计 算 机 研 究 与 发 展
Journal of Computer Research and Development

https://crad.ict.ac.cn
63(5): 1238−1248, 2026

BE-HB: 基于块浮点的混合位宽卷积处理单元

李东阳[1] 李泽洋[2] 唐忆滨[3,4] 刘博生[1] 武继刚[1]

[1](广东工业大学计算机学院 广州 510006)

[2](广东工业大学自动化学院 广州 510006)

[3](武汉数字工程研究所 武汉 430074)

[4](图形处理芯片及智能计算系统技术创新中心 武汉 430074)

(2112305291@mail2.gdut.edu.cn)

BE-HB: A Hybrid Bit-Width Convolution Processing Unit Based on Block Floating Point

Li Dongyang[1], Li Zeyang[2], Tang Yibin[3,4], Liu Bosheng[1], and Wu Jigang[1]

[1] (*School of Computer Science and Technology*, *Guangdong University of Technology*, *Guangzhou* 510006)

[2] (*School of Automation*, *Guangdong University of Technology*, *Guangzhou* 510006)

[3] (*Wuhan Digital Engineering Institute*, *Wuhan* 430074)

[4] (*Innovation Center for GPU and Intelligent Computing System Technology*, *Wuhan* 430074)

Abstract Hybrid bit-width block floating point (BFP) offers a flexible solution for low bit-width convolution computations, optimizing storage efficiency and computational precision. By assigning higher bit-widths to numerically sensitive layers while using lower bit-widths for redundant or stable regions, this approach preserves near-floating-point accuracy with substantially reduced computational and storage cost. Recent researches have deployed hardware solutions such as field programmable gate arrays (FPGAs) for hybrid bit-width BFP-based convolution accelerations, but they tend to underutilize FPGA resources by overlooking the full potential of digital signal processors (DSPs). Specifically, the underutilization of DSPs often leads to unnecessary resource waste and limited computational throughput, which restricts the overall performance of FPGA-based BFP convolution accelerators. This work develops a novel FPGA-based BFP convolution processing unit, termed "BE-HB", capable of coupling two sets of BFP convolution calculations in dual-mode bit-width (i.e., 8 b or 16 b) using a single DSP for high performance. We then introduce a novel mapping method that reuses the shared exponents and private mantissas of BFP representations to perform two sets of BFP convolution computations within 8 b or 16 b DSP data paths. By leveraging the exponent sharing, data packing and data reuse, the proposed approach significantly reduces hardware resource overhead. Compared with representative baseline designs, the proposed design achieves an average reduction of 61.4% in LUT utilization while maintaining model accuracy, thereby delivering superior performance and resource efficiency, which makes it more suitable for resource-constrained FPGA-based edge computing platforms.

Key words block floating point; hybrid bit-width; convolution computation; field-programmable gate array; data mapping

收稿日期: 2025-08-27; **修回日期**: 2026-01-05

基金项目: 国家自然科学基金项目(62302102); 广东省基础与应用基础研究基金 (2023A1515012844, 2025A04J5208)

This work was supported by the National Natural Science Foundation of China (62302102) and the Guangdong Basic and Applied Basic Research Foundation (2023A1515012844, 2025A04J5208).

通信作者: 武继刚(asjgwucn@outlook.com)

摘　要　混合位宽的块浮点（block floating point, BFP）技术为低比特位宽卷积计算提供了一种灵活的解决方案，能有效优化存储效率和计算精度。通过为数值敏感层分配更高的位宽，而为冗余或稳定区域分配较低的位宽，该方法在显著降低计算与存储开销的同时，仍能保持接近浮点精度的计算表现。尽管现有的研究已经部署了基于混合位宽的BFP卷积硬件方案，如现场可编程门阵列（field-programmable gate array, FPGA）等，但这些方案往往忽视了数字信号处理器（digital signal processor, DSP）的计算潜力，导致未能充分利用FPGA硬件资源，具体而言，DSP的利用不足往往导致不必要的资源浪费和计算吞吐量受限，从而限制了基于FPGA的BFP卷积加速器的整体性能。提出一种新颖的基于FPGA的BFP卷积处理单元，称为“BE-HB”，该设计利用单个DSP在双模式位宽（即8 b或16 b）下耦合2组BFP卷积计算，以实现高性能计算。接着，提供了一种映射方法，通过在计算中复用块浮点打包后的共享指数与私有尾数，实现减少在DSP内8 b或16 b数据宽度下2组BFP卷积计算的硬件资源开销。与具有代表性的基线方法相比，所提出的设计在保持模型精度的前提下，计算所需的LUT资源消耗平均降低了61.4%，实现了更好的性能和更低的资源消耗，使其更适合资源受限的基于FPGA的边缘计算平台。

关键词　块浮点；混合位宽；卷积计算；现场可编程门阵列；数据映射

中图法分类号　TP181

DOI: 10.7544/issn1000-1239.202550645　　**CSTR:** 32373.14.issn1000-1239.202550645

近年来，卷积神经网络（convolutional neural network, CNN）作为深度学习的典型代表，凭借着局部感知和权重共享机制，在图像识别、自然语言处理、智能对话等多个领域展现出卓越的性能，然而，随着网络层数和模型参数规模的不断增大，卷积神经网络的计算复杂度和资源消耗呈指数级增长，这也导致了它对于计算效率和硬件性能有着更高的需求。8b和16b卷积的广泛应用凸显了混合位宽在卷积计算中的潜力[1-3]。通过共享指数将低位宽浮点数的存储效率与高位宽浮点数的计算精度相结合，从而使得混合位宽块浮点（block floating point, BFP）表示能够支持具有动态特定位宽（例如8b或16b）的卷积操作，同时可以保持较高的精度和性能[4-6]。混合BFP方法有效地平衡了效率和精度，展现了在高精度数值计算领域中推动技术进步的巨大潜力，从而开辟了全新的计算路径[7-9]。

近期研究表明，通过借助于现场可编程门阵列（field-programmable gate array, FPGA）等专用硬件平台，混合位宽计算已经取得了显著进展[10-12]。具体而言，FPGA凭借其可定制硬件的灵活性，能够为特定的混合位宽卷积操作提供高效的并行处理[13-15]。例如，为了充分发挥混合位宽的优势，HFP8[16]在FPGA上实现了混合位宽浮点数据表示方式，有效减少了精度损失。通过为卷积层的不同层配置不同的精度级别，混合位宽卷积技术能够显著优化计算资源的利用[17-21]。FAST[22]采用了可变精度的BFP表示（即8b或16b），并结合随机舍入技术进行基于FPGA的卷积计算，实现了高性能。BiE[23]提出了一种具有双共享指数的低精度BFP数据格式，通过并行处理尾数和指数，大幅提升了基于FPGA计算的吞吐量。然而，现有的方法往往未能充分利用数字信号处理器（digital signal processor, DSP）的能力，导致FPGA资源未能够得到最佳利用[24-26]。

本文提出了一个基于FPGA的混合BFP卷积处理单元，称为“BE-HB”，该单元针对BFP卷积计算（8b和16b）进行了优化，以实现更高的性能和更少的资源开销。根据观察，DSP组件可以有效地通过定点乘法处理BFP中的尾数计算。然而，由于与输入数据的大小相比，DSP的寄存器宽度相对较大，因此直接使用可能无法充分利用DSP资源。为此，本文开发了一种基于FPGA的双模式卷积处理单元，将2组尾数计算耦合到一个DSP中，用于8b和16b模式的BFP计算。此外，本文引入了一种数据映射方法，使得在8b和16b BFP计算中能够简化DSP内的数据处理，提升计算速率并优化DSP的利用率。通过综合评估，实验结果表明，与具有代表性的基线方法相比，所提出设计（BE-8）在相似的精度条件下，计算所需的LUT资源消耗平均降低了61.4%，性能提高了2.0倍。

本文的主要贡献包括2个方面：

1) 提出了一种新颖的基于FPGA的混合块浮点卷积计算单元，旨在将多组混合的8b和16b BFP输入打包到一个DSP中，以实现高性能的计算处理。

2) 提供了一种数据映射方法，该方法能够同时

支持 8b 和 16b BFP 输入，并且能够通过数据复用的方式优化计算过程中的计算速率和 DSP 利用率。

1 相关工作

随着深度神经网络模型规模的不断扩大，其训练和推理过程中的计算负载和硬件开销呈指数级增长，传统的全精度浮点（例如 FP32）已难以满足资源受限平台对于高性能、低开销场景下的需求，在此背景下，量化技术成为深度学习优化的常用手段之一[27-29]。例如，Tender[30] 设计了一种基于 8 位整数的 MAC 计算架构，从而加快了模型推理中的卷积操作。Bucket[31] 整合了定点和浮点格式的优点，提出了基于桶的累加计算单元以及自适应循环桶累加器等技术，优化了现有的 BFP 处理引擎，显著降低了计算能耗并提高了面积效率。尽管整数计算和低位宽的 BFP 数计算能够降低存储开销和提高计算效率，但它们在某些场景下可能难以保持模型精度[32-33]。

为了能够在降低计算和存储开销的同时尽可能地保持模型精度，李鹏程等人[34] 提出了一种基于移位–加法协同算法与非对称截断机制的可重构单精度近似浮点乘法器，在精度损失极小的情况下显著降低了功耗与面积。吕倩茹等人[35] 针对 Vision Transformer 模型，提出基于分治策略的自适应低位宽量化方法，通过为不同子模块动态选择量化位宽，实现了精度与资源消耗间的平衡。Wang 等人[36] 设计了改进的动态 BFP 格式，通过引入可调共享指数与重定标机制，有效缓解了传统 BFP 在注意力与矩阵乘法中的精度损失。Han 等人[37] 进一步提出了双向 BFP 量化加速器，在前向与反向计算中自适应共享指数，提高了数值稳定性与硬件利用率。上述研究在不同层面上缓解了计算开销与模型精度之间的权衡问题，但在特定场景下仍难以同时兼顾两者。

在此基础上，研究者们进一步探索了混合精度在不同的神经网络模型中的适应性和高效性[38-39]。SmoothQuant[40] 采用了在计算密集型算子（如线性层）中使用 INT8 运算，而在其他轻量级层使用 FP16 运算的方式来均衡精度和硬件开销。FINGA[41] 通过将浮点数表示的激活值转换为 BFP 数格式，与量化为定点整数的权重进行计算，从而避免了每次 MAC 操作中对齐和归一化操作所带来的额外开销，从而优化了模型中的 FP-INT 矩阵乘法操作并提高了计算效率。近年来的研究表明，专用硬件解决方案（如 FPGA）在卷积计算方面取得了显著进展。Zhang 等人[42] 提出了一种基于 BFP 算术的 FPGA 加速技术，通过 FPGA 来完成浮点数分离后的指数加法和尾数乘法操作，大幅提高了计算速率。Fan 等人[43] 通过最小化 KL 散度来确定静态共享指数，并基于静态 BFP 的量化方式在 FPGA 上高效部署卷积神经网络，并且专门为之设计了一个基于静态 BFP 量化的 FPGA 硬件卷积核，显著减少了逻辑计算资源的使用并提高了硬件效率。

然而，现有研究在设计和优化过程中往往忽略了 FPGA 中 DSP 计算资源的深度利用，并且未能有效地将其与混合精度有效结合。针对这一问题，本文设计了一个支持数据打包技术的混合卷积计算单元，以此来解决数据计算的动态精度表示和资源开销之间的平衡问题。

2 基础知识

图 1 展示了基于 BFP 表示方法的卷积过程[44-45]。在该过程中，卷积核的权重（W_1 和 W_2）与输入特征图（I_1 和 I_2）进行卷积操作，以生成最终的输出。所有这些操作都是基于 BFP 数据表示进行编码的。具体来说，每个数据元素都有共享的指数部分，同时具有独立的尾数部分。例如，输入激活值 A_{21} 和 A_{22} 共享相同的指数值 E_{a2}。这意味着在进行计算时，它们的指数部分是相同的，而尾数部分则需要根据每个输入的尾数值以及原始指数值与共享指数值之间的差来共同确定。进一步地，给定一组 BFP 数 $(x_1, x_2, \cdots, x_n)$，这些数可以通过共享相同的指数并单独编码尾数来表示，如式（1）所示。

$$2^{E_S} \times \left[(-1)^{S_1} m_1, \cdots, (-1)^{S_n} m_n\right]。 \tag{1}$$

图 2 描述了 Xilinx FPGA Ultrascale 系列中 DSP 内部的基本结构[46-47]。具体来说，每个 DSP 由 1 个预加法器、1 个乘法器和 1 个加法器组成。DSP 具有各种寄存器和操作模式。寄存器 A, B, C, D, P 是用于数据存储的寄存器。inmode 和 opmode 表示可配置的操作模式。

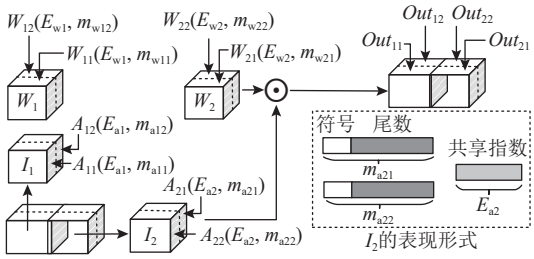


Fig. 1 BFP-based convolution

图 1 基于 BFP 的卷积

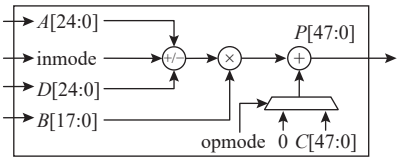


Fig. 2 Basic architecture of DSP

图 2 DSP 的基本架构

3 研究动机

根据图 1 和式(1)的分析，可以观察到 BFP 数的尾数和指数可以分开单独进行处理，而且 DSP 非常适合支持定点数的乘法计算。因此，本文利用 DSP 这一特性，将 BFP 数的尾数部分作为定点数输入到 DSP 中，使其能够适应 DSP 的计算架构。接着，根据共享指数对尾数进行处理，以恢复 BFP 数的完整表示。通过这种方式，所提出设计能够实现 DSP 对混合 BFP 卷积计算的有效支持，从而在保持高效计算的同时，提升卷积操作的性能。

此外，在计算混合位数据的尾数乘法时，可以清楚观察到 BFP 数的单个尾数的位宽远远小于 DSP 的固有输入位宽(即 25 b)，这无疑会导致 DSP 输入位宽的浪费和计算效率低。为了解决这个问题，本文通过部署 DSP 并采用数据打包技术，使得 DSP 能够同时处理 2 组基于 BFP 的乘法计算，从而提高数据的计算性能和吞吐量。值得注意的是，在数据打包的过程中，DSP 可能会误将打包结果中的低位有符号数识别为无符号数，这可能会导致计算结果出现偏差。为了避免这一潜在问题，本文引入了校正机制，通过加入校正值动态调整计算结果来修正偏差，确保最终的输出更加精确和可靠。

4 设计方法

4.1 硬件架构

图 3 描述了 BE-HB 的整体架构，它能够支持双模式(8 b 和 16 b 模式)的 BFP 卷积操作。它主要包括 3 个组件：输入打包(IPA)、计算(Cal)和移位(Shf)组件。IPA 耦合 2 组 8 b 或 16 b 的 BFP 数据作为输入；Cal 负责执行乘法和加法运算；Shf 则能够根据最终输出的共享指数来移动尾数得到最终的结果。

具体来说，IPA 组件同时支持将 8 b 和 16 b 的 BFP 输入数据打包成 2 组进行计算。例如，在 8 b 数据模式下，IPA 将 2 个权重的尾数 m_{w1} 和 m_{w2} 组合在一起，并将结果存储在寄存器 A 中，同时将激活值尾数 m_{a1}

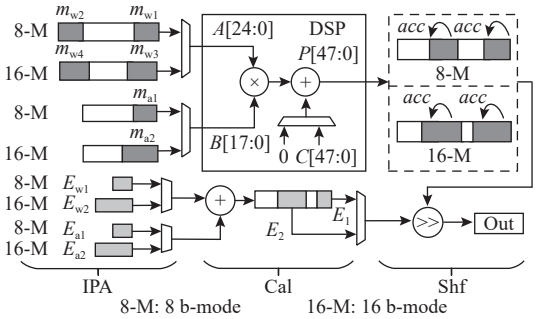


Fig. 3 BE-HB architecture

图 3 BE-HB 架构

分配给寄存器 B。同样，在 16 b 模式下，2 个权重的尾数被共同存储到寄存器 A 中。鉴于 16 b BFP 数据的尾数通常都小于完整的 16 b 宽度(例如，m_{w3} 和 m_{w4} 的位宽都为 8 b)，因此寄存器 A 可以容纳 16 b BFP 数据的多组尾数。

Cal 组件包括 2 个主要元件：加法器和 DSP，分别用于指数加法和尾数乘法。在加法器中，通过基于共享指数进行的加法操作来获得最终的共享指数。在 DSP 中，主要是让 2 组尾数权重(位于寄存器 A 中)和 1 组激活值尾数(存储在寄存器 B 中)分别同时进行乘法计算。值得注意的是，由于 2 组尾数内的低位数据(即 m_{w1} 和 m_{w3})会被 DSP 视为无符号数据进行运算，因此需要进行动态调整以纠正输出值，确保输出结果的准确性，这是通过寄存器 C 的偏移值加法实现的。值得注意的是，C 中的偏移值如式(2)所示。

$$C = -A[n-1] \times B \times 2^{n}, \tag{2}$$

其中 $A[n-1]$ 表示寄存器 A 中的低位宽度输入(例如 m_{w1} 或 m_{w3})的符号位，n 表示位宽。B 表示寄存器 B 中的值。

Shf 组件通过移位逻辑将结果连接起来，以获得最终输出。具体而言，Shf 以 8 b 和 16 b 模式从 DSP 中获取累加的 BFP 尾数乘法结果，以及 Cal 组件计算得出的指数加法结果作为输入。在这 2 种模式下，acc 会根据 Cal 组件输出的不同共享指数相应地移动(例如，$acc \gg E_a$ 或 $acc \gg E_b$)。

图 4 展示了 BE-HB 的一个具体数据输入示例，它在 8 b 和 16 b 模式下使用表 1 中所展示的 BE-HB 特定的位宽，尾数(包括符号位)分别使用 5 b 和 8 b，指数则分别使用 3 b 和 8 b 的位宽。DSP 输入和输出寄存器的固定位宽直接影响尾数在这 2 种模式下可以打包的最大组数。在图 4 中，8 b 和 16 b 模式下尾数可以打包的最大组数分别为 3 和 2。然而，通过后续的实验证明，8 b 和 16 b 模式的最佳打包组数量均为 2。

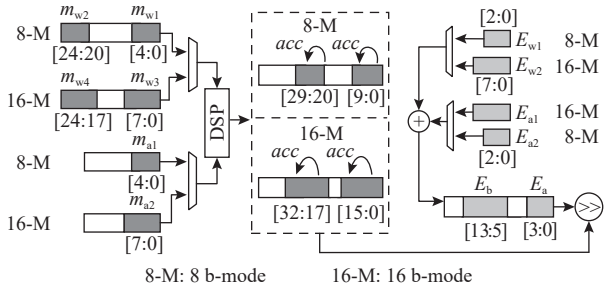


Fig. 4 Example of BE-HB with specific bit width input

图 4 采用具体位宽输入的 BE-HB 示例

Table 1 Data Bit-Width of Baseline and the Proposed Method

表 1 基线和本文方法的数据位宽

方法	数据位宽/b	具体位宽组成
FP16[13]	16	1*S*-5*E*-10*M*
BFloat16[12]	16	1*S*-8*E*-7*M*
Tender[30]	8	1*S*-7*M*
HFP8[16]	8	1*S*-4*E*-3*M*，1*S*-5*E*-2*M*
FAST[22]	7	1*S*-4*E*-2*M*
BiE[23]	10	1*S*-5*E*-4*M*
BE-HB（本文）	8/16	1*S*-3*E*-4*M*，1*S*-8*E*-7*M*

注：*S*，*E*，*M* 分别表示符号、指数和尾数的位宽，如 1*S*-3*E*-4*M* 表示符号、指数和尾数的位宽分别为 1，3，4。

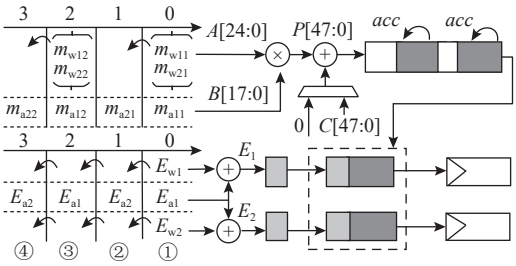
4.2 调度数据流

图 5(a)展示了基于图 1 的 BFP 卷积的映射数据流，该数据流由 2 部分组成：BFP 尾数和共享指数。BE-HB 采用 E_1 和 E_2 作为指数输入，将 A 和 B 作为尾数输入，并按从周期 0 到周期 3 的顺序提供数据。接着它按顺序输出基于 BFP 的乘法结果，如图 5(a)中的①~④所示。符号“{}”表示数据打包操作，弯曲箭头表示重用前一个周期的数据。例如，循环 1 可以重用循环 0 中的打包数据 m_{w11}，m_{w21}。循环 3 可以重用循环 2 中的打包数据 m_{w12}，m_{w22}。同样，循环 1~3 也能重用循环 0 中权重的 2 个共享指数 E_{w1} 和 E_{w2}，通过数据复用的方式能够减少数据传输所带来的开销并提高计算效率。

图 5(b)展示了图 1 中 BFP 卷积所需的计算，其中序号①~④表示乘法运算的顺序。BE-HB 可以在 1 个周期内对相同数量的乘法或加法操作进行计算，4 个周期可以完成 4 组数据的计算。以 Out_{11} 为例，序号①③表示 Out_{11} 是通过将①中计算的一组结果与③中获得的另一组结果相加而获得的。

4.3 具体实例

图 6(a)展示了 8 b 和 16 b 模式下的计算流程。其中权重张量和激活张量均采用 BFP 编码表示，每



(a) BE-HB映射数据流

$$Out_{11}=(m_{w11}\times m_{a11})\times 2^{E_1}+(m_{w12}\times m_{a12})\times 2^{E_2}\quad ①③$$

$$Out_{21}=(m_{w21}\times m_{a11})\times 2^{E_1}+(m_{w22}\times m_{a12})\times 2^{E_2}\quad ①③$$

$$Out_{12}=(m_{w11}\times m_{a21})\times 2^{E_3}+(m_{w12}\times m_{a22})\times 2^{E_4}\quad ②④$$

$$Out_{22}=(m_{w21}\times m_{a21})\times 2^{E_3}+(m_{w22}\times m_{a22})\times 2^{E_4}\quad ②④$$

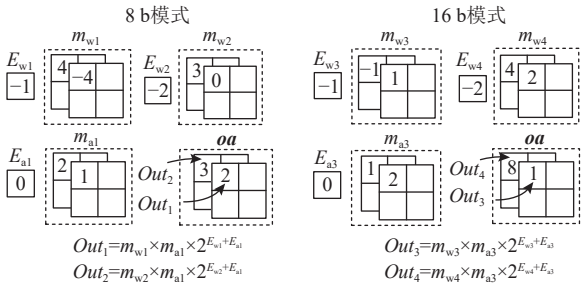
$$E_1=E_{w1}+E_{a1}\qquad E_2=E_{w2}+E_{a1}$$

$$E_3=E_{w1}+E_{a2}\qquad E_4=E_{w2}+E_{a2}$$

(b) BE-HB卷积计算

Fig. 5 Mapping dataflow

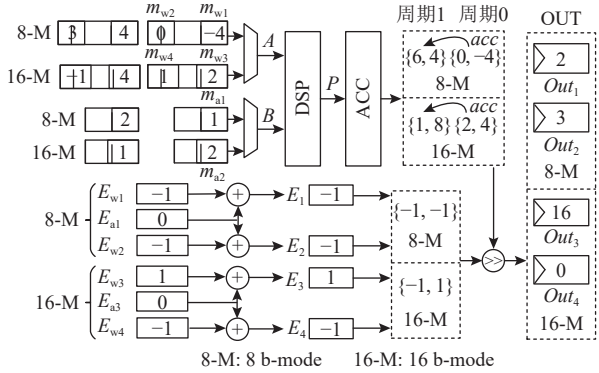
图 5 映射数据流



$$Out_1=m_{w1}\times m_{a1}\times 2^{E_{w1}+E_{a1}}\qquad Out_3=m_{w3}\times m_{a3}\times 2^{E_{w3}+E_{a3}}$$

$$Out_2=m_{w2}\times m_{a1}\times 2^{E_{w2}+E_{a1}}\qquad Out_4=m_{w4}\times m_{a3}\times 2^{E_{w4}+E_{a3}}$$

(a) 8 b和16 b模式下的计算流程



(b) BE-HB 2次卷积计算的完整计算流程

Fig. 6 Specific example of BE-HB

图 6 BE-HB 具体实例

个张量块由若干个尾数值(mantissa)和一个共享的指数值(exponent)构成。以 8b 模式为例，首先将 2 个权重块的尾数部分 m_{w1} 和 m_{w2} 分别乘以对应指数 E_{w1} 和 E_{w2} 值的幂，从而将其还原为近似的浮点数值；接着与还原为相同浮点形式的激活块相乘，得到 2 个乘积结果，并最终将它们累加至输出张量 ***oa*** 的相应位置。图 6(b)为 2 次卷积计算的完整计算流程，在第 1 个周期时，8b-mode 和 16b-mode 两种模式下的前 2 份权重数据被打包成 1 组数据，随后与对应的激活

值数据一同送入 DSP 模块进行计算；通过流水线设计，在第 2 个周期时，待第 1 个周期的数据在 DSP 中计算完成之后，后 2 份权重数据也同样被打包成 1 组，并与对应的激活值一同送入 DSP 模块中进行计算。DSP 模块的计算结果将会被送入 ACC 模块进行累加操作，这 2 种模式下的累加结果都需要进行移位恢复计算；共享指数的加法计算与尾数的计算可以并行执行，累加结果通过最终得到的共享指数的移位之后会送入输出寄存器，从而完成 2 次卷积计算的流程。

5 实验结果

5.1 实验设置

本文将所提出的设计与 6 个具有代表性的基线方法进行了比较，其中包括 FP16[13]，Bfloat16，BiE[23]。具体来说，FP16 使用 16 b 浮点数进行卷积计算；Bfloat16 提供了一个自定义的 16 b 浮点乘法累加单元，以减少硬件资源开销；Tender 通过将数据量化为 8 b 整数，从而实现 8 位定点整数乘法计算，能够在精度损失不大的情况下提高计算性能；HFP-8 能够支持 2 种不同位宽分配类型的 8 b 浮点卷积计算；FAST 提出了一个 BFP 卷积单元 FMAC，它能够通过共享指数的方式简化浮点计算，从而提高计算效率；BiE 则是通过采用具有双共享指数的 BFP 表示方法来执行基于 BFP 的卷积运算。

本文从 2 个方面开展评估。首先，将 BE-HB 与全精度浮点（FP）以及 3 种代表性的基线方法（Tender，FAST，BiE）进行了精度比较。为进行全面的评估，本文选取了一系列典型卷积神经网络模型作为基准，包括 ResNet-18[48]，VGG-16[49]，MobileNetV2，GoogleNet，AlexNet，ConvNeXt-Tiny，这些模型均基于 ImageNet-1K[50] 进行训练和优化。为了模拟精度数值，本文统一使用 Pytorch2.1.0 在 NVIDIA RTX 4 090 GPU 上量化激活和权重。

其次，为了验证加速单元架构在实际硬件环境中的性能表现，本文采用 Xilinx Vivado Design Suite 工具实现架构设计，并将平台部署在 ZNYQ XC7VX485 上，该平台具备丰富的逻辑资源以及高性能的 DSP 运算单元（DSP48E1）。对于所设计架构的计算资源和硬件开销的评估，分为 2 步。首先，本文提供 3 种类型的配置，分别是：所提出的设计仅能够支持 16 b 卷积计算，表示为 BE-16；所提出的结构仅支持 8 b 卷积计算，表示为 BE-8；所提出设计同时支持 8 b 和 16 b 混合卷积计算，称为 BE-HB。其次，为了更好地评估资源消耗，本文基于不同的 FPGA 资源实现了每种配置（BE-16，BE-8，BE-HB）：1）该设计仅使用 LUT 资源实现，表示为“NDSP”；2）该设计由 DSP 资源实现，但没有采用数据打包技术，称为“DSP-NP”；3）该设计是由 DSP 资源和数据打包技术共同实现的，表示为“DSP-P”。

5.2 实验结果

表 2 展示了在多个模型下的精度结果。与 FP 相比，BE-HB 保持了相似的精度水平，但未能明显超过 FP，主要是因为 FP16 在所有网络层统一使用 16 b 表示，虽然精度较高，但也导致较大的计算和存储开销。与此不同，所提出的方法能够灵活支持 8 b 和 16 b 的混合表示：对于精度要求较低的层，采用 8 b 表示；而对于精度要求较高的层，采用 16 b 表示，这种策略能够显著减少存储开销。虽然计算开销有所增加，但通过所提出的 DSP 数据打包技术，可以有效地减少计算开销，从而在性能和存储开销之间达到良好的平衡。与基于 INT8 的 Tender 和基于低比特 BFP 的 FAST 相比，BE-HB 在平均精度上分别提高了 2.45% 和 2.58%，这一提升主要得益于混合位宽策略所带来的优势，使用了类似计算资源的 BE-8 在精度也表现

Table 2 Accuracy Comparison of BE-HB and Baselines on Six Models

表 2 BE-HB 和基线在 6 个不同模型上的精度比较

方法	精度/%							
	(*I*/*W*)/b	ResNet-18	VGG-16	MobileNetV2	GoogleNet	AlexNet	ConvNeXt-Tiny	几何平均值
FP	16/16	71.47	71.59	71.87	62.46	56.56	82.13	68.87
Tender	8/8	68.47	69.21	69.45	59.67	54.68	79.76	66.39
FAST	7/7	68.21	68.91	68.72	59.57	54.28	80.99	66.26
BiE	10/10	71.12	70.82	69.76	62.01	56.01	81.72	68.10
BE-8（本文）	8/8	70.95	70.67	69.64	61.95	55.91	81.69	67.99
BE-HB（本文）	8, 16/8, 16	71.44	71.57	71.81	62.42	56.53	82.12	68.84

注：“8, 16”表示 8 b 和 16 b 的数据混合，即一部分数据用 8 b 表示，一部分数据用 16 b 表示。

出一定的提升, 分别提高了 1.60 个百分点和 1.73 个百分点。此外, 与 BiE 相比, BE-8 在计算开销更小的情况下, 保持了相似的精度, 而使用了略多一些计算资源的 BE-HB 的精度能够平均提高 0.85 个百分点, 这些结果表明混合位宽策略的有效性和在不同模型中的适用性。

表 3 显示了不同配置和 FPGA 资源下所提出设计的资源开销, 可以观察到: 1)DSP 资源的利用可以显著降低 LUT 资源开销, 例如, 对于 BE-16 和 BE-8, 使用 DSP(DSP-NP)的设计比仅使用 LUT(NDSP)的实现更有效地进行计算; 2)DSP 上的数据打包操作可以以较少的资源开销实现相同的性能, 例如, 对于 BE-HB, DSP-P(数据打包)的利用可以在仅占用一半 DSP 资源的情况下实现相同的性能, 但需要更多的 LUT 资源(1.15 倍), 额外的 LUT 资源开销来自对几个 BFP 数据的打包操作以及校正值的计算; 3)利用 DSP 资源可以有效降低 FF 资源开销, 例如, 在 BE-HB 中, NDSP 比 DSP-NP 消耗更多的 FF 资源(1.41 倍), 这是因为集成在 DSP 中的寄存器有助于大大减少数据的存储开销。此外, 由于 DSP 强大的计算能力, 利用 DSP 可以显著降低 LUT 资源开销。

Table 3 Resource Overhead of the Proposed Method (Sixteen Groups of Data)

表 3 本文方法的资源开销(16 组数据)

方法	类型	打包组数 (8-M/16-M)	资源个数		
			LUT	FF	DSP
BE-16	NDSP	0/0	1 289	601	0
	DSP-NP	1/1	295	345	16
	DSP-P	2/2	358	425	8
BE-8	NDSP	0/0	586	379	0
	DSP-NP	1/1	168	220	16
	DSP-P	2/2	243	276	8
BE-HB	NDSP	0/0	1 597	699	0
	DSP-NP	1/1	761	496	16
	DSP-P	2/2	874	608	8

注: 8-M 和 16-M 分别表示 8 b-mode 和 16 b-mode; LUT, FF, DSP 分别表示查找表、寄存器、数字信号处理器。

表 4 展示了本文通过与最近的代表性基线 FP, INT, BFP 的 MAC 设计进行比较来评估 BE-HB 的性能和效率。对于浮点数的 MAC 运算, 分别使用 FP16, BFloat16, HFP8 实现。而对于整数以及 BFP 的 MAC 计算, 本文选择 BiE 作为比较基准。为了确保公平比较, 使用 16 组数据来计算所有的基线设计, 包括所提出的设计。

Table 4 Results Compared with Baselines (Sixteen Groups of Data)

表 4 与基线的比较结果(16 组数据)

方法	资源个数		
	LUT	FF	DSP
FP16	1 514	753	-
BFloat16	1 305	684	-
HFP8	527	220	-
Tender	498	195	-
BiE	630	354	-
BE-16(本文)	358	425	8
BE-8(本文)	243	276	8
BE-HB(本文)	874	608	8

注: LUT, FF, DSP 分别表示查找表、寄存器、数字信号处理器; "-"表示不用 DSP 资源计算。

与 16 b 浮点数表示的基线(FP16, BFloat16)相比, 所提出的利用 DSP 资源和数据打包技术的 16 b 编码(BE-16)设计分别仅消耗其 23.7% 和 27.4% 的 LUT 资源, 减少的 LUT 资源来自所提出设计中利用数据打包和 DSP 的优势。BE-16 的 FF 资源消耗仅为 FP16 和 BFloat16 的 56.4% 和 62.1%, 这是由于 DSP 的内置寄存器减少了数据存储所带来的开销。此外, 与 FP16 和 BFloat16 相比, BE-HB 实现了更低的 LUT 资源开销, 分别为 FP16 和 BFloat16 资源开销的 57.7% 和 67%。与由 8 b 数据表示组成的基线 HFP8 和 INT8 相比, 所提出的设计 BE-8 仅消耗了 46.1% 和 48.8% 的 LUT 资源。

与基于 BFP 的 BiE 相比, 在相同的尾数位宽下, 由于 DSP 部署期间采用的数据打包策略, BE-8 仅消耗 38.6% 的 LUT 资源和 78.0% 的 FF 资源。另一方面, 由于 BE-HB 需要同时支持 8 b 和 16 b 模式的 BFP 卷积计算, 因此相对于 BiE 来说, BE-HB 需要多消耗 1.39 倍的 LUT 资源。

为了进一步分析资源影响, 本文进行了不同的打包分组。对于 16 b 的 BFP 输入, 所提出设计最多支持 2 组, 因为其尾数宽度较大, 3 组的输出结果会超过 DSP 自带的输出寄存器宽度; 对于 8 b 的 BFP 输入, 所提出的设计最多支持 3 组输入。表 5 展示了不同输入打包组下的资源开销。以 8 b 模式为例, 与打包成 2 组的策略(BE-8)相比, 打包成 3 组(BE-8T)可以节省少部分 DSP 资源, 但这也会导致 LUT 资源消耗增加 32.1%, 并引起 FF 资源的额外消耗。这一资源消耗的增加主要是由于打包 3 组的 DSP 内校正值计算会更加复杂, 需要关注 2 组值的纠正计算, 并且

Table 5 Resource Overhead of the Proposed Method Under Different Input Packing Groups (16 Groups of Data)

表 5 不同输入打包组下本文方法的资源开销（16 组数据）

方法	打包组数 (8-M/16-M)	资源个数		
		LUT	FF	DSP
BE-8	2/-	243	220	8
BE-8T	3/-	385	316	6
BE-HB	2/2	874	608	8
BE-HBT	3/2	1 019	636	8

注：LUT，FF，DSP 分别表示查找表、寄存器、数字信号处理器；“-”代表没有打包。

这些计算对资源的需求更高。此外，从表 5 中还可以观察到，在混合位宽 BFP 的卷积计算中，即使在相同的 DSP 资源条件下，打包 2 组 8 b 数据尾数（BE-HB）的总成本也低于打包 3 组（BE-HBT）。这进一步证明，在资源利用效率方面，2 组打包策略比 3 组打包策略更具有优势。

图 7 展示了在不同网络下的归一化资源效率。本文采用 GOPS/LUT 来评估，可以观察到，与 FP16 相比，BE-HB 的资源效率平均提升了 3.5 倍，这主要得益于其卓越的性能和较低的 LUT 消耗。与 BiE 相比，在相似的精度条件下，BE-8 的资源效率能够平均提高 5.2 倍，其中在性能上提升了 2.0 倍，这一性能的提升主要归因于其能够充分利用 DSP 内部自带的较大的数据位宽，从而实现双数据组并行处理的操作，并结合设计的数据调度流，以此来提升整体数据的吞吐能力。尽管 BE-HB 与 BiE 相比仅提高了 1.4 倍的资源效率，但它能够同时支持 2 种不同模式的卷积计算，并有着更低的模型精度损失。值得注意的是，对于这些网络本文采用了相同的数据流设计，因此所提出设计的性能提升在不同的网络间保持一致。

总的来说，BE-HB 架构通过混合精度的设计、映射数据流的优化以及数据打包操作的融合，显著提升了性能和计算效率，在不同的神经网络下均展现出优异的性能表现，验证了其在推理加速场景下的高效性和适应性。

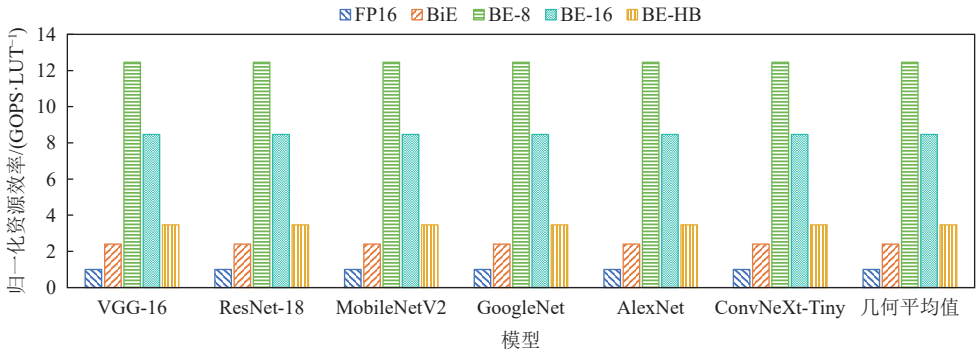


Fig. 7 Normalized resource efficiency comparison

图 7 归一化资源效率比较

本文提出的基于块浮点（BFP）的混合位宽卷积处理单元（BE-HB）在资源受限的硬件平台上展现了显著的性能和资源效率优势，尤其在大语言模型的应用场景中具有重要的应用价值。BE-HB 通过混合位宽的 BFP 表示，在保持模型精度的同时显著降低了计算和存储开销，使得大语言模型在 FPGA 等资源受限的硬件平台上能够更加高效地运行。以 GPT-3 为例，通过将模型的权重和激活值量化为 8 b 或 16 b 的 BFP 表示，并在 FPGA 上部署 BE-HB 卷积处理单元，可以显著优化模型的计算效率和资源消耗，同时保持高精度，满足实时文本生成等任务的需求。在 Opt 和 Llama 模型的多个数据集测试中，BE-HB 与原始精度几乎一致，相较于基于 INT8 的 Tender 平均提升了 1.52%，相较于基于 BFP 的 BiE 平均提升了 1.03%，进一步证明了其在实际应用中的高效性和适应性。

6 结束语

本文提出了一种新型的卷积处理单元，该单元通过巧妙低地利用混合块浮点技术，实现了卷积计算中计算效率与精度之间的动态平衡。此外，该单元以 DSP 为核心执行关键操作，并引入高效的打包策略，进一步优化了资源利用率和计算性能。值得注意的是，与目前先进的块浮点卷积技术相比，所提出的设计（BE-8）在相似的精度条件下，能够在推理性能上实现 2.0 倍的提升，并在资源效率上实现了 5.2 倍的提升。此外，BE-HB 在保持原始模型精度的同时，资源效率提升了 1.4 倍。这一计算方法的改进

显著提升了深度学习模型在资源受限的硬件平台上的部署效率，为其高效应用提供了强有力的支持。

作者贡献声明：李东阳设计了本文的研究方案，进行了硬件架构设计、主要实验的实现和验证以及论文初稿的撰写；李泽洋完成了精度实验和论文格式的调整；唐忆滨和刘博生参与了论文的修订工作；武继刚对论文的整体核心思想提供了指导，并对论文进行了最终的修改和定稿。

参 考 文 献

[1] Choi Y, El-Khamy M, Lee J. Universal deep neural network compression[J]. IEEE Journal of Selected Topics in Signal Processing, 2020, 14(4): 715−726

[2] Lü Junhuan. Research on mixed-bit-width convolutional neural network model quantization methods[D]. Beijing: Beijing Jiaotong University, 2022 (in Chinese)
(吕君环. 混合比特位宽卷积神经网络模型量化方法研究[D]. 北京：北京交通大学, 2022)

[3] Liu Hongda, Sun Xuhui, Li Yibin, et al. Survey on deep learning models for image classification based on convolutional neural networks[J]. Computer Engineering and Applications, 2025, 61(11): 1−21 (in Chinese)
(刘鸿达, 孙旭辉, 李沂滨, 等. 基于卷积神经网络的图像分类深度学习模型综述[J]. 计算机工程与应用, 2025, 61(11)：1−21)

[4] Zhao W, Dang Q, Xia T, et al. Optimizing FPGA-Based DNN accelerator with shared exponential floating-point format[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2023, 70(11): 4478−4491

[5] Tian Cheng, Li Zhengjie, Chen Gongfu, et al. Survey on low-bit quantization methods for deep neural networks[J]. Modern Information Technology, 2025, 9(10): 23−33,38 (in Chinese)
(田程, 李正杰, 陈功富, 等. 深度神经网络低比特量化方法综述[J]. 现代信息科技, 2025, 9(10)：23−33, 38)

[6] Darvish Rouhani B, Zhao R, Elango V, et al. With shared microexponents, a little shifting goes a long way[C]//Proc of the 50th Annual Int Symp on Computer Architecture. New York: ACM, 2023: 1−13

[7] Lü Junhuan, Xu Ke, Wang Dong. Low-bit quantization of neural networks via exponential moving average knowledge distillation[J]. Pattern Recognition and Artificial Intelligence, 2021, 34(12): 1143−1151 (in Chinese)
(吕君环, 许柯, 王东. 基于指数移动平均知识蒸馏的神经网络低比特量化方法[J]. 模式识别与人工智能, 2021, 34(12)：1143−1151)

[8] Bhatti U A, Tang Hao, Wu Guilu, et al. Deep learning with graph convolutional networks: An overview and latest applications in computational intelligence[J]. International Journal of Intelligent Systems, 2023, 2023(1): 8342104

[9] Junaid M, Arslan S, Lee T G, et al. Optimal architecture of floating-point arithmetic for neural network training processors[J]. Sensors, 2022, 22(3): 1230

[10] Liu Hongwei, Pan Ling, Wu Mingqin, et al. Design and implementation of a lightweight deep-learning computing architecture for FPGA clusters[J]. Telecommunication Engineering, 2024, 64(1): 14−21 (in Chinese)
(刘红伟, 潘灵, 吴明钦, 等. 一种 FPGA 集群轻量级深度学习计算架构设计及实现[J]. 电讯技术, 2024, 64(1)：14−21)

[11] Xu Yan, Wang Shuaishuai, Li Ning, et al. Design and implementation of an efficient CNN accelerator for low-cost FPGAs[J]. IEICE Electronics Express, 2022, 19(19): 20220370

[12] Wang Shibo, Kanwar P. BFloat16: The secret to high performance on cloud TPUs[EB/OL]. Google Cloud Blog, 2021 [2025-10-23]. https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus

[13] Ootomo H, Yokota R. Recovering single precision accuracy from Tensor Cores while surpassing the FP32 theoretical peak performance[J]. The International Journal of High Performance Computing Applications, 2022, 36(4): 475−491

[14] Wen Jiabao, Feng Yan, Li Zhiqiang. A high-throughput hardware architecture for bilateral filter with configurable convolution and cost-effective MAC unit[J]. IEICE Electronics Express, 2024, 21(13): 20240276

[15] Wang Zeyu, Ikeda M. High-throughput and fully-pipelined ciphertext multiplier for homomorphic encryption[J]. IEICE Electronics Express, 2024, 21(6): 20230628

[16] Sun Xiao, Choi J, Chen C Y, et al. Hybrid 8-bit floating point (HFP8) training and inference for deep neural networks[J]. Advances in Neural Information Processing Systems, 2019, 32: 4900−4909

[17] Oh Y H, Kim S, Jin Y, et al. Layerweaver: Maximizing resource utilization of neural processing units via layer-wise scheduling [C]//Proc of the 27th IEEE Int Symp on High-Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2021: 584−597

[18] Drumond M, Lin Tao, Jaggi M, et al. Training DNNs with hybrid block floating point[J]. Advances in Neural Information Processing Systems, 2018, 31: 451−461

[19] Tan Mingxing, Le Q V. Mixconv: Mixed depthwise convolutional kernels[J]. arXiv preprint, arXiv: 1907.09595, 2019

[20] Kim Y, Jang J, Lee J, et al. Winning both the accuracy of floating point activation and the simplicity of integer arithmetic[C/OL] // Proc of the 11th Int Conf on Learning Representations. 2023 [2025-10-23]. https://openreview.net/forum?id=z92lBy1ehjI

[21] Zeng Zhaoping, Yang Liangtao, Li Guangbin, et al. A high speed and high accurate floating-point EEG signal bandpass filter based on FPGA[C] // Proc of the 17th Int Conf on Complex Medical Engineering (CME). Piscataway, NJ: IEEE, 2023: 16−19

[22] Zhang Saiqian, McDanel B, Kung H T. FAST: DNN training under variable precision block floating point with stochastic rounding[C]//Proc of the 28th IEEE Int Symp on High-Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2022: 846−860

[23] Zou Lancheng, Zhao Wenqian, Yin Shuo, et al. BiE: Bi-exponent block floating-point for large language models quantization[C]//Proc

of the 41st Int Conf on Machine Learning. New York: ACM, 2024: 62978−62992

[24] Cheng Chao, Parhi K K. Fast 2D convolution algorithms for convolutional neural networks[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2020, 67(5): 1678−1691

[25] Ullah S, Rehman S, Shafique M, et al. High-performance accurate and approximate multipliers for FPGA-based hardware accelerators[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2021, 41(2): 211−224

[26] Huang Wenjin, Wu Huangtao, Chen Qingkun, et al. FPGA-based high-throughput CNN hardware accelerator with high computing resource utilization ratio[J]. IEEE Transactions on Neural Networks and Learning Systems, 2021, 33(8): 4069−4083

[27] Rokh B, Azarpeyvand A, Khanteymoori A. A comprehensive survey on model quantization for deep neural networks in image classification[J]. ACM Transactions on Intelligent Systems and Technology, 2023, 14(6): 1−50

[28] Xu Yuhui, Wang Yongzhuang, Zhou Aojun, et al. Deep neural network compression with single and multiple level quantization[C]//Proc of the 32nd AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2018: 4338−4346

[29] Zhao Yilong, Lin C Y, Zhu Kan, et al. Atom: Low-bit quantization for efficient and accurate LLM serving[J]. Proceedings of Machine Learning and Systems, 2024, 6: 196−209

[30] Lee J, Lee W, Sim J. Tender: Accelerating large language models via tensor decomposition and runtime requantization[C]//Proc of the 51st IEEE Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2024: 1048−1062

[31] Lo Yunchen, Liu Renshuo. Bucket getter: A bucket-based processing engine for low-bit block floating point (BFP) DNNs[C]//Proc of the 56th Annual IEEE Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2023: 1002−1015

[32] Potkins F, Langton O, Smirnov V, et al. Improve the accuracy and efficiency of large language models via dynamic token compression and adaptive layer pruning[J/OL]. Authorea Preprints, 2024 [2025-10-23]. https://www.techrxiv.org/doi/full/10.36227/techrxiv.173030653.35636983

[33] Singh G, George R P, Ahmad N, et al. A deep learning approach for evaluating the efficacy and accuracy of PoseNet for posture detection[J]. International Journal of System Assurance Engineering and Management, 2024: 1−10

[34] Li Pengcheng, Huang Libo, Chen Gang, et al. A reconfigurable single-precision approximate floating-point multiplier design[J]. Journal of Computer Research and Development, 2025, 62(6): 1581−1593 (in Chinese)
(李鹏程, 黄立波, 陈刚, 等. 一种可重构的单精度近似浮点乘法器设计[J]. 计算机研究与发展, 2025, 62(6): 1581−1593)

[35] Lü Qianru, Xu Jinwei, Jiang Jingfei, et al. DAQ: Adaptive vision Transformer low-bit quantization via divide-and-conquer[J]. Journal of Computer Research and Development, 2025, 62(6): 1530−1546 (in Chinese)
(吕倩茹, 许金伟, 姜晶菲, 等. DAQ: 基于分治策略的自适应 Vision Transformer 低位宽量化方法[J]. 计算机研究与发展, 2025, 62(6): 1530−1546)

[36] Wang Hui, Cheng Yuan, Han Xiaomeng, et al. Pushing the limits of BFP on narrow precision LLM inference[C]//Proc of the 39th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2025: 21099−21107

[37] Han Xiaomeng, Cheng Yuan, Wang Jing, et al. BBAL: A bidirectional block floating point-based quantisation accelerator for large language models[C]//Proc of the 62nd IEEE Design Automation Conf (DAC). Piscataway, NJ: IEEE, 2025: 1−7

[38] Li Boyang, Wang Longguang, Wang Yinqian, et al. Mixed-precision network quantization for infrared small target segmentation[J]. IEEE Transactions on Geoscience and Remote Sensing, 2024, 62: 1−12

[39] Frantar E, Castro R L, Chen Jiale, et al. Marlin: Mixed-precision auto-regressive parallel inference on large language models[C]//Proc of the 30th ACM SIGPLAN Annual Symp on Principles and Practice of Parallel Programming. New York: ACM, 2025: 239−251

[40] Xiao Guangxuan, Lin Ji, Seznec M, et al. SmoothQuant: Accurate and efficient post- training quantization for large language models[C]// Proc of the 40th Int Conf on Machine Learning. New York: PMLR, 2023: 38087−38099

[41] Jang J, Kim Y, Lee J, et al. FIGNA: Integer unit-based accelerator design for FP-int gemm preserving numerical accuracy[C]//Proc of the 30th IEEE Int Symp on High-Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2024: 760−773

[42] Zhang Heshan, Liu Zhenyu, Zhang Guanwen, et al. A block-floating-point arithmetic based FPGA accelerator for convolutional neural networks[C]// Proc of the 2019 IEEE Global Conf on Signal and Information Processing (GlobalSIP). Piscataway, NJ: IEEE, 2019: 1−5

[43] Fan Hongxiang, Wang Gang, Ferianc M, et al. Static block floating-point quantization for convolutional neural networks on FPGA [C]//Proc of the 2019 Int Conf on Field-Programmable Technology (ICFPT). Piscataway, NJ: IEEE, 2019: 28−35

[44] Nascimento M G, Prisacariu V A, Fawcett R, et al. Hyperblock floating point: Generalised quantization scheme for gradient and inference computation[C]//Proc of the IEEE Winter Conf on Applications of Computer Vision. Piscataway, NJ: IEEE, 2023: 6364−6373

[45] Chang Chengyang, Huang Chise, Chuang Yuchuan, et al. BFP-CIM: Data-free quantization with dynamic block-floating-point arithmetic for energy-efficient computing-in-memory-based accelerator[C]//Proc of the 29th Asia and South Pacific Design Automation Conf (ASP-DAC). Piscataway, NJ: IEEE, 2024: 545−550

[46] Nguyen D, Kim D, Lee J. Double MAC: Doubling the performance of convolutional neural networks on modern FPGAs[C]// Proc of the 20th Design, Automation & Test in Europe Conf & Exhibition (DATE). Piscataway, NJ: IEEE, 2017: 890−893

[47] Bharathi M, Shirur Y J M. Efficiency evaluation of scalable multiply and accumulate architectures in DSP: A comparative study of LUT based and LUT-Less based approaches[C]//Proc of the 1st Int Conf on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE). Piscataway, NJ: IEEE, 2024: 1−5

[48] Kang Zhen, Xiao Enhua, Li Zhen, et al. Deep learning based on ResNet-18 for classification of prostate imaging-reporting and data

system category 3 lesions[J]. Academic Radiology, 2024, 31(6): 2412−2423

[49] Heitmann K, Uram T D, Finkel H, et al. HACC cosmological simulations: First data release[J]. The Astrophysical Journal Supplement Series, 2019, 244(1): 17

[50] Liu Jiarun, Yang Hao, Zhou Hongyu, et al. Swin-umamba: Mamba-based unet with ImageNet-based pretraining[C]// Proc of the 27th Int Conf on Medical Image Computing and Computer-Assisted Intervention. Berlin: Springer, 2024: 615−625



Li Dongyang, born in 2001. Master candidate. His main research interest includes hardware-software co-design based on deep learning.

李东阳, 2001 年生。硕士研究生。主要研究方向为基于深度学习的软硬件协同设计。



Li Zeyang, born in 2005. Undergraduate. Her main research interest includes hardware-software co-design based on deep learning.

李泽洋, 2005 年生。本科生。主要研究方向为基于深度学习的软硬件协同设计。



Tang Yibin, born in 1991. PhD. His main research interests include the hardware-software co-design for deep learning, and computer architecture.

唐忆滨, 1991 年生。博士。主要研究方向为面向深度学习的软硬件协同设计、计算机系统结构。



Liu Bosheng, born in 1988. PhD, associate professor. His main research interests include hardware-software co-optimization for LLMs, and computer architecture. (liubosheng@gdut.edu.cn)

刘博生, 1988 年生。博士，副教授。主要研究方向为大语言模型软硬件协同优化、计算机体系结构。



Wu Jigang, born in 1963. PhD, PhD supervisor. His main research interests include AI, high-performance computing, and data science.

武继刚, 1963 年生。博士，博士生导师。主要研究方向为人工智能、高性能计算、数据科学。