PDPU：面向深度学习应用的开源Posit点积单元

Qiong Li、Chao Fang 和 Zhongfeng Wang(✉)ICAIS实验室，南京大学电子科学与工程学院，中国 邮箱： {qiongli, fantasysee}@smail.nju.edu.cn, zfwang@nju.edu.cn

arXiv:2302.01876v1 [cs.AR] 3 Feb 2023

摘要——由于 Posit 在动态范围与精度之间具有更好的权衡，因此它已成为深度学习应用中 IEEE 754 浮点格式的一种有前景的替代方案。然而，Posit 算术的硬件实现仍有待进一步探索，尤其是针对深度神经网络（DNN）中占主导地位的点积运算。现有实现采用乘法器与加法树组合，或级联融合乘加单元，导致计算效率低下且硬件开销过大。为解决这一问题，我们提出了一种开源Posit点积单元，即 PDPU，可实现资源高效且高吞吐量的点积硬件。PDPU 不仅采用融合混合精度架构，消除了冗余延迟和硬件资源，还配备了细粒度六级流水线，从而提高计算效率。我们进一步开发了可配置的 PDPU 生成器，以满足各种 DNN 对计算精度的不同需求。在 28纳米CMOS工艺下进行评估的实验结果表明，与现有实现相比，PDPU 的面积、延迟和功耗最高分别降低 43%、64% 和 70%。因此，PDPU 有望成为面向深度学习应用的基于 posit 的加速器的计算核心。

I. 引言

Posit [1] 被视为传统 IEEE-754 浮点（FP）格式的即插即用替代方案， [2] 因为它在动态范围与精度之间实现了更好的权衡 [3]。自 posit 数据格式问世以来，许多领域都从中受益，包括天气预报、 [4],图处理 [5] 和深度学习 [6]。尤其是在深度学习应用中，已有研究利用 posit 数据类型优化深度神经网络（DNN），以实现高效推理 [7] [8]和训练 [9] [10]。

为便于在硬件中实现面向深度神经网络的 posit 运算，研究人员提出了多种 posit 算术单元：加法器 [11] [12],乘法器 [13]–[16], 以及融合乘加（FMA）单元 [17]–[20]。然而，据我们所知，鲜有研究探讨如何高效实现基于 posit 的点积运算，

而这类运算占据了深度神经网络训练与推理任务中大部分计算负载 [21]。例如，AlexNet 和 ResNet50 [22]分别包含多达 724M 和 3.9G 次乘累加（MAC）运算，这些运算由深度神经网络层中的长点积运算分解而来。

目前，点积运算可以由如图所示的架构执行：该架构由现有乘法器和加法器，或融合乘加单元组成。1这些架构被称为离散点积单元（DPU）。然而，这两种实现都面临若干挑战。首先，大量冗余

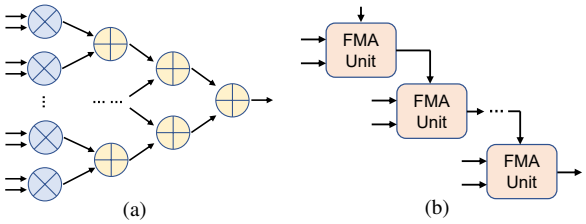

图1. 现有的离散点积架构通过 (a) 乘法器和加法器或 (b) 融合乘加单元实现。

运算（例如复杂的编码和解码过程）仍然保留在独立的算术单元中，导致延迟较高且硬件开销较大。其次，中间运算中频繁进行硬件舍入，很容易造成精度损失，从而可能降低深度神经网络模型的精度。最后，尽管混合精度算术在深度神经网络中具有显著优势， [23],现成的离散点积处理单元不支持这一策略，因此难以实现更高的计算效率。

为应对上述挑战，本文提出了一种开源 posit 点积单元（PDPU）[1] ，能够在深度神经网络中高效执行基于 posit 的点积运算。PDPU 通过在并行输入之间共享通用组件并移除不必要的逻辑，显著提高了硬件效率并优化了延迟。与离散点积处理单元相比，PDPU 较少进行硬件舍入，也能确保更高的数值精度。因此，PDPU 作为面向深度学习应用的基于 posit 的加速器的计算核心，具有巨大潜力。

总而言之，我们的贡献如下。

1) 提出的 PDPU 可高效执行具备融合与混合精度特性的点积运算。与传统的离散架构相比，PDPU 的面积、延迟和功耗最高分别降低 43%、64% 和 70%。2) 提出的 PDPU 配备了细粒度的 6 级流水线，可最大限度地缩短关键路径并提高计算效率。通过逐级分析延迟和资源使用情况，详细介绍了 PDPU 的结构。3) 开发了一种可配置的 PDPU 生成器，使 PDPU 能够灵活支持各种 Posit 数据类型、点积规模和对齐位宽。

[1]可在https://github.com/qleenju/PDPU获取。

II. 背景

Posit 数 [1]由字长 n 和指数位宽 es （即 P(n, es)）定义，由 4 个字段组成，分别是符号、区间、指数和尾数，如图 2所示。符号字段是最高有效位，其中 1 表示负数，0 表示正数。区间字段由 m 个连续相同的位 r 和一个相反的位 r组成，表示缩放因子 $2^{k\cdot 2^{es}}$，其中 $k=-m$ 若 r 为 $\bar{0}$，则 $m-1$ 若 r 为 1，则分别如此。紧随区间字段之后的是 es位指数字段，尾数字段则占据剩余的位。图 2中的最高有效 n 位2构成一个 Posit 数，其解码过程如下：

$$p=\begin{cases}\pm 0, & 000...000,\\ \pm\infty, & 100...000,\\ (-1)^s\times 2^{k\cdot 2^{es}}\times 2^{e_p}\times 1.m_p, & \text{otherwise},\end{cases} \tag{1}$$

其中， s、 e_p、 m_p 分别表示符号、指数和尾数字段的值。如果符号为负，则需要在解码前对数据取补码。

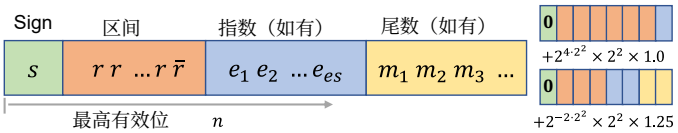

图2. Posit编码格式及两个P(8,2)解码示例。

Posit不包含非规格化数或非数表示，因此与浮点数相比，Posit数的异常处理大为简化。此外，其非均匀编码方式带来了对称的渐缩精度分布，这与深度神经网络参数的分布几乎完全一致。例如，ResNet18 [24] 第一卷积层中的激活值如图3所示。结果表明，在大多数计算中，Posit具有更高的十进制精度 [1] 以及更大的动态范围。

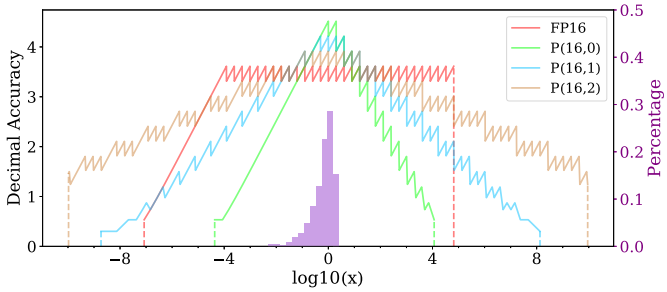

图3. Posit的渐缩精度与深度神经网络数据分布相匹配。

尽管具备上述优势，面向深度学习应用的基于posit的硬件设计仍有待进一步探索。例如，现有的若干基于posit的算术单元 [13] [17]不支持混合精度计算，而融合点积运算也鲜有讨论。本文提出的 PDPU 通过结合融合与混合精度策略，能够高效执行点积运算。

III. 所提出的 Posit DPU

A. 总体架构

图 4展示了提出的 PDPU 架构，该架构采用细粒度六级流水线。它以低精度格式对两个输入向量 V_a 和 V_b 执行点积运算，然后将点积结果与先前的输出 acc 累积为一个高精度值 out，如下所示：

$$\begin{aligned} out &= acc + V_a \times V_b \\ &= acc + a_0\cdot b_0 + a_1\cdot b_1 + ... + a_{N-1}\cdot b_{N-1}, \end{aligned} \tag{2}$$

其中 N 为参数化的点积规模。PDPU 在各流水级的数据流如下。

- **S1：解码**。Posit解码器并行提取输入的有效分量，随后在相应的硬件逻辑中计算 s_{ab} 和 e_{ab}，其中 s_{ab}和 e_{ab} 分别是 V_a与 V_b乘积的符号和指数。
- **S2：乘法**。尾数乘法由改进的基4 Booth乘法器执行 [25]，而所有指数（包括 acc （即 e_c的指数）均由比较器树处理，以获得最大指数 e_{max}。
- **S3：对齐**。S2的乘积结果根据各自指数与 e_{max}之间的差值进行对齐，然后转换为二进制补码，以便后续累加。
- **S4：累加**。对齐后的尾数在递归进位保留加法器（CSA）树中压缩为 sum 和 $carry$，随后将二者相加，得到累加结果 s_m 和最终符号 f_s。
- **S5：归一化**。根据前导零计数执行尾数归一化和指数调整，以确定最终指数 f_e 和尾数 f_m。
- **S6：编码**。posit编码器执行舍入，并将最终结果的各个组成部分打包为posit输出 out。

B. 融合与混合精度实现 n

PDPU 基于融合与混合精度算术实现：前者通过移除冗余逻辑，有助于提高计算效率和面积效率；后者则能在精度损失很小的情况下，进一步降低硬件开销和存储容量。

由于 posit 数的解码和编码过程受到动态regime字段影响，略显复杂，因此减少提取和打包、执行更多运算是合理的。不过，图 1(a) 的架构需要超过 $2N+2^{\lfloor log_2(N+1)}$ 」个解码器和 $N+2^{\lfloor log_2(N+1)\rfloor}$个编码器，而图 1(b) 的架构需要 $3N$ 个解码器和 N 个编码器，才能完成大小为 N 的点积。相比之下，得益于融合运算，PDPU 只需使用必需的 $2N+1$ 个解码器和 1 个编码器，便能大幅降低面积和延迟。此外，减少编码过程还可避免中间运算中的舍入，因此，与离散实现相比，PDPU 能实现更高的输出精度。

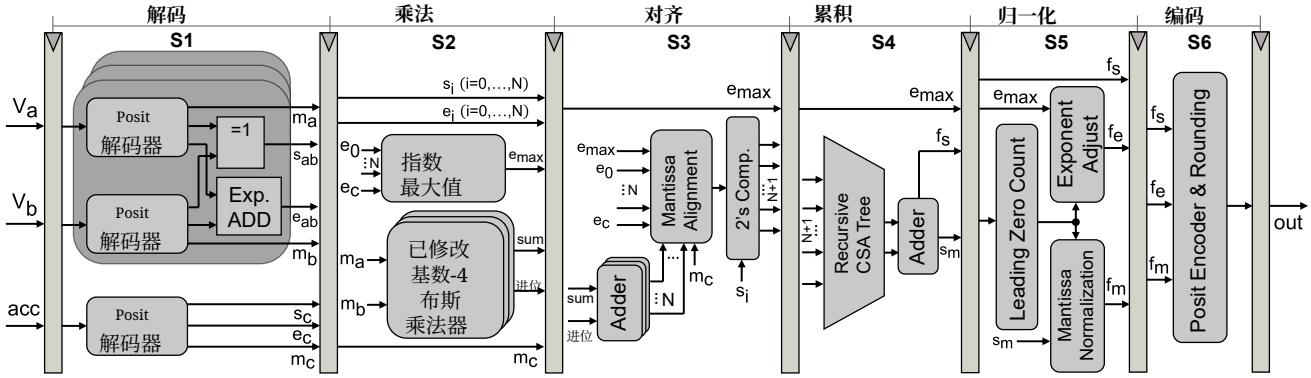

图4. 所提出的 posit 点积单元架构。

此外，PDPU 支持混合精度计算，该技术已广泛应用于深度神经网络训练和推理 [26]。具体而言，以统一的低精度 posit 格式替代 FP32 会带来精度下降的风险。然而，混合精度特性能够实现更灵活的量化策略，例如对输入采用低精度，而对点积结果采用稍高的精度；这既能在保持模型精度的同时，进一步降低计算复杂度和外部存储器需求。

C. 可配置 *PDPU* 生成器

我们开发了一种可配置的 PDPU 生成器，可从多个方面灵活配置 PDPU，即 posit 格式、点积规模和对齐位宽。

支持自定义 posit 格式：PDPU 支持输入和输出采用 n 和 es 的任意组合，这与某些面向若干固定精度的硬件设计 [27] 或软件库 [28] [29] 不同。灵活的格式支持也使混合精度策略成为可能，因为所提出的解码器和编码器分别能够提取和打包任意 posit 格式的数据。

支持多种点积规模：深度神经网络中的点积运算通常会被划分为较小的分块，并通过基于分块的累加来完成 [30]。点积分块大小记为 N，根据面积、延迟或功耗方面的硬件约束确定。

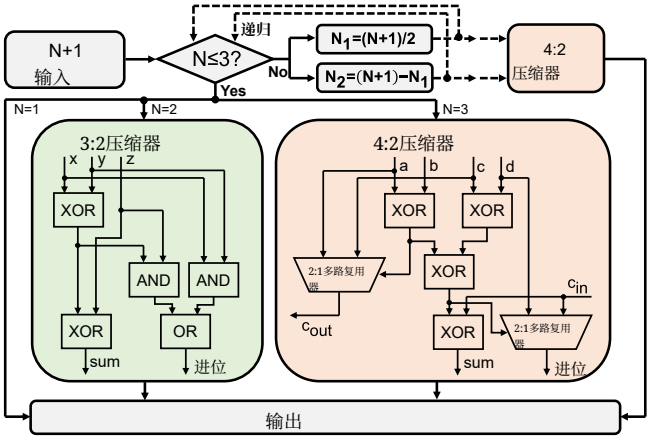

图5. 进位保存加法器树的递归设计方法。

PDPU 能够支持多种 N，而非仅支持特定的分块大小（例如 N=2 [31]、 N=4 [32]），因此能够更好地适应各种硬件约束。为适应可变规模，如图4所示，PDPU 的若干子模块以并行方式实例化，其他一些子模块则按树结构递归生成。图5 展示了如何使用 3:2 和 4:2 压缩器设计递归进位保存加法器树，将 $N+1$ 个输入的累加压缩为等效的sum和 $carry$，以便进行最终加法。

采用合适的对齐位宽：在若干设计中 [8] [19], quire [33] 格式用于表示两个 posit 向量的精确点积，且无需舍入或溢出。然而，相关硬件开销过高 [34]，因为中间操作数以 quire 值的形式保存，位宽很大，导致后续操作消耗过多计算资源。相比之下，PDPU 对齐后的尾数位宽进行参数化，即 W_m，其值可根据输入的分布特征和 DNN 精度要求确定。配置合适的对齐位宽后，PDPU 可在满足精度要求的同时，将硬件成本降至最低。

IV. 实验结果

PDPU 使用 SystemVerilog 实现，并采用 Synopsys Design Compiler，在典型工作条件（1.05 V，25°C）下，使用台积电 28纳米CMOS工艺标准单元库进行综合。我们使用由扩展版 SoftPosit库生成的测试向量对其进行了全面验证；该库 [28] 支持任意 posit 格式。

A. 与最先进技术的比较

为评估融合与混合精度算术在深度神经网络点积运算中的有效性，我们将具有这些特性的提出的 PDPU 与分别基于 FPnew [35] 和 PACoGen [13]库的传统离散点积处理单元进行比较。我们还评估了配备 quire 精确累加功能的 PDPU 所受到的影响。此外，我们还将若干融合乘加单元（包括 IEEE-754 浮点融合乘加单元 [35] 和 posit FMA 单元 [17], ）与 PDPU 进行比较，以评估其面积效率和能效。为确保比较公平，我们提取了 ResNet18 [24] 第一卷积层的激活值、权重和输出，并以 FP64格式表示，以评估所有单元的精度；这些数据也

表 I 提出的 PDPU 与 SOTAS 的比较

架构	格式	N	W_m	精度	Area (um^2)	延迟 (ns)	功耗 (mW)	性能 (GOPS)†	面积效率 (每瓦十亿次运算/mm^2)	能效 (每瓦十亿次运算)
FPnew DPU [35]	FP32	4	\	100%	28563.19	3.45	7.60	1.16	40.59	152.65
	FP16	4	\	91.21%	13448.99	2.75	4.29	1.45	108.15	338.85
PACoGen DPU [13]	P(16,2)	4	\	98.86%	13433.11	4.45	12.21	0.90	66.91	73.59
提出的 PDPU	P(16/16,2)	4	14	99.10%	9579.15	1.62	4.49	2.47	257.76	550.37
	P(13/16,2)‡	4	14	98.69%	**7694.82**	**1.60**	**3.66**	2.50	324.89	682.82
	P(13/16,2)	8	14	98.68%	13560.37	1.69	5.80	**4.73**	**349.09**	**816.16**
	P(10/16,2)	8	14	89.58%	10006.42	1.70	4.24	4.71	470.29	1110.95
	P(13/16,2)	8	10	88.90%	12157.11	1.66	5.06	4.82	396.42	953.14
Quire PDPU	P(13/16,2)	4	256	98.79%	29209.45	2.10	5.87	1.90	65.21	324.50
FPnew FMA [35]	FP32	1	\	100%	6668.17	1.20	3.97	0.83	124.97	210.00
	FP16	1	\	92.93%	3713.72	1.00	2.51	1.00	269.27	398.61
Posit FMA [17]	P(16,2)	1	\	99.23%	7035.34	1.35	3.79	0.74	105.29	195.52

†为便于比较，此处将一次乘加运算计为一次运算。‡混合精度算术，即，对 V_a 和 V_b采用 P(13,2)，对 $outs$ 和 acc 采用 P(16,2)，分别对应公式2。

有助于确定 PDPU 的适当数据格式和对齐位宽。此外，为避免不同流水线方案造成影响，比较中的所有单元均采用组合逻辑实现。

实验结果见表I。结果表明，FP32 和 P(16,2) 几乎能够保持 FP64 的精度，而采用 FP16 时精度约下降 8%，这充分体现了 posit 格式在深度神经网络中的优势。此外，与基于 posit 的 PACoGen DPU 相比，我们的混合精度 PDPU 在 W_m=14和 N=4 下，面积、延迟和功耗分别最多可降低 43%、64% 和 70%。如表I所示，与 Quire PDPU 相比，P(13/16,2) PDPU 在精度损失可忽略不计的情况下，面积和能效分别显著提升 5。0×和 2。1×。此外，与每周期执行一次乘加运算的 Posit FMA 单元相比，PDPU 凭借其融合算术，在1×下的面积和能效分别提升 3。5× 和 3。 N 倍输入。表I 还表明，增大 PDPU 的点积规模 N 可提升其性能和能效。请注意，不合适的数据格式或对齐位宽可能导致高出 10%

计算精度损失，表明应根据目标深度学习应用确定合适的 PDPU 配置至关重要。

B. 六级流水线评估

我们评估了 PDPU 的六级流水线性能，实验结果如图6所示。具有不同点积规模的 PDPU 均采用相同的混合精度格式 P(13/16,2) 和 14 位对齐位宽进行配置。

如图 6所示，所提出的流水线策略使各级的关键路径延迟趋于均衡，从而将 PDPU 的吞吐量提升了 4。4× 和 4。6×具体而言，六级流水线 PDPU 的最坏延迟仅约为 0.37 ns，因此其运行频率最高可达 2.7 GHz，能够嵌入高速人工智能加速器中。相比之下， [19] 中的 5级 posit MAC 单元在相同的 28纳米

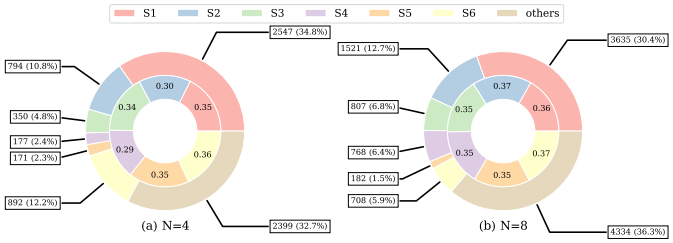

图6。PDPU 的六级流水线分解图。请注意，内圈和外圈分别表示每个流水级的延迟（ns）和面积占用（um^2）。

CMOS 工艺下的延迟为 0.8 ns。随着 N 的增加，PDPU 中 S2 和 S4 的延迟迅速增加，因为它们的树结构变得更加复杂。在面积方面，S1 的并行 posit解码器在 PDPU 中占比较大，这是由于其前导零计数和动态移位模块较为复杂，这凸显了采用融合算术消除冗余解码操作的重要性。

V. 结论

本文提出了一种可配置开源 posit 点积单元（PDPU），能够在深度学习应用中高效执行点积运算。PDPU 采用均衡的六级流水线，并具备融合精度和混合精度特性，在面积和功耗效率方面表现出色。此外，我们开发了可配置的 PDPU 生成器，以支持深度学习应用的多样化计算需求。与现有的传统点积硬件实现相比，PDPU 在面积、延迟和功耗方面分别实现了 43%、60% 和 70% 的显著降低。

致谢

本研究部分由中国国家自然科学基金项目（批准号：62174084、62104097）、江苏省高层次人才项目（批准号：JSSCBS20210034）以及江苏省研究生科研与实践创新计划（项目编号：149）资助。

参考文献

[1] J. L. Gustafson 和 I. T. Yonemoto，“以其人之道还治其身：Posit 算术”，《Supercomput. Front. Innov.》，第 4 卷，第 2 期，第 71–86 页，2017年。[2] “IEEE浮点算术标准”，IEEE Std 754-2008，第 1–70 页，2008年。[3] P. Lindstrom、S. Lloyd 和 J. Hittinger，“实数通用编码：IEEE浮点数的替代方案”，载于《下一代算术会议（CoNGA）论文集》。ACM，2018年，第 1–14 页。[4] M. Kl¨ower、P. D¨uben 和 T. Palmer，“使用浅水模型分析天气与气候建模中的数值格式、误差缓解及16位算术的适用范围”，《J. Adv. Model. Earth Syst.（JAMES）》，第 12 卷，第 10 期，2020年。[5] N. Shah、L. I. G. Olascoaga、S. Zhao、W. Meert 和 M. Verhelst，“DPU：采用精度可扩展Posit算术、面向非规则图的DAG处理单元，基于28纳米工艺”，《IEEE J. Solid State Circuits（JSSC）》，第 57 卷，第 8 期，第 2586–2596 页，2022年。[6] N. Ho、D. T. Nguyen、H. D. Silva、J. L. Gustafson、W. Wong 和 I. J. Chang，“用于生成对抗网络训练与部署的Posit算术”，载于《2021年欧洲设计、自动化与测试会议暨展览会（DATE）》。IEEE，2021年，第 1350–1355 页。[7] S. H. F. Langroudi、T. Pandit 和 D. Kudithipudi，“嵌入式设备上的深度学习推理：定点与Posit”，载于《2018年第1届面向嵌入式应用的节能机器学习与认知计算研讨会（EMC2）》。IEEE，2018年，第 19–23 页。[8] Z. Carmichael、H. F. Langroudi、C. Khazanov、J. Lillie、J. L. Gustafson 和 D. Kudithipudi，“Deep positron：使用Posit数制的深度神经网络”，载于《2019年欧洲设计、自动化与测试会议暨展览会（DATE）》。IEEE，2019年，第 1421–1426 页。[9] J. Lu、C. Fang、M. Xu、J. Lin 和 Z. Wang，“使用Posit数制对深度神经网络训练进行评估”，《IEEE Trans. Computers（TC）》，第 70 卷，第 2 期，第 174–187 页，2020年。[10] Y. Wang、D. Deng、L. Liu、S. Wei 和 S. Yin，“PL-NPU：采用基于Posit的对数域计算的节能边缘设备DNN训练处理器”，《IEEE Trans. Circuits Syst. I Regul. Pap.（TCAS-I）》，第 69 卷，第 10 期，第 4042–4055 页，2022年。[11] R. Chaurasiya、J. Gustafson、R. Shrestha、J. Neudorfer、S. Nambiar、K. Niyogi、F. Merchant 和 R. Leupers，“参数化Posit算术硬件生成器”，载于《2018年IEEE第36届国际计算机设计会议（ICCD）》。IEEE，2018年，第 334–341 页。[12] M. K. Jaiswal 和 H. K. So，“3型unum Posit加减器的架构生成器”，载于《2018年IEEE国际电路与系统研讨会（ISCAS）》。IEEE，2018年，第 1–5 页。[13] M. K. Jaiswal 和 H. K. So，“PACoGen：硬件Posit算术核心生成器”，《IEEE Access》，第 7 卷，第 74 586–74 601 页，2019年。[14] R. Murillo、A. A. D. Barrio 和 G. Botella，“使用FloPoCo核心生成器定制Posit加法器与乘法器”，载于《2020年IEEE国际电路与系统研讨会（ISCAS）》。IEEE，2020年，第 1–5 页。[15] H. Zhang 和 S. Ko，“节能Posit乘法器设计”，《IEEE Trans. Circuits Syst. II Express Briefs（TCAS-II）》，第 67 卷，第 5 期，第 861–865 页，2020年。[16] C. J. Norris 和 S. Kim，“一种面向FPGA的近似迭代式Posit乘法器架构”，载于《2021年IEEE国际电路与系统研讨会（ISCAS）》。IEEE，2021年，第 1–5 页。[17] H. Zhang、J. He 和 S. Ko，“面向深度学习应用的高效Posit乘累加单元生成器”，载于《2019年IEEE国际电路与系统研讨会（ISCAS）》。IEEE，2019年，第 1–5 页。[18] R. Murillo、D. Mallas´en、A. A. Del Barrio 和 G. Botella，“面向融合Posit算术的节能MAC单元”，载于《2021年IEEE第39届国际计算机设计会议（ICCD）》。IEEE，2021年，第 138–145 页。[19] L. Crespo、P. Tom´as、N. Roma 和 N. Neves，“用于跨精度计算的统一Posit/IEEE-754向量MAC单元”，《IEEE Trans. Circuits Syst. II Express Briefs（TCAS-II）》，第 69 卷，第 5 期，第 2478–2482 页，2022年。[20] J. Lu、S. Lu、Z. Wang、C. Fang、J. Lin、Z. Wang 和 L. Du，“使用Posit数制训练深度神经网络”，载于《2019年第32届IEEE片上系统会议（SOCC）》。IEEE，2019年，第 62–67 页。[21] J. Lee、L. Mukhanov、A. S. Molahosseini、U. I. Minhas、Y. Hua、J. M. del Rinc´on、K. Dichev、C.-H. Hong 和 H. Vandierendonck，“资源高效深度学习：关于模型、算术的综述——及实现层面的技术，”arXiv预印本 arXiv:2112 15131，2021年。.[22] V. Sze、Y. Chen、T. Yang 和 J. S. Emer，《深度神经网络的高效处理：教程与综述》，Proc. IEEE，第105卷，第12期，第2295–2329页，2017年。[23] P. Micikevicius、S. Narang、J. Alben、G. F. Diamos、E. Elsen、D. Garc´ıa、B. Ginsburg、M. Houston、O. Kuchaiev、G. Venkatesh 和 H. Wu，《混合精度训练》，载于《国际学习表征会议》（ICLR），2018年。[24] K. He、X. Zhang、S. Ren 和 J. Sun，《用于图像识别的深度残差学习》，载于《2016年IEEE计算机视觉与模式识别会议》（CVPR），2016年，第770–778页。[25] G. W. Bewick，《快速乘法：算法与实现》。斯坦福大学，1994年。[26] G. Raposo、P. Tom´as 和 N. Roma，《PositNN：使用混合低精度Posit训练深度神经网络》，载于《2021年IEEE国际声学、语音与信号处理会议》（ICASSP）。IEEE，2021年，第7908–7912页。[27] S. Tiwari、N. Gala、C. Rebeiro 和 V. Kamakoti，《PERI：可配置的Posit支持型RISC-V核心》，ACM Trans. Archit. Code Optim.（TACO），第18卷，第3期，第1–26页，2021年。[28] C. Leong，《Softposit》，https://gitlab.com/cerlane/SoftPosit，2018年。[29] R. Murillo、A. A. D. Barrio 和 G. Botella，《Deep PeNSieve：基于Posit数制的深度学习框架》，Digit. Signal Process.（DSP），第102卷，文章编号102762，2020年。[30] N. Wang、J. Choi、D. Brand、C. Chen 和 K. Gopalakrishnan，《使用8位浮点数训练深度神经网络》，《神经信息处理系统进展》（NeurIPS），第31卷，2018年。[31] J. Sohn 和 E. E. Swartzlander，《浮点融合点积单元的改进架构》，载于《第21届IEEE计算机算术研讨会》（ARITH）。IEEE，2013年，第41–48页。[32] J. Sohn 和 E. E. Swartzlander，《融合浮点四项点积单元》，IEEE Trans. Circuits Syst. I Regul. Pap.（TCAS-I），第63卷，第3期，第370–378页，2016年。[33]《Posit算术标准（2022年）》，https://posithub.org/docs/posit standard-2.pdf，2022年。[34] L. Forget、Y. Uguen 和 F. de Dinechin，《Posit数制的硬件成本评估》，载于Compas’ 2019-Conf´erence d’informatique en Parall´elisme, Architecture et Syst`eme，2019年，第1–7页。[35] S. Mach、F. Schuiki、F. Zaruba 和 L. Benini，《FPnew：面向能效成比例的跨精度计算的开源多格式浮点单元架构》，IEEE Trans. Very Large Scale Integr. Syst.（TVLSI），第29卷，第4期，第774–787页，2020年。