可重构计算: 软件可定义的计算引擎

魏少军*, 李兆石, 朱建峰, 刘雷波

清华大学微电子学研究所, 北京 100084

* 通信作者. E-mail: wsj@tsinghua.edu.cn

收稿日期: 2020-05-11; 接受日期: 2020-08-04; 网络出版日期: 2020-09-23

国家自然科学基金重点 (批准号: 61834002) 和国家重点研发计划 (批准号: 2018YFB2202100) 资助项目

摘要 随着科技快速进步, 新兴应用不断涌现. 无法响应软件变化的芯片, 如专用集成电路(application-specific integrated circuit, ASIC), 将因为生命周期过短, 面临一次性工程成本(non-recurring engineering, NRE) 过高的难题. 与此同时, 随着摩尔定律 (Moore's law) 和迪纳德定律 (Dennard scaling) 走向终结, 未来集成电路工艺更新带来的能效收益越来越小, 通用处理器可实现的计算能力被芯片功耗约束. 近几年兴起的领域定制加速器 (domain-specific accelerator, DSA) 通过针对特定应用领域的计算模式, 定制芯片架构, 以期兼顾能量效率和特定领域内的灵活性. 但目前 DSA 面向硬件定制软件, 这导致软件生态碎片化, 程序员学习成本增大. 未来芯片设计需要兼顾灵活性、能量效率和可编程性. 软件定义芯片 (software-defined chip, SDC) 在这一需求下成为了研究热点. 可重构芯片通过融合处理器的高灵活性、ASIC 的高能效, 并通过重构提供了在运行时根据软件定制芯片架构的能力, 是当前 SDC 的研究热点. 本文首先回顾 SDC 的研究动机, 然后分析可重构芯片如何满足 SDC 的需求, 之后探讨当前可重构芯片面临的挑战, 最后阐述为了实现 SDC, 可重构芯片未来的发展方向.

关键词 软件定义芯片, 可重构计算, 领域定制架构, 能量效率, 可编程性

1 引言

随着摩尔定律 (Moore's law) 逐渐逼近物理极限^[1], 集成电路的功耗约束在许多应用中变得更加严格. 芯片设计人员必须将注意力从性能转向能量效率. 另一方面, 大量新兴应用 (如神经网络和生物信息学) 的更新速度越来越快, 灵活性也成为芯片设计中的重要考虑因素. 由于软件随着新兴应用程序和用户需求迅速发展, 如果硬件无法适应软件, 专用集成电路 (application-specific integrated circuit, ASIC) 将面临生命周期短和一次性工程费用 (non-recurring engineering, NRE) 成本高的问题. 因此, 高能效和高灵活性间的权衡已成为一款计算芯片最重要的设计考虑. 然而, 对于现行的主流计算芯片来

引用格式: 魏少军, 李兆石, 朱建峰, 等. 可重构计算: 软件可定义的计算引擎. 中国科学: 信息科学, 2020, 50: 1407–1426, doi: 10.1360/SSI-2020-0130
Wei S J, Li Z S, Zhu J F, et al. Reconfigurable computing: toward software defined chips (in Chinese). Sci Sin Inform, 2020, 50: 1407–1426, doi: 10.1360/SSI-2020-0130

说,同时满足这两个需求极具挑战:ASIC 能效虽高,但却牺牲了灵活性;而冯·诺依曼(von Neumann)处理器(如通用处理器(CPU)、图形处理单元(GPU)、数字信号处理器(digital signal processor, DSP))虽灵活性高,但能效太低.为了解决这一挑战,芯片设计领域涌现出大量新兴计算架构.

近年来兴起的领域定制加速器(domain-specific accelerator, DSA),如谷歌的TPU,面向特定应用领域的计算特征,定制芯片的逻辑通路和存储系统,从而在高效和高灵活性间达到平衡.但目前主流 DSA 基于硬件架构定制软件接口.由于芯片设计时可能存在的对某些应用特征进行的过度优化,用户需要对他们的需求进行裁剪,使需求匹配这些应用特征后,才能充分利用硬件的性能.这使得当前 DSA 的可编程性难以满足新兴应用不断迭代的需求.

高能量效率、高灵活性和高可编程性的需求催生了“软件定义芯片”(software-defined chip, SDC)的理念.顾名思义,软件定义芯片就是让硬件根据软件进行定制化设计,提高了能量效率.这样设计出的芯片能够根据产品的需求改变功能,实现了高灵活性的芯片设计.此外,SDC 基于软件框架,而不是某些应用领域计算特征,进行硬件的定义,从而避免了硬件设计时对特定计算特征的“过拟合”,提高了芯片的可编程性.

可重构计算是 SDC 理念的出发点.2017 年由美国国防部高级研究计划局(Defense Advanced Research Projects Agency, DARPA)提出了电子产业振兴计划(Electronics Resurgence Initiative, ERI),试图推进后摩尔定律时代的计算芯片新材料、架构与设计流程的研究.ERI 的主要研究目标“软件定义芯片”期望实现接近 ASIC 的性能(10 倍以内),同时不牺牲数据密集型算法的灵活性的芯片.DARPA ERI 的项目说明中大量引述了可重构计算芯片的研究结果¹⁾.

可重构计算兴起于 20 世纪 90 年代^[2],21 世纪以来发展迅速.可重构计算能够持续吸引工业界和学术界关注,是因为它具有接近 ASIC 的能效和性能,以及由制造后的硬件可以重构带来的灵活性^[3~5].按照可重构芯片进行重构时的粒度,可重构芯片可以进一步分为以现场可编程逻辑门阵列(field-programmable gate array, FPGA)为代表的细粒度可重构芯片,和粗粒度可重构芯片(coarse-grained reconfigurable array, CGRA).FPGA 经过 30 年的发展,已经成为了通信、网络、航天、国防等领域不可或缺的核心电子器件.例如,微软的 Catapult 项目使用 FPGA 加速其数据中心的 Bing 搜索引擎,带来了显著的能效提升,同时保留了系统灵活性和算法的可更新性^[6].IBM、英特尔和百度也做出类似选择^[7,8].但 FPGA 能效与 ASIC 仍存在较大差距.更严重的是,目前 FPGA 的可编程性极差,这造成程序员对硬件足够了解才能开发 FPGA 的性能和能效潜力,严重限制了它的推广.近年来学术界有大量研究面向 SDC 探索 CGRA 的研究^[9,10].而在工业界,CGRA 正在被应用到越来越多的生产系统中.例如,英特尔启动了一个将 CGRA 纳入其 Xeon 处理器的项目²⁾,三星将 CGRA 加速器集成在它的 8 K 高清电视和 Exynos 片上系统(system-on-chip, SoC)中³⁾⁴⁾.虽然 CGRA 在易用性方面的不成熟使得它在商业应用上仍受到一些限制,但这也为我们以 SDC 为导向,开展 CGRA 相关研究提供了机会窗口.

本文从 SDC 的需求出发,从设计方法、挑战和潜在解决方案等角度全面介绍可重构芯片,主要贡献如下:

(1) 系统地分析可重构芯片如何实现 SDC 的需求.

1) DARPA. <https://www.darpa.mil/>.

2) Intel. <https://newsroom.intel.com/news-releases/intel-tsinghua-university-and-montage-technology-collaborate-to-bring-indigenous-data-center-solutions-to-china/>.

3) Samsung. <http://www.samsung.com/semiconductor/minisite/exynos/products/mobileprocessor/exynos-5-octa-5430/>.

4) PACT. <http://www.pactxpp.com/>.

Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8 KB)	5
32b DRAM Read	640

图 1 通用处理器中典型操作的能耗

Figure 1 Energy consumption of typical operations in CPU

- (2) 对可重构芯片在当今遇到的挑战进行了自顶向下的全面分析.
- (3) 调研了当前研究中解决这些挑战所采用的技术. 基于这些技术, 我们对未来可重构芯片的发展趋势做出预测.

本文其他小节组织如下. 第 2 节介绍软件定义芯片的需求, 第 3 节介绍可重构芯片如何实现软件定义芯片的需求, 第 4 节分析了当前可重构芯片的设计方法, 第 5 节介绍了当今可重构芯片遇到的挑战, 第 6 节基于当今最先进的技术对于上述挑战提出潜在的解决方案, 第 7 节总结本文.

2 软件定义芯片

本节从如今计算芯片面临的挑战出发, 分析软件定义芯片的需求. 具体而言, “功耗墙”的约束促使通用计算架构走向定制化; 专用集成电路的 NRE 费用引出了芯片灵活性的需求, 从而催生了领域定制加速器; 领域定制后引出的可编程性问题, 催生了软件定义芯片的概念.

2.1 功耗墙的约束

通用处理器是最为灵活的计算芯片. 在自动化编译工具的支持下, 通用处理器可以方便地实现不同类型的算法. 同时, 通用处理器的灵活性使其可以拥有较大的出货量, 进而使得每颗芯片分担的 NRE 费用较低. 然而, 为了确保灵活性, 通用处理器设计时包含了大量取指、译码等非算术逻辑单元逻辑, 因此它在计算过程中的额外能量开销非常大, 计算的能效比偏低. 图 1 展示了通用处理器中典型操作的能耗对比. 可以看到平均而言一条精简指令集 (reduced instruction set computer, RISC) 指令实际消耗在算术逻辑单元 (arithmetic-logic unit, ALU) 上的功耗不到 20%. 大量的功耗都消耗在了取指译码或寄存器堆访问上. 而对于访存操作, 又有额外的 20% 高速数据缓存 (cache) 访问开销^[11].

额外的能耗开销引发了当今通用处理器的“功耗墙”问题^[12]: 随着集成电路工艺的发展, 通用处理器的性能在过去的数十年间获得了极大的提升, 然而在 2005 年以后, 因为全比例缩放技术的终结^[13], 功耗成为了通用处理器的关键设计约束^[14].

2.2 灵活性的需求

相比于通用处理器, ASIC 将指令集架构中的取指译码和寄存器堆的访问等步骤转化为芯片内部

图 2 (网络版彩图) 80 nm 后各工艺节点单个晶体管的费用随时间的变化趋势

Figure 2 (Color online) Normalized cost per transistor after 80 nm technology

固化的电路, 并通过数据流驱动整个芯片的运行. 由于芯片专注于应用的运算部分, ASIC 具有极高的能量效率. 然而, 也正是由于 ASIC 对电路的功能进行了固化, 芯片只能支持单一的应用. 图 2 展示了 80 nm 后各工艺节点单个晶体管的费用随时间的变化趋势⁵⁾. 可以观察到, 在 28 nm 工艺之后, 新的工艺将很难再降低单个晶体管的费用, 芯片流片费用将随着芯片上晶体管的数量指数增长. 为了分担高昂的 NRE 费用, ASIC 设计往往局限于通信基带、网络流处理等成熟的、出货量极大的应用. 这一要求严重限制了 ASIC 在新兴应用中的使用. 当代新兴应用, 如深度神经网络、生成式对抗网络等, 算法更新迭代速度非常快. 针对单一应用或单一算法设计的 ASIC, 一方面难以获得足够的出货量分担 NRE 费用, 另一方面会由于较长的研发周期错过了最佳上市时间.

2.3 可编程性的问题

能量效率和灵活性间的矛盾促使学术界和工业界开始研究领域定制加速器. 在 DSA 的设计中, 计算芯片由两部分组成: 一个通用处理器, 负责传统的资源管理和用户交互任务, 如操作系统、编译器; 以及一系列领域定制处理单元, 他们针对一个或多个特定领域的应用提供高效能的解决方案. 如图 1 所示, ALU 计算浮点加法和 32-bit 整型加法的功耗远小于一条指令的功耗, 而 8-bit 整型加法的功耗更是远远低于 32-bit 整型加法. 如果一个应用领域的运算和访存模式相对固定, 那么该领域相应的 DSA 可以将通用处理器中冗余控制、访存逻辑替换成该领域常用的算术逻辑单元以提高计算能量效率.

但是, DSA 面临着可编程性的困境. 为了提高能量效率, DSA 面向特定应用领域的计算特征, 在芯片设计时定制其逻辑通路和存储系统. 然而, 相同应用领域的不同的算法对 DSA 的芯片设计需求会有很大区别. 图 3 展示了给定 DSA 上的总存储容量, 不同算法的最优化存储系统设计下对 cache, 共享存储 (shared memory) 和寄存器堆容量的需求^[15]. 由于目前 DSA 的芯片架构设计选择需要流片前确定, 因此大多数 DSA 芯片只能针对特定应用领域的特定计算模式选择最优实现. 对于其他应用来说, 这些选择通常不会是最优的. 由于芯片设计时的过度优化, 程序员需要对需求进行裁剪, 使其匹配 DSA 支持的应用特征后才能充分利用硬件的性能. 目前市面上常见的 DSA 都有类似的可编程性

5) Beyond Moore's law. The Economist, 2015. <https://www.economist.com/science-and-technology/2015/05/26/beyond-moores-law>.

图3 (网络版彩图) 不同算法对 DSA 的存储系统不同的要求
Figure 3 (Color online) Optimal hardware design differs for different algorithms

困境: DSA 架构设计者要教用户写程序, 而不是由用户自行描述应用需求. DSA 的适用空间完全由架构设计者规划, 其可编程性难以满足新兴应用不断变化的需求.

为了解决这一困境, SDC 吸收了 DSA 的高能效和特定领域灵活性的特点, 尝试走出可编程性的困境. SDC 的思路是按照软件, 动态地、实时地改变硬件的架构. 这样, SDC 可以将部分设计选择推迟到运行时, 避免在芯片设计时对于特定应用的过度优化, 这一思路与可重构芯片的特性不谋而合.

3 可重构芯片的发展历程

随着计算芯片的功耗约束愈加严苛, 以及集成电路设计与制造成本也在急剧增加, 传统的计算芯片越来越难以兼顾高灵活性和高能效. 体系结构的研究正在迎来一场革命. 诸多历史上没有进入主流通用处理器体系结构的设计被逐一挖掘出来. 可重构计算正是其中的一种. 本节简要从可重构计算的发展历程出发, 重点介绍两类可重构芯片 FPGA 和 CGRA 的异同, 然后分析 CGRA 为何是 SDC 的最佳选择.

3.1 可重构计算模型的提出

可重构计算概念的提出可以上溯到 20 世纪 60 年代. 美国加州大学洛杉矶分校 (University of California, Los Angeles) 的 Estrin 等^[16] 提出一种特殊的体系结构, 包括一个主处理器和一个可重构硬件, 前者控制后者的行为, 后者通过重构的方式加速特定任务. 由于过于超前, 该概念直到 40 年以后才获得系统性的研究. 加州大学伯克利分校 (University of California, Berkeley) 的 DeHon 等^[17] 将可重构计算定义为具有下面特征的体系结构:

- (1) 在其制造后, 芯片的功能仍然可以定制, 形成加速特定任务的硬件功能;
- (2) 计算功能的实现主要依靠任务到芯片的空间映射.

满足以上两个特征的计算方式都可以被称为可重构计算. 其中, 第 1 个特征说明了可重构计算的灵活性, 即它在生产后仍然可以进行编程, 拥有类似于通用处理器的灵活性; 第 2 个特征则指明了可重构计算的能效优势的来源, 即计算主要以空域并行计算的模式完成, 避免了通用处理器在时域上不断取指译码新指令的额外开销, 从而可以逼近 ASIC 的高能效.

3.2 可重构芯片的兴起

本小节介绍目前主流的两类可重构计算芯片与传统芯片架构的异同, 并着重介绍它们如何在能效和灵活性之间进行权衡.

图 4 可重构计算架构与现有主流计算架构在能量效率和灵活性上的对比

Figure 4 Comparison between reconfigurable architectures and traditional computer architectures in terms of energy efficiency and flexibility

图 4 定性地从灵活性和能量效率两个角度对比了 CGRA 与当今其他主流计算芯片^[5]. 可重构计算芯片的灵活性和能量效率介于通用处理器和 ASIC 之间, 其设计理念也来源于前文所述的灵活性和能量效率的折衷.

可重构计算兼顾高灵活性和高能效的关键在于它的执行过程是由配置信息驱动的. 配置信息不仅能表达处理单元的操作, 还能通过显式设定芯片的互联功能表达数据在处理单元间的流动. 配置信息的设定体现了可重构计算的空域映射特性. 相比而言, 通用处理器受指令驱动, 而 ASIC 由应用的数据流驱动. 配置信息既结合了时域上指令的灵活性, 又结合了空域上数据流的高能效^[18]. 因此使用配置信息进行计算的 CGRA 结合了通用处理器和 ASIC 的优点^[19].

根据可重构芯片的组织结构更像 ASIC 还是更像通用处理器, 又可以将其细分为静态可重构和动态可重构两类^[9, 20, 21], 如图 5^[22] 所示.

静态可重构芯片以 FPGA 为代表, 其计算功能在计算开始前的初始化阶段进行定制, 随后计算过程中不再改变. 因此, 静态可重构计算芯片的控制器只需要在初始化阶段加载配置信息, 运行过程中不再需要工作. 因此图 5(a) 中可以将控制器省略. 与图 5(c) 的 ASIC 相比, 静态可重构芯片在运行时工作原理相近, 因而它具有很高的能量效率. 然而, 为了在固化逻辑的基础上提供灵活性, 静态可重构计算芯片需要以比特流的形式逐一指定各个查找表和多路选择器的功能. 这导致它的配置信息加载的时间非常大, 计算功能受限于硬件资源数目.

动态可重构芯片以 CGRA 为代表, 与通用处理器更为相似, 其架构中包含可重构数据通路和可重构控制器. 如图 5(b) 所示. 前者的定位类似于通用处理器的算术逻辑单元. 但他们间的一个显著区别在于可重构数据通路通常采用许多基本处理单元 (processing element, PE) 组成的阵列结构. 其中每个 PE 的功能都和 ALU 类似, 并通过一个灵活的互连网络实现 PE 间中间计算结果的转发. 因为数据通路的功能相比单一计算逻辑单元变得更加复杂, 传统指令集很难完全地描述它的功能. 所以动态可重构计算芯片使用由配置字组成的配置信息控制其行为. 在数据通路计算的同时, 可重构控制器加载并组织配置信息, 并将它们以合适的顺序发送给数据通路, 这极大地增强了动态可重构计算芯片的灵活性. 与比特流形式的静态可重构配置信息相比, 配置字以更粗的粒度指定 PE 和互连网络的功能.

图5 (网络版彩图) 静态可重构和动态可重构的基本组织结构及其与ASIC和通用处理器的对比^[22]

Figure 5 (Color online) Comparison between FPGA/CGRA with ASIC/CPU

这使得计算通路可以在时域上快速重构以适应更大规模的应用。

从上面的比较可以看出, 相比于FPGA, CGRA与SDC的“高效能、高灵活、高可编程性”的设计目标契合度更高。本文之后将集中讨论CGRA的设计与挑战。

4 CGRA 设计方法

至今为止, CGRA的设计方法尚未统一, 存在多样性。为了解和比较不同架构, 将CGRA设计在多个系统级抽象层次上进行阐释, 是一种有效途径。本节采用文献[23]提出的多维度分类方法根据以下几个系统级抽象层次描述当前主流CGRA的设计方法。图6展示了该分类方法下通用处理器和CGRA的对比。

(1) 微架构模型描述了芯片上各个模块的实现, 及模块间的组织方式。对于CGRA, 微架构模型主要考虑PE、存储系统、内部互联等芯片功能的实现方式。不同CGRA的微架构差异很大。

(2) 执行模型是微架构的抽象表示, 它定义计算模型生成的中间表示的调度方案。该模型需要表征硬件的核心工作机制, 例如指令/配置的触发、执行和结束。因此, 执行模型确立了微架构设计的基本框架, 决定了计算模型所表示的底层运行顺序和并行模式的实现。

(3) 计算模型是计算语义的抽象表示, 它定义了一个应用程序的高层次组合成分^[24]。该模型提供一种计算引擎, 能够为使用相应编程模型描述的任务生成解决方案。计算模型必须能够反映物理计算平台的主要计算特征^[25]。同时, 计算模型确定了底层计算架构能够支持的算法并行模式。例如, 单线程计算模型通常不能利用线程级并行度。

图 6 (网络版彩图) 使用文献 [23] 提出的多维度分类方法比较通用处理器和 CGRA

Figure 6 (Color online) Comparison of an CGRA and CPU under the guidance of the multidimensional taxonomy

(4) 编程模型是底层计算机系统的抽象表示, 主要用于算法和数据结构描述^[25]. 该模型连接了底层硬件和程序可用的软件支持层. 事实上, 所有编程语言和应用程序编程接口 (application programming interface, API) 都是编程模型的实例. 编程模型可以描述软件程序, 也可用于编程硬件^[26]. 它对底层硬件细节进行抽象, 便于程序员并行编程, 而无需考虑并行计算过程在硬件上的具体实现. 同时, 编程模型也限制了程序员可以显示利用的算法级并行度类别, 从而简化编译算法. 例如, 典型多线程编程模型, 例如 PThread, 将硬件资源抽象为线程, 使得程序员可将应用程序的并行性表示为相互协调的多个线程.

在计算模型上, 根据 2.3 小节的描述, CGRA 同时在空域和时域上具有较高的自由度, 而经典冯·诺依曼架构仅有时域的高自由度. 这是 CGRA 与其他计算形式的根本区别. 因此, 本节从计算模型出发, 系统描述 CGRA 在各个抽象层次上的设计空间.

4.1 计算模型

图 7 细化了图 5(b), 展示了 CGRA 计算模型的形式^[27]. CGRA 的高能效性主要表现在下面两个方面.

- **空域上可重构计算依靠处理单元间的互联显式实现依赖关系.** 应用中的基本操作及它们之间的依赖关系可以用控制数据流图 (control-data flow graph, CDFG) 表示. 每个应用中都会存在多种数据依赖关系和控制依赖关系, 对应的应用程序中需要确保依赖关系正确执行. 目前主流编译器都会提取应用的 CDFG 用于优化. 然而由于冯·诺依曼计算模型只有时域一个维度, 编译器在生成其程序时需要将二维的 CDFG 改写成一维的指令流, 利用寄存器名称隐式表达指令间的依赖关系. 之后通用处理器在执行程序时要依赖多端口寄存器堆或流水线旁路逻辑重发现这些依赖关系. 相比而言, 可重构计算模型加入了空间维度, 这使得 CDFG 中的依赖关系可以通过配置数据通路上处理单元的互连网络显

图7 (网络版彩图) CGRA 在时域和空域上协同计算

Figure 7 (Color online) CGRA executes cooperatively in time domain and space domain

式表达, 避免了重发现依赖关系的能量和性能额外开销。

- **时域上可重构计算功能单次配置, 多次执行。**冯·诺依曼计算模型下指令的加载和执行是耦合的, 每条指令执行前都需要经过取指、译码等步骤。即便编译器发现一段指令需要重复执行, 它也无法在程序中省掉这段指令在第1次执行后的取指和译码等步骤。相比而言, 可重构计算模型的配置信息功能更为自由。对于需要重复执行的功能, 配置信息中可以指定数据通路多次执行该功能; 对于连续变化的功能, 配置信息可以利用配置管理器使数据通路快速连续重构以满足功能要求。

4.2 编程模型

可重构计算模型在时域和空域的自由度上需要以编程模型的形式向程序员提供。可重构计算模型下编程模型需要开发应用中的并行性以充分利用时域和空域的计算资源。本小节根据并行性的开发方式将编程模型做如下分类。

- **命令式编程模型和命令式语言 (如 C/C++)。**命令式模型使用顺序化的语句、命令或指令序列控制程序运行, 因此难以表达并行语义。一些命令式语言的扩展支持显式表达并行性, 例如 PThreads, 能够用特殊指令显式或部分显式表达并行性。所有支持顺序命令执行的硬件 (例如大多数处理器) 都可以使用此模型。CGRA 的运行流程由配置序列控制, 因此也可以使用命令式模型。由于命令式编程语言对于程序员来说门槛相对较低, 且便于与通用处理器集成, 许多 CGRA 都采用了命令式编程模型 [28~30]。

- **声明性编程模型 (如函数式语言、数据流语言和硬件描述语言)。**声明性编程模型使用声明或表达式代替命令式语句描述计算逻辑。由于该模型不包含任何控制流, 故而计算并行性是隐式表达的。命令式编程模型依赖编译器或硬件挖掘并行性, 相较而言, 并行编程模型允许编程者使用提供的接口显式地表达并行性, 从而减轻编译器和硬件的负担, 提高编译和运行效率。虽然目前仅有少数的 CGRA 采用并行编程模型 [19, 31], 但大多数的 CGRA 都适用于这类并行模型。原因在于 CGRA 的底层硬件执行并不是命令式的, 而是并行地计算一个配置内包含的多个操作。

- **透明编程,**即对一个特定 CGRA 架构, 不进行任何静态编译。这类编程采用动态编译技术, 依赖硬

件在程序运行时实时的翻译和优化常见的程序表示 (如指令序列). 因此, 它的底层计算模型、执行模型或微架构对编程者来说是完全透明的. 例如, 对于 DORA^[32], CCA^[33] 以及 DynSPAM^[34] 等架构, 它们的配置信息是根据运行时的指令流动态生成的. 采用透明编程模型的 CGRA 通常无需修改源程序即可移植到新计算架构, 可大大提高生产效率, 并且硬件还可使用部分运行时信息执行编译器无法实现的动态优化. 但是, 由于需要额外硬件模块执行运行时并行度挖掘, 支持透明编程模型的 CGRA 通常面临性能不足、能效降低、设计困难等问题.

不同的编程模型下同一个应用在 CGRA 上的性能会截然不同. 目前学术界主流观点认为在编程模型层面提升性能的最有效途径是根据应用领域定制编程模型.

4.3 执行模型

执行模型指定了可重构计算模型时域和空域自由度的实现方式, 约束了下一层的硬件架构设计. 同时, 执行模型可以看作编译器面向的硬件架构特性: 程序员在编写应用时不需要考虑执行模型, 之后在编译时编译器根据执行模型自动将应用映射到目标架构上.

与通用处理器相比, CGRA 目前对于什么样的执行模型最佳还没有定论. 因此, CGRA 执行模型设计空间很大, 设计维度众多. 本研究认为最重要的两个设计维度如下.

- **配置信息调度方案.** 即配置信息加载到数据通路的顺序和时机. CGRA 可以是静态调度, 即编译器决定配置信息如何加载; 也可以是动态调度, 即 CGRA 在运行时决定配置信息如何加载. 2.3 小节曾将 CGRA 分为“静态可重构”与“动态可重构”, 那里的“静态”和“动态”指的就是配置信息的加载方式.

- **配置信息执行方案.** 即配置信息在数据通路上执行的顺序和时机. CGRA 可以是顺序执行, 即 CGRA 按照配置信息加载顺序执行; 也可以是数据流执行, 即 CGRA 的各个运算资源根据其配置信息, 检查它绑定的操作的各个操作数是否准备就绪, 然后执行那些所有操作数都已就绪的操作. 数据流执行又可以进一步细分为静态数据流执行和动态数据流执行. 静态数据流执行中每个运算资源在一段时间只绑定单一操作, 而动态数据流执行中每个运算资源在一段时间可以绑定多个操作.

将以上两个设计维度组合, 我们可以得到 4 种常见 CGRA 执行模型, 按照其时域 - 空域自由度由小到大排列如下: 静态调度顺序执行、动态调度顺序执行、动态调度静态数据流执行、动态调度动态数据流执行. 需要注意的是, 执行模型的自由度不是越大越好. 实际设计一个 CGRA 时需要根据应用领域、编程模型和架构设计复杂度审慎地选择执行模型.

4.4 微架构

CGRA 的微架构按照功能可以分为 3 个模块.

- **数据通路.** 可重构数据通路一般由 PE 单元和互连网络两部分组成. PE 单元是可重构计算的基本单元. 与众核通用处理器中的各个核相比, PE 单元不是独立的处理器. 它的功能更接近于通用处理器中执行单元及相关支持硬件结构, 用于完成计算操作. 按照 PE 单元的数据位宽, 可重构数据通路可以分为细粒度 (1-bit) 和粗粒度 (大于 1-bit) 两种. 互连网络是 CGRA 区别于通用处理器的重要特征. 按照 PE 单元操作间的“生产者 - 消费者”关系, PE 运算数据在空间利用互连网络快速流动. 互连网络具体组织形式不固定, 一般而言越灵活的互联实现代价越大. PE 单元和互连网络组成了 PEA, 是可重构计算模型中空域 - 时域自由度的主要载体.

- **配置通路.** 配置通路一般由配置控制器和配置缓存两部分组成. 配置控制器通常是一个可编程的有限状态机, 按照主控制器的控制, 以及数据通路的计算结果, 选择后续的配置信息. 为了减少配置信

息加载对性能的影响, CGRA 通常都会有片上的配置缓存. 尤其对于支持快速动态重构的动态 CGRA, 配置缓存必不可少.

- **访存通路.** 访存通路一般由数据缓存和数据控制器两部分组成. 与通用处理器类似, CGRA 频繁的片外数据访问会严重影响性能. 因而类比通用处理器, CGRA 也常常会采用数据 Cache 的设计. 然而, 由于计算资源众多, CGRA 对数据的需求量相比通用处理器更大. 为了满足 PEA 的数据需求, CGRA 还经常采用数据暂存器 (scratchpad) 结合数据控制器的设计. 数据控制器原理与 SoC 设计中的直接存储器存取 (direct memory access, DMA) 控制器类似, 但在 CGRA 上数据控制器可以根据目标应用领域做更定制化的设计.

需要特别指出的是, 编程模型、计算模型和执行模型的划分在层次化系统设计中是自顶向下的方法, 不同层次的模型之间存在着特定的对应关系. 例如, 动态调度要求配置控制器的状态机执行时可以受到数据通路计算结果的控制, 数据流执行要求 PE 单元具有检查操作数可用性的功能. 以上模型揭示了其对应的不同 CGRA 设计的本质差异, 表明了对 CGRA 采用多维度分类方法的合理性.

4.5 CGRA 应用现状

本小节将会展示近年较为成熟的 CGRA 应用领域. 目前 CGRA 提供足够的粗粒度计算资源, 但对控制流的支持较弱, 因此主要用于计算和数据密集型领域.

- **信息安全.** 信息安全领域的应用, 如加密和解密, 对底层硬件的灵活性和针对抵抗物理攻击的安全性提出更高要求. 这类应用通常包含数百个不断变化的密文, 并且需要承受比传统攻击更具威胁的物理攻击, CGRA 符合这些要求. 除灵活性之外, CGRA 还具有较强抵御物理攻击的能力, 因为它具有的软件定义芯片的运行特点比独立专有硬件实现或软件算法实现更具安全性. 近年来, 基于 CGRA 的加密处理器发展迅速, 引起广泛关注. 例如, COBRA^[35] 是一种适用于 40 多种对称密码的定制 CGRA; Cryptoraptor^[36] 是用于对称密钥加密算法和标准的可重构加密处理器, 支持多种加密算法, AES-128 算法实现的峰值吞吐量为 128 Gbps.

- **信号和图像处理.** 信号和图像处理应用具有典型流计算特征. 由于 CGRA 在流处理上性能优异, 因此许多针对这类应用的 CGRA 架构被提出. 经典加速架构 ADRES 已应用于视频处理 (H.264 /AVC 解码^[37])、图像处理^[38]、SDR 信号处理 (SDM-OFDM) 接收机^[39], 以及 MIMO SDM-OFDM 基带处理^[40].

- **深度学习.** 自 2010 年以来, DNN 已在当代人工智能应用 (包括计算机视觉、语音识别、医疗、游戏和机器人等) 中被广泛采用. DNN 通过频繁层间通信对大量数据执行密集计算. 由于 CGRA 具有高吞吐率计算和片上通信系统, 可被用作 DNN 核心加速引擎. 例如, 文献 [41, 42] 提出的加速架构具有可重构数据路径和可扩展映射方法, 以支持多种混合数据重用模式; 文献 [43] 提出的动态可重构架构以脉动阵列作为二维数据流计算引擎, 支持多种 CNN 中的计算模式.

- **其他.** 上述所有领域的应用都包含计算密集和数据密集特点. 其他具有类似特点的应用领域, 如网络处理^[44]等, 也可使用 CGRA 作为加速平台.

5 CGRA 面临的挑战

CGRA 实现动态可重构计算的主流形式, 近年来受到学术界和工业界越来越多的关注, 无论计算机体系架构领域、固态电路领域, 抑或电子设计自动化领域, 都是如此. 然而, 由于 CGRA 在易用性和适用性方面仍然不成熟, 直到今天它在商业应用上仍受到一些限制. 首先, CGRA 难以用高层次语

言编写高效的程序, 自动化编译也很难实现. 其次, CGRA 在面向不同应用领域的设计中差别较大, 其在实际系统中的集成成本依旧不低. 最后, CGRA 的应用仍然严格受限于纯粹的计算或数据密集型内核, 在涉及不规则代码或复杂控制流的情况下, 它的性能会严重下降. 这些问题之所以产生不只是与微架构有关, 必须在系统级别上全面分析与审视, 以此确定这些致命问题的根本原因所在. 本节在上文提出的分类方法指导下分析了 CGRA 存在这些不足的根本原因, 提出目前 CGRA 面临的 3 项关键挑战.

5.1 挑战 1: 高层次编程模型的易用性与效率的矛盾

编程模型作为描述应用的高层次抽象, 会对于底层计算架构的性能和易用性产生重要的影响^[26]. 编程模型层次过低会使编程语言对程序员不友好、编程时间过长. 而编程模型层次过高则会使底层计算架构难以实现高性能.

理想的 CGRA 编程模型必须在易用性和效率之间做出权衡, 这是非常大的挑战. (1) 从硬件角度来看, 其根本原因在于 CGRA 复杂的底层计算架构. CGRA 中主要的计算资源是一个有着很强计算能力的二维处理单元阵列, 它比传统的顺序处理器架构更加复杂. 因为 CGRA 具有时域计算的特性, 在 CGRA 上进行任务划分也比在 FPGA 上更加困难, 而且使用高级语言编程 FPGA 仍然是非常困难的, 所以复杂的硬件架构使得 CGRA 的编程变得非常有挑战性. (2) 从软件角度来看, 需挖掘大量细粒度并行性的复杂 CGRA 架构与目前流行的顺序风格的软件编程模型之间存在根本冲突. 为实现易用性, 编程模型应该是高层次抽象的, 这样才能使程序员免于学习架构特征细节. 高层次抽象只能提供粗粒度并行, 这不足以完全开发硬件的潜能. 为实现高效率, 编程模型应明确公开体系结构中对实现效率有影响的关键元素, 以便程序员探索体系结构设计空间以确定性能瓶颈. 例如, FPGA 可以使用高层次模型 (如 OpenCL) 和高层次综合工具编程, 从而产生效率低的设计^[45]. 或者, FPGA 可用低层次模型 (如 VHDL) 且通过综合、布局和布线工具编译, 从而生成更高性能的实现, 但需要更长设计和编译时间^[46]. 难点在于在提高抽象层次的同时提高性能^[26]. 换言之, 设计适合 CGRA 的编程模型, 其挑战在于如何在隔离其他硬件细节的同时, 仅揭示对性能有严重影响的架构特性. 类似架构的成功示例如 GPGPU 的 CUDA.

目前, CGRA 还没有完美的编程范式. 大多数 CGRA 及其编译器仍需大量手工工作. 具体来说, 这些 CGRA 使用高层次语言编程, 并依赖编译器前端将高层次输入转换为较低层次的中间表示 (intermediate representation, IR), 随后将 IR 进行优化并转换为机器代码. 在这些情况下, 编译器结果和手动优化结果之间存在显著差距^[47], 这意味着向程序员公开更多架构细节 (采用较低层次的编程模型) 将比仅依靠面向编译器的优化更有效. 总之, 让高层次编程模型可以充分开发 CGRA 的性能和能效潜能, 是 CGRA 在易用性上的主要挑战.

5.2 挑战 2: 多层次并行的性能收益与面积功耗开销的矛盾

良好的计算模型能够利用算法并行性, 并支持编程模型各种并行语义的有效实现. 实际的应用程序总是包含一些算法并行性. 一个有能力利用更多算法并行性的架构可以实现更好的性能. 挖掘更多的并行性可以通过支持多级并行的架构实现, 包括指令级并行 (instruction-level parallelism, ILP)、数据级并行 (data-level parallelism, DLP)、内存级并行 (memory-level parallelism, MLP)、任务级并行 (task-level parallelism, TLP) 和推测并行. 例如, 由于利用了 ILP, 超标量处理器比标量处理器性能更好; 由于利用了 TLP, 多核处理器比超标量处理器性能更好; 由于利用了推测并行, 乱序处理器的性能优于顺序处理器.

CGRA 在计算模型中已经支持很多并行性. 例如, TRIPS 支持 3 个执行模型, 每一种执行模式都适用于不同的并行, 如 ILP, DLP 和 TLP^[48]. 多态流水线阵列支持具有软件流水线的细粒度并行和来自 ILP 和 TLP 的粗粒度流水线并行^[49]. MLP 和 DLP 支持的并行已成为最近针对数据密集型域的 CGRA 的主题^[20]. 然而, 很少有 CGRA 能够很好地支持推测并行性. 推测并行性是并行性开发的重要资源. 实现推测并行性的主要障碍在于面积和功耗限制. 尽管在通用处理器中已经开发推测性执行, 但是当将该技术应用于 CGRA 时, 会出现难以克服的问题.

通用处理器主要通过分支预测和乱序执行这两个技术利用推测并行性. (1) 分支预测通常与推测相结合, 以错误预测惩罚为代价来消除控制依赖性. (2) 乱序执行是一种针对内存依赖的推测技术. 模糊内存访问是并发执行的, 就好像不存在依赖关系一样, 因此依赖于他们的指令将会被立即执行. 虽然这些技术在提高性能方面取得显著成功, 但它们的面积和功耗开销也很大. 根据 RISC 处理器 (90 nm) 的功耗分析, 在计算中执行算术运算中有有用工作的能耗仅占媒体处理应用中总功耗的不到 1%^[11]. 事实证明, 乱序核的能效低于顺序核.

与 GPP 不同, CGRA 是能进行时域计算的空间架构, 这使得 CGRA 难以利用推测并行性. 首先, 分支预测在配置级别执行时具有较低的准确性, 因为分支历史比指令级预测短得多^[50]. 虽然通过捕获配置中的所有与分支相关的信息, 可以提高准确性, 但由于阵列规模相当大, 硬件成本变得难以负担. 其次, 由于检查和验证处于配置级别, 分支错误预测惩罚激增, 因为预测错误意味着刷新和重新填充相应配置, 并对这些配置中所有内存访问进行缓冲和重新排序. 因此, 与 GPP 相比, CGRA 性能损失、功耗开销和面积开销可能更大. 考虑到预测准确度降低和误预测惩罚增加, 分支预测技术变得不那么有效. 最后, 乱序执行技术也面临着缓冲配置级存储器访问的显著开销 (配置内的存储器访问是原子的). 总而言之, 利用 CGRA 推测并行性的主要挑战是如何最大限度减少错误推测率和惩罚.

5.3 挑战 3: 多种非规则应用特征的定制化执行机制间的矛盾

目前流行的 DSA, 实际上大多是“规则应用领域定制架构”, 即针对基于稠密矩阵计算、谱分析算法等领域的 DSA. 这一现象背后的原因是当前 CGRA 的执行模型对于非规则应用支持不足. 大量非规则应用需要特殊的执行机制支持. 这些执行机制是面向通用处理器设计的, 在 CGRA 上需要专门空间计算资源的支持, 从而与追求普遍性的 CGRA 执行机制设计产生矛盾.

根据通用处理器上执行机制的不同, 非规则性可以被分为两类.

第 1 类不规则性是由复杂控制流引入的控制依赖关系. 在 C 语言中表现为分支语句、定界/不定界循环和子函数调用等. 因为控制流表达的控制依赖关系决定了实际的语句执行顺序, 复杂控制流将会导致语句的串行化执行. 此外, 控制流产生的不同分支间的语句经常需要互斥执行, 难以简单地同时执行. 所以, CGRA 难以高效执行应用中的复杂控制流.

第 2 类不规则性是由共享数据结构引入的运行期依赖关系. 多种常用的数据结构, 如树、图等, 都会在算法中引入不规则访问的共享数据. 这些访问的并发性需要在运行期根据具体访问地址确定, 无法在编译期确定. 通用处理器提供了锁、信号量等同步原语. 这些同步原语为通用处理器共享内存执行模型定制, 在 CGRA 上实现代价远高于通用处理器.

目前, CGRA 上主要有两种方案对非规则数据访问进行优化: 第 1, 使用对开发者透明的片上多级 cache 来缓存主存中的数据; 第 2, 需要开发者使用底层硬件原语控制数据在片外主存和片上 scratchpad 之间的搬移. 这两个方案的关键问题是: 前者在维持 cache 状态时会有很大功耗开销, 而后者需要开发者理解可重构芯片存储架构的设计细节, 从而会导致上层的编程模型易用性变差. 而对于非规则控制流, 目前 CGRA 上缺少执行机制上的支持. 总之, 开发 CGRA 在非规则应用上的潜力是 CGRA 在

适用性上面临的主要挑战.

6 CGRA 发展趋势

为了应对第 4 节提出的 3 项挑战, 我们调研了当前研究中在 CGRA 上解决这些挑战所采用的技术. 基于这些技术, 我们对未来 CGRA 的发展趋势做出预测.

6.1 趋势 1: 编程模型驱动的 CGRA 设计

目前研究者已经意识到, 应将编程模型作为首要设计问题引入 CGRA 的设计流程中. Prabhakar 等^[18]提出一种专用于并行模式编程模型的 CGRA, 其中计算单元为计算嵌套模式而设计, 类似于 SIMD PE 的多级流水线; 内存单元为数据局部性而设计, 类似于 scratchpad 内存和可配置的地址解码器. Li 等^[51]将具有线程级推测的编程模型引入到 FPGA 硬件生成中, 以适应那些在运行时才能确定其数据依赖或控制依赖的不规则应用程序. Nowatzki 等^[19]提出一种面向流数据的流编程模型的架构, 该架构由一个 CGRA、一个控制核以及一个流内存接口组成. 结果表明, 通过该方法实现的微架构能效比 ASIC 高 50%.

根据上述工作, 我们认为编程模型在很大程度上决定了底层计算模型、执行模型以及微架构. 编程模型的使用者, 即程序员, 在设计一个易用且高效的编程模型过程中起着关键的作用. 在 CGRA 的设计过程中, 程序员需要考虑两个关键因素: 第 1, 编程模型的抽象模型的抽象层次应该足够高以至于可以将硬件细节抽象化, 如此一来, 即使程序员不了解任何硬件知识, 仍然可以使用它. 第 2, 编程模型的抽象应该显式地表达那些能够提高目标应用性能的特征, 这些特征至少应包括以下 5 方面.

(1) 独立的任务级并行性易于使用, 并且不影响其他并行性. 因此编程模型设计应支持这种并行性.

(2) 字级并行在计算密集型应用程序中很常见, 并且大多数 CGRA 硬件支持字级并行.

(3) 位级并行在某些加解密应用程序 (如 SHA) 中可能存在, 但与常规 CGRA 粒度不兼容. 因此, 如果应用需要, 需要对整个设计进行大量修改实现这种并行性. 当然, 如果针对密码学的混合粒度 CGRA 能够支持位级并行, 那么这种并行性将带来很大益处.

(4) 各种应用中的数据局部性是多种多样的. 如果在编程中可以显式地表达数据的局部性, 然后通过空间分布的有效的 CGRA 互连实现, 则外部存储器带宽的负担将减轻.

(5) 应分析目标领域的内存访问模式, 以帮助减轻底层架构设计中可能的内存限制.

总而言之, 编程模型的设计是 CGRA 设计流程的第 1 步. 如何设计一个面向程序员或面向应用的有效编程模型是 CGRA 设计者应首先考虑的问题.

6.2 趋势 2: 多种并行性融合的 CGRA 计算模型

ILP 指的是并行地执行一个任务中多个独立的指令或者操作. 这种并行可以通过很多架构技术实现. 例如, 在通用处理器中, 指令流水线技术可以重叠多个指令同时执行. CGRA 的 ILP 与其他架构的 ILP 主要区别在于面积和功耗. 第 1, CGRA 提供分布式互连, 这比 CPU, GPU, DSP 中的多端口寄存器堆能效更高、功耗更小; 第 2, CGRA 同时依赖于编译器和数据流调度方法, 因此消耗的面积和功耗比乱序处理器更小.

DLP 指的是在一个大的数据结构的不同子集上执行相同的操作. 该术语通常指 SIMD 计算, 例如由 GPGPU 执行的计算. 与 SIMD 相比, CGRA 可以实现更高效的 DLP, 因为 CGRA 可以提供可编

程接口, 整合存储器访问并避免冗余地址生成.

任务级并行或线程级并行指的是并行地执行来自不同任务的不相关指令或操作. 最近的 CGRA 采用分布式控制方案, 使得多个任务操作可同时执行^[52]. 数据流技术可用于实现 TLP. 静态数据流使多个 DFG 能够以空间并行方式执行 (例如 TRIPS^[48]). CGRA 中的任务间通信有多种实现形式, 例如共享内存、分布式 FIFO 和电路开关互连.

推测并行指的是并行地执行多个具有依赖关系的操作, 就好像它们是独立的或不相关的一样. 目前少数 CGRA 支持指令级或基本块级的推测并行性. 部分谓词执行^[53]通过与条件计算并行地推测执行分支路径探索 ILP. 未采用路径的影响必须限制在阵列内部, 因此该方法的应用受到限制. TFlex^[54]为 CGRA 提出块级融合谓词/分支预测器, 它使用块中的所有谓词预测下一个块. 如果检测到错误预测, 则从其流水线中冲刷掉推测的块.

我们认为, 未来的 CGRA 计算模型需要融合指令级并行、内存级并行、数据级并行、任务级并行和推测并行, 通过开发应用中的多样的并行性, 充分利用 CGRA 的计算资源, 提高灵活性和能量效率. 对于数据级并行, CGRA 使用空域计算技术实现编译器静态探索的 ILP, 同时 CGRA 可以使用数据流技术动态探索编译器无法跟踪的 ILP. 对于内存级并行, 它可以被视为一种特殊类型的 ILP, 在 CGRA 上可以使用数据流技术实现. 对于数据级并行, CGRA 可以为数据重用定制高效内存访问原语. 对于任务级并行, CGRA 可以利用数据流技术实现细粒度 TLP, 避免了复杂的多发射和多线程逻辑开销, 还提供了比通常的共享内存更高能效的任务间通信. 对于推测并行, CGRA 需要探索 PE 阵列和独立 PEs (或具有集中控制方案的 CGRA 的外部控制器) 中的推测并行性, 因为单独采用这些方法都不能在所有情况下实现更好的性能. CGRA 需要动态地考虑控制流的总执行时间等于预测配置的执行时间、重新配置代价乘以错误预测率、替代配置的执行时间乘以错误预测率之和, 启发式地选择在 PE 阵列层次和独立 PE 中开发推测并行.

6.3 趋势 3: 面向非规则性的 CGRA 流式执行机制

通用处理器的并发执行机制以线程为中心. 多个线程竞争性地访问共享数据, 导致了大量非规则数据访问. 尽管线程的抽象可以以有限状态机或流水线的形式移植到 CGRA, 但先前的研究表明, 将这些同步原语的语义直接移植到 CGRA 可能会导致远低于通用处理器的性能^[55~57].

直接移植基于线程的非规则应用非常困难. 我们可以换一个角度, 以数据流为中心, 参考规则应用如何在 CGRA 上实现. 目前规则应用可通过流式处理, 抽象为数据流上的一系列操作. 规则应用编程范式的核心难点是如何针对规则应用中的复杂数据流访存行为, 结合动态重构存储系统的特性扩展现有的流式处理编程范式^[58]. 而在处理面向以图计算为代表的非规则应用时, 流式处理会带来大范围的随机访存, 严重影响系统性能, 需要考虑采用对数据进行分块处理的方法^[59]. 非规则应用的编程范式的核心难点是如何利用数据分块减少随机访存的范围, 通过存储系统的动态重构充分复用分块.

我们认为, 流式执行机制在 CGRA 处理非规则应用时也有很好的普适性. 未来 CGRA 执行机制设计可以利用可重构芯片存储系统的灵活性, 拓展流式计算范式的应用领域: 通过在开发框架中利用片上存储控制器的动态重构功能, 使控制器可以按照程序员的需求检测数据传输状态, 从而更好地支持大规模流式计算并发, 实现栅栏同步等复杂规则应用所需的数据访问原语; 通过在开发框架中支持存储系统的部分重构, 片上的存储结构可以灵活地在流数据缓存和 cache 间切换, 从而支持在非规则应用中对各个数据分块的选择性重用.

7 结束语

随着集成电路工艺尺寸缩小、新的应用需求的不断涌现, 以及快速迭代的软件对硬件可编程性的要求, 以 CGRA 为代表的可重构芯片在功能效率、灵活性核可编程性方面的优势相比当前主流计算芯片更加显著. 可重构芯片将在主流计算架构中扮演更重要角色.

参考文献

- 1 Bohr M. A 30 year retrospective on Dennard's MOSFET scaling paper. *IEEE Solid-State Circ News*, 2007, 12: 11–13
- 2 Hartenstein R W, Hirschbiel A G, Riedmuller M, et al. A novel ASIC design approach based on a new machine paradigm. *IEEE J Solid-State Circ*, 1991, 26: 975–989
- 3 Horowitz M. 1.1 computing's energy problem (and what we can do about it). In: *Proceedings of IEEE International Solid-state Circuits Conference (ISSCC)*, 2014
- 4 Liu L, Li Z, Yang C, et al. HReA: an energy-efficient embedded dynamically reconfigurable fabric for 13-Dwarfs processing. *IEEE Trans Circ Syst II*, 2018, 65: 381–385
- 5 Nicol C. A coarse grain reconfigurable array (CGRA) for statically scheduled data flow computing. *Wave Computing White Paper*, 2017. https://wavecomp.ai/wp-content/uploads/2018/12/WP_CGRA.pdf
- 6 Putnam A, Caulfield A M, Chung E S, et al. A reconfigurable fabric for accelerating large-scale datacenter services. *IEEE Micro*, 2015, 35: 10–22
- 7 Ouyang J, Lin S, Qi W, et al. SDA: software-defined accelerator for large-scale DNN systems. In: *Proceedings of IEEE Hot Chips 26 Symposium (HCS)*, 2016
- 8 Chen F, Shan Y, Zhang Y, et al. Enabling FPGAs in the cloud. In: *Proceedings of the 11th ACM Conference on Computing Frontiers*, 2014. 1–10
- 9 Tessier R, Pocek K L, DeHon A. Reconfigurable computing architectures. *Proc IEEE*, 2015, 103: 332–354
- 10 Bondalapati K, Prasanna V K. Reconfigurable computing systems. *Proc IEEE*, 2007, 90: 1201–1217
- 11 Qadeer W, Hameed R, Shacham O, et al. Convolution engine: balancing efficiency and flexibility in specialized computing. *Commun ACM*, 2015, 58: 85–93
- 12 Taylor M B. Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse. In: *Proceedings of Design Automation Conference*, 2012. 1131–1136
- 13 Dennard R H, Gaensslen F H, Yu H N, et al. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE J Solid-State Circ*, 1974, 9: 256–268
- 14 Sutter H, Larus J. Software and the concurrency revolution. *Queue*, 2005, 3: 54–62
- 15 Fuchs A, Wentzlaff D. The accelerator wall: limits of chip specialization. In: *Proceedings of 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Washington, 2019. 1–14
- 16 Estrin G. Organization of computer systems: the fixed plus variable structure computer. In: *Proceedings of Western Joint IRE-AIEE-ACM Computer Conference*, New York, 1960. 33–40
- 17 DeHon A, Adams J, Delorimier M, et al. Design patterns for reconfigurable computing: field-programmable custom computing machines. In: *Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2004. 13–23
- 18 Prabhakar R, Zhang Y, Koeplinger D, et al. Plasticine: a reconfigurable architecture for parallel patterns. In: *Proceedings of ACM/IEEE International Symposium on Computer Architecture*, 2017
- 19 Nowatzki T, Gangadhar V, Ardalani N, et al. Stream-dataflow acceleration. In: *Proceedings of ACM/IEEE Interna-*

- tional Symposium on Computer Architecture, 2017
- 20 Hartenstein R. A decade of reconfigurable computing: a visionary retrospective. In: Proceedings of the Conference on Design, Automation and Test in Europe, 2001
 - 21 DeHon A. Fundamental underpinnings of reconfigurable computing architectures. *Proc IEEE*, 2015, 103: 355–378
 - 22 Li Z S. Research on programming model and architectural design for reconfigurable processor with high flexibility. Dissertation for Ph.D. Degree. Beijing: Tsinghua University, 2018 [李兆石. 高灵活可重构处理器的编程模型和硬件架构关键技术研究. 博士学位论文. 北京: 清华大学, 2018]
 - 23 Liu L, Zhu J, Li Z, et al. A survey of coarse-grained reconfigurable architecture and design. *ACM Comput Surv*, 2020, 52: 1–39
 - 24 Zain-ul-Abdin, Svensson B. Evolution in architectures and programming methodologies of coarse-grained reconfigurable computing. *Microprocessors Microsyst*, 2009, 33: 161–178
 - 25 Maggs B M, Matheson L R, Tarjan R E. Models of parallel computation: a survey and synthesis. In: Proceedings of the 28th Hawaii International Conference on System Sciences, 1995
 - 26 Asanovic K, Bodik R, Catanzaro B C, et al. The Landscape of Parallel Computing Research: A View from Berkeley. Technical Report UCB/EECS-2006-183. 2006
 - 27 Liu L, Wang D, Yin S, et al. SimRPU: a simulation environment for reconfigurable architecture exploration. *IEEE Trans VLSI Syst*, 2014, 22: 2635–2648
 - 28 Mei B, Vernalde S, Verkest D, et al. ADRES: an architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix. In: Proceedings of International Conference on Field Programmable Logic and Application (FPL), 2003
 - 29 Baumgarte V, Ehlers G, May F, et al. PACT XPP—a self-reconfigurable data processing architecture. *J Supercomput*, 2003, 26: 167–184
 - 30 Liu L, Wang D, Zhu M, et al. An energy-efficient coarse-grained reconfigurable processing unit for multiple-standard video decoding. *IEEE Trans Multimedia*, 2015, 17: 1706–1720
 - 31 Dutta H, Kissler D, Hannig F, et al. A holistic approach for tightly coupled reconfigurable parallel processors. *Microprocessors Microsyst*, 2009, 33: 53–62
 - 32 Watkins M A, Nowatzki T, Carno A. Software transparent dynamic binary translation for coarse-grain reconfigurable architectures. In: Proceedings of International Symposium on High Performance Computer Architecture (HPCA), 2016
 - 33 Clark N, Kudlur M, Park H, et al. Application-specific processing on a general-purpose core via transparent instruction set customization. In: Proceedings of International Symposium on Microarchitecture, 2004
 - 34 Liu F, Ahn H, Beard S R, et al. DynaSpAM: dynamic spatial architecture mapping using out of order instruction schedules. In: Proceedings of ACM/IEEE International Symposium on Computer Architecture, 2015
 - 35 Elbirt A J, Paar C. An instruction-level distributed processor for symmetric-key cryptography. *IEEE Trans Parallel Distrib Syst*, 2005, 16: 468–480
 - 36 Fronte D, Perez A, Payrat E. Celator: a multi-algorithm cryptographic co-processor. In: Proceedings of 2008 International Conference on Reconfigurable Computing and FPGAs, 2008
 - 37 Mei B, Veredas F J, Masschelein B. Mapping an H.264/AVC decoder onto the ADRES reconfigurable architecture. In: Proceedings of International Conference on Field Programmable Logic and Applications, 2005
 - 38 Hartmann M, Pantazis V V, Aa T V, et al. Still image processing on coarse-grained reconfigurable array architectures. *J Sign Process Syst*, 2010, 60: 225–237

- 39 Novo D, Moffat W, Derudder V, et al. Mapping a multiple antenna SDM-OFDM receiver on the ADRES coarse-grained reconfigurable processor. In: *Proceedings of IEEE Workshop on Signal Processing Systems Design and Implementation*, 2005
- 40 Palkovic M, Cappelle H, Glassee M, et al. Mapping of 40 MHz MIMO SDM-OFDM baseband processing on multi-processor SDR platform. In: *Proceedings of IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, 2008
- 41 Tu F, Yin S, Ouyang P, et al. Deep convolutional neural network architecture with reconfigurable computation patterns. *IEEE Trans VLSI Syst*, 2017, 25: 2220–2233
- 42 Yin S, Ouyang P, Tang S, et al. A 1.06-to-5.09 TOPS/W reconfigurable hybrid-neural-network processor for deep learning applications. In: *Proceedings of Symposium on VLSI Circuits*, 2017
- 43 Farabet C, Martini B, Corda B, et al. NeuFlow: a runtime reconfigurable dataflow processor for vision. In: *Proceedings of Computer Vision and Pattern Recognition Workshops*, 2011
- 44 Li B, Tan K, Luo L, et al. ClickNP: highly flexible and high performance network processing with reconfigurable hardware. In: *Proceedings of Conference on ACM SIGCOMM*, 2016
- 45 Cong J, Liu B, Neuendorffer S, et al. High-level synthesis for FPGAs: from prototyping to deployment. *IEEE Trans Comput-Aided Des Integr Circ Syst*, 2011, 30: 473–491
- 46 Wang Z, He B, Zhang W, et al. A performance analysis framework for optimizing OpenCL applications on FPGAs. In: *Proceedings of IEEE International Symposium on High PERFORMANCE Computer Architecture*, 2016
- 47 Windh S, Ma X, Halstead R J, et al. High-level language tools for reconfigurable computing. *Proc IEEE*, 2015, 103: 390–408
- 48 Sankaralingam K, Nagarajan R, Liu H M, et al. TRIPS: a polymorphous architecture for exploiting ILP, TLP, and DLP. *ACM Trans Archit Code Opt*, 2004, 1: 62–93
- 49 Park H, Park Y, Mahlke S. Polymorphic pipeline array: a flexible multicore accelerator with virtualized execution for mobile multimedia applications. In: *Proceedings of IEEE/ACM International Symposium on Microarchitecture*, 2009
- 50 Robotmili B, Li D, Esmaeilzadeh H, et al. How to implement effective prediction and forwarding for fusible dynamic multicore architectures. In: *Proceedings of IEEE International Symposium on High PERFORMANCE Computer Architecture*, 2013
- 51 Li Z, Liu L, Deng Y, et al. Aggressive pipelining of irregular applications on reconfigurable hardware. In: *Proceedings of 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, 2017. 24–28
- 52 Pager J, Jeyapaul R, Shrivastava A. A software scheme for multithreading on CGRAs. *ACM Trans Embed Comput Syst*, 2015, 14: 1–26
- 53 Chang K, Choi K. Mapping control intensive kernels onto coarse-grained reconfigurable array architecture. In: *Proceedings of International SoC Design Conference*, 2009
- 54 Kim C, Sethumadhavan S, Govindan M S, et al. Composable Lightweight Processors. In: *Proceedings of IEEE/ACM International Symposium on Microarchitecture*, 2007
- 55 Ramanathan N, Fleming S T, Wickerson J, et al. Hardware synthesis of weakly consistent C concurrency. In: *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, New York, 2017. 169–178
- 56 Ramanathan N, Wickerson J, Winterstein F, et al. A case for work-stealing on FPGAs with OpenCL atomics. In: *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. New York: ACM, 2016. 48–53

- 57 Winterstein F, Bayliss S, Constantinides G A. High-level synthesis of dynamic data structures: a case study using Vivado HLS. In: Proceedings of International Conference on Field-Programmable Technology (FPT), 2013. 362–365
- 58 Thomas J, Hanrahan P, Zaharia M. Fleet: a framework for massively parallel streaming on FPGAs. In: Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems, Lausanne, 2020. 639–651
- 59 Zhou S, Kannan R, Prasanna V K, et al. HitGraph: high-throughput graph processing framework on FPGA. IEEE Trans Parallel Distrib Syst, 2019, 30: 2249–2264

Reconfigurable computing: toward software defined chips

Shaojun WEI*, Zhaoshi LI, Jianfeng ZHU & Leibo LIU

Institute of Microelectronics, Tsinghua University, Beijing 100084, China

* Corresponding author. E-mail: wsj@tsinghua.edu.cn

Abstract With the rapid development of technology, software is rapidly evolving with emerging applications. Chips that fail to adapt to software (e.g., application-specific integrated circuits, ASICs) suffer from a short lifecycle and high nonrecurring engineering (NRE) costs. Meanwhile, as the projection of Moore's law and Dennard scaling are decreasing, energy efficiency has shown a diminishing return with new technologies. The computing capacity of general-purpose processors is limited due to power budgets. Consequently, future chips must jointly optimize flexibility, power efficiency, and ease of programmability. Reconfigurable chips combine the high flexibility of a general-purpose processor and high energy efficiency of ASIC by providing on-demand customization of their architectures. This article thoroughly reviews the development and architecture of reconfigurable chips. Moreover, the future challenges of reconfigurable chips are analyzed. Based on these challenges, future directions are also discussed.

Keywords software defined chips, reconfigurable computing, domain-specific architecture, energy efficiency, programmability

Shaojun WEI was born in 1958 in Beijing, China. He received his Ph.D. degree from the Faculte Polytechnique de Mons, Belgium, in 1991. He became a professor at the Institute of Microelectronics of Tsinghua University in 1995. He is a senior member of the Chinese Institute of Electronics (CIE) and a fellow of IEEE. His main research interests include VLSI SoC design, EDA methodology, and communication ASIC design.

Zhaoshi LI received his B.S. degree in Electronic Engineering from Tsinghua University, Beijing, China, in 2012 and his Ph.D. degree from the Institute of Microelectronics, Tsinghua University, in 2018. His current research interests include programming models and programming models for reconfigurable computing.

Jianfeng ZHU received his B.S. degree in Electronic Engineering from Tsinghua University, Beijing, China, in 2008 and his Ph.D. degree from the Institute of Microelectronics, Tsinghua University, in 2014. His current research interests include reconfigurable computing, low-power design, and design for testability.

Leibo LIU received his B.S. degree in Electronic Engineering from Tsinghua University, Beijing, China, in 1999, and his Ph.D. degree from the Institute of Microelectronics, Tsinghua University, in 2004. He is currently a professor at the Institute of Microelectronics, Tsinghua University. His research interests include reconfigurable computing, mobile computing, and VLSI DSP.