。处理器为调用加速器而发出的存储器映射输入/输出，可能会使集中式 CPU-FPGA 互连发生拥塞。Duet 的可扩展集成使工作负载能够分散到多个嵌入式 FPGA 上，而影子寄存器则可进一步降低存储器映射输入/输出的延迟。

B. 硬件增强

1）概述： 硬件增强允许应用程序开发者在芯片制造完成后向系统添加各种硬件组件。这些组件有助于提升处理器效率，和/或缓解某些执行模型中的软件开销，从而实现应用无关加速。不同的硬件增强组件往往表现出独特的行为，并提供各异的软件 API。例如，解耦访问-执行引擎 [49]通过解引用软件指定指针并将数据收集到硬件队列中，来隐藏内存重定向的延迟；硬件垃圾回收器 [32] 则在后台自主运行，减轻运行时环境的软件开销。由于硬件增强组件与处理器紧密协作，Duet 的可扩展、紧密型缓存一致性集成对于最大限度提升性能至关重要。

2) 示例：离散事件仿真（DES）一直是并行执行领域公认的难题 [18]。所有仿真

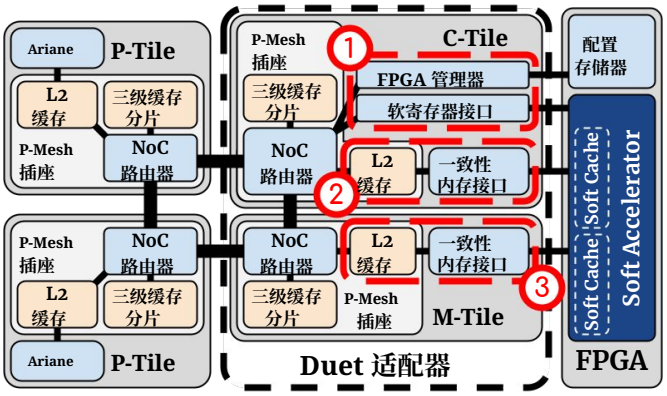


图 8：Dolly-P2M2 的架构

Dolly-P2M2 包含 2 个处理器、1 个嵌入式 FPGA 和 2 个内存枢纽。P-Tile、C-Tile 和 M-Tile 是二维网格网络中的物理瓦片。P-Mesh 插槽是所有物理瓦片中通用组件的物理封装，包括 L2 缓存、片上网络路由器和共享 L3 缓存的一个分片。① 是控制枢纽（第II-E节）。② 和③ 是两个内存枢纽（第II-B节）。请注意，①和② 位于同一个物理 C-Tile 中。

线程都必须要么按锁步方式推进（保守方法），但这会带来很高的同步开销；要么利用推测执行（乐观方法），但这需要精心设计的软件运行时，或支持线程级推测的架构，例如 SWARM [27]。

Duet 通过硬件增强开辟了另一种可能：可以将由嵌入式 FPGA 模拟的任务调度器作为模拟器软件的一部分加载，同时提供硬件级性能。下面考虑这种任务调度器的一种可能实现。处理器通过将事件的内存指针推送到面向 *FPGA* 的 *FIFO*中来调度新事件；随后，任务调度器从共享内存中获取事件数据，并将指针添加到相应的事件队列中。当某些事件准备好处理时，任务调度器会将指针推送到面向 *CPU* 的 *FIFO*中，使处理器能够以最少的调度器通信延迟来简化事件处理。任务调度器可以通过获取可能被推测执行事件修改的缓存行，并将其版本化副本存储在其非一致性内存中，来支持任务推测。发生推测错误时，任务调度器会将缓存行回滚到最新的非推测版本，然后重新调度推测错误的事件。

IV. DOLLY

为高保真地评估 Duet，我们构建了 Dolly[1],原型系统，采用 RTL 级（Verilog/SystemVerilog）实现，并配备了用于软件开发和加速器设计的完整工具链。Dolly 可配置处理器数量、嵌入式 FPGA 的逻辑容量、内存枢纽数量等。为简便起见，我们将每个 Dolly 实例命名为 PpMm，其中p 表示处理器数量，m 表示

[1]得名于《Dolly组曲》，作品56号，这是加布里埃尔·福雷创作的一组钢琴四手联弹曲。

嵌入式 FPGA 可用的内存枢纽数量。例如，图8 展示了 Dolly-P2M2 的架构。**为了推动对紧密集成的硬件缓存一致性 CPU-嵌入式 FPGA 系统的进一步研究，我们已将 Dolly 及其工具链开源，可通过以下地址获取：https://github.com/PrincetonUniversity/Duet。**

Dolly基于 OpenPiton P-Mesh缓存一致性系统 [6] ，采用二维网格配置。缓存系统分为三级：L1缓存与处理器紧密交织；每个物理瓦片配有一个私有的 8KB L2 缓存，采用回写策略，并通过事务响应接口（TRI） [5]与对应的 L1 缓存相连；共享 L3 缓存分布在所有物理瓦片中，每个分片为 64KB，并与私有 L2 缓存共同运行基于目录的MESI协议。片上网络提供点对点的消息传递顺序保证，并支持一致性消息之外的其他消息类型，从而支持 Dolly 所需的片上存储器映射输入/输出。

Dolly包含三种类型的物理瓦片和一个嵌入式 FPGA。每个 *P-Tile* 均搭载一颗 Ariane [57] 处理器，这是一款六级、单发射、顺序执行的 CPU，实现了 64 位 RISC-V指令集。每颗 Ariane处理器均配有一个私有硬件浮点运算单元、一个 8KB L1 指令缓存（L1I）和一个 8KB L1 数据缓存（L1D）。*C*瓦片 和 *M*瓦片 共同构成 Duet 适配器：每个 *C-Tile* 包含一个控制枢纽（第 II-E节）和一个内存枢纽（第II-B节）；每个 *M-Tile* 包含一个内存枢纽。为展示 Duet 的适应性，Dolly通过为未经修改的 一致性内存接口 添加一个 未经修改的 P-Mesh 二级缓存，实现了代理缓存。所有物理瓦片类型中的 L2 缓存、L3 缓存分片和片上网络路由器均相同，因此我们将它们封装在一个模块中，即 *P-Mesh* 插槽。

该嵌入式 FPGA 使用 PRGA [31] 构建，采用由逻辑块和布线块组成的标准岛式架构。该嵌入式 FPGA 包含现代 FPGA 常见的所有逻辑资源，包括查找表（LUT）、可旁路触发器、硬加法器链、块随机存取存储器（BRAM）和硬乘法器。Dolly 使用不可综合的时钟发生器生成嵌入式 FPGA 时钟，其频率可通过软件指定。所有异步先进先出队列均使用双时钟 RAM 和格雷码两级同步器实现。

V. 评估

A. 概述

我们通过对 Dolly 实例进行寄存器传输级仿真来评估 Duet，并从三个方面展开：首先，报告 Dolly 硬件组件的面积消耗和典型频率（第V-B节）；其次，运行综合基准测试，测量不同条件下 Duet 适配器实现的 CPU-FPGA 通信延迟和带宽（第 V-C节）；第三，运行七项应用基准测试，并将其性能与仅使用处理器的基线和 FPSoC 基线进行比较（第V-D节）。

为了展示 Duet 所带来的改进幅度下限，我们在整个评估过程中为处理器提供了各种优势。具体而言，我们将

表 I：Dolly 组件的面积和典型频率

组件	器件工艺	面积 (mm^2)	频率 (兆赫)	**缩放面积*** **(mm^2)**	**缩放频率*** **(兆赫)**
Ariane [58]	GlobalFoundries 22nm FDX	0.39	910	**1.56**	**455**†
P-Mesh 插座 [6]	IBM 32nm SOI	0.55	1000	**1.1***	**711***†
FPGA 管理器 + 软寄存器接口	FreePDK45	0.21	925	**0.21***	**925***
一致性内存接口		0.04	1250	**0.04***	**1250***

*按线性 MOSFET 缩放模型缩放至 45nm †为模拟性能更高的多核处理器，我们以 1 GHz 的频率对 Ariane 内核和硬件缓存系统进行仿真

Ariane 处理器和 P-Mesh 缓存系统的时钟频率提升至 1GHz，尽管以 45nm 为基准进行缩放后，先前研究报告的最高频率分别为 455MHz 和 711MHz。对于应用基准测试，仅处理器基线始终以热缓存启动，而软加速器始终以冷缓存启动。

B. 硬组件的面积和典型频率

表 I总结了 Dolly 各硬件组件的面积和典型频率。Ariane 和 P-Mesh 组件的数据取自先前的研究。控制枢纽和内存枢纽的子模块使用 Synopsys Design Compiler 和 Silvaco 的 Open-Cell Library 进行综合 [48]，基于 NCSU FreePDK45 工艺设计套件 [50]，面积利用率为 70%。总之，控制枢纽和内存枢纽所需的硬件资源极少。嵌入式 FPGA 的面积消耗和仿真加速器的时钟频率按基准测试报告，详见第V-D节。

C. CPU-eFPGA 通信延迟和带宽

本节通过合成基准测试评估 Duet 的峰值性能。嵌入式 FPGA 仿真一个简单的暂存存储器，处理器则使用不同机制访问该存储器。为展示嵌入式 FPGA 的时钟频率对性能有显著影响，我们将嵌入式 FPGA 时钟从 20 兆赫逐步提高到 500 兆赫，同时将系统时钟固定为 1 吉赫。具体而言，我们研究以下 CPU-eFPGA 通信机制，并展示 Duet 如何在现有 CPU-FPGA 系统的基础上改进这些机制：

软寄存器是由嵌入式 FPGA 仿真的内存映射寄存器，处理器可通过非一致性内存映射I/O对其进行读写。这些软寄存器可以在嵌入式 FPGA 内部实现（**普通寄存器**），在这种情况下，处理器访问必须承担跨时钟域的开销。**影子寄存器**解决了这一低效问题。在本研究中，我们使用面向 FPGA 的 FIFO 处理处理器写入，并使用面向 CPU 的 FIFO 处理处理器读取。

或者，处理器可以将数据存储在共享内存中，并通过软寄存器写入将存储器地址传递给嵌入式 FPGA。软寄存器写入不仅会通知嵌入式 FPGA 加载数据（**eFPGA 拉取**），还可用作同步，确保处理器的所有存储操作在嵌入式 FPGA 开始加载数据之前都已提交到缓存系统

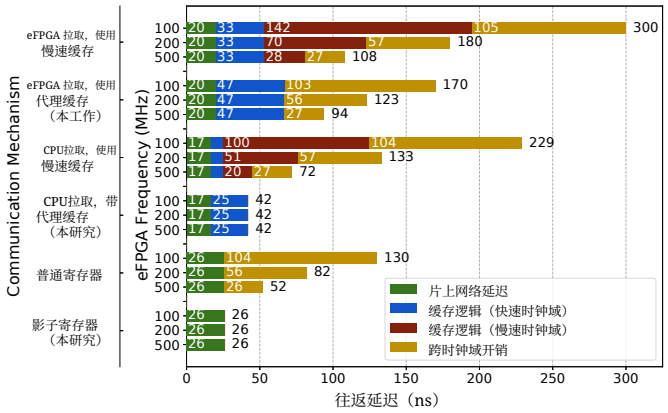


图 9：CPU-eFPGA 通信延迟（单处理器；单次事务；越低越好）

。嵌入式 FPGA 可以通过类似的方式向处理器发送数据（**CPU拉取**）。如II-C节所述，商用 FPSoC 通常使用嵌入式 FPGA 资源来模拟 FPGA 侧缓存（**慢速缓存**），而 Duet 则采用新颖的**代理缓存**来提升缓存性能。

延迟研究

我们首先测量了上述通信机制在 Dolly-P1M1 上的最小往返延迟。图 9展示了 CPU-eFPGA 通信延迟的四部分构成：片上网络延迟、快速时钟域中的缓存处理时间、慢速时钟域中的缓存处理时间，以及 CDC 开销。所有数据均在理想情形下采集，即单个处理器（无争用或 NoC 拥塞）、单次事务（无缓冲区阻塞）。请注意，eFPGA 拉取和 CPU 拉取均保证在请求方缓存中未命中，并在另一方的私有缓存中以已修改状态命中。片上网络延迟和快速时钟域中的缓存处理时间均包含分布式目录发送和处理二次回写请求所需的时间。

如图所示，即使嵌入式 FPGA 的运行频率达到 CPU 时钟频率的 50%，CDC 开销和缓存处理过程中因时钟较慢而产生的惩罚，仍占慢速缓存往返延迟的一半以上。对于嵌入式 FPGA 拉取，代理缓存可将延迟降低 13% 至 43%，且嵌入式 FPGA 运行得越慢，降幅越大。对于 CPU 拉取，无论嵌入式 FPGA 时钟频率如何，代理缓存都能实现恒定延迟，将延迟降低 42% 至 82%。影子寄存器也能实现固定延迟，将 CPU-eFPGA 通信延迟降低 50% 至 80%。请注意，代理缓存和影子寄存器只是将嵌入式 FPGA 移出关键路径；嵌入式 FPGA 仍需花时间发起内存访问并处理软寄存器访问。

单处理器带宽研究

接下来，我们研究 Dolly-P1M1 上单处理器通信的最大带宽。在这项研究中，我们将合成基准测试配置为将 512 个四字（8 字节）整数从一个处理器传递到嵌入式 FPGA，然后再取回。使用软寄存器时，处理器通过执行循环，在每次迭代中写入/读取一个整数来完成这一操作

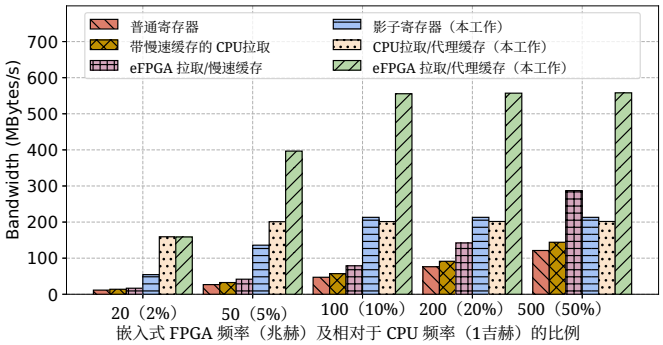


图 10：处理器与嵌入式 FPGA 通信带宽与嵌入式 FPGA 时钟频率的关系（单处理器；越高越好）

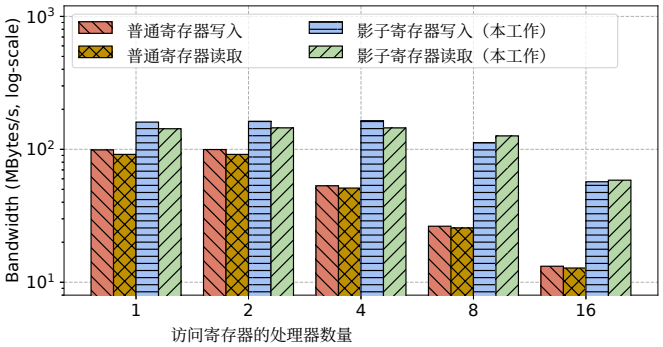


图 11：处理器与嵌入式 FPGA 通信带宽（每个处理器）与竞争处理器数量的关系

。使用共享内存时，处理器首先在存储器中分配两个 4 KB 缓冲区，并通过两个普通影子寄存器将基地址传递给嵌入式 FPGA。然后，处理器将所有整数存储到其中一个缓冲区，并向另一个普通软寄存器发送 读 请求，以唤醒嵌入式 FPGA，同时使自身进入阻塞状态。嵌入式 FPGA 主动将整个数组加载到其暂存存储器中，再将其存储到另一个缓冲区，随后通过确认处理器的读请求来解除处理器的阻塞。最后，处理器从共享内存中加载该数组。

图 10 展示了本研究的结果。在所有嵌入式 FPGA 频率下，代理缓存都能提供最高带宽。事实上，当嵌入式 FPGA 的运行速度高于 100MHz（CPU 时钟频率的 10%）时，代理缓存的嵌入式 FPGA 拉取峰值带宽可达 558MB/s；对于 CPU 拉取，峰值带宽（ 201MB/s ）则在 50MHz 时达到。相比之下，即使嵌入式 FPGA 以 500MHz 运行，慢速缓存的嵌入式 FPGA 拉取带宽也仅为 287MB/s，CPU 拉取带宽也仅为 144MB/s。当嵌入式 FPGA 以 100MHz 运行时，带宽差距最大，为 9.5 倍。基于缓存的通信上限取决于片上网络带宽，以及代理缓存所支持的并发在途内存请求数量。请注意，CPU 拉取的上限远低于嵌入式 FPGA 拉取。这是因为缓存行大小为 16 字节，嵌入式 FPGA 每个周期最多可加载一行；但 Dolly 中的 L2 缓存一次最多只支持存储 8 字节，因此嵌入式 FPGA 必须发送两次请求才能存储一个缓存行，导致异步 FIFO 未得到充分利用。

当嵌入式 FPGA 时钟频率超过 CPU 时钟频率的 10% 后，影子寄存器也能实现稳定的带宽（213MB/s）。相比之下，普通寄存器在 500MHz 时只能达到 121MB/s。软寄存器访问的上限受存储器映射输入/输出的严格一致性模型限制。如果放宽输入/输出一致性模型，上限则取决于处理器的加载存储队列所支持的并发在途存储器映射输入/输出数量。

多处理器可扩展性研究

Duet 与现有 CPU-FPGA 系统的一个关键区别在于可扩展性。在架构层面，Duet 对处理器数量、嵌入式 FPGA 结构数量或每个嵌入式 FPGA 的内存枢纽数量没有限制。本节研究多处理器争用对软寄存器的影响。具体而言，我们在多个 Dolly 实例上运行合成基准测试，这些实例配备不同数量的处理器，并持续对同一个软寄存器进行写入和读取。嵌入式 FPGA 时钟固定为 500 兆赫（CPU 时钟频率的 50%）。

图 11 展示了结果。影子寄存器能够稳定支持多达 8 个处理器，之后每个处理器的带宽才开始下降；而普通寄存器最多只能支持 2 个处理器。考虑到处理器在软寄存器访问之间还需要执行其他任务，因此在实际应用中，影子寄存器能够支持更多处理器。

总结

总而言之，延迟显著降低且带宽稳定提升，这充分证明了将软寄存器和私有缓存提升至高速时钟域的合理性。

D. 应用基准测试

在本节中，我们使用七个应用基准测试评估 Duet。对于每个基准测试，我们首先运行 C 实现，将其作为**仅处理器**基线，并记录总运行时间。请注意，由于 RTL 级仿真速度有限，所有基准测试均在裸机上运行。随后，我们为每个基准测试设计软加速器，可以直接在 RTL 级设计，也可以使用 Catapult HLS [47]对另一种 C 实现进行综合。每个加速器都通过 PRGA [31] 工作流进行综合、布局和布线，以准确获取嵌入式 FPGA 时钟频率，并估算所用 FPGA 资源的面积。具体而言，我们将每个基准测试映射到 VTR [38]提供的旗舰 FPGA 模型上（`k6_frac_N10_frac_chain_mem32K_40nm`），该模型与 Altera Stratix IV FPGA 相似。获得准确的嵌入式 FPGA 时钟频率后，我们在各种**Dolly** 实例和类**现场可编程片上系统**架构上重新运行基准测试，并记录总运行时间，其中包括 CPU-FPGA 通信和同步的所有开销。具体而言，FPSoC 模型将 P-Mesh 二级缓存移入嵌入式 FPGA 的（低速）时钟域，并将所有影子软寄存器降级为普通寄存器。随后，我们计算相对于仅处理器基线的加速比，并比较 Dolly 和 FPSoC 模型取得的结果。

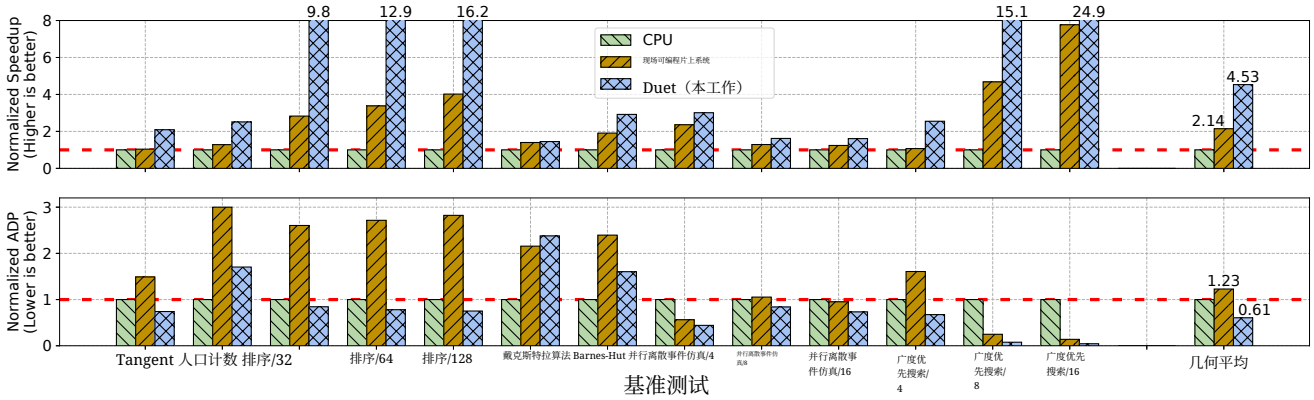


图12：应用基准测试的归一化加速比和面积-延迟积。sort/N 表示排序加速器的数组大小；pdes/N 和 bfs/N 表示共享加速器的核心数量。

我们使用面积-延迟乘积（ADP）量化并比较面积效率。计算面积消耗时，仅处理器基线只计入处理器和硬件缓存系统。类 FPSoC 架构在仅处理器基线的基础上增加 FPGA 的芯片面积，而 Dolly 还进一步计入 Duet 适配器。

七个基准测试、各自对应的 Dolly 实例以及加速范式（**细粒度加速** 和**硬件增强** ）如下：

- **Tangent** （P1M0，细粒度加速）：使用 Catapult HLS 实现浮点正切加速器，采用分段线性近似算法，与 C数学库（libm）相比最大误差率为 0.3%。使用面向 FPGA 的 FIFO 将参数传递给加速器并调用加速器。结果通过面向 CPU 的 FIFO 返回。
- **人口计数** （P1M1，细粒度加速）：人口计数用于计算一个长位向量（512 位）中 1 的数量。由于 Ariane处理器不支持 RISC-V位操作扩展，我们为仅处理器基线采用字节查找算法。该加速器使用 Verilog 手工编写，并通过一个内存枢纽从一致性内存中加载位向量。
- **排序** (P1M2，细粒度加速)：我们使用 SPIRAL项目 [60] 生成 3 个 Verilog 排序网络，用于对 32、64、128 个双字（4 字节）整数进行排序。该加速器使用两个内存枢纽，一个用于从一致性内存读取输入数组，另一个用于将已排序数组写回，因此，加速器可以通过流水线方式对较大数组中的定长切片进行排序，随后由处理器对这些切片进行归并排序。仅处理器基线则对整个数组运行快速排序。
- **戴克斯特拉算法** (P1M1，细粒度加速)：我们使用 Catapult HLS 实现了一个用于迪杰斯特拉最短路径算法的加速器，并使用软缓存来利用加速器连续调用之间的数据局部性。
- **Barnes-Hut** (P4M1，细粒度加速)：如第 III-A2 节所述，Barnes-Hut 算法中的两个关键内核通过 Catapult HLS 实现为软加速器。两个加速器均采用流水线设计，并由四个处理器进行分时复用。仅处理器基线也会使用四个处理器并行计算不同粒子的作用力。
- **并行离散事件仿真** (P4M1/P8M1/P16M1, 硬件增强)：如第III-B2节所述，我们使用 Verilog 设计了一个非推测式硬件任务调度器，以加速数字电路的并行离散事件仿真（PDES）。仅处理器基线使用 MCS 锁 [35] 来仲裁对共享事件队列的访问；随着核心数量增加，锁竞争可能会变得十分严重。
- **广度优先搜索** (P4M0/P8M0/P16M0，硬件增强)：我们使用 Verilog 实现了多个硬件无锁队列，以降低并行广度优先搜索（广度优先搜索）中的同步开销。处理器以屏障同步的步骤遍历图，并使用队列存储当前和下一轮的搜索前沿。与 PDES 基准测试类似，仅处理器基线受同步瓶颈的影响。

表II：软加速器的时钟频率和面积

基准测试	最高频率（兆赫）	归一化面积*	资源利用率 可配置逻辑块†	块随机存取存储器
Ariane + P-Mesh 插槽	1000	1.0	-	-
Tangent	282	0.47	0.84	0
人口计数	189	2.77	0.83	0.56
排序（32）	228	6.29	0.30	0.76
排序（64）	234	8.10	0.27	0.92
排序（128）	228	10.27	0.27	0.92
戴克斯特拉算法	127	1.94	0.96	0.31
Barnes-Hut	85	14.22	0.99	0.05
BFS	208	1.24	0.61	0.75
PDES	126	2.77	0.47	0.56

* 嵌入式 FPGA 实现该加速器所需的总硅面积，以 1x Ariane + 1x P-Mesh 插槽为基准进行归一化
† 可配置逻辑块，包括查找表和触发器

表。II 列出了软加速器的最高时钟频率和嵌入式 FPGA 资源利用率。请注意，软加速器的运行频率仅为处理器时钟频率的 8%–28%。正如第V-C节的研究所示，Duet 在这一频率范围内可实现峰值带宽，性能至少是传统现场可编程片上系统通信机制的 2 倍。

图12 展示了应用程序基准测试的归一化加速比和面积-延迟积。Duet 在所有基准测试中均优于 FPSoC，相较于仅处理器基线，几何平均加速比达到 4.53 倍，而

采用相同加速器的 FPSoC 加速比为 2.14 倍。排序加速器在 Duet 上的加速比最高可达 16.2 倍，而在 FPSoC 上仅为 4.0 倍，这是因为 FPSoC 中 FPGA 侧缓存受到了较慢时钟周期的影响。有人可能会认为 FPGA 侧缓存并非必需，因为已排序数组是连续的数据块，适合通过直接内存访问高效处理。然而，由于数组较小（128-512B），即使采用末级缓存一致性直接内存访问，缓存刷新也会产生过多的控制开销。广度优先搜索中的无锁队列在 Duet 上的加速比最高可达 24.9 倍，而在 FPSoC 上仅为 7.8 倍，这是因为 FPSoC 缺少影子寄存器，处理器与加速器通信延迟要高得多。值得注意的是，广度优先搜索基准测试在系统从 4 核（Dolly-P4M0）扩展到 8 核（Dolly-P8M0）时，呈现出超线性加速。这是因为仅处理器基线从 4 核扩展到 8 核时，锁竞争加剧，导致性能下降。

在面积效率方面，Duet 的面积-延迟积几何平均值比仅处理器基线低 0.39 倍（越低越好），而 FPSoC 则高 0.23 倍。除戴克斯特拉算法外，Duet 在大多数基准测试中的表现都优于 FPSoC。这是因为 FPSoC 模型将慢速时钟域中的 FPGA 侧缓存固化为硬件，因此不再需要软缓存，可以将其移除以节省嵌入式 FPGA 资源。

总之，Dolly 使用相同的嵌入式 FPGA 仿真加速器，实现了优于 FPSoC 的加速比。因此，我们可以得出结论：Duet 能更好地支持细粒度加速和硬件增强。

VI. 相关工作

A. 独立式 *FPGA* 加速器

由于 FPGA 具有可编程性，且性能接近专用集成电路，因此其应用领域正迅速扩大。在较小的应用规模中，独立 FPGA 用于加速数据库查询 [51]，图像处理 [14]，网络过滤 [52]，以及最广泛应用的深度学习应用 [21], [46], [59]。在另一端，FPGA 被大规模部署在云端，既可以作为按需计算服务直接提供，例如亚马逊 AWS F1 实例 [2]，，也可以用于构建云规模加速器，例如 Microsoft Catapult [10] 和 BrainWave [15]。在这些系统中，FPGA 通常集成为 PCIe 外设，或作为独立设备接入数据中心网络。如第I节所述，这类系统擅长在应用层进行加速，但由于 CPU 与 FPGA 通信开销较高，无法胜任细粒度加速。

B. 同封装 *CPU-FPGA* 混合架构与 *FPSoC*

为解决 CPU-FPGA 通信瓶颈，学术界和工业界一直在探索如何让二者实现更紧密的集成。Intel Harp [13] 将一颗 Intel Xeon 多核 CPU 和一块 Altera FPGA 集成在同一封装中，并通过 QPI 将二者连接起来。QPI 是一种支持缓存一致性的点对点互连。现场可编程片上系统（FPSoC）则通过在同一芯片上集成处理器和 FPGA 实现了更紧密的集成，从总体上看与 Duet 类似。许多商用 FPSoC [26], [36], [37], [42], [55]和学术界 FPSoC [28], [45], [53] 支持完整或部分缓存一致性。例如，Xilinx Zynq-7000 采用 AXI4 ACP 接口 [3]，支持单向缓存一致性（输入/输出一致性）。Xilinx Zynq UltraScale+ MPSoC [56]和 Versal ACAP [19] 通过 ACE 协议 [3]提供完整的一致性支持。

Duet 与 FPSoC 之间有两个关键区别。首先，如图所示 1b，FPSoC 中的处理器子系统与嵌入式 FPGA 由集中式互连隔开，硬件缓存层级则位于处理器子系统中。对于以 FPGA 为中心的双核架构而言，这种组织方式是合理的，但无法扩展到更多核心。其次，由于没有 FPGA 侧硬件缓存，设计并验证符合一致性协议（例如 ACE）的软缓存便成了加速器设计者的责任 [3]。软缓存除了增加设计复杂度，还会因运行在慢速时钟域中并直接参与缓存一致性，而拖慢整个缓存系统。此外，由于软缓存能够访问片上网络的微架构状态，系统无法遏制软缓存的故障行为或恶意行为。

C. 近处理器与处理器内可重构计算

早在可重构计算发展的初期，计算机架构师就提出将可重构结构 (RF) 集成在处理器附近，甚至集成到处理器内部。Garp [9]将 RF 放置在处理器与其私有缓存之间，使两个计算单元共享整个存储器系统。Chimaera [24] 和 PRISC [43]将可重构功能单元 (RFU) 直接嵌入处理器的数据通路，从而能够在制造完成后定制指令集架构 (ISA)。制造后微架构 [29]将 RF 与超标量核心耦合，使 RF 能够在关键流水线阶段进行观察并实施微架构级干预。这些系统各有优势，但与我们的即插即用集成 目标相冲突，因为它们要求重新设计处理器。

D. 加速器一致性协议

硬件缓存一致性已广泛应用于多处理器系统，因为它不仅提供可编程性，也带来性能提升。在硬件加速领域，近期研究也展示了采用硬件缓存一致性的优势，并提出了多种解决方案。FUSION [30] 是一种分层的、基于时间戳的一致性协议，着重于加速器之间的数据传输；Crossing Guard [39]提出使用硬缓存转换器，将加速器一致性协议与处理器（主机）协议解耦，同时纳入容错和内存转换；Spandex [1] 提供灵活的写入策略和粒度，以适应高度异构的架构。尽管这些方案都实现了硬件缓存一致性，但它们并未针对基于FPGA的加速器进行优化，因为它们都要求加速器直接参与。不过，借助代理缓存，Duet 可以适配这些一致性协议，在片上网络看来伪装成一款“快速”加速器，同时向一致性协议隐藏慢速时钟域。

，并向片上网络伪装成一款“快速”加速器，同时向一致性协议隐藏慢速时钟域。

业界也正在推出加速器一致性标准和互连技术，例如 CCIX [11]，OpenCAPI [40]，和 CXL [25]。这些方案面向电路板级或更高层级，针对粗粒度加速进行了优化。

VII. 结论

在本研究中，我们提出了 Duet——一种可扩展的众核 FPGA 架构，集成了非侵入式、具备一致性的嵌入式 FPGA，并针对广泛的通用领域中的细粒度加速和硬件增强进行了优化。通过将嵌入式 FPGA 提升为片上一级公民，Duet 使嵌入式 FPGA 能够访问片上网络，并像其他处理器一样参与双向缓存一致性。新颖且轻量的 Duet 适配器将代理缓存和影子寄存器置于更快的处理器时钟域中，从而降低关键通信开销。我们的评估表明，Dolly（Duet 的 RTL 级实现）可将处理器 - 加速器通信的延迟降低最多 82%，并将带宽提升最多 9.5 倍，且在广泛的嵌入式 FPGA 时钟频率范围内保持稳定。利用 Duet 的软加速器进行的选定基准测试显示，其加速比最高可达相较于仅处理器基线提升 24.9 倍，相较于 FPSoC 提升最多 4 倍。Dolly 及其工具链已开源，可从 https://github.com/PrincetonUniversity/Duet获取。

确认消息

本材料基于由空军研究实验室 (AFRL) 和国防高级研究计划局 (DARPA) 根据协议编号 FA8650-18-2-7852 赞助的研究。本材料基于美国国家科学基金会资助编号 CNS-1823222 以及美国国家科学基金会研究生研究奖学金计划资助编号 DGE-2039656 所支持的工作。尽管相关文件上注明了版权，美国政府仍有权为政府用途复制和分发重印件。本文所述观点和结论仅代表作者意见，不应被视为必然代表空军研究实验室 (AFRL)、国防高级研究计划局 (DARPA) 或美国政府的官方政策或认可，无论是明示还是暗示。本文所表达的任何意见、发现、结论或建议均为作者个人观点，不一定反映美国国家科学基金会的看法。

参考文献

[1] J. Alsop、M. D. Sinclair 和 S. V. Adve，“Spandex：用于高效异构一致性的灵活接口”，载于第 45 届国际计算机体系结构研讨会论文集，系列：ISCA ’18。IEEE Press，2018 年，第 261–274 页。 [在线]。可访问：https://doi.org/10 1109 2018 00031 。/ISCA。亚马逊，“Amazon EC F 实例”，https://aws.amazon.com/ec2/instance-types/f1/。[3] ARM Limited，“AMBA AXI 和 ACE 协议规范”，https://developer.arm.com/documentation/ihi0022/e/。[4] ——，“AMBA CHI 体系结构规范”，https://developer.arm.com/documentation/ihi0050/c/。[5] J. Balkind、K. Lim、M. Schaffner、F. Gao、G. Chirkov、A. Li、A. Lavrov、T. M. Nguyen、Y. Fu、F. Zaruba、K. Gulati、L. Benini 和 D. Wentzlaff，“BYOC：异构指令集架构研究的‘自带核心’框架”，载于第二十五届国际编程语言与操作系统架构支持会议论文集，系列：ASPLOS ’20。美国纽约州纽约市：计算机协会，2020 年，第 699–714 页。 [在线]。可访问：https://doi.org/10 1145 3373376 3378479 。/。J. Balkind、M. McKeown、Y. Fu、T. Nguyen、Y. Zhou、A. Lavrov、M. Shahrad、A. Fuchs、S. Payne、X. Liang、M. Matl 和 D. Wentzlaff，“OpenPiton：开放源代码众核研究框架”，载于第二十一届国际编程语言与操作系统架构支持会议论文集，系列：ASPLOS ’16。美国纽约州纽约市：ACM，2016 年，第 217–232 页。 [在线]。可访问：http://doi.acm.org/10 1145 2872362 2872414 。/。J. Barnes 和 P. Hut，“一种分层 (N 对数 N) 力计算算法”，《自然》，第 324 卷，第 446–449 页，1986 年。[8] A. Boutros、S. Yazdanshenas 和 V. Betz，“无法改进未曾测量之物：卷积神经网络推理中 FPGA 与专用集成电路的效率差距”，ACM 重构技术系统汇刊，第 11 卷，第 3 期，2018 年 12 月。 [在线]。可访问：https://doi.org/10 1145 3242898 。/ T. Callahan、J. Hauser 和 J. Wawrzynek，“Garp 体系结构与 C 编译器”，《计算机》，第 33 卷，第 4 期，第 62–69 页，2000 年。[10] A. M. Caulfield、E. S. Chung、A. Putnam、H. Angepat、J. Fowers、M. Haselman、S. Heil、M. Humphrey、P. Kaur、J.-Y. Kim、D. Lo、T. Massengill、K. Ovtcharov、M. Papamichael、L. Woods、S. Lanka、D. Chiou 和 D. Burger，“云规模加速体系结构”，载于 2016 年第 49 届 IEEE/ACM 国际微体系结构研讨会 (MICRO)，2016 年，第 1–13 页。[11] CCIX 联盟，“加速器缓存一致性互连 (CCIX)”，https://www.ccixconsortium.com/。[12] Chen, Yu-Hsin、Krishna, Tushar、Emer, Joel 和 Sze, Vivienne，“Eyeriss：用于深度卷积神经网络的能效型可重构加速器”，载于 IEEE 国际固态电路会议，ISSCC 2016，技术论文摘要，2016 年，第 262–263 页。[13] Y.-k. Choi、J. Cong、Z. Fang、Y. Hao、G. Reinman 和 P. Wei，“现代 CPU-FPGA 平台微体系结构的定量分析”，载于 2016 年第 53 届 ACM/EDAC/IEEE 设计自动化会议 (DAC)。IEEE Press，2016 年，第 1–6 页。 [在线]。可访问：https://doi.org/10 1145 2897937 2897972 。/。N. Chugh、V. Vasista、S. Purini 和 U. Bondhugula，“用于在 FPGA 上加速图像处理流水线的领域特定语言编译器”，载于 2016 年并行体系结构与编译国际会议论文集，系列：PACT ’16。美国纽约州纽约市：计算机协会，2016 年，第 327–338 页。 [ ] 10 1145 2967938 2967969 在线。可访问：https://doi.org/。/。E. Chung、J. Fowers、K. Ovtcharov、M. Papamichael、A. Caulfield、T. Massengill、M. Liu、D. Lo、S. Alkalay、M. Haselman、M. Abeydeera、L. Adams、H. Angepat、C. Boehn、D. Chiou、O. Firestein、A. Forin、K. S. Gatlin、M. Ghandi、S. Heil、K. Holohan、A. El Husseini、T. Juhasz、K. Kagi、R. K. Kovvuri、S. Lanka、F. van Megen、D. Mukhortov、P. Patel、B. Perez、A. Rapsang、S. Reinhardt、B. Rouhani、A. Sapek、R. Seera、S. Shekar、B. Sridharan、G. Weisz、L. Woods、P. Yi Xiao、D. Zhang、R. Zhao 和 D. Burger，“借助 Brainwave 项目在数据中心规模上实时提供深度神经网络服务”，IEEE Micro，第 38 卷，2018 年 20，2，8，第 – 页。[16] E. S. Chung、P. A. Milder、J. C. Hoe 和 K. Mai，“单芯片异构计算：未来是否包含定制逻辑，FPGA和通用图形处理器？”载于《2010年第43届IEEE/ACM国际微体系结构研讨会》，2010年，第225–236页。[17] J. Coole、J. Wernsing 和 G. Stitt，《遍历缓存框架：通过 FPGA 加速指针数据结构——Barnes-Hut n 体模拟案例研究》，载于《2009年国际可重构计算与FPGA会议》，2009年，第143–148页。[18] R. M. Fujimoto，《并行离散事件仿真》，《ACM通讯》，第33卷，第10期，第30–53页，1990年10月。 [在线]。获取地址：https://doi.org/10 1145 84537 84545 。/。B. Gaide、D. Gaitonde、C. Ravishankar 和 T. Bauer，《赛灵思自适应计算加速平台：VersalTM 架构》，载于《2019年ACM/SIGDA国际现场可编程门阵列研讨会论文集》，系列：FPGA ’19。美国纽约：计算机协会，2019年，第84–93页。 [在线]。获取地址：https://doi.org/10 1145 3289602 3293906 。/。Z. Guo、W. Najjar、F. Vahid 和 K. Vissers，《FPGA 相较于处理器的加速比定量分析》，载于《2004年ACM/SIGDA第12届国际现场可编程门阵列研讨会论文集》，系列：FPGA ’04。美国纽约：计算机协会，2004年，第162–170页。 [ ] 10 1145 968280 968304 在线。获取地址：https://doi.org/。/。S. Han、J. Kang、H. Mao、Y. Hu、X. Li、Y. Li、D. Xie、H. Luo、S. Yao、Y. Wang、H. Yang 和 W. B. J. Dally，《ESE：基于 FPGA 的稀疏长短期记忆网络高效语音识别引擎》，载于《2017年ACM/SIGDA国际现场可编程门阵列研讨会论文集》，系列：FPGA ’17。美国纽约：计算机协会，2017年，第75–84页。 [在线]。获取地址：https://doi.org/10 1145 3020078 3021745 。/。S. Han、X. Liu、H. Mao、J. Pu、A. Pedram、M. A. Horowitz 和 W. J. Dally，《EIE：基于压缩深度神经网络的高效推理引擎》，载于《第43届国际计算机体系结构研讨会论文集》，系列：ISCA ’16。IEEE Press，2016年，第243 254 [ ] 10 1109 2016 30 –页。在线。获取地址：https://doi.org/。/ISCA。/。Handel Jones，《基于知识产权杠杆优化半导体供应商市场地位的策略》，https://www.ibs-inc.net/white-papers，2014年。[24] S. Hauck、T. Fry、M. Hosler 和 J. Kao，《Chimaera 可重构功能单元》，载于《论文集：第五届IEEE年度现场可编程定制计算机研讨会（目录号97TB100186）》，1997年，第87–96页。[25] 英特尔公司，《高速计算互连TM (CXL)》，https://www.intel.com/content/www/us/en/io/cxl-cache-mem-protocol-interface-cpi.html。[26] ——，《Cyclone V片上系统》，https://www.intel.com/content/www/us/en/products/details/fpga/cyclone/v.html。[27] M. C. Jeffrey、S. Subramanian、C. Yan、J. Emer 和 D. Sanchez，《有序并行的可扩展架构》，载于《2015年第48届IEEE/ACM国际微体系结构研讨会（MICRO）》，2015年，第228–241页。[28] D. Koch、N. Dao、B. Healy、J. Yu 和 A. Attwood，《FABulous：嵌入式FPGA框架》，载于《2021年ACM/SIGDA国际现场可编程门阵列研讨会》，系列：FPGA ’21。美国纽约：计算机协会，2021年，第45 56 [ ] 10 1145 3431920 3439302 –页。在线。获取地址：https://doi.org/。/。C. Kumar、A. Seshadri、A. Chaudhary、S. Bhawalkar、R. Singh 和 E. Rotenberg，《制造后微架构》，载于《MICRO-54：第54届IEEE/ACM国际微体系结构研讨会》，系列：MICRO ’21。美国纽约：计算机协会，2021年，第1270–1281页。 [在线]。获取地址：https://doi.org/10. 1145/3466752.3480119[30] S. Kumar、A. Shriraman 和 N. Vedula，《FUSION：加速器一致性缓存层次结构中的设计权衡》，载于《2015年ACM/IEEE第42届年度国际计算机体系结构研讨会（ISCA）》，2015年，第733–745页。[31] A. Li 和 D. Wentzlaff，《PRGA：开源FPGA研究与原型设计框架》，载于《2021年ACM/SIGDA国际现场可编程门阵列研讨会》，系列：FPGA ’21。美国纽约：计算机协会，2021年，第127 137 [ ] 10 1145 3431920 3439294 –页。在线。获取地址：https://doi.org/。/。M. Maas、K. Asanovic 和 J. Kubiatowicz，《追踪式垃圾回收硬件加速器》，《IEEE Micro》，第39卷，46，2019 3，38第。页–。[33] C. A. Mack，《摩尔定律五十年》，《IEEE半导体制造汇刊》，第24卷，第2期，第202–207页，2011年。

[34] I. Magaki、M. Khazraee、L. V. Gutierrez 和 M. B. Taylor，《ASIC 云：数据中心专用化》，载于《2016年ACM/IEEE第43届年度国际计算机体系结构研讨会（ISCA）》，2016年，第178–190页。[35] J. M. Mellor-Crummey 和 M. L. Scott，《共享内存多处理器上的可扩展同步算法》，《ACM计算机系统汇刊》，第9卷，第1期，第21–65页，1991年2月。 [在线]。获取地址：https://doi.org/10 1145 103727 103729 。/。Microchip Technology Inc.，《[36] SmartFusion 2 片上系统》，https://www.microsemi.com/product-directory/soc-fpgas/1692-smartfusion2。[37] Microsemi Corporation，《PolarFire 片上系统》，https://www.microsemi.com/product-directory/soc-fpgas/5498-polarfire-soc-fpga。[38] K. E. Murray、O. Petelin、S. Zhong、J. M. Wang、M. Eldafrawy、J.-P. Legault、E. Sha、A. G. Graham、J. Wu、M. J. P. Walker、H. Zeng、P. Patros、J. Luu、K. B. Kent 和 V. Betz，《VTR 8：高性能计算机辅助设计与可定制 FPGA 架构建模》，《ACM可重构技术与系统汇刊》，第13卷，第2期，2020年5月。 [在线]。获取地址：https://doi.org/10 1145 3388617 。/。L. E. [39] Olson、M. D. Hill 和 D. A. Wood，《Crossing Guard：协调主机-加速器一致性交互》，载于《第二十二届国际编程语言与操作系统架构支持会议论文集》，系列：ASPLOS '17。美国纽约：计算机协会，2017年，第163 176 [ ] 10 1145 3037697 3037715 –页。在线。[40] 获取地址：https://doi.org/。/。OpenCAPI 联盟，《OpenCAPITM》，https://opencapi.org/。[41] 甲骨文公司，《OpenSPARCTM T1 微架构规范》，https://www.oracle.com/servers/technologies/opensparc-t1-page.html。[42] QuickLogic Corporation，《EOS S3》，https://www.quicklogic.com/products/soc/。[43] R. Razdan 和 M. Smith，《具有硬件可编程功能单元的高性能微架构》，载于《MICRO-27论文集：第27届IEEE/ACM年度国际微体系结构研讨会》，1994年，第172–180页。[44] K. Rupp，《48年微处理器趋势数据》，https://github.com/karlrupp/microprocessor-trend-data，2019年。[45] P. D. Schiavone、D. Rossi、A. D. Mauro、F. Gurkaynak、T. Saxe、M. Wang、K. C. Yap 和 L. Benini，《Arnold：面向灵活、低功耗物联网终端节点的增强型 eFPGA 的 RISC-V 片上系统》，2020年。[46] A. Shawahna、S. M. Sait 和 A. El-Maleh，《用于学习与分类的深度学习网络 FPGA 加速器：综述》，《IEEE Access》，第7卷，第7823–7859页，2019年。[47] 西门子数字化工业软件，《Catapult 高层次综合与验证》，https://eda.sw.siemens.com/en-US/ic/catapult-high-level-synthesis/。[48] Silicon Integration Initiative, Inc.，《15NM开放单元库和45NM FreePDK》，https://si2.org/open-cell-library/。[49] J. E. Smith，"解耦访存/执行计算机体系结构"，SIGARCH计算机体系结构新闻，第10卷，第3期，第112–119页，1982年4月。[ ] 10 1145 1067649 801719 在线。可查阅：[50] https://doi.org/。/。J. E. Stine、I. Castellanos、M. Wood、J. Henson、F. Love、W. R. Davis、P. D. Franzon、M. Bucher、S. Basavarajaiah、J. Oh 和 R. Jenkal，"Freepdk：一种开源变异感知设计套件"，载于2007年IEEE微电子系统教育国际会议（MSE'07），2007年，第173–174页。[51] B. Sukhwani、H. Min、M. Thoennes、P. Dube、B. Iyer、B. Brezzo、D. Dillenberger 和 S. Asaad，"使用FPGA加速数据库分析"，载于第21届并行体系结构与编译技术国际会议论文集，PACT '12系列。美国纽约州纽约市：计算机协会，2012年，第411 420 [ ] 10 1145 2370816 2370874 –页。在线。[52] 可查阅：https://doi.org/。/。N. Weaver、V. Paxson 和 J. M. Gonzalez，"The Shunt：用于网络入侵防御的基于FPGA的加速器"，载于2007年ACM/SIGDA第15届国际现场可编程门阵列研讨会论文集，FPGA '07系列。美国纽约州纽约市：计算机协会，2007年，第199–206页。[ ] 10 1145 1216919 1216952 在线。可查阅：[53] https://doi.org/。/。P. N. Whatmough、S. K. Lee、M. Donato、H.-C. Hsueh、S. Xi、U. Gupta、L. Pentecost、G. G. Ko、D. Brooks 和 G.-Y. Wei，"一款16纳米、25平方毫米的片上系统，其灵活性-能效范围为54.5倍，涵盖双核Arm Cortex-A53、嵌入式 FPGA和缓存一致性加速器"，载于2019年VLSI电路研讨会，2019年，第C34–C35页。[54] C. Wolf，"Yosys开放综合套件"，http://www.clifford.at/yosys/。[55] 赛灵思公司，"Zynq-7000片上系统"，https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html。[56] ——，"Zynq UltraScale+ MPSoC"，https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html。[57] F. Zaruba 和 L. Benini，"应用级处理的成本：采用22纳米FDSOI工艺的支持Linux、1.7吉赫、64位RISC-V内核的能耗与性能分析"，《IEEE超大规模集成电路系统汇刊》，第27卷，第11期，第2629–2640页，2019年11月。[58] F. Zaruba、F. Schuiki、S. Mach 和 L. Benini，"浮点三位一体：实现极致能效与性能的多模态方法"，载于2019年第26届IEEE电子、电路与系统国际会议（ICECS），2019年，第767–770页。[59] C. Zhang、P. Li、G. Sun、Y. Guan、B. Xiao 和 J. Cong，"优化深度卷积神经网络的基于FPGA的加速器设计"，载于2015年ACM/SIGDA国际现场可编程门阵列研讨会论文集，FPGA '15系列。美国纽约州纽约市：计算机协会，2015年，第161 170 [ ] 10 1145 2684746 2689060 –页。在线。[60] 可查阅：https://doi.org/。/。M. Zuluaga、P. Milder 和 M. Püschel，"流式排序网络"，《ACM设计自动化与电子系统汇刊》，第21卷，第4期，2016年5月。[在线]。可查阅：https://doi.org/10.1145/2854150