arXiv:2301.04767v1 [cs.AR] 12 Jan 2023

RAD-Sim：新型可重构加速设备的快速架构探索

Andrew Boutros[1,2],Eriko Nurvitadhi[2] 和 Vaughn Betz[1]1多伦多大学和人工智能向量研究所 2可编程解决方案事业部，英特尔公司 电子邮件：andrew.boutros@mail.utoronto.ca，eriko.nurvitadhi@intel.com，vaughn@eecg.utoronto.ca

摘要——**随着现场可编程门阵列（现场可编程门阵列）的容量不断增长，并被应用于数据中心等新环境，我们见证了一类新型可重构加速设备（RAD）的出现，它们突破了传统 FPGA 架构的局限。这些设备将可重构逻辑结构与粗粒度领域专用加速器模块相结合，并通过高性能分组交换片上网络（NoC）连接，实现高效的全系统通信。然而，目前缺少必要工具，无法高效探索 RAD 庞大的设计空间、研究其不同组件之间复杂的相互作用，并评估各种设计选择组合。在这项工作中，我们开发了 RAD-Sim，这是一款周期级架构模拟器，可支持以应用为导向，快速探索新型 RAD 的设计空间。为展示 RAD-Sim 的能力，我们将一个先进的深度学习（深度学习）推理覆盖层映射到一个 RAD 实例并进行模拟。该实例包含现场可编程门阵列逻辑 fabric 和一组硬化矩阵-向量乘法引擎，各组件通过全系统片上网络进行通信。通过这一示例，我们展示了 RAD-Sim 如何帮助架构师量化特定架构参数的变化对端到端应用性能的影响。**

索引词—**现场可编程门阵列、片上网络、加速器模块、架构模拟器、深度学习**

I. 引言

过去三十年来，现场可编程门阵列（FPGA）已显著演进，从简单的可重构逻辑与布线阵列，发展为集成了片上存储器（块 RAM）、数字信号处理模块和高速收发器的复杂异构器件 [1]。近年来，我们见证了超越 *FPGA* 的可重构加速设备（RAD）的兴起。这类设备将传统 FPGA 逻辑 fabric 与多个粗粒度专用加速器模块相结合，并通过高性能片上网络（NoC）进行通信，如图 1 所示；Xilinx Versal 架构便是其中一个典型示例 [2]。随着多芯粒集成技术的发展，RAD 还可跨越多个芯粒，并由系统级片上网络充当芯粒之间连续的通信平面。

RAD 中这些不同组件的组合形成了巨大的设计空间，也引出了诸多研究问题：鉴于各组件之间复杂的相互作用，我们应如何设计这些器件？尽管多年来人们已对 FPGA 逻辑结构架构进行了广泛研究，但用于探索和评估逻辑结构架构的工具与方法，并不适用于新型 RAD 的架构探索。首先，这些工具和方法依据与应用无关的

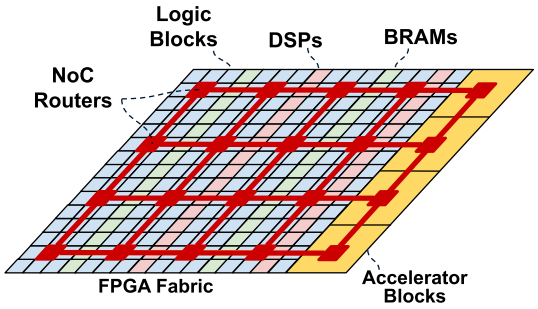

图 1：RAD 实例示例，包含传统的 FPGA 逻辑 fabric、一侧的粗粒度加速器模块组，以及用于全系统通信的分组交换硬片上网络。

性能指标评估候选逻辑结构架构，例如基准电路的最高工作频率。对于采用粗粒度加速器模块和延迟不敏感型 NoC 通信的 RAD，所用性能指标必须超越 FPGA 逻辑 fabric 上实现的逻辑的工作频率，并能反映端到端应用性能。

其次，FPGA架构探索流程主要由以硬件描述语言（HDL）编写的基准测试驱动，并依赖寄存器传输级（RTL）仿真进行功能验证。为对应用和片上网络路由器、硬加速器模块等系统组件进行系统级仿真，以开展功能验证和性能估算，需要开发大量 RTL 基础设施。这种缓慢且耗费人力的流程不仅阻碍了对 RAD架构进行广泛探索，也限制了架构师协同优化应用与 RAD 平台的能力。最后，RAD架构探索工具还需要评估新指标，例如在拟议架构上运行不同应用时的片上网络流量和拥塞情况。

在这项工作中，我们首先介绍 RAD-Sim，这是一款面向新型 RAD 的系统级应用驱动架构模拟器，可集成不同的片上网络、加速器模块和可编程逻辑模块。RAD-Sim 的输入包括应用模块和加速器模块的高层 SystemC 描述，以及可重构加速器设计架构参数、片上网络规格和路由器布局约束。它执行系统级仿真，并生成端到端应用性能和片上网络流量报告。此外，提供用户指定的测试输入和预期输出后，还可用它对在给定 RAD 实例上实现的应用进行功能验证。随后，我们展示一个示例设计，以说明

RAD-Sim 的功能：我们将最先进的深度学习（DL）推理现场可编程门阵列覆盖层——神经处理单元（NPU）——映射到一个示例 RAD 实例，该实例集成了现场可编程门阵列、硬化矩阵向量乘法加速器模块和系统级片上网络。本文的贡献如下：

- RAD-Sim 是一款开源工具[1]，用于快速探索采用 FPGA架构、加速器模块和系统级片上网络的新型 RAD 的架构。
- 一个来自深度学习领域的设计示例，展示了 RAD-Sim 如何帮助架构师量化不同设计选择对端到端应用性能的影响。

II. 背景与相关工作

A. RAD 的兴起

在许多 FPGA 数据中心部署中，FPGA 位于不同服务器端点之间的数据传输路径上。Microsoft Catapultv2 项目 [3] 将 FPGA 部署在网络与服务器CPU之间，充当线路中间的透明设备。在这种场景下，可以将不同的网络功能（例如数据包处理和密码学）卸载到 FPGA 上，从而释放中央处理器资源。此外，连接到网络的 FPGA 构成了一个同构的数据中心规模加速平面，可灵活重新配置，以加速深度学习工作负载等各种关键数据中心应用 [4]。在这些部署中，FPGA 的价值不仅来自其可重构逻辑，也来自其高带宽输入输出。

然而，关键工作负载的数据流持续增长，给细粒度可编程路由架构带来了压力，尤其是在 FPGA 连接多个高带宽外部接口时。先前的研究表明，采用硬化的分组交换片上网络可以缓解这些片上带宽挑战 [5], [6]。此外，关键应用中的一些计算操作在许多工作负载中都很常见，将它们硬化为粗粒度加速器模块可以显著提高效率。以深度学习加速为例，不同工作负载中的层组成、层间数据处理、向量运算以及预处理和后处理阶段可能存在显著差异。然而，它们都包含大量点积运算，将其硬化为高性能张量核心可以提高效率 [7]。

由于这些趋势，我们开始见证超越 FPGA 的 RAD 的兴起。这类设备兼具 FPGA 的灵活性、硬化片上网络在数据调度方面的效率，以及专用加速器模块的高性能。Xilinx Versal 架构就是一种 RAD，它结合了传统的可重构 FPGA 逻辑 fabric、通用 ARM内核和用于深度学习加速的向量处理器，所有这些组件都通过全系统片上网络进行通信 [2]。

B. 传统 FPGA 架构探索流程

FPGA架构探索工具（例如 VTR [8],）已在 FPGA 研究社区中得到广泛应用。典型的 FPGA 架构探索流程由三个主要部分组成：(1) 一套代表 FPGA 应用领域关键特征的基准电路 [9], [10]；(2) 一种架构描述，用于定义 FPGA 模块、布线架构及其面积/延迟模型；以及 (3) 一个可重定向的计算机辅助设计系统，能够将给定的基准测试集映射到指定的 FPGA 架构，并生成面积、时序和功耗指标。该流程仅关注 FPGA 逻辑 fabric 的设计，主要依据与应用无关的指标，例如基准电路的最高工作频率或底层现场可编程门阵列电路的面积成本。这不足以探索和评估包含其他复杂组件（例如片上网络和硬加速器模块）的 RAD架构，也无法生成片上网络拥塞和应用吞吐量等关键系统级信息。虽然也有片上网络模拟器 [11], ，但由于缺少模拟耦合 FPGA fabric 的功能，它们同样无法全面评估 RAD。

C. 架构模拟器

架构模拟器广泛用于对经典冯·诺依曼架构以及新兴计算技术进行快速架构探索。例如，gem5 [12]模拟器能够对现代中央处理器进行高保真周期级建模，并可针对不同指令集架构运行完整应用。GPGPU-Sim [13] 是另一款面向当代 Nvidia GPU架构的学术模拟器，可运行 CUDA 或 OpenCL 工作负载，并支持张量核心和 CUDA 动态并行等高级功能。SIAM [14] 是一款较新的模拟器，专注于面向深度神经网络的新兴基于芯粒的存内计算。它整合了架构、片上网络、封装内网络和动态随机存取存储器模型，以模拟端到端系统。此外，人们通常也会构建专用架构模拟器，以评估定制加速器架构，例如 in [15]–[17]。我们的 RAD-Sim 与上述模拟器一样，采用应用驱动的架构探索方法，但专注于可重构计算领域。与 gem5 或 GPGPU-Sim 等其他模拟器不同，评估 RAD架构时，模拟器的输入并非只有编译后的应用指令，而可以是软件可编程粗粒度加速器模块的指令与在 FPGA 逻辑 fabric 上实现的用户自定义模块的混合体。另一个关键区别在于，由于现场可编程门阵列可重构性，计算模块的布局及其与片上网络路由器的连接方式都具有灵活性（即可以在应用设计时进行编程）。

III. RAD架构探索流程

A. 流程概述

图 2 展示了我们完整的 RAD 架构评估流程概览，该流程由三个主要部分组成。第一部分也是本文的重点是 RAD-Sim，它支持快速开展 RAD 设计空间探索，并评估不同 RAD 组件的设计选择之间的相互作用。其输入包括以架构参数、片上网络规格以及一组 RAD 硬加速器模块的 SystemC 模型形式提供的 RAD 架构描述。此外，它还接收另一组应用模块的 SystemC 模型，这些模块将在 FPGA 逻辑 fabric 上实现；如果这些模块需要访问系统级片上网络，还需提供它们与特定片上网络路由器的分配关系。

[1]代码可从以下地址下载：https://github.com/andrewboutros/rad-flow

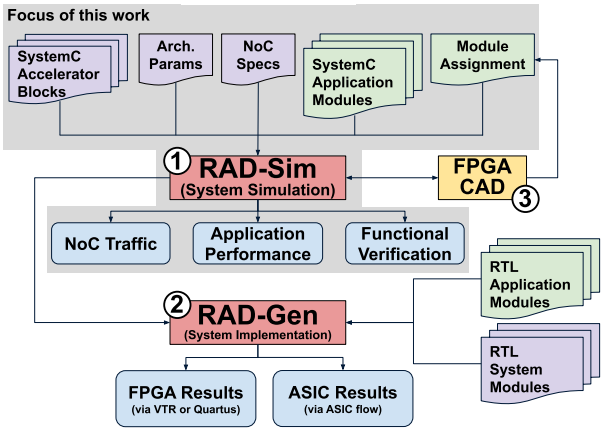

图 2：RAD架构探索与评估流程。

随后，它会对整个系统进行周期级仿真，以生成应用性能结果和片上网络流量报告。提供测试输入和预期输出集合后，它还可用于验证映射到指定 RAD 的应用功能。在架构探索的早期阶段协同设计 RAD 和应用时，这一功能尤为实用。

使用 RAD-Sim 快速缩小目标应用的设计空间后，我们可以使用流程的第二个组件 RAD-Gen，对少数候选 RAD架构进行更详细的评估。该工具会为完整系统生成骨架 RTL代码，其中包括片上网络路由器、适配器和模块封装器，设计人员可以在其中加入应用模块和硬加速器模块的 RTL 实现。随后，它会将可编程逻辑上实现的设计部分通过 FPGA CAD 流程[2]，以获取设计的最高工作频率和资源利用率。它还会将片上网络路由器和任何硬加速器模块通过 ASIC实现流程，以获得硅面积和时序结果。

我们流程的第三个也是最后一个组成部分，是传统 FPGA CAD 工具与 RAD-Sim 之间的连接。FPGA 上的硬化片上网络给布局带来了新的挑战：模块不仅必须放置在具有充足可编程逻辑资源、且能最大限度减少传统可编程布线的位置，还必须确保其与附近路由器上的片上网络适配器相连时不会造成过度的片上网络拥塞。RAD-Sim 可充当预言机，在布局期间评估可编程逻辑模块与特定路由器之间的连接。例如，FPGA CAD 工具可以提出特定的模块分配方案，并将其连同用户指定的预期片上网络流量模式一起传递给 RAD-Sim。随后，RAD-Sim 可以快速模拟这一场景，并生成报告，给出不同流量流的预期延迟。布局引擎可依据该报告调整模块分配方案；如果未满足延迟约束，还可以再次迭代。这类似于在传统 FPGA CAD 流程的布局阶段调用静态时序分析。本研究仅关注第一个组成部分——

[2]VTR 可以直接对嵌入式路由器建模；要在 Quartus 中对其建模，我们会在适当的位置创建具有适当尺寸的预留逻辑锁定区域。

RAD-Sim。第二和第三个组成部分仍在开发中，将在未来的研究中介绍。

B. RAD-Sim 实现细节

RAD-Sim 使用 SystemC 开发，使设计人员能够以不同抽象级别对硬加速器模块和应用模块建模，在模型保真度与设计效率之间进行权衡。例如，特定模块可以用 SystemC 以高级行为级描述方式进行描述，以便快速开发；也可以采用更详细（更接近 RTL）的方式描述，并将其输入高级综合工具以生成硬件。RAD-Sim 使用 BookSim 2.0 [11] 执行周期精确的片上网络仿真。BookSim 是一款开源片上网络模拟器，已被许多系统模拟器（如 GPGPU-Sim）采用。它具有丰富的参数配置，可用于模拟各种具有不同拓扑结构、路由函数、仲裁机制和路由器微架构的互连网络。

RAD-Sim 在三个主要方面建立在 BookSim 之上。首先，RAD-Sim 为 BookSim 添加了 SystemC 封装层，使设计人员能够轻松地将片上网络与采用 SystemC 建模的不同加速器模块和应用模块结合起来。其次，它通过跟踪数据包内容来补充 BookSim，从而支持对 RAD 上实际应用进行功能验证。之所以需要这样做，是因为 BookSim 主要关注性能估算，因此对数据包到达时间进行建模，而不关注其内容。最后，RAD-Sim 还实现了 SystemC 片上网络适配器，使 RAD 架构师能够独立于底层片上网络协议，尝试不同的面向用户的片上网络抽象。这些适配器还负责在应用模块或硬加速器模块与片上网络之间进行时钟域跨越和位宽适配。例如，我们为用户提供 AXI 流式接口（AXI-S）和 AXI 存储器映射接口（AXI-MM）适配器；同时，RAD-Sim 的模块化架构也使架构师能够实现自定义或标准化的片上网络适配器协议，并将其轻松集成到模拟器中。

图 3 以 RAD-Sim 中实现的 AXI-S 主从片上网络适配器为例进行了展示。它们由三个主要阶段组成：模块接口、编码/解码和片上网络接口。对于从设备适配器，输入仲裁器会从连接到同一路由器的 AXI-S 接口（可能有多个）中选择一个。事务缓冲后，会被分组化为若干片上网络流控单元，并映射到特定的片上网络虚拟通道（VC）。随后，根据路由器通道仲裁和交换分配机制，这些流控单元会被推入异步 FIFO，以便注入片上网络。主设备适配器的工作方式类似，但方向相反：流控单元从片上网络弹出；收到尾部流控单元后，它们会被解组为 AXI-S 事务，随后转发到预期的模块接口。RAD-Sim 中实现的适配器均可参数化，从而支持对不同的仲裁机制、虚拟通道映射表以及 FIFO/缓冲区大小进行实验。它们还支持多达三个不同的时钟域，在这些时钟域中，相连的模块、适配器和片上网络均以不同的时钟频率运行。

表 I 列出了一些用户可以调整的参数，以试验不同的 RAD架构。其他更

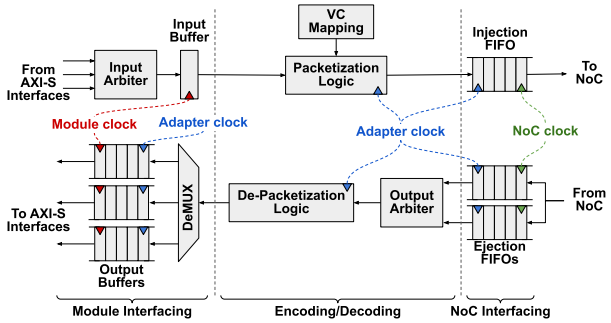

图 3：AXI-S 从设备（上）和主设备（下）片上网络适配器。

表 I：RAD-Sim 架构参数。

用户输入	描述
num_nocs	全系统片上网络的数量
片上网络_载荷_位宽	流控单元载荷的片上网络链路位宽
片上网络_频率	片上网络工作频率
片上网络_拓扑	片上网络拓扑（例如网格、环面）
片上网络维度_	片上网络维度（适用于某些拓扑结构）
片上网络_布线_函数	片上网络路由算法（例如 XY、最少跳数）
noc_vcs	片上网络虚拟通道数量
片上网络虚通道缓冲区大小_ _ _	虚拟通道缓冲区深度（字）
适配器_接口	每个适配器连接的接口数量
适配器_fifo_大小	适配器弹出/注入 FIFO 深度
适配器_obuff_大小	适配器输出缓冲区深度（字）
适配器_输入_仲裁器	适配器输入仲裁机制
适配器_输出_仲裁器	适配器输出仲裁机制
适配器_虚拟通道_映射	Flit 类型到虚拟通道的映射
适配器_频率	适配器工作频率
模块_频率	每个模块的工作频率
轨迹数量_	记录的事件轨迹数
轨迹 _名称	已记录事件轨迹的标识符

详细的 NoC 特定选项，例如延迟参数、路由器微架构以及交换机/VC 分配机制，也可以直接通过 BookSim 配置文件进行指定。此外，RAD-Sim 接受一个模块分配文件作为输入，该文件指定了所有硬加速器模块和可编程逻辑模块在 NoC 中的布局（即每个模块/模块端口连接到哪个片上网络路由器）。目前，这一分配由用户手动指定。不过，也可以将其自动化，以满足用户指定的流量延迟约束，或优化整体应用性能。如第 III-A 节所述，FPGA CAD 流程可以调整 FPGA 逻辑 fabric 上实现的模块在 NoC 中的布局，并调用 RAD-Sim，以量化这些调整对整体性能的影响。

此外，RAD-Sim 还提供遥测工具，用于记录特定的仿真事件和轨迹，并配有不同的脚本来可视化收集到的数据。这有助于分析 RAD 各组件之间复杂的交互，并了解改变各种架构参数对系统整体性能的影响。图 4 展示了 RAD-Sim 生成的示例可视化结果，用于表征采用 4×4 网格片上网络、且每个路由器连接两个模块的 RAD 在无负载情况下的通信延迟。在此示例实验中，

单个模块依次向连接到每个路由器的第一个模块发送两个 AXI-MM 事务（15 个路由器对应 × 2 个事务），期间片上网络中没有其他流量。随后

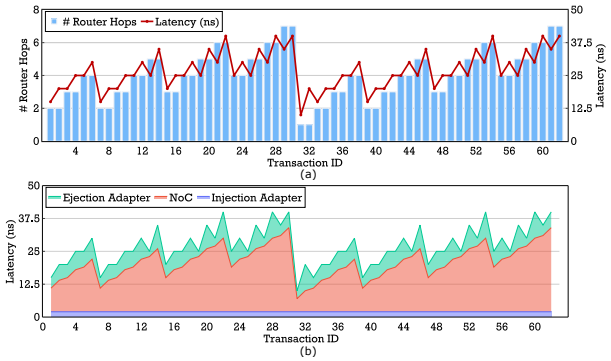

图 4：RAD-Sim 为无负载的 4×4 网格片上网络生成的示例可视化结果，展示：(a) 整体通信延迟和跳数；(b) 延迟细分。

对连接到每个路由器的第二个模块重复上述操作。模块、适配器和片上网络的工作频率分别设为 200 MHz、800 MHz 和 1 GHz。使用 RAD-Sim 遥测工具记录事务生命周期中的各类时间戳，例如源模块发起事务、分组化、注入/弹出、解分组以及目的模块接收事务的时间。图 4a 显示了 62 个已发起事务各自的延迟（纳秒）和经过的片上网络路由器跳数。图中显示，随着源模块与目的模块之间的距离增加，跳数和通信延迟也随之增加；当移动到路由器网格的下一行时，这些数值则会下降。图 4b 展示了 RAD-Sim 生成的另一种可视化结果，将每个事务的延迟细分为注入适配器、片上网络和弹出适配器所耗费的时间。这有助于凸显在试验不同适配器实现和协议时引入的开销。

IV. 神经网络处理器示例设计

A. 神经网络处理器（*NPU*）叠加层

在本研究中，我们将神经处理单元叠加层作为深度学习应用领域的一项关键基准。NPU 是一款先进的 FPGA 软处理器，面向多层感知机（MLP）、循环神经网络（RNN）、门控循环单元（GRU）和长短期记忆模型（LSTM）等内存密集型深度学习模型，旨在实现低延迟推理。它在配备深度学习优化型张量块的英特尔 Stratix 10 NX FPGA 上实现了先进的性能。平均而言，与同代 Nvidia T4 和 V100 图形处理器相比，其性能分别高出 24× 和 12× [18]。

图 5 展示了 NPU 覆盖架构的概览，该架构由五个串联的模块组成，每个模块的输出都会直接转发到下一个模块。矩阵-向量乘法单元（MVU）由 T 个瓦片组成，每个瓦片包含 D 组长度为 L 个乘法通道的 C 个点积引擎（DPE）。每个瓦片计算矩阵-向量乘法运算的一部分，随后将各瓦片的部分结果在多个时间步中归约并累加，以生成 MVU 的最终输出。接下来是外部向量寄存器文件（eVRF），用于跳过不包含矩阵-向量乘法的指令所对应的 MVU；然后是两个相同的多功能单元（MFU），用于执行向量逐元素

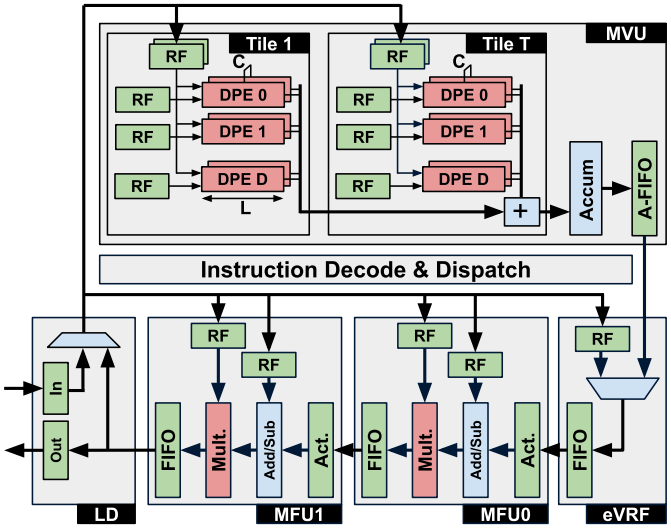

图 5：NPU 覆盖架构概览。红色标出的连接是对延迟敏感的通道。

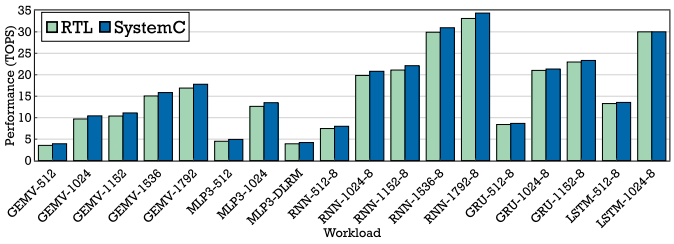

图 6：RTL 和 SystemC 仿真得出的 NPU 性能结果。

运算，例如激活函数、加法/减法和乘法。最后是加载器模块（LD），它会将流水线结果写回 NPU 的任意寄存器文件（RF），并与其他系统组件（例如其他模块或外部接口）通信。所有这些模块均由超长指令字进行协调；中央控制单元负责对这些指令进行解码，并将其分发到不同模块，详情见 [18], [19]。

B. 基准 *SystemC NPU* 模型

为了将 NPU 用作 RAD-Sim 的案例研究，我们为其各个模块开发了 SystemC 仿真模型，以便之后在 RAD-Sim 中将它们用作硬加速器模块或互连结构应用模块。这些模型采用参数化设计，因此我们可以尝试不同的 NPU 架构参数（T、D、C和 L），并根据模块底层实现细节调整模块延迟。为了评估 NPU SystemC 仿真模型的速度和准确性，我们将其与 NPU SystemVerilog 实现的周期精确 RTL 仿真进行比较。在实验中，RTL 仿真使用 Synopsys VCS v2016.06，SystemC 和 RTL 仿真均在同一台 Intel Xeon Gold 6146 24 核中央处理器上运行。我们采用与 [18] 中类似的 NPU 配置，包括 2 个核心、7 个瓦片、40 个 DPE 和 40 个通道；本文其余实验也使用此配置。我们针对多种 NPU 工作负载运行仿真，包括简单的矩阵向量乘法（矩阵向量乘法）、循环神经网络、门控循环单元、长短期记忆网络，以及不同规模的多层感知机，并在图 7 中以每秒万亿次运算（每秒万亿次运算）为单位报告结果。结果表明，我们的 SystemC 仿真模型能够以很高的准确度估算 NPU 性能：与周期精确 RTL 仿真相比，平均误差仅为 5.1%，最大误差为 10.8%。然而，SystemC 仿真平均比 RTL 仿真快 26× ，加速比根据工作负载规模介于 6.5× 和 100× 之间。这凸显了 SystemC 与 RTL 仿真之间显著的速度差异；这一差异是 RAD-Sim 的关键支柱，也增强了我们对使用该 NPU 模型生成的其余实验性能估算结果的信心。

C. 在 *RAD* 实例上映射和仿真 *NPU*

我们对 NPU 进行了修改，使其采用延迟无关接口，从而能够通过 RAD 实例的系统级片上网络连接这些接口。这样便将应用计算与模块间通信完全解耦，并提高了互连抽象级别，使我们能够探索包含硬加速器模块的复杂 RAD。此时，传统 FPGA CAD工具无需优化跨越模块边界或试图连接到硬加速器模块可编程布线接口的信号的时序和可布线性。只要每个应用模块分别满足时序要求，并且能够连接到片上网络适配器，那么在给定 RAD 实例上评估端到端应用性能，就转化为对软/硬模块和片上网络延迟进行周期级仿真；这正是 RAD-Sim 所实现的功能。

我们将神经网络处理器映射到一个示例 RAD 实例上，该实例包含 FPGA 逻辑 fabric 和一个独立的硬加速器模块群，如图 1 所示，并使用 RAD-Sim 评估其整体性能。在此例中，我们将类似于神经网络处理器中 MVU 瓦片的矩阵-向量乘法单元（见图 5）实现为硬加速器模块，这些模块只能通过片上网络从逻辑 fabric 访问。由于这些模块实现了几乎所有深度学习工作负载中共有的功能，因此是硬化的理想候选；而神经网络处理器的其余模块则可针对不同工作负载进行专门设计，以提高效率 [20]，从而受益于 FPGA 的可重构性。

我们将*FPGA* 扇区 定义为一个 FPGA 资源区域，其中心设有片上网络路由器/适配器。例如，具有 8×5 个扇区的 FPGA，其逻辑 fabric 中共有 40 个片上网络路由器/适配器。类似地，我们将*ASIC* 扇区 定义为一块硅区域，其占地面积与 FPGA 扇区相同，并包含一个硬加速器模块（可能还包含其他硬化组件）和一个片上网络路由器。本实验使用的示例 RAD 实例包含一个 8×5 的 FPGA 扇区网格，以及一侧由 2×5 个 ASIC 扇区组成的区域。所有 FPGA 扇区合计拥有与基准 Intel Stratix 10 NX 2100 器件相同的资源（702k 个自适应逻辑模块、6,847 个块 RAM、3,960 个张量块）。

我们将神经网络处理器映射到示例 RAD 实例上，并使用 RAD-Sim 评估其性能。我们将 FPGA 逻辑 fabric 的工作频率设为 300 兆赫（与 [18] 中神经网络处理器的工作频率相同），并保守地假设硬加速器模块仅以 600 兆赫运行。我们将

表 II：在 RAD FPGA 逻辑资源上实现的神经网络处理器模块的资源利用率。

ALMs	块 RAM	张量块
550,0930 (78%)	2,632 (90%)	3,200 (81%)

Stratix 10 的 14 纳米工艺技术中，28 纳米片上网络路由器的工作频率从 [21] 提高到 1.5 吉赫兹；我们假设片上网络适配器以 4× fabric 的速度运行，与 [21] 类似。在实验中，我们使用网格片上网络拓扑，其维度等于 FPGA 扇区和 ASIC 扇区的总数（即 10×5 网格），并采用 3 个虚通道和维序路由。片上网络适配器注入/弹出 FIFO 和输出缓冲区（见图 3）的深度分别设为 16 和 2。我们手动将 FPGA 逻辑 fabric 上实现的神经网络处理器向量逐元素模块（外部向量寄存器文件、乘法函数单元、加载单元、指令分发器）分配给特定的片上网络路由器，采用合理（但可能并非最优）的布局。

D. 实现结果

为确定 FPGA 资源利用率，我们使用 Intel Quartus Prime Pro 21.2，在 Stratix 10 NX 2100 器件上对映射到 FPGA 逻辑 fabric 的 NPU 模块进行综合、布局和布线。我们在适当位置为片上网络路由器和适配器预留逻辑锁定区域，将其标记为空设计分区，并根据手动将模块分配给不同路由器的方案，将 NPU 模块与这些区域连接起来。与 [22]，中使用的 3×3 LAB 区域相比，我们为每个逻辑锁定区域采用较为保守的尺寸，将其设为包含 10×10 个逻辑阵列块 (LAB) 的网格，因为我们使用的是 128 位宽链路，而 [22] 使用的是 32 位宽链路。表 II 展示了在 FPGA 逻辑 fabric 上实现的 NPU 模块的资源利用率。

我们还验证了，选定作为硬加速器模块实现的矩阵-向量乘法单元，能够利用来自 [23]、[24] 和 [25] 的 FPGA 资源、硅面积和 FPGA 到 ASIC 的面积缩放比例，适配可用的 ASIC 区块面积占用。我们的估算表明，硬矩阵-向量单元占用的 ASIC 区块面积不到 55%，因此有充足的剩余面积用于片上网络路由器、适配器、链路及任何额外的硬化功能。未来，我们流程中的 RAD-Gen 组件（见第 III-A 节）将自动完成获取 FPGA 结果所需的手动步骤，并将硬加速器模块的寄存器传输级实现送入 ASIC 设计流程，以获得精确的面积和时序结果。

E. 性能结果

图 7 展示了来自 [18] 的 Stratix 10 NX 上基准神经网络处理器与映射到我们示例 RAD 实例后的性能对比。通过利用硬化的矩阵-向量单元粗粒度加速器模块，并使用腾出的 FPGA 逻辑 fabric 资源在软逻辑中实例化更多逐元素向量引擎，基于 RAD 实现的神经网络处理器平均性能比传统 Stratix 10 NX 基准方案高出 1.32× 。RAD-Sim 还使我们能够研究不同架构参数的选择对端到端应用性能的影响。图 8 展示了改变

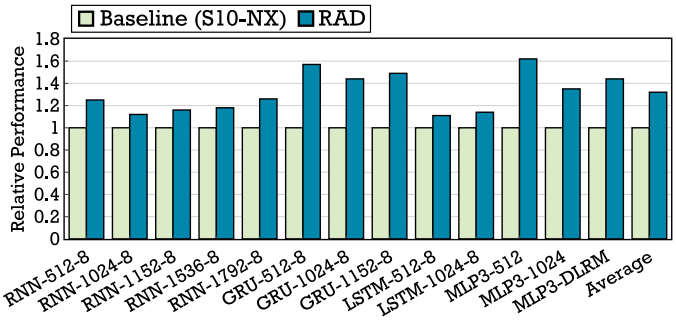

图 7：Stratix 10 NX 上的神经网络处理器与我们的示例 RAD 实例的相对性能比较。

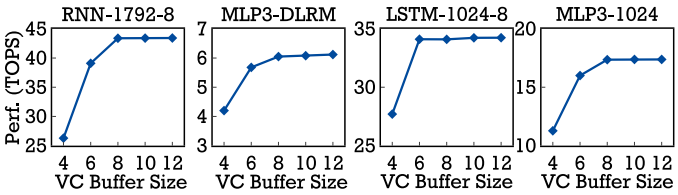

图 8：对于选定工作负载，改变片上网络虚通道缓冲区大小对神经网络处理器性能的影响。

我们示例 RAD 实例中片上网络路由器的虚通道缓冲区大小所产生的影响。在指定的片上网络规格和神经网络处理器模块布局下，若虚通道缓冲区深度小于 8 个流控单元，则可能因神经网络处理器的流量模式而限制性能。另一方面，虚通道缓冲区深度超过 8 个流控单元后，性能提升微乎其微，甚至没有额外收益。

V. 结论

随着 FPGA 容量不断增长并进入数据中心，市场对更快获得解决方案以及提升关键工作负载加速能力的需求日益增加。这些压力正推动业界转向新型 RAD，将 FPGA 的硬件可重构性与特定领域加速器模块及片上网络相结合，以实现系统范围内功能完备的通信。然而，目前尚无用于探索此类器件庞大设计空间的工具。在本研究中，我们介绍了 RAD-Sim，这是一款基于 SystemC 的应用驱动模拟器，可用于快速探索采用传统 FPGA、高性能分组交换片上网络和粗粒度硬加速器模块的 RAD 架构。该周期级模拟器能够研究不同的 RAD 架构，并量化特定设计选择对端到端应用性能的影响。为展示 RAD-Sim 的能力，我们给出了一个示例设计，将最先进的神经网络处理器深度学习推理覆盖层映射到一个示例 RAD 实例上。RAD-Sim 和神经网络处理器示例设计均为开源，研究社区可借此推动 RAD 架构的进一步创新。

致谢

作者感谢 Intel/VMware Crossroads 3D-FPGA学术研究中心和 NSERC/Intel 可编程硅工业研究主席项目提供资金支持。

参考文献

[1] A. Boutros 和 V. Betz，《FPGA架构：原理与演进》，IEEE电路与系统杂志，第 21 卷，29，2021 2，4第。页 – 。[2] B. Gaide 等，《赛灵思自适应计算加速平台：Versal架构》，载于 ACM/SIGDA国际现场可编程门阵列研讨会（FPGA），2019。[3] A. Caulfield 等，《云规模加速架构》，载于国际微体系结构研讨会（MICRO），2016。[4] J. Fowers 等，《面向实时人工智能的可配置云规模深度神经网络处理器》，载于国际计算机体系结构研讨会（ISCA），2018。[5] S. Yazdanshenas 和 V. Betz，《虚拟化现场可编程门阵列的互连解决方案》，IEEE Access，第 6 卷，第 10 497–10 507 页，2018。[6] M. S. Abdelfattah 等，《现场可编程门阵列上的嵌入式片上网络设计与应用》，IEEE计算机汇刊，第 66 卷，第 6 期，第 1008–1021 页，2016。[7] M. Langhammer 等，《Stratix 10 NX架构与应用》，载于 ACM/SIGDA国际现场可编程门阵列研讨会（FPGA），2021。[8] K. E. Murray 等，《VTR 8：高性能计算机辅助设计与可定制现场可编程门阵列架构建模》，ACM可重构技术与系统汇刊（TRETS），第 13 卷，第 2 期，第 1–55 页，2020。[9] ——，《Titan：在学术计算机辅助设计中支持大型复杂基准测试》，载于国际现场可编程逻辑与应用会议（FPL），2013。[10] A. Arora 等，《Koios：面向现场可编程门阵列架构与计算机辅助设计研究的深度学习基准测试套件》，载于国际现场可编程逻辑与应用会议（FPL），2021。[11] N. Jiang 等，《一种详细且灵活的周期精确片上网络模拟器》，载于国际系统与软件性能分析研讨会（ISPASS），2013。[12] N. Binkert 等，《gem5模拟器》，ACM SIGARCH计算机体系结构新闻，第 39 卷，第 2 期，第 1–7 页，2011。[13] M. Khairy 等，《Accel-Sim：用于经验证的GPU建模的可扩展仿真框架》，载于 ACM/IEEE国际计算机体系结构研讨会（ISCA），2020。[14] G. Krishnan 等，《SIAM：采用网格的基于芯粒、可扩展的深度神经网络存内加速》，ACM嵌入式计算系统汇刊（TECS），第 20 卷，第 5s 期，第 1–24 页，2021。[15] J. Albericio 等，《Cnvlutin：无无效神经元深度神经网络计算》，ACM SIGARCH计算机体系结构新闻，第 44 卷，第 3 期，第 1–13 页，2016。[16] S. Angizi 等，《MRIMA：基于磁阻式随机存取存储器的存内加速器》，IEEE集成电路与系统计算机辅助设计汇刊，第 39 卷，第 5 期，第 1123–1136 页，2019。[17] M. Yan 等，《HyGCN：采用混合架构的图卷积网络加速器》，载于 IEEE国际高性能计算机体系结构研讨会（HPCA），2020。[18] A. Boutros 等，《超越峰值性能：比较人工智能优化现场可编程门阵列与图形处理器的实际性能》，载于 IEEE国际现场可编程技术会议（FPT），2020。[19] E. Nurvitadhi 等，《既然可以协同工作，何必竞争：现场可编程门阵列与专用集成电路集成以实现持久化循环神经网络》，载于现场可编程定制计算机器国际研讨会（FCCM），2019。[20] A. Boutros 等，《专为效率而优化：定制现场可编程门阵列上的人工智能推理处理器》，载于 IEEE国际微电子会议（ICM），2021。[21] M. S. Abdelfattah 等，《走高速公路：现场可编程门阵列上嵌入式片上网络的设计》，载于 ACM/SIGDA国际现场可编程门阵列研讨会（FPGA），2015。[22] M. S. Abdelfattah 和 V. Betz，《基于现场可编程门阵列的硬核与软核片上网络的设计权衡》，载于 IEEE国际现场可编程技术会议（FPT），2012。[23] H. Wong 等，《比较现场可编程门阵列与定制互补金属氧化物半导体及其对处理器微体系结构的影响》，载于 ACM/SIGDA国际现场可编程门阵列研讨会（FPGA），2011。[24] A. Boutros 等，《无法改进未曾度量之物：卷积神经网络推理中现场可编程门阵列与专用集成电路的效率差距》，ACM可重构技术与系统汇刊（TRETS），第 11 卷，第 3 期，第 1–23 页，2018。

[25] ——，“拥抱多样性：面向 FPGA 低精度深度学习的增强型 DSP 模块”，载于现场可编程逻辑与应用国际会议（*FPL*），2018。