BARVINN：由 RISC-V CPU 控制的任意精度深度神经网络加速器

Mohammadhossein Askarihemmat[1], Sean Wagner[2], Olexa Bilaniuk[3],Yassine Hariri[4], Yvon Savaria[1], Jean-Pierre David[1]1蒙特利尔理工学院，加拿大， [2] IBM，多伦多，加拿大， [3] Mila，蒙特利尔，加拿大，[4] CMC Microsystems，金斯顿，加拿大
{mohammad.hossein.askari-hemmat,yvon.savaria,jpdavid}@polymtl.ca,
wagnerse@ca.ibm.com, olexa.bilaniuk@mila.quebec, hariri@cmc.ca

arXiv:2301.00290v1 [cs.AR] 31 Dec 2022

摘要
我们提出了一种深度神经网络加速器，可通过专用处理单元实现任意精度推理，这些处理单元可在位级进行配置。我们的深度神经网络加速器包含 8 个处理单元，由 RISC-V 控制器控制；在采用最新的 Alveo U250 FPGA 平台实现时，其综合计算能力为 8.2 TMAC。我们开发了一款代码生成器工具，可读取 ONNX 格式的卷积神经网络模型，并为 RISC-V 控制器生成可执行的命令流。我们通过运行不同的深度神经网络内核和模型，并选择不同的量化级别，展示了该加速器可扩展的吞吐量。与其他低精度加速器相比，我们的加速器无需硬件重新配置即可在运行时进行编程，并且无论目标 FPGA 的规模如何，都能加速具有多个量化级别的深度神经网络。BARVINN 是一个开源项目，可通过 https://github.com/hossein1387/BARVINN 获取。

关键词
神经网络、硬件加速、FPGA、低精度

ACM 引用格式：
MohammadhosseinAskarihemmat[1], Sean Wagner[2], Olexa Bilaniuk[3],,Yassine Hariri[4], Yvon Savaria[1], Jean-Pierre David[1]。2023。BARVINN：由 RISC-V CPU 控制的任意精度深度神经网络加速器。载于第*28*届亚洲及南太平洋设计自动化会议（*ASPDAC '23*），*2023*年*1*月*16*日至*19*日，日本东京。ACM，纽约，NY，美国，7页。https: //doi.org/10.1145/3566097.3567872

1 引言
深度神经网络（DNN）传统上依赖浮点运算。与定点/整数运算相比，这些操作速度较慢，且需要消耗更多功耗和硅片面积。加速深度神经网络计算的一种方法是通过量化[12]降低计算精度。这还能减少内存消耗和能耗。例如，在45纳米工艺中，8位整数乘法和加法分别消耗0.2 pJ和0.03 pJ，而相同的操作若使用32 位浮点数，则乘法和加法分别需要3.7 pJ和0.9 pJ [11]。在运行频率为3.4GHz的Intel Core i7 4770上，定点乘法比浮点乘法快3倍以上 [15]。借助近期的量化技术，这些优势得以实现，同时模型性能和准确率几乎不会降低，甚至完全不受影响。在 [9, 13], 中，他们的量化方案在大多数分类和目标检测模型上以2位精度量化时，准确率损失为1-3%。表1展示了学习尺度量化（LSQ） [9]应用于不同模型和任务、采用不同位精度时的结果。量化模型的准确率与全精度模型相近，同时体积更小。

表 1：量化对准确率和模型大小的影响。

Task	数据集	模型	精度 A/W	Acc/ MAP	Size (MB)
分类	CIFAR 100	ResNet18	LSQ(2/2)	76.81	2.889
			LSQ(4/4)	76.92	5.559
			LSQ(8/8)	78.45	10.87
			FP32	76.82	42.8
目标检测	VOC-2007	SSD300-ResNet18	LSQ(2/2)	0.61	10.34
			LSQ(4/4)	0.60	11.81
			LSQ(8/8)	0.68	14.77
			FP32	0.59	32.49

混合精度量化 [7, 16, 21, 23, 24] 通过学习网络中每一层的不同精度，进一步实现更精细的控制，以达到最优解。在 [23],中，作者展示了使用其混合精度框架，与 8 位量化模型相比，在准确率略有下降的情况下，模型延迟和能耗降低了近 2× 倍。

要在深度神经网络中充分发挥低精度的优势，硬件必须原生支持低精度计算。商用硬件可以通过转换数据布局并使用位运算指令 [8] 执行任意精度算术。然而，这种方法对通用处理器而言代价极高，因为要将数据移位、掩码并进行位打包，才能转换为正确格式。在撰写本文时，尚无可高效处理任意精度数据的商用通用处理器（CPU 或 GPU）。

本文提出了一种任意低精度的深度神经网络硬件加速器 BARVINN。我们的加速器可由软件编程，并可集成到 RISC-V 标准开发流程中。它被设计为面向深度神经网络的高度优化计算流水线，具有较低的硬件开销，并支持低功耗运行。本文的贡献如下：

允许免费制作本作品全部或部分内容的数字或纸质副本，供个人或课堂使用，但不得为营利或商业利益而制作或分发副本，且副本必须附有本声明和首页的完整引用。必须尊重非 ACM 所有的本作品组件的版权。注明出处后允许摘录。若要以其他方式复制、再版、发布到服务器或重新分发至列表，须事先获得具体许可和/或支付费用。请通过 permissions@acm.org 申请许可。*ASPDAC '23，2023*年*1*月*16*日至*19*日，日本东京© 2023 计算机协会。ACM ISBN 978-1-4503-9783-4/23/01...$15.00
https://doi.org/10.1145/3566097.3567872

- 实现了一种用于矩阵向量乘法运算的深度神经网络硬件加速器，支持任意定点低精度，并具备高吞吐量和低功耗特性。
- 通过软件控制深度神经网络加速器阵列的自定义嵌入式 RISC-V CPU 实现。
- 用于高吞吐量串行计算的权重和激活值高效存储与处理的数据结构。
- 开发软件代码生成器，将深度神经网络转换为可在我们的加速器上执行的 RISC-V 代码。

第 2 节回顾了文献中相关的深度神经网络加速器。第 3 节介绍 BARVINN 的架构。第 4 节对 BARVINN 进行详细的性能分析，并将其与其他深度神经网络加速器进行比较。

2 相关工作

近年来，面向 FPGA 和 ASIC 的支持量化与低精度的多种深度神经网络硬件加速器已经被提出。在此，我们讨论与本文工作最相关的加速器架构。

近期基于 FPGA 的加速器包括 FINN [6, 22],DNNBuilder [25], 和 FILM-QNN [20]。在 FINN 和 DNNBuilder 中，软件工具链用于将训练好的深度神经网络映射为生成的逻辑模块，并将这些模块集成在一起。随后生成完整的处理流水线，并针对目标器件进行综合。这种方法的优势在于，逻辑模块能够高效实现特定的深度神经网络，同时在可于不同时刻重新配置为不同深度神经网络的器件上仅产生极小的开销。然而，这种方法要求所有深度神经网络层同时全部实现在逻辑中，因此深度神经网络的规模受到给定 FPGA 上可用逻辑资源数量的限制。FINN 支持低至二进制的低精度，而 DNNBuilder 支持低至 4 位的精度，但二者都不支持不同深度神经网络层采用任意精度和混合精度。相比之下，FILM-QNN 支持任意规模的深度神经网络模型，以及采用混合精度的量化深度神经网络。然而，由于 FPGA 中 DSP 模块所采用的位打包方案，它仅限于使用 4 位或 8 位权重以及 5 位激活值。

多种 ASIC 加速器设计支持任意精度。Bit Fusion [19] 使用由大量 2 位处理单元组成的阵列，这些处理单元可以融合在一起，以执行最高 8 位的运算。Loom[18], 和 BitBlade [17] 采用位串行计算方案，以提升灵活性。尽管任何单个数学运算（例如乘法）的位串行计算，相较于位并行电路，固有地需要额外的时钟周期并带来更高的延迟，但这些设计利用了深度神经网络中大量可并行执行的计算。具体而言，它们实现了大量同时运行的位串行计算单元，以高吞吐量弥补高延迟。例如，Loom引擎由 $128 \times 16 = 2048$ 个串行内积单元（SIP）组成，每个单元每周期执行 $16\ 1 \times 1$位乘法。

3 架构

BARVINN 旨在提供高吞吐量和软件可编程性，同时支持任意尺寸和类型的深度神经网络。BARVINN 的高层架构如图 1 所示。它由以下主要组件组成：1）一个由矩阵向量单元（MVU）构成的阵列 [5], ；以及 2）一个名为 Pito 的 RISC-V CPU [4] ，用作 MVU 阵列的控制器。矩阵向量单元可加速常见的深度神经网络计算，例如通用矩阵向量乘法、通用矩阵乘法和卷积，以及批归一化、ReLU 激活和量化等其他操作。Pito 协调 MVU 阵列中的计算，同时处理与主机系统之间的数据传输。

由于无法预见未来文献中可能出现的所有神经网络，BARVINN 中张量运算的高层级时序由软件完成。为了控制处理单元阵列，与前述加速器不同，BARVINN 使用标准的 RISC-V RV32I 指令集架构。这使我们能够利用现有的软件生态系统。此外，通过采用支持知名指令集架构的 CPU，BARVINN 具备更高的灵活性，并且能够适配以支持新的深度神经网络架构。

3.1 矩阵向量单元

BARVINN 的基础配置由一个包含 8 个矩阵向量单元的阵列实现。图 1 展示了每个 MVU 都是一个包含多个模块的 64 元素向量流水线：a) 矩阵向量乘积单元（MVP）；b) 用于存储激活值／权重／缩放因子／偏置的随机存取存储器；c) 缩放单元；d) 池化／激活单元；以及 e) 量化器。MVU 每个时钟周期可计算 64 个输出向量元素，使用激活值 RAM 中的 64 元素输入数据向量，以及权重 RAM 中的一个 64×64元素矩阵。激活值 RAM 和权重 RAM 以低精度存储数据。MVP 单元以低精度运行，而流水线中的后续单元则以高精度定点运行。

为了说明我们选择处理 64 元素向量的设计方案是合理的，我们分析了 ONNX 模型库 [2] 中提供的 50 多个模型，以检查卷积层的输入通道大小。图 2 展示了这些模型中所有层的输入通道大小分布。我们发现，其中 79% 的模型使用的卷积层输入通道大小是 64 的倍数。

3.1.1 矩阵向量乘积单元。矩阵运算由 MVP 单元执行。它们对位数为 1 至 16 位的任意精度定点操作数进行计算。每个 MVP 包含 64 条向量向量乘积（ VVP ）流水线。每条 VVP 流水线有 64 个输入通道，每个通道配备一个 1 位乘法器，随后接入加法树，输出为 8 位，如图 4 所示。每个周期，激活值 RAM 中的 64 位数据会广播到 64 条 VVP 流水线，而权重 RAM 中则会读取一个 64×64 矩阵块，并将块中的每一行分别发送至不同的 VVP 流水线。每条流水线中的 VVP 都会对 1 位操作数计算 64 元素点积。每个 MVP 配备 64 条 VVP 流水线，因此整体输出为一个 64 元素向量。

MVP 使用 [5] 的位串行方案计算任意位精度的点积。权重和激活值可以是无符号定点数，也可以是二进制补码有符号定点数。二者的位深度可独立设置，因此支持混合精度。如算法 1 所示，位串行点积是一个多周期序列，从激活值张量和权重张量各 64 个元素的最高有效位（MSB）开始。在每个通道中对位进行相乘，然后在加法树中跨通道相加，生成一个 8 位点积。该结果会加到累加器/移位器中（见图 4）。MSB× MSB 结果表示整个点积中数量级最高的部分和。随后，MVP 通过提取操作数所需的位组合，计算下一个更低数量级的部分和。当数量级发生变化时，累加器左移 1 位，以便对齐

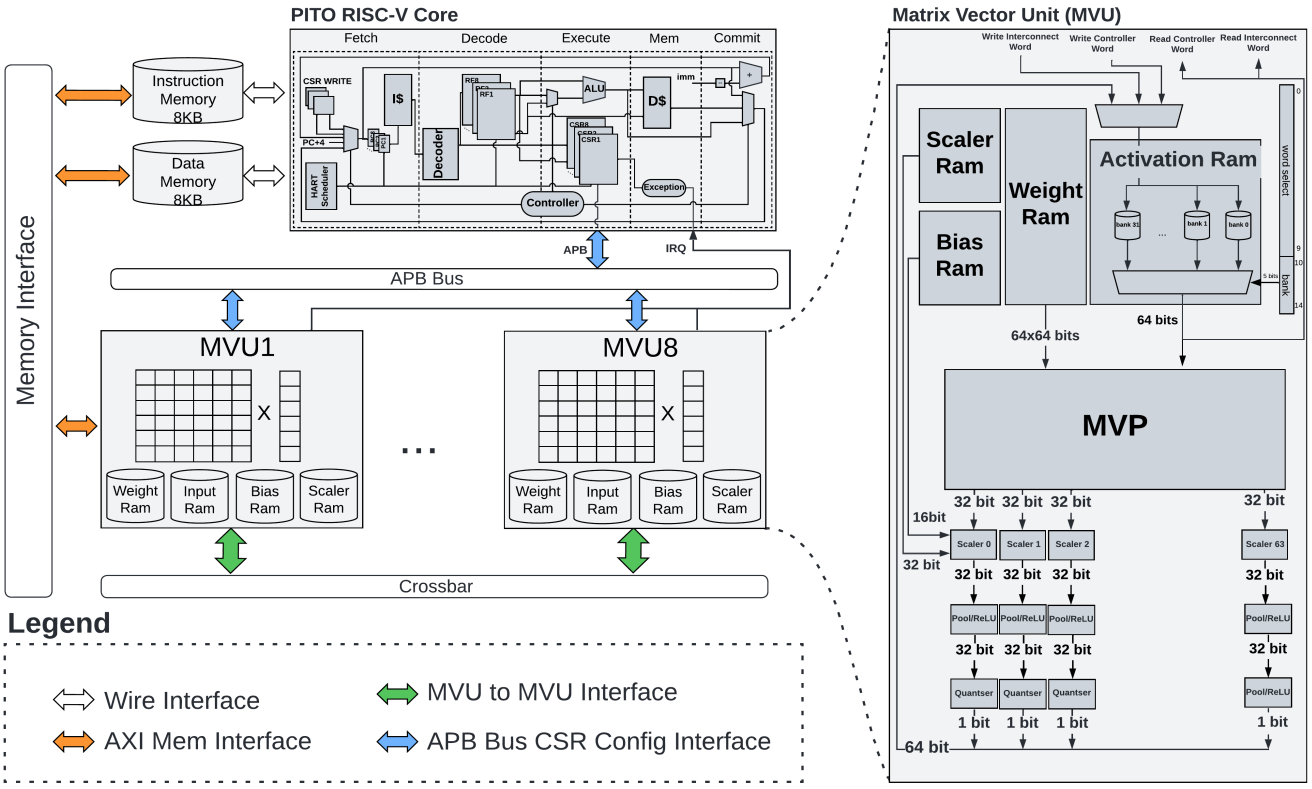

图 1：BARVINN硬件架构，包含 MVU 阵列和 Pito RISC-V控制器。右侧为 MVU 细节图。

到加法树输出相加前的数量级。MVP 采用全流水线设计，因此可以在不同阶段处理不同的位组合，而不会发生停顿。当操作数最低有效位（LSB）的点积计算并累加完成后，运算即告完成。对于 b_w 位权重和 b_a 位激活值，计算输出向量的一个图块整体需要 $b_w b_a$ 个周期。操作数的精度可针对每个 MVU 分别配置，因此每个 MVU 都能以不同的位精度处理不同的层。

算法 1位串行点积

b_a, b_w：激活值和权重的位精度
x, w：大小为 n 的激活值和权重向量
j， k：激活值和权重的位位置
$accumulator \leftarrow 0$
对于 $i \leftarrow b_w + b_a$ 到 1 **执行**
  对于所有 $(j,\ k)$ 其中 $j + k == i$ **执行**
    对于 $l \leftarrow 0$ 到 $n - 1$ **执行**
      $onebitprod = x_j[l] \times w_k[l]$
      $accumulator \leftarrow accumulator + onebitprod$
    结束循环
  结束循环
  将 $accumulator$ 左移 1 位
结束循环
$output \leftarrow accumulator$

我们的位串行点积方案与其他架构不同。BitFusion 的计算方案基于计算整个点积中的单个乘积，然后将它们相加。这需要大量移位寄存器来对齐并累加部分积。BARVINN 和 BitBlade 则改变计算顺序，先计算所有单个乘积中幅值相同的部分积，再将它们相加。这样可以减少所需的移位器数量。此外，BARVINN 会将不同幅值部分积的计算串行化，因此只需要一个固定移位器和一棵加法器树；而 BitBlade 则需要 16 个可变移位器和 17 棵加法器树。尽管采用了这种串行化方案，BARVINN 仍能通过对更多输入操作数进行并行化，并在每个时钟周期生成更多输出乘积来维持吞吐量。BitFusion 和 BitBlade 还受限于 2、4 和 8 位的操作数大小，而 BARVINN 中的 MVU 和 Loom 中的 SIP 则支持位深度低至 1 位的任意位宽操作数。不过，当权重位深度低于 16 位时，Loom 的数据加载方案会限制其在通用矩阵乘法中的效率，而 BARVINN 即使在 1 位时也能保持满吞吐量。

*3.1.2 存储器与数据布局。*激活值 RAM 和权重 RAM 以图 3 所示的位转置格式存储数据，以利用位串行计算。当精度大于 1 位时，张量元素按块组织，同一数量级的位存储在同一个存储字中，最高有效位存储在最低地址处。一个包含 n 个元素、精度为 b

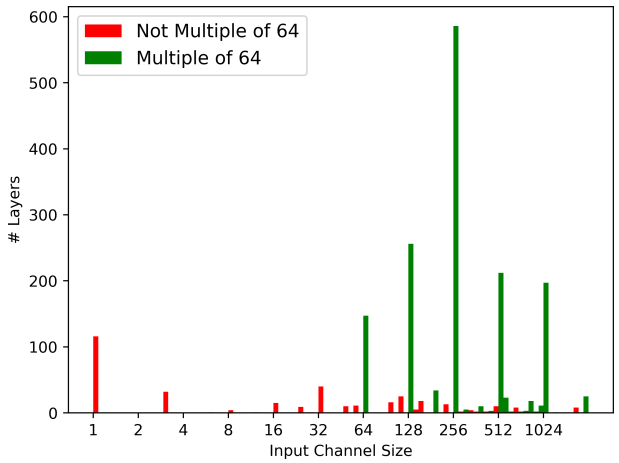

图 2：ONNX 模型库中模型的通道大小。

需要 b 个宽度为 n 的存储字。激活向量元素按 64 位一块组织，而权重矩阵元素按每块 4096 位组织，以便加载一个 64×64 矩阵块。转置模块将来自主机的输入数据转换为所需的位转置格式。由于 MVU 会以位转置格式将数据写回激活值 RAM，因此只有深度神经网络的第一层需要进行转置。权重由主机上的工具链预处理，并以预期的位转置格式加载到权重 RAM 中。

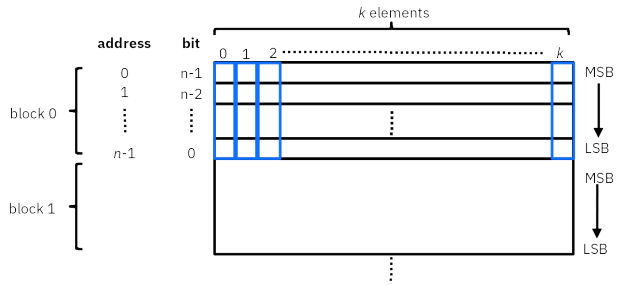

图 3：用于任意精度的位转置数据格式。

随机存取存储器中张量的布局取决于要执行的操作。对于通用矩阵向量乘法，激活值按每块 64 个元素的向量组织，而权重矩阵则组织为一组 64×64 块。对于二维卷积，激活值的布局为 $NHWC$，其中通道维度 C 位于最内层，其后依次是宽度 W、高度 H和批次大小 N。C维度位于最内层，是因为 ResNet 等几种常见的深度神经网络，其隐藏层通道深度通常是 2 的幂，因此与 VVP 的 64 个输入通道对齐。当通道数超过 64 时，前 64 个通道存储在第一个块中，接下来的 64 个通道存储在第二个块中，依此类推。例如，一个具有 [N=1、H=8、W=8、C=256] 且位精度为 2 位的输入张量，将包含 4 个通道块，每个块有 64 行，每行包含 2 个 64 位元素。

我们针对二维卷积设计的权重张量内存布局，通过交错输入通道维度 C_i 和输出通道维度 C_o，支持高效执行。每个权重存储字包含来自C_o维度的 64 个子集，每个子集包含来自 C_i 维度的 64 个元素。一个连续的

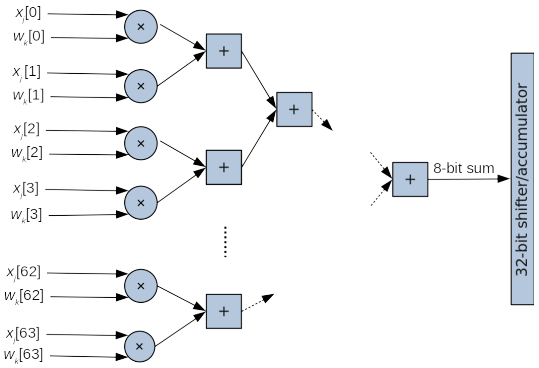

图 4：带移位累加器的 VVP 单元。激活张量 x 中 64 个元素的第 j 位，以及权重张量 w 中 64 个元素的第 k 位，以位串行方式输入。请注意，部分输入位以及 5 级加法树中的部分层未显示。

由 b_w 个存储字组成、存储所需权重精度的完整位数集合的块称为通道块 C_b。二维卷积权重的布局为 $C_o,\ _sF_HF_WC_b$，其中 $C_o,\ _s = C_o/64$ 表示输出通道集合，卷积核尺寸为（F_W，F_H）。

3.1.3 作业配置与执行。矩阵向量单元通过编程执行通用矩阵向量乘法或 Conv2D 等作业。控制器会设置配置寄存器，以编排完成矩阵向量单元中一项操作所需的计算序列和存储器读取。作业完成后，MVU 会向控制器产生中断，表明作业已完成且结果已就绪，可发送回主机，或触发同一 MVU 或其他 MVU 上的后续操作。MVU 忙碌时，也可以对其进行编程以准备下一项作业，从而尽量缩短空闲时间。

每个 MVU 都包含地址生成单元（AGU），用于控制激活值和权重 RAM 的存储器访问模式。访问模式由一组最多包含五层的嵌套循环管理，每层的参数设定迭代次数，以及每次迭代时地址向前或向后跳转的步长。地址跳转方案将逻辑简化为一组小型累加器，用于控制循环，并使用小型加法器计算地址。最内层循环通常设为按步长遍历激活值和权重的位深度。外层循环用于遍历串行点积过程中的位组合，以及张量的各个维度。对于通用矩阵向量乘法，激活值和权重各需要两个嵌套循环。Conv2D 操作通过编程设定为每个作业计算输出激活图的一行，这需要四层嵌套循环。

3.1.4 流水线模块。 每个 MVU 都在 MVP 下游设有模块，用于实现其他 DNN 运算，包括乘法器/加法器单元、池化/ReLU 单元以及量化器/串行器单元。这些模块以高精度运算。定点乘法器/加法器单元（图 1 中的 缩放器 ）用于计算批归一化和量化缩放等 DNN 运算，如 LSQ [9] 中所示。缩放器将 MVP 的输出乘以一个来自缩放器 RAM 的 16 位操作数。在 FPGA 中，乘法器为 27×16，与片上定点 DSP 单元的端口宽度相匹配。后续的加法器会将来自偏置 RAM 的 32 位定点偏置项相加。缩放器

和偏置 RAM 各自拥有独立的 AGU。后续模块将最大池化与 ReLU 结合起来（图 1 中的 池化/*ReLU*），其实现方式是带有内部寄存器的比较器。对于 ReLU，系统会将输入值与初始设为 0 的寄存器值进行比较。组合的最大池化/ReLU 通过对 MVU 编程来实现，使其按最大池化窗口所需的顺序生成数据。

流水线结束于量化/串行化单元（图 1 中的 *QuantSer*）。该单元接收来自 64 条数据通路的 32 位定点数据，并将其串行化为 64 个 1 位输出。通过编程，该单元可根据输入字设置输出位深度和最高有效位位置。结合缩放单元，可实现 LSQ [9] 等量化方案。每条数据通路的串行化输出会组合成一个 64 位字，并发送到同一 MVU 的激活值 RAM，或经由互连网络发送到另一 MVU。

3.1.5 互连网络。 MVU 之间可通过互连网络相互发送数据。该网络由一个具备广播功能的 8 路交叉开关实现。源 MVU 经编程后，会将其输出结果以串行方式发送到目标 MVU 激活值存储器中的指定地址。在目标 MVU 处，目标 MVU 激活值 RAM 的写端口采用固定优先级仲裁方案。互连网络拥有最高优先级，其次是控制器，最后是 MVU 本身。当多个 MVU 尝试写入同一目标 MVU 时，固定优先级方案决定哪个 MVU 可以写入其存储器。

3.1.6 深度神经网络映射。 每个 MVU 都可以分配给深度神经网络中的不同层，例如卷积层和全连接层。或者，也可以将单个层拆分到多个 MVU 上，由每个矩阵向量单元处理一部分输入激活值和/或权重。部分结果通过互连网络从一个 MVU 转发到另一个 MVU，以处理网络后续的层，从而在整个阵列中形成处理流水线。通过将部分结果从一个 MVU 发送到另一个 MVU，后续 MVU 一旦从前面的层收到足够的数据，就可以开始处理。例如，处理 3×3 卷积的 MVU 只需从前一层接收 3 行激活值，即可生成其所处理层的一行输出。这样就无需等待某一层的所有输出都生成，从而降低延迟并减少空闲时间。此外，后续 MVU 能够立即处理部分层输出，因此片上存储需求较低，因为只需存储生成下一层部分输出所需的部分激活值。

根据性能目标，BARVINN 可以通过流水线模式或分布式模式执行深度神经网络。在流水线模式（图 5.a）下，MVU 阵列最多可同时处理 8 个卷积层和全连接层。每个 MVU 都可以配置为使用不同的精度。如果深度神经网络模型包含超过 8 层，则可以对 MVU 阵列进行编程，将整个模型划分为若干个子集，每个子集最多包含 8 层，从而完成处理。每个 MVU 都可以加载各子集中的层的权重；如果每个 MVU 的权重存储器足够，则可在处理开始时一次性加载所有权重，否则可在处理过程中从外部存储器动态加载。链中最后一个 MVU 的输出激活值也可以暂时存储在片外存储器中，并在第一个 MVU 仍在处理当前轮次的数据时稍后读取。在分布式模式下，为尽可能降低延迟，目标是尽快处理单批次输入。正如可以看到的

在图 5.b 中，在此模式下，单层的计算被拆分为 8 个独立的计算区域。所有 MVU 都将被编程为共享同一组权重。为确保某个 MVU 的计算不依赖于其他 MVU 执行的计算，用户可能需要复制各计算单元之间共享的输入区域。BARVINN 的可编程性使用户能够针对不同的层和模型灵活组合这些执行模式，以实现最高性能。

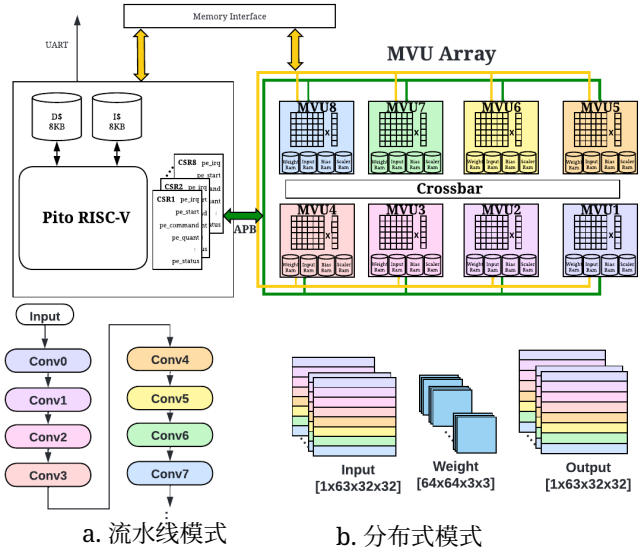

图 5：深度神经网络在流水线模式 (a) 和分布式模式 (b) 下于 MVU 阵列上的执行流程。在流水线模式下，每个 MVU 一次处理一层。在分布式模式下，单层计算由多个 MVU 分担。

3.2 Pito：基于 RISC-V 的控制器

要将 MVU 用于神经网络，需要一个控制单元。该控制器是一个桶式 RISC-V 处理器，旨在通过彼此独立但相互通信的硬件线程（硬件线程）控制 8 个 MVU；每个线程分别管理对应的 MVU。深度神经网络层根据编译时是为了最大化吞吐量还是最小化延迟，采用分布式或流水线方式执行。这种设计使各个 MVU 能够彼此独立地完成张量运算。其缺点是需要 8 个微处理器来执行这 8 个程序。我们采用桶式处理，分摊处理器的固定成本。借助八路线程处理器，我们可以分配一个线程来控制每个 MVU。由于每个线程每 8 个时钟周期才获得一次执行机会，因此五个流水线阶段（取指、译码、执行、数据读写和提交）可以完全隐藏。无需分支预测单元。由于张量运算在一个 MVU 上可能需要数百个周期才能完成，桶式处理器在此期间可以完成数十轮调度，让每个线程只需执行几条指令，就能向其 MVU 发出下一条指令。

我们采用哈佛架构，将指令 RAM 和数据 RAM 分开，各为 8 KB，并由所有硬件线程共享。这提供了

1K 字的空间，用于存储数据以及控制各个 MVU 的指令。处理器按照编译顺序执行指令，不进行任何额外调度。硬件线程调度器会在各个阶段为硬件线程提供所需资源。在取指阶段，每个硬件线程都会从指令 RAM 中加载指令。每个硬件线程都有各自的程序计数器（PC）和寄存器文件，硬件线程调度器则指示在特定时刻应访问哪个寄存器。译码阶段会对指令进行译码，并加载源寄存器或立即数操作数。我们的 RISC-V 控制器兼容 RV32I RISC-V 指令集架构，仅对特权级规范提供最低限度的支持，以便通过控制与状态寄存器（CSRs）和中断与 MVU 阵列进行接口连接。除基础控制与状态寄存器外，我们还增加了 74 个 MVU 专用控制与状态寄存器，使软件能够控制处理单元阵列。这些控制与状态寄存器用于控制 MVU 内的不同设置，例如权重和激活精度、AGU 的跳转设置、输入、权重和输出的存储器地址，以及流水线模块选择，如 3.1.4 节所述。

3.3 代码生成器

BARVINN 执行通用矩阵乘法/通用矩阵向量乘法、卷积、最大池化和激活（ReLU）。然而，用户需要通过软件来安排深度神经网络中的运算顺序。为此，我们开发了一种代码生成器，它接收以 ONNX[3] 描述的深度神经网络以及配置设置（权重/输入/输出精度），并为每项运算生成 RISC-V 代码。代码生成器会将权重导出为第 3.1.2 节所述的位转置格式。由于每个 MVU 都以 64 位字为单位进行处理，代码生成器会将每个权重张量划分为 64×64 大小的块。当无法做到这一点时（张量输入通道或输出通道不是 64 的倍数），我们会对相应的块进行填充。目前，代码生成器不会应用图优化技术。此外，代码生成器目前支持流水线模式执行。在下一节中，我们使用代码生成器将 PyTorch 模型映射为微内核代码，这些代码随后可直接用于 BARVINN。

4 性能分析与结果

4.1 实验设置

为了展示 BARVINN 的性能，我们选择了用于 CIFAR10 数据集的 ResNet9 图像分类模型。我们使用 LSQ [9]在 CIFAR10 上训练并量化了 ResNet9 模型，并采用残差蒸馏 [14] 技术移除捷径连接（普通卷积神经网络模型）。在许多图像分类深度神经网络模型中，例如 ResNet [10], , 第一层的输入通常包含少于 64 个通道。此外，由于第一层和最后一层对信息损失较为敏感，大多数最先进的压缩和量化方法都不会对输入层和输出层进行优化 [9], , 因此这些层保持不变并采用全精度。我们采用相同的方法，在主机或 RISC-V 控制器上计算第一层和最后一层。

表2展示了 ResNet9 在 PyTorch 框架下运行 CIFAR10 的性能。我们对量化模型的性能感到满意后，便将训练模型导出为 ONNX 格式，然后使用我们的代码生成器。表 3 展示了使用 2 位激活值和权重在 BARVINN 上运行 ResNet9 的逐层计算成本。所有卷积均采用 1 的填充。如前所述，我们跳过了在 BARVINN 上运行第一层和最后一层，

表2：CIFAR10 上不同位精度的 ResNet9

ResNet9 模型	精度	准确率	大小（字节）
原始模型	Fp32	90.8%	19605141
Plain-CNN	Fp32	91.1%	18912487
量化Plain-CNN	Int2	89.2%	1181360

表 3：CIFAR10 数据集的 ResNet9 层及计算成本。除第一层和最后一层外，所有层的激活值和权重均量化为 2 位。

层	输入	卷积核	输出	周期
conv0	[3, 32, 32]	[64, 3, 3, 3]	[64, 32, 32]	N/A
conv1	[64, 32, 32]	[64, 64, 3, 3]	[64, 32, 32]	34560
conv2	[64, 32, 32]	[64, 64, 3, 3]	[64, 32, 32]	34560
conv3	[64, 32, 32]	[128, 64, 3, 3]	[128, 16, 16]	17280
卷积层4	[128, 16, 16]	[128, 128, 3, 3]	[128, 8, 8]	32256
卷积层5	[128, 8, 8]	[256, 128, 3, 3]	[128, 8, 8]	16128
卷积层6	[128, 8, 8]	[256, 256, 3, 3]	[256, 4, 4]	27648
卷积层7	[256, 4, 4]	[512, 256, 3, 3]	[256, 4, 4]	13824
卷积层8	[256, 4, 4]	[512, 512, 3, 3]	[512, 4, 4]	18432
fc	[512, 4, 4]	[10, 512]	[10]	N/A
			总计：	194688

并将它们保留为原始格式。整体计算需要 194,688 个周期才能完成。

我们的设计采用 Verilog 编写，并使用 Xilinx Vivado 2021.1 为 Xilinx Alveo U250 加速卡进行综合。RISC-V 控制器、处理阵列和加速器的综合结果见表 4。功耗通过 Vivado 中的软件工具进行估算。

4.2 讨论

我们将 BARVINN 与 FINN [22], 进行了比较。FINN 是一个基于模板的 Vivado HLS C++ 库，包含常见的深度神经网络层。与 BARVINN 一样，FINN 可以生成任意精度的硬件，但不支持软件编程。因此，FINN 硬件生成后，用户无法更改计算数据流。我们尝试使用此前采用的 ResNet9 模型比较 BARVINN 与 FINN 的性能。然而，在本文撰写时，FINN 仅支持简单的线性拓扑，因此我们无法获得该模型的性能指标。作为替代，我们采用了 FINN 仓库中可用的 CIFAR10-CNV 模型进行比较，该模型针对 FINN 数据流进行了调优。表 5 展示了 BARVINN 和 FINN 的性能。本次实验中，我们为权重和激活值采用了不同的精度。对于这两种工具，我们使用其性能估算的每秒帧数（FPS）数据。对于 FINN，我们采用了 FINN 示例代码库 [1]中公开提供的默认折叠配置。如表 5 所示，尽管查找表使用量更高，我们的吞吐量仍提升了 7 至 15 倍。另一方面，在更高位精度下，FINN 的每秒帧数/查找表表现更好，这表明它能够为更大的模型提供可扩展的解决方案。

我们还比较了 ResNet-50 模型上的性能。表 6 展示了 BARVINN 在流水线模式下运行时估算的每秒帧数，以及 FINN [1] 为 Xilinx U250 综合后的报告性能和 FILM-QNN [20] 为 Xilinx ZCU102 FPGA 综合后的报告性能。尽管 FINN 的每秒帧数最高，BARVINN 的

表 4：BARVINN 的综合后资源利用率。

资源	Pito RISC-V	MVU 阵列	总体
LUT	10454	190625	201079
BRAM	15	1312	1327
DSP	0	512	512
动态功耗	0.410 W	21.066 W	21.504 W
频率	250 MHz	250 MHz	250 MHz

表 5：在 AlveoU250 上使用不同位精度运行 CIFAR10 的 CNV模型时的预估性能。

	Bits (W/A)	kLUT	BRAM	DSP	FPS	FPS/ kLUT
Ours	1/1	201.1 (15.0%)	1327	512	61035	303.5
	1/2	201.1 (15.0%)	1327	512	30517	151.7
	2/2	201.1 (15.0%)	1327	512	15258	75.8
FINN	1/1	28.2 (2.1%)	150	0	7716	273.6
	1/2	19.8(1.47%)	103	0	2170	109.6
	2/2	24.3(1.81%)	202	0	2170	89.3

表 6：ImageNet 上 ResNet-50 模型的性能。

	位宽（权重/激活）	时钟频率	FPS	每瓦每秒帧数
Ours	1/2	250 MHz	2296	106.8
FINN-R [1][6]	1/2	178 兆赫兹	2873	41.0
FILM-QNN [20]	4(8)/5	150 兆赫兹	109	8.4

每瓦性能却是最佳的。根据 FINN 示例代码库 [1], , 微调的 ResNet50 模型需要占用 Alveo U250 加速器 87% 以上的资源。这表明 FINN 在处理更大型模型时存在局限。无论模型大小和位宽如何，BARVINN 所需的查找表用量都相同。

5 结论

本文介绍了一种支持任意位精度计算的基于 FPGA 的深度神经网络加速器。我们测试了 BARVINN 在不同位精度下处理各种深度神经网络内核和模型时的性能。为部署模型，我们开发了一种代码生成器工具，可接收 ONNX 格式的模型，并为控制器生成 RISC-V 汇编代码。与其他低精度加速器相比，我们提供了一种可编程解决方案，使 BARVINN 更加灵活。借助可编程的矩阵向量单元，用户可以运行不同模型，而不受模型规模限制。BARVINN 可通过以分布式或流水线模式运行深度神经网络层，在吞吐量和延迟之间进行权衡。与其他低精度加速器不同，我们提出的解决方案可通过软件实现多种权衡方式，最终用户无需在运行时进行 FPGA 重配置，即可针对每个独立层进行控制。研究表明，与可编程加速器相比，BARVINN 的每瓦吞吐量性能更佳。

致谢

作者感谢 IBM AI Horizons Network、CMC Microsystems、魁北克自然与技术研究基金（FRQNT）、MITACS 以及 NSERC COHESA 战略研究网络对本项目的支持。

参考文献

[1] 2021. FINN 数据流加速器示例。(2021)。https://github.com/Xilinx/finn-examples [2] 2021。ONNX 模型库。(2021)。https://github.com/onnx/models [3] 2021。开放神经网络交换。(2021)。https://onnx.ai/ [4] MohammadHossein AskariHemmat、Olexa Bilaniuk、Sean Wagner、Yvon Savaria 和 Jean-Pierre David。2021。用于深度神经网络加速的 RISC-V 桶式处理器。载于 2021 年 IEEE 国际电路与系统研讨会（ISCAS）。 1 5 10 1109 51556 2021 9401617 –。https://doi.org/。/ [5] ISCAS。。Olexa Bilaniuk、Sean Wagner、Yvon Savaria 和 Jean-Pierre David。2019。面向量化神经网络的 FPGA 位切片加速器。载于 2019 年 IEEE 国际电路与系统研讨会（ISCAS）。1–5。[6] Michaela Blott、Thomas B. Preußer、Nicholas J. Fraser、Giulio Gambardella、Kenneth O'brien、Yaman Umuroglu、Miriam Leeser 和 Kees Vissers。2018。FINN-R：用于快速探索量化神经网络的端到端深度学习框架。ACM 可重构技术系统汇刊 11，3，文章 16（12 月 2018 23 10 1145 3242897），页码。https://doi.org/。/ [7] 2021 Adrian Bulat 和 Georgios Tzimiropoulos。。Bit-Mixer：具有运行时位宽选择功能的混合精度网络。载于 IEEE/CVF 国际计算机视觉会议（ICCV）论文集。5188–5197。[8] Meghan Cowan、Thierry Moreau、Tianqi Chen、James Bornholt 和 Luis Ceze。2020。高性能量化机器学习内核自动生成。计算机协会，纽约州纽约市，美国，305–316。https://doi.org/ 10 1145 3368826 3377912 。/。Steven K. Esser、[9] Jeffrey L. McKinstry、Deepika Bablani、Rathinakumar Appuswamy 和 Dharmendra S. Modha。2020。学习步长量化。载于国际学习表征会议。[10] Kaiming He、Xiangyu Zhang、Shaoqing Ren 和 Jian Sun。2016。用于图像识别的深度残差学习。载于 IEEE 计算机视觉与模式识别会议（CVPR）论文集。[11] Mark Horowitz。2014。计算能耗问题（以及我们能为此做些什么）。国际固态电路会议（2014）。[12] Itay Hubara 等。2018。量化神经网络：使用低精度权重和激活值训练神经网络。JMLR 18，187（2018），1–30。[13] B. Ham J. Lee、D. Kim。2021。采用逐元素梯度缩放的网络量化。载于 IEEE/CVF 计算机视觉与模式识别会议论文集。[14] Guilin Li 等。2020。残差蒸馏：迈向无需捷径连接的便携式深度神经网络。载于《神经信息处理系统进展》，第 33 卷。8935–8946。[15] Nicolas Limare。2014。浮点数学：速度与精度。(2014)。http://nicolas.limare.net/pro/notes/2014/12/16_math_speed/ [16] Paulius Micikevicius 等。2017。混合精度训练。arXiv 预印本 arXiv:1710.03740（2017）。[17] Sungju Ryu、Hyungjun Kim、Wooseok Yi 和 Jae-Joon Kim。2019。Bitblade：采用按位求和的面积与能效兼优的精度可扩展神经网络加速器。载于第 56 届年度设计自动化会议（2019）论文集。1–6。[18] Sayeh Sharify、Alberto Delmas Lascorz、Kevin Siu、Patrick Judd 和 Andreas Moshovos。2018。Loom：利用权重和激活值精度加速卷积神经网络。载于 2018 年第 55 届 ACM/ESDA/IEEE 设计自动化会议（ DAC ）。 1 6 10 1109 2018 8465915 –。https://doi.org/。/ [19] 2018 DAC。。Hardik Sharma 等。。Bit Fusion：用于加速深度神经网络的位级动态可组合架构。载于 2018 年 ACM/IEEE 第 45 届年度国际计算机体系结构研讨会（ISCA）。IEEE，764–775。[20] Mengshu Sun 等。2022。FILM-QNN：采用层内混合精度量化的深度神经网络高效 FPGA 加速。载于 2022 年 ACM/SIGDA 现场可编程门阵列国际研讨会论文集。134–145。[21] Stefan Uhlich 等。2020。混合精度深度神经网络：只需良好的参数化。载于第八届国际学习表征会议（ICLR 2020），埃塞俄比亚亚的斯亚贝巴，2020 年 4 月 26 日至 30 日。[22] Yaman Umuroglu 等。2017。FINN：快速、可扩展的二值化神经网络推理框架。载于 2017 年 ACM/SIGDA 现场可编程门阵列国际研讨会论文集。ACM，65–74。[23] Kuan Wang、Zhijian Liu、Yujun Lin、Ji Lin 和 Song Han。2019。HAQ：具有混合精度的硬件感知自动量化。载于 2019 年 IEEE/CVF 计算机视觉与模式识别会议（CVPR）。8604–8612。https://doi.org/ 10 1109 2019 00881 。/CVPR。。Haichao Yu、Haoxiang Li、[24] Humphrey Shi、Thomas S. Huang 和 Gang Hua。2021。任意精度深度神经网络。载于第三十五届 AAAI 人工智能会议。AAAI Press，10763–10771。https://ojs.aaai.org/index.php/ AAAI/article/view/17286[25] Xiaofan Zhang 等。2018。DNNBuilder：用于在 FPGA 上构建高性能深度神经网络硬件加速器的自动化工具。载于（ICCAD）。