IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 2 September 2026, accepted 26 September 2026, date of publication 30 September 2026,
date of current version 7 October 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3738900

RESEARCH ARTICLE

MBA: Efficient Hardware Accelerator for MXFP4 Quantization via Learnable Block-Wise Affine Transformation

SANGKI PARK, (Graduate Student Member, IEEE), CHANHOON KIM, (Graduate Student Member, IEEE), SOOMIN RHO, (Graduate Student Member, IEEE), MINYOUNG LEE, (Graduate Student Member, IEEE), AND KI-SEOK CHUNG, (Member, IEEE)
Department of Electronic Engineering, Hanyang University, Seoul 04763, South Korea

Corresponding author: Ki-Seok Chung (kchung@hanyang.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by Korean Government [Ministry of Science and Information and Communication Technology (MSIT)] under Grant RS-2024-00409492.

ABSTRACT Large language models (LLMs) have achieved remarkable performance across a wide range of natural language processing tasks; however, their rapidly growing model sizes pose critical challenges for energy-efficient inference, increasing the demand for low-bit quantization. Although transformation-based methods have shown superior performance in conventional low-bit formats, their efficacy degrades substantially under the MXFP4 format. This degradation arises because these transformations spread outliers across MX block boundaries, inadvertently introducing new outliers within individual blocks. Furthermore, applying these transformations to activations at runtime requires implementing high-precision arithmetic units that incur significant hardware design overhead in terms of area and power. To address these challenges, we propose MBA, a software-hardware co-design framework that enables hardware-efficient and accurate affine transformation-based MXFP4 quantization. MBA learns affine transformation matrices constrained to a block-diagonal structure aligned with the MX block size, and reduces both the number of parameters and computational complexity while maintaining high accuracy. We also implement a dedicated hardware accelerator featuring dual-mode processing elements (PEs) that efficiently support both BF16 and MXFP4 computations. Hardware evaluations on linear-layer matrix multiplication, including the online affine transformation, demonstrate that MBA achieves 2.66× higher TOPS/W and 1.9× higher TOPS/mm^2 than the state-of-the-art MX accelerator. Additionally, in the LLaMA3-8B model, our method achieved an average of 65.69% on reasoning tasks, outperforming LATMiX by 2.72%p and demonstrating better performance compared to existing methods across various LLaMA models.

INDEX TERMS Low-bit quantization, LLM quantization, rotation quantization, microscaling data formats, hardware acceleration.

I. INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language processing tasks, yet their deployment remains challenging due to prohibitive computational and memory demands during inference. Model quantization has emerged as one of the most effective techniques to address this bottleneck, enabling significant reductions in memory footprint and arithmetic complexity.

Recently, the Microscaling (MX) family of data formats [1] was introduced as an open, interoperable standard for low-bit precision, driven by a collaborative effort among major technology companies, including AMD, Arm, Intel, Meta, Microsoft, NVIDIA, and Qualcomm. MX data formats employ block-level shared scaling factors to enable

The associate editor coordinating the review of this manuscript and approving it for publication was Maurizio Casoni.

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

fine-grained quantization. Among the MX formats, MXFP4 has been widely adopted by industry and is natively supported by the latest generation of commercial accelerators, including NVIDIA's Blackwell GPU [2], AMD's MI350x/MI355x [3], and Microsoft's Maia 100 [4]. By exploiting shared exponents and streamlined dequantization, these formats deliver high throughput with minimal hardware overhead. However, the limited precision of the 4-bit format and the coarse granularity of power-of-two (E8M0) scaling render models highly sensitive to outliers, often leading to severe accuracy degradation that prevents practical deployment.

To mitigate the impact of outliers, prior works [5], [6], [7] proposed transformation-based pre-processing methods. QuaRot [5] and SpinQuant [6] apply orthogonal rotation matrices to redistribute outliers across channels before quantization. FlatQuant [7] replaces the orthogonality constraint with learnable invertible affine transformations, relaxing the energy-conservation limitation of rotation-based methods and achieving improved accuracy. However, while effective for INT4 formats, these methods are fundamentally misaligned with MXFP4's block-wise structure because they tend to spread outliers across MX block boundaries, inadvertently introducing new outliers within individual blocks. BRQ [8] proposed a block-rotation method as a structure-aware alternative, yet it inherently induces a bimodal distribution within each block, which significantly degrades performance.

From a hardware perspective, transformations tailored to each layer require the online transformation of every input activation. This necessitates a BFloat16 (BF16) matrix multiplication (MM) between the unquantized activation and the transformation matrix, imposing a dual-precision requirement alongside the MXFP4 MM. A naive implementation requires two separate multiply-accumulate (MAC) units, incurring significant area and power overhead.

In summary, existing methods fail to satisfy at least one of the three requirements for MXFP4 quantization: (1) The transformation must respect MX block boundaries to avoid inter-block energy transfer of outliers, ensuring that each block's scaling factor accurately captures its local dynamic range. (2) The transformation must be adaptive to the data distribution to be MXFP4-friendly. (3) Hardware for transformation-based quantization must support both high-precision MM and quantized MM with minimal area and power overhead.

In this paper, we propose MBA, a software-hardware co-design framework that effectively addresses these limitations. In MBA, we introduce MX-aware affine training, a novel training method for block-wise affine transformation aligned with the MX block size that adapts the distribution to be MXFP4-friendly while avoiding bimodal artifacts induced by Hadamard rotations. MBA uses a block-diagonal matrix to ensure that each MX block is transformed independently without crossing the block boundaries, thereby reducing both the parameter count and computational complexity from $O(N^2)$ to $O(Ng)$, where N is the hidden dimension and g is the block size, making the transformation both parameter-efficient and hardware-amenable. We also implement a dedicated hardware accelerator, featuring a dual-mode processing element (PE) inspired by BitFusion [9] and Jack Unit [10] that reconfigures the same hardware resources for both BF16 affine transformation and MXFP4 MM, minimizing area and power overhead.

The main contributions of this paper are as follows:

- We analyze the structural incompatibility with MXFP4's block-wise format in the existing transformation-based methods and theoretically demonstrate their accuracy degradation.
- We propose a learnable block-wise affine transformation aligned with MX block size, which improves MXFP4 quantization accuracy while reducing parameter and computational complexity from $O(N^2)$ to $O(Ng)$.
- We design a dedicated hardware accelerator that natively supports the proposed transformation through a dual-mode PE with negligible area and power overhead.

Experimental results demonstrate that MBA achieves superior accuracy under W4A4KV4 MXFP4 quantization, improving zero-shot task accuracy by 2.72%p over LATMiX, the previous MXFP4-targeted method, in the LLaMA3-8B model, and achieving the lowest WikiText-2 perplexity among all existing methods across all LLaMA models. Furthermore, for the evaluated linear-layer MM workload including the online affine transformation, our hardware accelerator outperforms state-of-the-art MX accelerators, achieving up to 1.21× speedup, 1.4× energy reduction, 2.66× higher TOPS/W, and 1.9× TOPS/mm^2 improvement, with negligible area and power overhead.

II. BACKGROUND

A. TRANSFORMATION-BASED LLM QUANTIZATION

Quantization reduces the bit-width of weights and activations from full-precision floating-point representations (e.g., FP32 or BF16) to lower-precision formats (e.g., INT8, FP8, FP4), yielding significant reductions in memory footprint, required memory bandwidth, and computational cost. Formally, given a tensor $\mathbf{X} \in \mathbb{R}^n$, each element is quantized as:

$$\tilde{x}_i = \text{round}\left(\frac{x_i}{s}\right), \quad s = \frac{\max(|\mathbf{X}|)}{Q_{\max}}, \tag{1}$$

where s is the scale factor and $Q_{\max}$ is the maximum representable value of the target format.

A key challenge in LLM quantization is the presence of activation outliers. While weights exhibit relatively smooth distributions amenable to low-bit representation, activations are generated at runtime and often contain large outliers that cause significant quantization error in low-bit formats.

Recent works employ rotation methods using orthogonal matrices to mitigate outliers and perform quantization. QuaRot [5] and SpinQuant [6] apply orthogonal rotation matrices to redistribute outlier energy uniformly across channels using the *computational invariance* [11] property. Furthermore, because $\mathbf{R}\mathbf{R}^\top = \mathbf{I}$, the rotation can be absorbed

into adjacent weight matrices offline, reducing the number of transformations applied at runtime. FlatQuant [7] and LATMiX [12] further relax the orthogonality constraint by using learnable invertible affine transformations.

B. MICROSCALING (MX) DATA FORMATS

Microscaling (MX) formats, standardized under the Open Compute Project (OCP) MX specification [1], define a family of low-precision formats that achieve a favorable balance between quantization accuracy and hardware efficiency. The central idea is *block-level shared scaling*, where a contiguous group of 32 elements shares a common scale factor S encoded as an 8-bit E8M0, and each element is represented individually in a low-precision format. This can be computed as follows:

$$\text{shared_exp} = \max\left(\lfloor \log_2(|x_i|) \rfloor\right) - e_{\max}, \quad S = 2^{\text{shared_exp}}, \tag{2}$$

where $e_{\max}$ is the maximum representable exponent value of the element format. Among element formats defined by the OCP MX specification, MXFP4 offers the most aggressive compression using only 4 bits per element. The block-shared scaling structure enables hardware-efficient dequantization by applying a single scale to all elements within a block, reducing element-wise dequantization to a simple exponent adjustment rather than a per-element FP multiplication. However, this method also introduces a fundamental limitation. The E8M0 shared scale is determined solely by the block maximum, so a single outlier dominates the scale assignment and introduces relatively large quantization error for the other elements with a small magnitude. Existing transformation-based methods attempt to mitigate this by reducing outlier magnitudes, but they do not consider MX's block granularity. Therefore, the redistributed energy inflates small values within the same block, inducing new outliers.

III. MOTIVATION AND CHALLENGES

Although existing methods [5], [6], [7] achieve strong performance by dispersing outliers through transformation, their performance degrades substantially when directly applied to MXFP4, as summarized in Fig. 1. We identify four quantization challenges and one hardware challenge that motivate our approach.

MXFP4 adopts a per-block quantization granularity and a block-wise shared exponent that is determined by the maximum absolute value within each block. Globally applied transformations are fundamentally misaligned with this block-wise structure. They suppress outliers, but simultaneously amplify previously small values within blocks, inducing a large block-wise maximum and shared exponent (**Challenge #1**). Since MXFP4 adopts an E8M0 shared scale determined by the block maximum, this increases the rounding error of the maximum values due to the coarse granularity of E8M0 and incurs significant quantization error across elements. This suggests that transformation must be applied at the MX block level to prevent inter-block outlier energy transfer and ensure that each block's maximum remains small.

Although LATMiX [12] also initializes its transformation matrix in a block-diagonal form, it does not enforce this structure as a hard constraint during training, so the learned matrix can still evolve into a dense form that redistributes energy across block boundaries and fails to escape Challenge #1. BRQ [8] structurally preserves the block boundary through a fixed Hadamard rotation, but its orthogonal sign-flipping cannot suppress the block-wise maximum, leaving the same challenge unresolved through a different mechanism. Both limitations persist empirically, as shown in Fig. 9. LATMiX's massive outliers remain comparable in magnitude to the untransformed baseline, while BRQ's fixed rotation suppresses the outlier inconsistently, leaving a large residual outlier; MBA instead suppresses the outliers substantially and consistently. FlatQuant, despite a lower maximum value than MBA, exhibits numerous scattered outliers across many blocks; both therefore retain a large overall deviation even though their single largest value is controlled. MBA is the only method that suppresses both the block-wise maximum and the overall spread simultaneously, remaining close to zero across nearly all blocks and tokens.

Besides outlier suppression, the block distribution critically affects MXFP4 quantization accuracy (**Challenges #2 & #3**). We further mathematically analyze the MSE of each distribution, as discussed in Sec. IV: the MSE upper bound under a Laplace distribution is higher than that under a Gaussian distribution by $O(\log 2g)$, indicating that a Gaussian distribution is more suitable for MXFP4 quantization. This distribution mismatch is particularly pronounced when transformation is applied at the block level, a design choice motivated by Challenge #1 to prevent inter-block outlier energy transfer. Specifically, a naive block-wise Hadamard rotation introduces a new problem. Its sign-flipping nature concentrates values at two extremes, producing a bimodal distribution within each block. The two extreme peaks span a large range, wasting the representable FP4 values. This results in excessive variance and large quantization error. These observations indicate that effective MXFP4 quantization requires not only block-level outlier suppression, but also explicit optimization of the intra-block distribution. These fundamental incompatibilities with MXFP4 directly manifest as quantization error in practice. In the W4A4KV4 quantization setting, rotation-based methods, SpinQuant and BRQ, yield average MSEs of $9.41e^{-3}$ and $1.76e^{-3}$, respectively, confirming that global rotations fail to address the distribution mismatch inherent to MXFP4. Furthermore, the affine-based FlatQuant has an average MSE of $2.50e^{-3}$, it remains suboptimal because its transformations are not explicitly optimized for MXFP4's block-wise structure, and the MXFP4-targeted method LATMiX also shows a high average MSE of $9.42e^{-3}$. (**Challenge #4**). In contrast, MBA achieves $2.78e^{-5}$, empirically validating that jointly addressing outlier suppression and distribution shaping is essential for MXFP4.

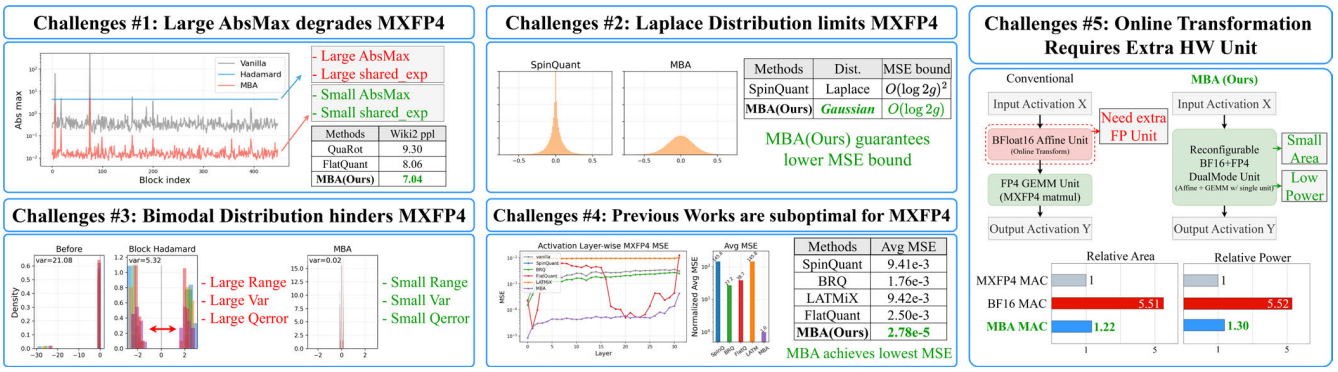


FIGURE 1. Limitations of previous works and challenges in MXFP4 quantization and HW accelerator design.

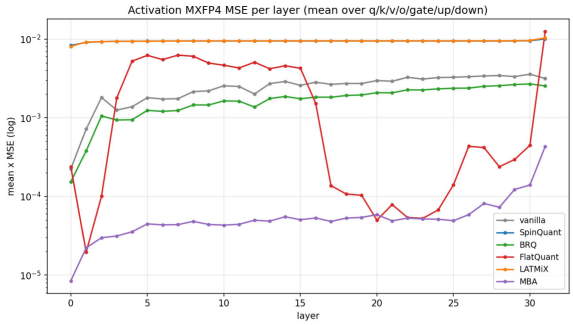


FIGURE 2. Layer-wise quantization error on LLaMA3-8B for various methods.

This distribution-shape argument is confirmed directly by the block-wise statistics in Fig. 10 and Fig. 11. At the target MX block size $g = 32$, the baseline and the globally-mixing methods (SpinQuant, LATMiX) still retain a substantial population of blocks with strongly super-Gaussian kurtosis in Fig. 10, whereas MBA's block-wise kurtosis concentrates tightly around the Gaussian reference value of 3. Fig. 11 shows a complementary pattern for the block-wise mean. At $g = 32$, FlatQuant and MBA achieve the narrowest interquartile range among all compared methods. However, FlatQuant retains a considerably larger population of outlier blocks with substantial nonzero mean, whereas MBA's outlier blocks are both fewer and smaller in magnitude. MBA therefore attains the most consistent zero-mean, near-Gaussian block statistics at the granularity over which MXFP4 actually quantizes.

In addition to these algorithmic challenges, efficiently implementing transformation-based quantization in hardware introduces a critical challenge (**Challenge #5**). Prior works [5], [6] necessitate the design of a rotation unit for FP16 addition and subtraction, while FlatQuant [7] requires an FP16 MM unit for the online transformation. In either case, supporting online transformation alongside MXFP4 MM inevitably imposes a *dual-precision requirement*: BF16 for the affine transformation and MXFP4 for the quantized MM. A naive implementation would require two separate dedicated MAC units, where the additional BF16 MAC unit incurs substantial area and power overhead, **5.51**× and **5.52**× relative to an MXFP4 MAC unit in area and power, respectively. To overcome this challenge, we propose a dual-mode PE design that supports both precisions within a single MAC unit to reduce the hardware cost of dedicated logic through hardware reconfiguration.

IV. METHODS

To overcome the limitations of transformation-based approaches discussed in Sec. III, we propose MBA, an MXFP4 quantization method via learnable block-wise affine transformation. MBA improves performance by learning block-wise affine transformations optimized for MXFP4. Specifically, we introduce an affine transformation matrix with a block-diagonal structure, which prevents outliers from crossing MX block boundaries while enabling block-level optimized transformations and reducing both computational complexity and the number of trainable parameters during affine transformation. In addition, MBA introduces a new loss function that incorporates both the MX quantization error and a moment-regularization term for enforcing block-wise distributional properties, thereby supporting stable training of quantized models. Furthermore, we implement a hardware accelerator that efficiently supports MBA's affine transformation and minimizes the additional hardware overhead incurred when applying the transformation.

A. THEORETICAL ANALYSIS FOR ERROR BOUND

We mathematically analyze the MSE of each distribution to support the claims made in Sec. III. Let us define an arbitrary transformation matrix $R \in \mathbb{R}^{N\times N}$, where R is invertible. Let $Q(\cdot)$ denote the MX quantization function. Then, for an arbitrary vector $\boldsymbol{x} \in \mathbb{R}^N$, the upper bound of the MX quantization error $Err_{mx}(R, \boldsymbol{x})$ induced by the transformation is derived as follows [12]:

$$Err_{mx}(R,\boldsymbol{x}) \lesssim \frac{||R^{-1}||_2}{G} \sum_{i=1}^{G} \mathbb{E}\left[\left(\max_{j\in\theta_i} |R\boldsymbol{x}_j|\right)^2\right], \quad (3)$$

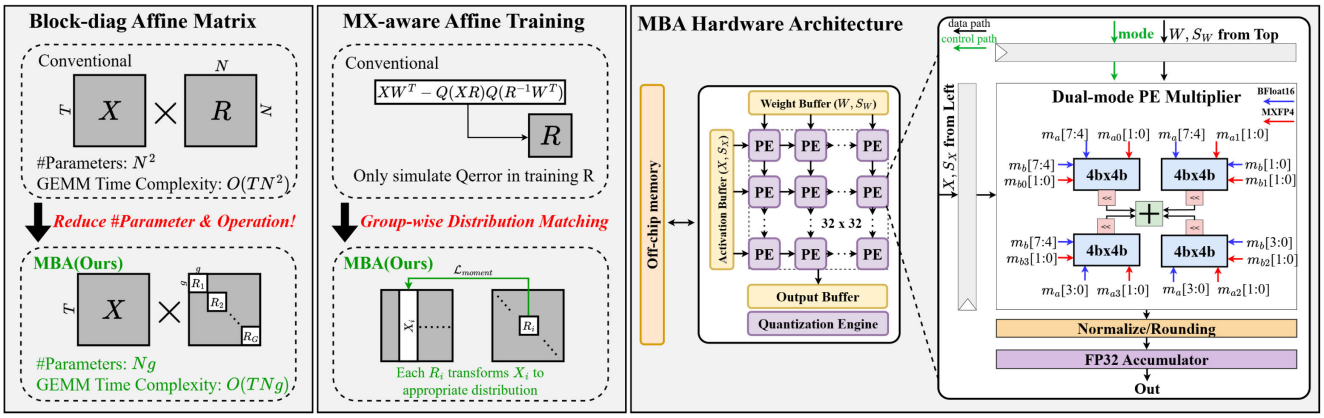


FIGURE 3. Overview of the major contributions of the MBA algorithm and its hardware implementation.

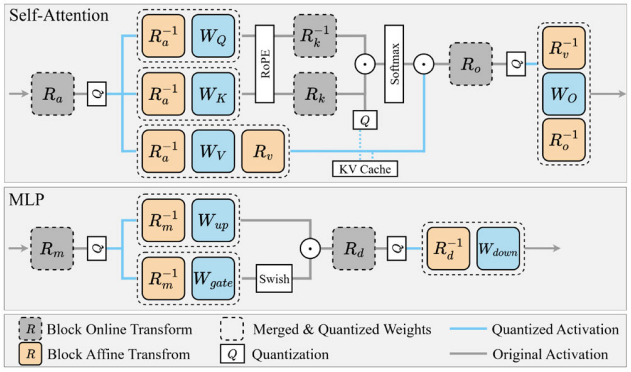


FIGURE 4. Overview of the affine transformation used in MBA.

where $G = N/g$ denotes the number of MX groups, $||R^{-1}||_2$ is the spectral norm of R^{-1}, $j \in \theta_i$ denotes an index belonging to the i-th MX group θ_i, and $\boldsymbol{Rx}_i$ denotes the collection of channel vectors corresponding to the i-th MX block. Since MBA does not employ a bias term, the transformation is expressed simply as $\boldsymbol{Rx}$. Equation (3) indicates that the quantization error of each MX block is upper-bounded by the product of the spectral norm of R^{-1} and the squared maximum value within the block, $M_i \triangleq \mathbb{E}\left[\left(\max_{j\in\theta_i} |\boldsymbol{Rx}_j|\right)^2\right]$. Therefore, block-level optimization is essential in MX quantization. However, as pointed out by [12], block-wise optimization alone may increase the overall quantization error. Because the upper bound of the MX quantization error depends on the distribution, we address this issue in Sec. IV-C by proposing a method that guides transformed activations toward a target distribution.

Since the theorem in [12] covers both the sub-Gaussian and sub-exponential cases, the corresponding error bound for each distribution can be derived from it. When x is a ψ_α random vector with mean μ, M_i is as follows:

$$M_i \lesssim (A + h_\alpha(g)K)^2,$$
$$\text{where } h_\alpha = \left(\frac{4g}{\alpha}\cdot\Gamma(2/\alpha, \log(2g)) + (\log(2g))^{2/\alpha}\right)^{1/2},$$

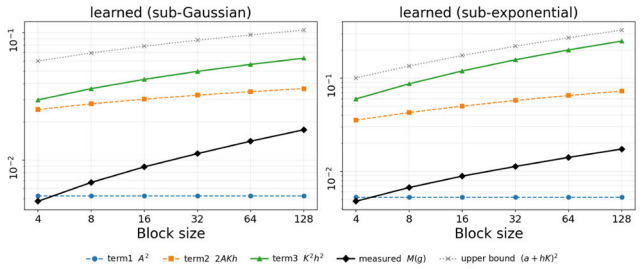


FIGURE 5. Average M_i and bound analysis on the LLaMA3.2-1B model.

$$A = \max_{j\in\theta_i} |[R\boldsymbol{\mu}]_j|,$$
$$K = \max_{j\in\theta_i} ||[R\boldsymbol{x} - R\boldsymbol{\mu}]_j||_{\psi_\alpha}, \tag{4}$$

where $||\cdot||_{\psi_\alpha}$ denotes the Orlicz norm of a random variable, and $\Gamma(s,x) = \int_x^\infty t^{s-1}e^{-t}dt$ is the upper incomplete gamma function. Here, $\alpha = 1$ corresponds to the sub-exponential case and $\alpha = 2$ to the sub-Gaussian case, and the value of h_α for each α is given as follows:

$$h_\alpha = \begin{cases} \left((\log 2g)^2 + 2\log(2g) + 2\right)^{1/2} & \text{if } \alpha = 1 \\ \left(1 + \log(2g)\right)^{1/2} & \text{if } \alpha = 2. \end{cases} \tag{5}$$

Since MBA does not employ a bias term, $A = 0$ cannot be guaranteed. To determine the actual bound on M_i, we therefore conducted an experiment on the LLaMA3.2-1B model comparing the empirical value of M_i with the value of each term in the expansion of $(A + h_\alpha(g)K)^2$. As shown in Fig. 5, A in MBA remains roughly constant regardless of the value of g, whereas the ratio $(h_\alpha(g)K)^2/A^2$ increases monotonically with block size, from 11.3× to 47.4× for the sub-exponential case and from 5.6× to 12× for the sub-Gaussian case, indicating that M_i closely follows the growth rate of $(h_\alpha(g)K)^2$. Therefore, the upper bound of M_i for distributions following sub-exponential and sub-Gaussian tails can be summarized as follows:

$$M_i \lesssim \begin{cases} O((\log 2g)^2) & \text{if } \alpha = 1 \\ O(\log 2g) & \text{if } \alpha = 2. \end{cases} \tag{6}$$

Laplace-distributed and Gaussian-distributioned random variables are sub-exponential and sub-Gaussian random variables, respectively. For a Laplace-distributed variable with scale parameter b, the sub-exponential norm satisfies $K = \Theta(b)$; for a Gaussian-distributed variable with standard deviation σ, the sub-Gaussian norm satisfies $K = \Theta(\sigma)$. Here, b and σ are treated as constants with respect to the block size g. We assume that the transformation matrix R is optimized under comparable conditions in both cases, so that $||R^{-1}||_2$ remains of a similar order. Furthermore, assuming the post-transformation variances are approximately equal in the two cases, b and σ are of the same order, since the variances of the Laplace and Gaussian distributions are $2b^2$ and σ^2, respectively. Consequently, the MSE upper bound under a Laplace distribution is higher than that under a Gaussian distribution by $O(\log 2g)$. Therefore, under the idealized assumptions of this analysis, the Gaussian reference case provides a more favorable error-bound scaling than the Laplace reference case. This result motivates reducing excessive tail dominance within MX blocks; it does not require the post-transformation activations to exactly follow a Gaussian distribution. However, since sub-Gaussian and sub-exponential random variables are closed under linear transformation, an additional regularization term is required to induce a favorable distribution. To this end, this paper introduces a moment-matching loss that guides each MX block toward a distribution suitable for quantization; as Fig. 1 shows, MBA achieves the lowest MSE.

B. MX-AWARE AFFINE TRANSFORMATION

Given activation X and weight W, the linear projection in a typical LLM can be expressed as $Y = XW^T$. If X and W are quantized directly, this becomes $\tilde{Y} = Q(X)Q(W^T)$, or by introducing a transformation, $\tilde{Y} = Q(XR)Q(R^{-1}W^T)$. Therefore, it is essential to find R that minimizes the quantization error in (3). However, when existing methods [5], [6] are applied directly, the MXFP4 quantization error becomes significantly larger, as shown in Fig. 1. In MX, each block needs to maintain a consistent distribution. Although conventional full-matrix learning is effective at globally reducing outliers, it does not consider the block-wise characteristics required by MX. Accordingly, we construct the affine matrix $R \in \mathbb{R}^{N\times N}$ as a block-diagonal matrix:

$$R = \mathrm{diag}(R_1, R_2, \ldots, R_G), R_i \in \mathbb{R}^{g\times g}, \tag{7}$$

where g is the block size, and $G = N/g$ is the number of blocks.

Because each block component is learned independently in this block-diagonal form, optimized transformations can be applied to each corresponding activation or weight group. Moreover, due to the block-diagonal structure, the computational complexity of the affine transformation is reduced from $O(N^2)$ to $O(Ng)$, and the number of trainable parameters is likewise reduced from N^2 to Ng.

This design departs from LATMiX [12] in two respects. First, LATMiX optimizes a full $N \times N$ matrix, using the block-diagonal form only as an initialization, and its training objective minimizes $|\det(A)|$ as a proxy for reducing quantization error; this does not directly constrain the distribution-dependent term M_i that governs the error bound in (3) and (4). In contrast, MBA enforces R to remain block-diagonal throughout training and, as described in Sec. IV-C, directly regularizes the moments of each block's distribution toward the target identified by our error-bound analysis. Second, LATMiX defines transformations only for the query, key, and value projections and does not extend to the KV cache, leaving its quantization error unmitigated; MBA additionally introduces R_k to support KV cache quantization under the W4A4KV4 setting, as described below.

Fig. 4 illustrates the overall structure of the affine transformation used in the MBA algorithm. Following a similar design to prior transformation-based approaches [7], the affine transformations are applied to both the self-attention and MLP modules, consisting of the sets $\{R_a, R_v, R_o, R_k\}$ and $\{R_m, R_d\}$, respectively. R_a is applied to the input of the self-attention module, R_v is applied to the value projection, and R_k is applied to the output of the key projection for use in KV cache quantization. R_o is applied to the output projection. In the MLP module, R_m is applied to the input activation, while R_d is applied to the input of the down projection. All the inverse affine matrices except R_k are absorbed into the weights, while the forward transformation on the activations is computed at runtime.

C. OBJECTIVE FUNCTION FOR STABLE TRAINING

Each block component R_i in the block-diagonal matrix should be learned to minimize the MX quantization error. At the same time, it is crucial to preserve a distribution that is favorable for MX quantization. Accordingly, the loss function should include both an MX quantization-error term and a regularization term that induces the desired distribution. To enable stable training of the MBA method, we propose a moment-matching loss $\mathcal{L}_{moment}$ that includes regularization terms on the third and fourth moments of each activation-channel group corresponding to an MX block. Therefore, the final loss function, which jointly considers MX quantization error and moment-matching loss, can be written as follows:

$$\mathcal{L}_{moment} = \frac{1}{G}\sum_{i=1}^{G}\left[(\gamma_i - \gamma^*)^2 + (\kappa_i - \kappa^*)^2\right], \tag{8}$$

$$\mathcal{L}_{final} = \mathcal{L}_{MSE}(\boldsymbol{y}, \tilde{\boldsymbol{y}}) + \lambda\mathcal{L}_{moment}, \tag{9}$$

where λ denotes the weighting coefficient, $\boldsymbol{y}$ is the output of the non-quantized model, and $\tilde{\boldsymbol{y}}$ is the output of the model quantized with MX. γ_i and κ_i denote the third and fourth moments (skewness and kurtosis) of the i-th block, respectively, while γ^* and κ^* denote their target values. As shown in Fig. 10 and Fig. 11, MBA attains stable kurtosis

and mean values at $g = 32$, closely converging to the Gaussian distribution.

Although the error-bound analysis above indicates that a Gaussian target minimizes the theoretical MSE bound, moment-matching regularizes only the kurtosis and skewness rather than the full distribution shape, and hence does not impose a strict Gaussian constraint on each block. In practice, we find that setting κ^* to the exact Gaussian kurtosis, $\kappa^* = 3$, under-constrains training and yields a block distribution that is less favorable for MXFP4, whereas setting $\kappa^* = 6$, more effectively suppresses the heavy-tailed outliers that arise during training. We therefore adopt $\gamma^* = 0$ and $\kappa^* = 6$ as the target moments throughout this work. Details about ablation study of kurtosis target is in Section VI-D2.

In addition to the proposed block-wise affine transformation, we investigate per-channel scaling [13], learnable weight clipping (LWC), and learnable activation clipping (LAC) [7] as optional components for stabilizing the calibration optimization and mitigating quantization outliers.

Unlike previous work [7], however, MBA applies clipping thresholds at the block level to ensure that the quantization process within each block remains stable. Accordingly, for $\boldsymbol{x}$ belonging to an arbitrary block, each LWC α_w and LAC α_a is applied as follows: $\hat{\boldsymbol{x}} = \text{clip}(\boldsymbol{x}, \sigma(\alpha^{min}), \sigma(\alpha^{max}))$, where σ is the sigmoid function, and α_a^{max} and α_a^{min} denote the maximum and minimum clipping thresholds for the block, respectively. Our ablation study in Section VI-D shows that the combination of per-channel scaling and LWC yields the best overall accuracy, whereas enabling LAC does not provide additional improvement. Therefore, unless otherwise specified, MBA denotes the configuration with per-channel scaling and LWC, without LAC.

V. ARCHITECTURE

Fig. 3 illustrates the top-level architecture of our MBA accelerator. The accelerator includes standard components such as vector processing units, on-chip buffers, and a controller unit. The MBA hardware accelerator introduces two components to support the proposed method: (1) **Dual-mode PE** that supports both BF16 and MXFP4 computation via hardware reconfiguration, and (2) **Quantization Engine** that performs online MXFP4 quantization of activations after the affine transformation and the MXFP4 MM.

A. PROCESSING ELEMENT

Each PE is designed to operate in two modes: **BF16** mode for the affine transformation and **MXFP4** mode for quantized MM. The key insight is that an 8-bit mantissa multiplication in BF16 decomposes into four independent 4b×4b partial products; the multipliers are reconfigured to process four parallel FP4 mantissa multiplications in the MXFP4 mode, yielding a 4× throughput improvement over the BF16 mode.

1) BF16 MAC

As shown by the datapath marked in blue in Fig. 3, the 8-bit mantissas $m_a[7:0]$ and $m_b[7:0]$ are split into the most significant 4 bits and the least significant 4 bits, yielding four partial products: $(m_a[7:4] \times m_b[7:4])$, $(m_a[7:4] \times m_b[3:0])$, $(m_a[3:0] \times m_b[7:4])$, and $(m_a[3:0] \times m_b[3:0])$. Each partial product is computed by a dedicated 4b×4b multiplier and subsequently left-shifted by 8, 4, 4, and 0 bits, respectively, to reflect their positional weights. The shifted partial products are then accumulated by an adder tree to produce the final 16-bit mantissa result. The output of the adder tree is sent to a normalization and rounding unit, and the normalized mantissa is combined with the exponent and the sign.

2) FP4 MAC

As shown by the datapath marked in red in Fig. 3, in MXFP4 mode, the four 4b×4b multipliers are reconfigured to process four independent FP4 operand pairs concurrently. Each unit receives a 2-bit mantissa pair ($m_{a,i}[1{:}0]$, $m_{b,i}[1{:}0]$), which is zero-padded at the MSB side to form a 4-bit input $(00, m[1{:}0])$. Each mantissa product is then left-shifted by the corresponding exponent sum $e_{a,i} + e_{b,i}$, encoding the magnitude of each product in a common positional representation. The four shifted products are accumulated by the adder tree. Since only four products are accumulated, a 16-bit integer width is sufficient to prevent overflow, avoiding the need for wider accumulators. This enables the PE to execute four FP4 multiply-accumulate operations per cycle, achieving 4× the arithmetic throughput of the BF16 mode per PE.

3) DEQUANTIZATION AND ACCUMULATION

In the BF16 mode, no dequantization step is required as the normalization and rounding stage directly produces the FP32 result. For the MXFP4 mode, the adder tree output is passed to the normalization and rounding unit, where dequantization is performed by applying the shared block scale. Because MX formats employ an E8M0 shared scale, this dequantization is efficiently implemented as a simple exponent adjustment, eliminating the need for costly FP multiplications. In both modes, the output is forwarded to a shared FP32 accumulator within each PE, which locally accumulates partial sums, and the final result is converted to BF16 and written into the output buffer. This unified accumulation path eliminates the need for separate accumulation hardware per mode, reducing area overhead while maintaining full-precision accumulation fidelity.

B. QUANTIZATION ENGINE

A key challenge in deploying MX-format quantization within an inference accelerator is that the quantization of activations must be carried out online, immediately after the preceding layer's computation, before the results are consumed by the next MM operation. To address this, we integrate a dedicated quantization engine that is fused with the output path of the systolic array. Specifically, once the systolic array completes a MM tile, the BF16 activation results are written to the output buffer. The quantization engine continuously reads

from the output buffer and performs MXFP4 quantization in a streaming fashion, converting the BF16 activations into the MXFP4 format.

VI. EVALUATION

A. EXPERIMENTAL SETUP

1) MODELS AND DATASETS

To validate the MBA algorithm, we evaluated the LLaMA2 [14] and LLaMA3 [15] model families. We use 128 random samples from WikiText-2 [16] for calibration data. For zero-shot commonsense reasoning tasks, we used PIQA [17], HellaSwag [18], WinoGrande [19], ARC [20], OpenBookQA [21], BoolQ [22], SocialIQA [23], and LAMBADA [24].

2) ALGORITHM IMPLEMENTATION

We implemented the MBA framework using PyTorch and the HuggingFace Transformers library. All experiments were conducted on a single NVIDIA RTX 5090 GPU. MBA is trained for 15 epochs on Wikitext-2 with a learning rate of $1e^{-3}$, $\lambda = 0.05$, $\gamma^*a = 0$, and $\kappa^*a = 6$; the total calibration time was approximately 53 minutes for LLaMA3-8B and approximately 21 minutes for the smallest model, LLaMA3.2-1B. We use $ga = 32$ for all MX quantization. For all models, the reported MBA results, including hardware results, use per-channel scaling and learnable weight clipping (LWC) as the selected optional components. To benchmark MBA, we compared it against state-of-the-art quantization baselines, QuaRot [5], SpinQuant [6], FlatQuant [7], BRQ [8], and LATMiX [12], under the same MXFP4 setting. QuaRot, SpinQuant, and FlatQuant were retrained from scratch under MXFP4 using their official implementations, while LATMiX and BRQ were reproduced following the descriptions in their respective papers. All experiments were conducted under the W4A4KV4 configuration with GPTQ [25] for weight quantization.

3) HARDWARE IMPLEMENTATION

We implemented MBA's dual-mode PE and the integrated quantization engine in Verilog RTL, and synthesized them using Synopsys Design Compiler with the FD-SOI 28 nm standard cell library at 1 GHz. An on-chip buffer was also synthesized using the same CMOS process, with area and power consumption estimated using Synopsys Memory Compiler and PrimePower, respectively. We extended the open-source cycle-level simulator DNNWeaver [26], integrated with CACTI [27], to model our accelerator. Following the evaluation methodology of prior quantization accelerator studies built on the same simulator, including ANT [28], OliVe [29], and AxCore [30], our simulator measures the latency and energy of the linear layer MM, including the online affine transformation, and does not include the attention computation. All evaluations are conducted under a batch size of 32 and a sequence length of 2048, using an output-stationary dataflow. In addition, all designs share identical on-chip SRAM size and DRAM interface.

We evaluate the proposed MBA accelerator against representative baselines. For perplexity comparison, we include all five baselines: M^{2}XFP [31], BlockDialect [32], MX+ [33], MX-LightRot, and MX-GyRot. For hardware evaluation, we compare against M^{2}XFP, MX-LightRot, and MX-GyRot, which have dedicated accelerator designs amenable to direct hardware comparison. MX-LightRot and MX-GyRot are adapted from LightRot [34] and GyRot [35], originally designed for non-MX formats, to operate under MXFP4 quantization. We also denote *Effective Bitwidth (Eff. bit)* [32], the average bit width per data accounting for the scaling factor and metadata overhead, to provide a more comprehensive comparison of the quantization efficiency of different methods.

TABLE 1. WikiText-2 perplexity under W4A4KV4 MXFP4 quantization on various LLaMA models (lower is better).

Method	Eff. bit	2-7B	3-8B	3.2-3B	3.2-1B
FP16	16	5.47	6.14	7.81	9.75
MXFP4	4.25	6.65	8.68	11.99	23.33
QuaRot	4.25	7.70	9.30	15.83	37.54
SpinQuant	4.25	6.52	8.06	10.32	14.79
FlatQuant	4.25	5.97	7.24	9.18	13.30
BRQ	4.25	6.05	7.61	10.00	13.82
LATMiX	4.25	7.43	7.78	9.16	12.86
MX-LightRot	4.25	6.64	8.08	10.02	15.31
MX-GyRot	4.25	6.75	8.12	10.11	15.29
BlockDialect	4.28	6.57	8.48	11.68	22.15
MX+	4.50	6.00	7.47	9.63	14.24
M^{2}XFP	4.50	5.93	7.14	9.04	12.96
MBA (Ours)	4.25	**5.90**	**7.04**	**8.84**	**12.58**

B. LANGUAGE MODEL PERFORMANCE

1) COMPARISON WITH BASELINE METHODS

To evaluate the performance of the MBA algorithm, we measured perplexity and accuracy on WikiText-2 and nine zero-shot commonsense reasoning benchmarks, and compared MBA against state-of-the-art transformation-based quantization methods, namely SpinQuant [6], QuaRot [5], FlatQuant [7], BRQ [8], and LATMiX [12].

Table 1 summarizes the perplexity of each method on WikiText-2, and MBA achieves the lowest perplexity across all models. In particular, the conventional rotation-based methods, SpinQuant and QuaRot, exhibit performance degradation across all models, indicating that existing approaches are not suited to MXFP4 quantization. Notably, MBA achieves lower perplexity than FlatQuant, the prior state-of-the-art method, across all LLaMA models, demonstrating the effectiveness of our approach in overcoming the limitations of transformation-based methods for MXFP4 quantization. LATMiX and BRQ, the previous MXFP4-specific methods, also achieve lower performance than MBA across all models, with the largest gaps observed on the LLaMA2 2-7B model

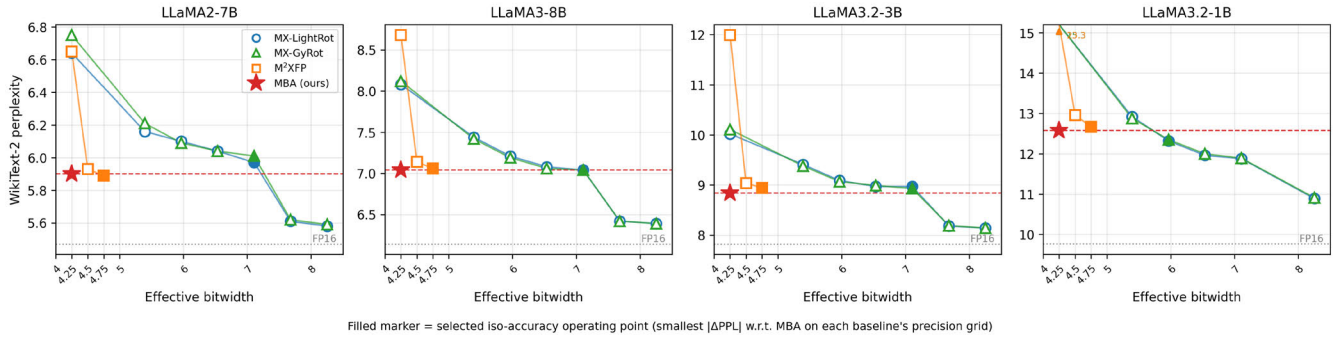


FIGURE 6. WikiText-2 perplexity versus effective bitwidth for iso-accuracy comparison. Filled markers denote the selected iso-accuracy operating points of each baseline.

TABLE 2. Zero-shot commonsense reasoning benchmark comparison of MBA and other algorithmic methods under W4A4KV4 MXFP4 quantization on various LLaMA models across 10 different seeds (higher is better).

Model	Method	PIQA	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	BoolQ	SIQA	LAMBADA	Average
2-7B	FP16	78.89	76.06	69.22	74.54	45.90	44.00	77.92	46.01	73.59	65.13
	MXFP4	76.30 ± 0.13	70.82 ± 0.17	65.68 ± 0.79	68.56 ± 0.33	40.67 ± 0.81	39.84 ± 0.73	71.87 ± 0.82	43.77 ± 0.49	69.81 ± 0.36	60.81 ± 0.22
	QuaRot	72.86 ± 0.62	63.08 ± 0.61	63.28 ± 1.33	64.59 ± 2.06	36.40 ± 0.74	36.48 ± 1.35	70.56 ± 1.21	41.77 ± 1.08	64.82 ± 2.00	57.09 ± 0.75
	SpinQuant	75.54 ± 0.52	69.63 ± 0.21	65.78 ± 0.71	67.51 ± 0.67	40.34 ± 0.72	38.28 ± 1.17	73.35 ± 0.66	42.84 ± 0.61	68.40 ± 0.50	60.19 ± 0.28
	FlatQuant	76.20 ± 0.26	71.45 ± 0.12	65.89 ± 0.84	70.20 ± 0.72	41.23 ± 0.56	40.58 ± 0.73	74.03 ± 0.64	43.65 ± 0.41	68.98 ± 0.35	61.36 ± 0.19
	BRQ	77.23 ± 0.48	72.40 ± 0.25	67.28 ± 0.63	71.43 ± 0.61	42.81 ± 0.70	42.02 ± 1.07	74.95 ± 0.78	44.42 ± 0.86	72.13 ± 0.64	**62.74 ± 0.26**
	LATMiX	72.59 ± 0.51	61.65 ± 0.33	57.72 ± 0.85	57.97 ± 0.60	33.37 ± 0.57	34.40 ± 1.18	66.01 ± 0.59	39.25 ± 0.59	54.31 ± 0.40	53.03 ± 0.23
	MBA	77.98 ± 0.19	72.36 ± 0.30	67.30 ± 0.73	71.44 ± 1.06	42.31 ± 0.63	41.04 ± 0.77	74.69 ± 0.63	44.62 ± 0.47	71.88 ± 0.42	62.63 ± 0.26
3-8B	FP16	80.90	79.12	73.40	77.69	53.33	45.00	80.95	47.29	75.74	68.16
	MXFP4	74.92 ± 1.48	72.35 ± 0.18	68.03 ± 1.59	67.41 ± 2.42	42.06 ± 2.51	41.68 ± 0.73	75.16 ± 1.15	45.12 ± 0.30	67.75 ± 0.64	61.61 ± 0.46
	QuaRot	73.98 ± 0.84	67.94 ± 0.40	68.36 ± 0.60	70.86 ± 1.50	43.40 ± 1.54	40.22 ± 1.53	75.96 ± 2.08	44.20 ± 0.96	72.19 ± 0.80	61.90 ± 0.47
	SpinQuant	77.33 ± 0.69	73.83 ± 0.25	68.88 ± 1.01	71.63 ± 1.13	46.48 ± 0.56	42.40 ± 0.68	78.58 ± 0.43	45.77 ± 0.57	69.02 ± 0.72	63.77 ± 0.28
	FlatQuant	78.05 ± 0.40	75.24 ± 0.23	70.64 ± 0.98	74.44 ± 0.94	47.44 ± 1.12	42.94 ± 0.94	79.36 ± 0.53	44.54 ± 0.52	71.91 ± 0.34	64.93 ± 0.30
	BRQ	77.83 ± 0.58	74.78 ± 0.33	70.48 ± 0.90	74.56 ± 1.61	47.91 ± 1.30	43.16 ± 0.87	78.52 ± 0.95	45.58 ± 0.84	73.11 ± 0.64	65.10 ± 0.38
	LATMiX	77.56 ± 0.44	73.09 ± 0.17	68.01 ± 0.76	72.98 ± 0.85	45.19 ± 1.00	40.90 ± 0.67	76.75 ± 0.63	44.26 ± 0.56	68.00 ± 0.66	62.97 ± 0.24
	MBA	78.66 ± 0.63	75.28 ± 0.26	71.00 ± 0.86	75.12 ± 1.10	48.63 ± 0.51	43.62 ± 1.21	79.43 ± 0.83	45.55 ± 0.90	73.91 ± 0.61	**65.69 ± 0.27**
3.2-3B	FP16	77.31	73.59	68.51	71.59	46.08	42.80	73.06	47.24	70.10	63.36
	MXFP4	73.28 ± 0.33	66.06 ± 0.17	60.54 ± 1.69	62.78 ± 1.04	38.53 ± 1.35	36.64 ± 1.13	67.92 ± 1.34	43.08 ± 0.36	59.18 ± 0.79	56.45 ± 0.42
	QuaRot	68.38 ± 0.60	52.99 ± 1.10	57.13 ± 1.49	56.58 ± 1.77	32.52 ± 1.16	31.68 ± 1.19	66.48 ± 0.79	40.20 ± 0.48	48.10 ± 2.19	50.45 ± 0.60
	SpinQuant	73.43 ± 0.95	66.42 ± 0.27	62.58 ± 0.56	63.14 ± 1.45	39.36 ± 1.01	37.66 ± 1.22	69.93 ± 0.75	43.68 ± 0.51	60.87 ± 0.71	57.45 ± 0.45
	FlatQuant	74.99 ± 0.63	68.02 ± 0.18	64.94 ± 0.71	65.91 ± 0.58	40.27 ± 0.78	39.69 ± 1.32	69.27 ± 0.78	44.56 ± 0.59	61.46 ± 0.64	58.79 ± 0.23
	BRQ	73.72 ± 0.44	66.43 ± 0.28	62.99 ± 1.39	65.61 ± 1.05	39.80 ± 1.29	37.30 ± 0.95	68.46 ± 2.16	43.80 ± 0.49	63.34 ± 0.84	57.94 ± 0.33
	LATMiX	74.48 ± 0.75	66.48 ± 0.18	62.03 ± 1.60	67.33 ± 0.82	40.34 ± 0.85	37.58 ± 0.55	69.14 ± 1.34	42.70 ± 0.56	60.26 ± 0.72	57.82 ± 0.29
	MBA	75.19 ± 0.45	68.35 ± 0.30	64.75 ± 1.08	66.82 ± 1.47	41.54 ± 0.69	39.48 ± 0.81	70.72 ± 1.37	44.72 ± 0.56	65.74 ± 0.68	**59.70 ± 0.28**
3.2-1B	FP16	74.54	63.63	61.01	60.35	36.43	37.20	63.79	42.99	62.06	55.78
	MXFP4	67.58 ± 0.92	51.74 ± 0.22	52.93 ± 1.03	50.29 ± 0.71	29.83 ± 0.66	31.64 ± 0.59	60.32 ± 0.25	40.26 ± 0.31	37.62 ± 0.53	46.91 ± 0.05
	QuaRot	61.33 ± 0.82	37.88 ± 0.58	52.57 ± 1.16	44.80 ± 1.08	25.91 ± 0.96	27.14 ± 1.40	61.43 ± 0.81	36.62 ± 0.62	18.41 ± 1.93	40.68 ± 0.30
	SpinQuant	68.57 ± 0.53	51.67 ± 0.35	54.32 ± 1.08	53.69 ± 0.66	30.90 ± 0.66	31.88 ± 1.02	61.27 ± 0.62	39.29 ± 0.57	40.72 ± 0.96	48.04 ± 0.20
	FlatQuant	69.74 ± 0.47	54.70 ± 0.21	55.30 ± 0.96	52.68 ± 0.59	30.80 ± 1.10	32.72 ± 0.99	60.38 ± 0.42	39.86 ± 0.66	48.29 ± 0.96	49.38 ± 0.20
	BRQ	68.88 ± 0.69	53.65 ± 0.48	54.77 ± 1.11	52.50 ± 0.78	30.61 ± 0.98	33.28 ± 1.30	60.86 ± 0.95	40.39 ± 0.61	47.16 ± 1.44	49.12 ± 0.35
	LATMiX	69.44 ± 0.61	53.64 ± 0.22	56.21 ± 0.84	55.01 ± 0.64	31.70 ± 0.90	32.64 ± 1.30	61.11 ± 0.30	39.96 ± 0.47	45.49 ± 1.48	49.47 ± 0.38
	MBA	69.81 ± 0.70	55.01 ± 0.40	55.93 ± 1.04	53.91 ± 1.62	32.80 ± 1.38	33.44 ± 1.04	61.63 ± 1.61	41.34 ± 0.59	49.40 ± 1.63	**50.37 ± 0.57**

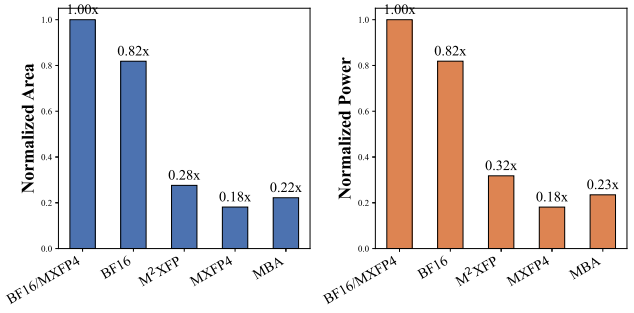


FIGURE 7. Normalized hardware comparison of different PE designs.

for LATMiX, 1.53, and on the LLaMA3.2-1B model for BRQ, 1.24.

Table 2 reports the mean and standard deviation of accuracy on the nine zero-shot commonsense reasoning benchmarks, obtained by repeating each experiment over 10 different random seeds. MBA achieves the highest average accuracy across all models except LLaMA2-7B. Compared with the conventional rotation-based methods QuaRot and SpinQuant, MBA achieves better performance, outperforming them by 3.79%p and 1.92%p, respectively, on LLaMA3-8B, the largest model in our experiments, and by 9.69%p and 2.33%p, respectively, on LLaMA3.2-1B, the smallest model. Furthermore, compared with FlatQuant, another affine-transformation-based method, MBA achieves a higher average accuracy, with the gap between the two averaging 0.98%p across the four evaluated models. These results support our claim that rotation-based methods suffer

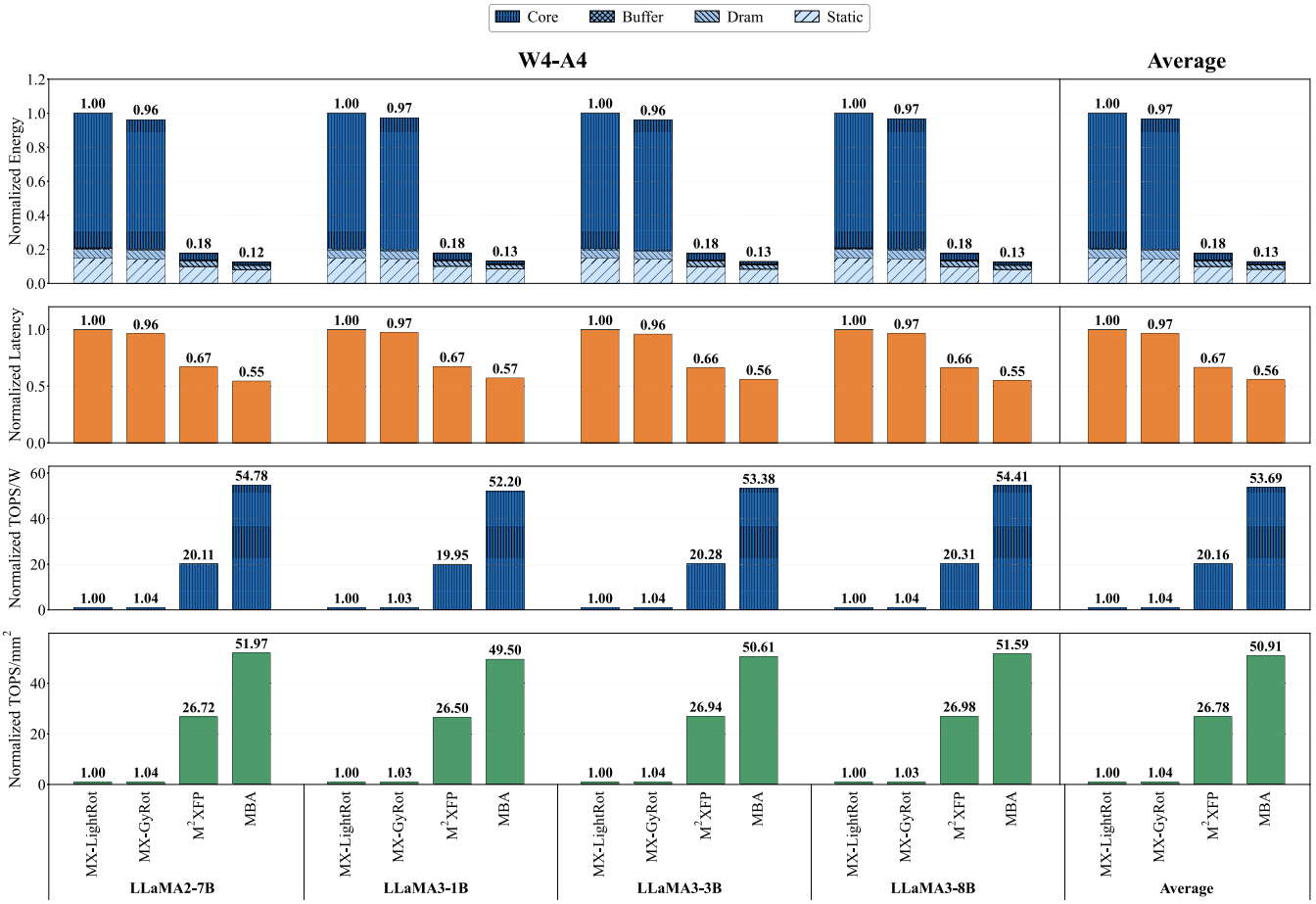


FIGURE 8. Normalized energy, latency, TOPS/W, and TOPS/mm^2 comparison between MBA and baseline accelerators.

from a significant increase in quantization error due to distribution shifts under MXFP4 quantization, and that the affine transformation must also incorporate block-wise optimization that considers the characteristics of MX data. In addition, MBA outperforms LATMiX, which likewise targets MXFP4 quantization as in this paper, with an average-accuracy gap that averages 3.78%p across the four models. Although BRQ marginally outperforms MBA by 0.11%p on LLaMA2-7B, MBA achieves higher average accuracy than BRQ on the remaining models.

2) COMPARISON WITH BASELINE ACCELERATORS

We evaluated the perplexity on Wikitext-2 across several LLM models, comparing MBA with MX-LightRot, MX-GyRot, BlockDialect, MX+, and M^{2}XFP under W4A4KV4 quantization with a block size of 32 and an E8M0 shared scaling factor. As shown in Table 1, MBA achieves the lowest perplexity across all models. MX-LightRot and MX-GyRot show accuracy improvements over naive MXFP4 quantization, but still suffer from worse degradation than MBA. This shows the incompatibility of rotation-based methods with MXFP4 quantization, as the data distribution after rotation cannot be captured by its block scaling factor, leading to significant quantization error. BlockDialect and MX+ achieve better accuracy than QuaRot, but worse than MBA, especially for smaller models, such as 1B and 3B, where the presence of outliers is more severe. While M^{2}XFP narrows the accuracy gap through additional mantissa bits for subgroup-level scaling factors and elements, this incurs non-trivial storage and memory bandwidth overhead and still cannot match the accuracy of MBA. Notably, MBA achieves these results while maintaining an effective bitwidth of only 4.25, which is lower than that of M^{2}XFP (4.50), MX+ (4.50), and BlockDialect (4.28). This demonstrates the effectiveness of MBA, which directly suppresses outliers through block-wise affine transformation, achieving higher accuracy at lower effective bitwidth with no additional metadata cost.

C. HARDWARE EVALUATION

1) PERFORMANCE, POWER, AND AREA

Table 3 summarizes the area and power breakdown of the MBA accelerator synthesized at 1 GHz in 28 nm FD-SOI technology. The dual-mode PE array dominates both area and power. The quantization engine accounts for only 0.2% of the total area and 0.6% of the total power, demonstrating

TABLE 3. Area and power breakdown of the proposed accelerator (28 nm FD-SOI, 1 GHz).

Component	Area (mm^2)	Power (mW)
PE Array (1024 PEs)	0.4737 (59.3%)	379.52 (84.1%)
Quantization Engine	0.0018 (0.2%)	2.50 (0.6%)
On-chip Buffer (360 KB)	0.3238 (40.5%)	69.18 (15.3%)
Total	**0.7993**	**451.20**

the low overhead of MX quantization in E8M0 format. The design includes 360 KB of on-chip buffer: 144 KB for both activations and weights, and 72 KB for output activations, including the MX shared scale.

To quantify the hardware overhead of the MBA PE design, we also synthesized a separate-unit PE, a single-mode BF16 PE, a single-mode MXFP4 PE, and an M^2XFP PE using the same 28 nm process. Fig. 7 shows the normalized area and power of each design relative to the separate-unit PE. Separately implementing dedicated hardware for each precision mode incurs substantial area and power overhead, and the MBA PE achieves 0.22× area and 0.23× power consumption compared to the separate PE design. Also, compared with a single-mode MXFP4 PE design, the MBA PE shows negligible area and power overhead. This demonstrates that our MBA PE design achieves the flexibility to support both BF16 and MXFP4 computation with negligible hardware overhead, enabling efficient execution of the proposed method.

2) ISO-ACCURACY PERFORMANCE COMPARISON

Fig. 8 compares normalized latency, energy, TOPS/W, and TOPS/mm^2 against MX-LightRot, MX-GyRot, and M^2XFP under identical hardware configurations. Since each of the MBA PEs can process four MXFP4 MAC operations per cycle, a fair-comparison configuration uses 256 PEs rather than the full 1,024-PE array. The designs differ only in the PE design, decoder, and rotation unit, while 1024 MXFP4 MAC units are identical across all designs. Fig. 6 shows the accuracy-versus-effective bitwidth curves of all baselines on all four LLaMA models, with the selected iso-accuracy operating points marked. Throughout the comparison, accuracy is measured by WikiText-2 perplexity. To match MBA's accuracy, MX-GyRot and MX-LightRot require promoting linear layers (both weights and activations) from MXFP4 to MXFP8 at layer granularity: the down/gate/up projections for LLaMA3.2-1B, and the down/gate/up/out/q projections for the other three models. In contrast, MBA operates entirely in MXFP4 precision without the need for higher-precision, which results in a large performance gap. Beyond precision fallback, both rotation-based designs incur additional area and power overhead from their dedicated rotation units, whereas MBA executes its affine transformations on the same reconfigurable PEs in BF16 mode, requiring no auxiliary transformation logic. For M^2XFP, matching MBA's accuracy requires increasing the effective bitwidth from 4.5 to 4.75 bits for the activations, which translates into higher memory traffic and energy cost. Furthermore, processing the extra mantissa bits increases latency and energy in the decoder and PE. In contrast, MBA achieves higher accuracy with no additional metadata cost, enabling it to fully leverage the efficiency of MXFP4 quantization and demonstrating negligible overhead of the affine transformation without any dedicated units. Overall, MBA achieves 1.21× speedup, 1.4× energy savings, 2.66× higher TOPS/W, and 1.9×TOPS/mm^2 improvement over state-of-the-art M^2XFP on average.

Regarding the sensitivity of our hardware results to the evaluation configuration, within the linear-layer MM, batch size and sequence length only scale the MM dimensions, and this scaling applies identically to all compared methods; the normalized speedup and energy efficiency ratios reported in Fig. 8 therefore remain unchanged across these parameters.

D. ABLATION STUDY

This section investigates the sensitivity of MBA to its key hyperparameters and training cost. Unless otherwise specified, all ablation experiments use 128 sequences randomly sampled from the WikiText-2 training split as the calibration set, following the calibration procedure described in Section VI-A. Hyperparameters are selected based on perplexity measured on the WikiText-2 test set. We note that this WikiText-2 test-set perplexity is also reported in Table 1 as an evaluation metric, so the WikiText-2 results should be interpreted with this overlap in mind; in contrast, the zero-shot benchmarks in Table 2 were never used for hyperparameter selection and thus provide an independent check of generalization.

1) COMPONENT ANALYSIS

Table 4 reports the performance difference on LLaMA3-8B due to clipping and per-channel scaling. When LWC, LAC, and scaling are individually enabled, the WikiText-2 perplexities are 7.14, 7.21, and 7.19, respectively, while using all three together yields a perplexity of 8.01. Overall, LAC adversely affects MBA performance, whereas the combination of LWC and scaling achieves the best WikiText-2 perplexity of 7.04.

TABLE 4. Ablation study of block-wise learnable clipping thresholds on LLaMA3-8B. LWC and LAC denote the learnable weight and activation clipping thresholds, respectively. 0-Avg. denotes the average zero-shot accuracy.

LWC	LAC	Scaling	Wiki2	C4	0-Avg.
			7.15	11.70	65.28
✓			7.14	11.87	65.73
	✓		7.21	11.76	65.48
		✓	7.19	11.75	65.65
✓	✓		7.15	11.90	**65.88**
	✓	✓	7.23	11.85	65.15
✓		✓	**7.04**	**11.75**	65.83
✓	✓	✓	8.01	13.51	64.31

2) MOMENT REGULARIZATION

Table 5 shows the effect of the moment regularizer on LLaMA3-8B. Without moment regularization, the model is optimized only with the output MSE loss and obtains a WikiText-2 perplexity of 7.88. Adding the moment regularizer improves the quantization performance, and the best result is obtained with $\lambda = 0.05$, $\gamma^* = 0$, and $\kappa^* = 6$.

TABLE 5. Ablation study of moment-matching loss design on LLaMA3-8B. 0-Avg. denotes the average zero-shot accuracy.

λ	Kurtosis	Wiki2	C4	0-Avg
	Baseline	7.88	13.36	64.04
0.01	3	8.19	14.07	63.64
	6	8.25	14.29	64.40
0.05	3	7.32	12.20	64.73
	6	**7.04**	**11.75**	**65.83**
0.1	3	7.73	12.93	64.56
	6	7.10	11.91	65.26
1.0	3	7.61	12.93	64.69
	6	7.54	12.59	64.42

The distribution comparison in Section IV is intended as a directional theoretical motivation, rather than as a requirement that the transformed activations exactly follow a Gaussian distribution. Under the idealized assumptions of the error-bound analysis, the Gaussian reference distribution has a more favorable bound than the Laplace reference distribution, indicating that excessive tail dominance is undesirable for MX quantization. However, the practical objective in (8) matches only the third and fourth standardized moments and does not constrain the complete activation distribution. Consequently, setting $\kappa^* = 3$ does not guarantee that the learned block-wise activation distributions become Gaussian. Likewise, $\kappa^* = 6$ should not be interpreted as enforcing a Laplace distribution. We therefore use $\kappa^* = 6$ as an empirically selected moderate-kurtosis regularization target for the finite-block MXFP4 setting.

Tables 6-8 provide quantization-aware diagnostics for the $\kappa^* = 3$ and $\kappa^* = 6$ settings. $MBA_{\kappa 3}$ and $MBA_{\kappa 6}$ denote models trained with $\kappa = 3$ and $\kappa = 6$, respectively. Although $MBA_{\kappa 3}$ concentrates more values around zero, it retains more severe standardized outliers. In Table 6, compared with $MBA_{\kappa 6}$, $MBA_{\kappa 3}$ exhibits 1.59×, 2.50×, and 4.64× higher outlier ratios at 3σ, 4σ, and 6σ, respectively. Thus, matching the Gaussian kurtosis value in expectation does not by itself control the extreme-tail behavior that is relevant to a finite-range quantizer.

This distinction is particularly important for MXFP4, where 32 values share a single E8M0 scale. As shown in Table 7, $MBA_{\kappa 3}$ yields a block kurtosis closer to 3 on average, but its 32-element blocks are more strongly dominated by individual large values. In particular, the median and 90th-percentile crest factors are 2.469 and 3.126 for $MBA_{\kappa 3}$, compared with 2.300 and 2.801 for $MBA_{\kappa 6}$, respectively.

TABLE 6. Ratio of outlier distribution for each method. σ denotes the standard deviation.

Outlier Ratio	$MBA_{\kappa 6}$	$MBA_{\kappa 3}$	$\frac{MBA_{\kappa 3}}{MBA_{\kappa 6}}$
$\geq 1\sigma$	25.98%	15.89%	0.61
$\geq 2\sigma$	4.87%	4.30%	0.88
$\geq 3\sigma$	1.04%	1.65%	1.59
$\geq 4\sigma$	0.33%	0.83%	2.50
$\geq 5\sigma$	0.14%	0.48%	3.51
$\geq 6\sigma$	0.07%	0.31%	4.64

TABLE 7. Block-wise statistical metrics for each method. μ denotes the mean.

Block-wise Metric	$MBA_{\kappa 6}$	$MBA_{\kappa 3}$
κ	2.68 ± 0.768	2.97 ± 1.274
μ	1.8E-5 $\pm$ 0.011	-2.3E-5 $\pm$ 0.046
$\lvert\mu\rvert$	5.6E-3 $\pm$ 0.009	16.1E-3 $\pm$ 0.043
$\lvert\mu\rvert$ $p90$	1.77E-02	6.19E-02
CF_g	2.34 ± 0.34	2.53 ± 0.43
CF_g median	2.30	2.47
CF_g $p90$	2.80	3.13
$M_{\lvert\mu\rvert}$	2.91 ± 0.53	3.26 ± 0.78

TABLE 8. The proportion of overflow occurrences caused by clipping during quantization for each method.

Overflow Ratio	$MBA_{\kappa 6}$	$MBA_{\kappa 3}$
Element-wise	2.38%	2.19%
Block-wise	39.15%	40.21%

Here, the crest factor is defined as:

$$\mathrm{CF}_g = \frac{\max_{i \in g} |x_i|}{\sqrt{\frac{1}{g}\sum_{i \in g} x_i^2 + \epsilon}}, \tag{10}$$

where $g = 32$ is the MXFP4 block size. A larger crest factor indicates that a small number of values disproportionately determine the dynamic range of a shared-scale block. Consistently, the max-mean ratio $M_{|\mu|} = \frac{\max |x|}{\sum |x| * 1/|g|}$ is also larger for $MBA_{\kappa 3}$ than for $MBA_{\kappa 6}$ (3.256 versus 2.908).

Table 8 reports two complementary saturation statistics under the actual E8M0 block scale. The element-wise overflow ratio denotes the fraction of activation elements whose scaled values exceed the representable range of the quantization format and are consequently saturated to an endpoint value. The block-wise overflow ratio is the fraction of 32-element blocks that contain at least one such saturated element. While the former measures the frequency of individual saturation events, the latter measures how often a shared-scale MX block is affected by saturation. The latter is especially relevant in MXFP4 because a single extreme element can influence the range utilization of all values sharing the same E8M0 scale. $MBA_{\kappa 3}$ slightly lowers the element overflow ratio (2.19% versus 2.38%), but it does not reduce the block overflow ratio (40.21% versus 39.15%). Therefore, the small reduction in the aggregate number of

saturated elements does not translate into fewer MXFP4 blocks affected by saturation. Together, these results explain why directly using the Gaussian kurtosis target is not optimal in our finite-block setting. The target $\kappa^* = 6$ is not selected because a Laplace distribution is theoretically preferable. It empirically provides the most favorable trade-off among the tested targets by reducing severe tail events and avoiding max-dominated shared-scale blocks while preserving the end-to-end performance shown in Table 5.

3) BLOCK SIZE

Table 9 presents WikiText-2 perplexity for different block sizes. The best result is obtained at block size 32, which matches the MX block size. Reducing the block size to 16 degrades performance (7.12), since a transformation block smaller than the MX block cannot jointly control the maximum and the distribution of the 32 elements that share a single E8M0 scale. Increasing the block size beyond 32 provides no further accuracy improvement (7.05-7.06), while the parameter count and computational cost of the transformation grow linearly with g. Therefore, the MX block size of 32 is the optimal transformation granularity in terms of both accuracy and cost, and the transformation should respect the MX block size.

TABLE 9. Ablation study of block size on LLaMA3-8B. $O(\cdot)$ denotes computational complexity.

Block Size	16	32	64	128	256	512	1024
Wiki2	7.12	7.04	7.06	7.05	7.05	7.05	7.06
Norm. $O(\cdot)$	1	2	4	8	16	32	64

4) CALIBRATION DATASET

To evaluate the sensitivity of MBA to the calibration set, we replace the WikiText-2 calibration samples with 128 sequences randomly sampled from C4 while keeping the remaining calibration procedure and hyperparameters unchanged. The impact of the calibration dataset and sequence length is summarized in Table 10. When the sequence length is reduced to 1024, the WikiText-2 perplexity increases from 7.04 to 7.39 and from 7.51 to 7.72, respectively. Using C4 as the calibration set increases the WikiText-2 perplexity from 7.04 to 7.51 and from 7.39 to 7.72. Nevertheless, both the perplexity on C4 and the average zero-shot performance remain stable, indicating that the method is robust with respect to the choice of calibration set.

TABLE 10. Ablation study of calibration set and sequence length on LLaMA3-8B. 0-Avg. denotes the average zero-shot accuracy.

Calib.Set	seq_len	Wiki2	C4	0-Avg.
Wiki2	2048	7.04	11.75	65.83
C4	2048	7.51	11.82	65.54
Wiki2	1024	7.39	11.94	64.77
C4	1024	7.72	11.84	64.98

5) TRAINING COST

MBA is trained for 15 epochs on 128 samples from WikiText-2 (sequence length 2048, batch size 2). The training time and GPU memory usage on each model are summarized in Table 11. Optimization steps are summarized in Table 12, showing that 15 epochs is optimal on both WikiText-2 and zero-shot tasks.

TABLE 11. Per-model training time (min) and approximated GPU memory usage (GB) on a single NVIDIA RTX 5090 GPU.

Model	Time	Mem
LLaMA2-7B	51	24
LLaMA3-8B	53	24
LLaMA3.2-3B	50	15
LLaMA3.2-1B	21	13

TABLE 12. Ablation study of optimization steps on LLaMA3-8B. 0-Avg. denotes the average zero-shot accuracy.

epoch	Wiki2	C4	0-Avg.
5	7.31	12.14	64.73
10	7.08	11.67	65.70
15	7.04	11.75	65.83

6) IMPLEMENTATION DETAILS

To validate our algorithm, the MBA framework was implemented on top of the official FlatQuant [7] implementation. The transformation set of MBA, $\{R_a, R_v, R_o, R_k, R_m, R_d\}$, is applied in the same manner as FlatQuant's $\{P_a, P_h, P_v, P_o, P_{ug}, P_d\}$, except that MBA does not employ the Kronecker decomposition. For calibration, we use 128 randomly sampled sequences of 2048 tokens from the WikiText-2 training set. All experiments were conducted on a single NVIDIA RTX 5090, with a calibration batch size of 2, 15 epochs, and a default random seed of 0. The calibration objective combines the MSE loss and the moment-matching loss, optimized with AdamW using a cosine learning-rate schedule and a weight decay of 0.01, without warmup. The learning rate is set to $1e^{-3}$ for the transformation matrices and per-channel scaling factors, and $1e^{-2}$ for LWC and LAC, when enabled. Calibration is performed in a layer-wise manner. The block-diagonal matrices are initialized as random orthogonal matrices using QR decomposition, and the per-channel scaling follows the default scheme of SmoothQuant [13].

As baselines, we selected LLM quantization methods based on rotation or affine transformation. Since most of them do not consider MX quantization, we either implemented them on top of their official implementations to support MX quantization or reproduced them from their paper descriptions. QuaRot, SpinQuant, and FlatQuant were implemented by adding MXFP4 quantization to their official implementations, and both calibration and evaluation were performed using the default settings reported in their

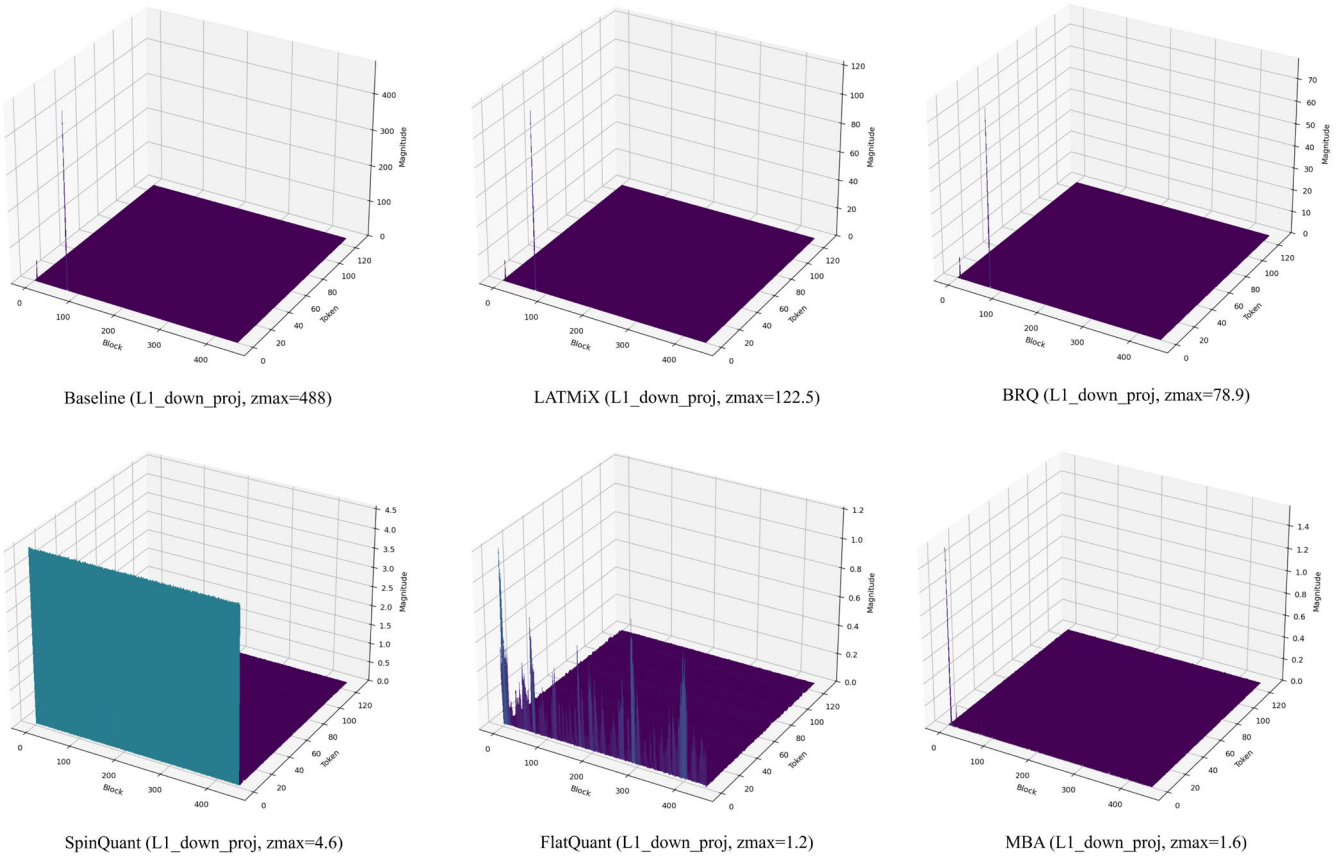


FIGURE 9. **Comparison of activation distributions across MBA and other methods on LLaMA3-8B (layer 1 down projection).**

respective papers. Since the official implementation of BRQ is not publicly available, we implemented BRQ on top of the official QuaRot [5] implementation by replacing the global Hadamard rotation with the block-wise Hadamard rotation described in [8], applied at the MX block granularity ($g = 32$); as BRQ employs a fixed rotation, no additional training is involved. LATMiX was reproduced following the implementation details specified in [12] and implemented on top of the official SpinQuant [6] implementation. The invertible affine matrices were trained with a teacher-student distillation loss on 256 random samples from the WikiText-2 training set with a sequence length of 1024, using the AdamW optimizer with a learning rate of $1e^{-4}$ and a cosine learning-rate schedule, for 500 optimization steps with a batch size of 4. All experiments were conducted under the W4A4KV4 configuration with GPTQ [25] quantization.

7) GENERALIZATION TO OTHER MODEL FAMILIES

To evaluate whether MBA generalizes beyond the LLaMA family, we additionally evaluate MBA and LATMiX, the previous MXFP4-targeted method, on the Qwen3-1.7B and Qwen3-4B models under the same W4A4KV4 MXFP4 setting and calibration procedure described in Section VI-D. As shown in Table 13, MBA achieves lower WikiText-2 perplexity and higher average zero-shot accuracy than LATMiX on both models, indicating that the proposed block-wise affine transformation is not specific to the LLaMA architecture and generalizes to a different model family.

TABLE 13. **Average zero-shot commonsense reasoning accuracy and WikiText-2 perplexity comparison of MBA and LATMiX on Qwen3 models under W4A4KV4 MXFP4 quantization.**

Model	Method	Wiki2 (↓)	0-Avg (↑)
Qwen3-1.7B	FP16	16.71	57.40
	LATMiX	25.08	46.77
	MBA	**19.41**	**51.88**
Qwen3-4B	FP16	13.66	64.33
	LATMiX	15.55	59.21
	MBA	**15.04**	**60.81**

E. DISCUSSIONS

While our current design optimizes linear layers and KV cache quantization, the attention mechanism becomes the dominant bottleneck at longer contexts, accounting for

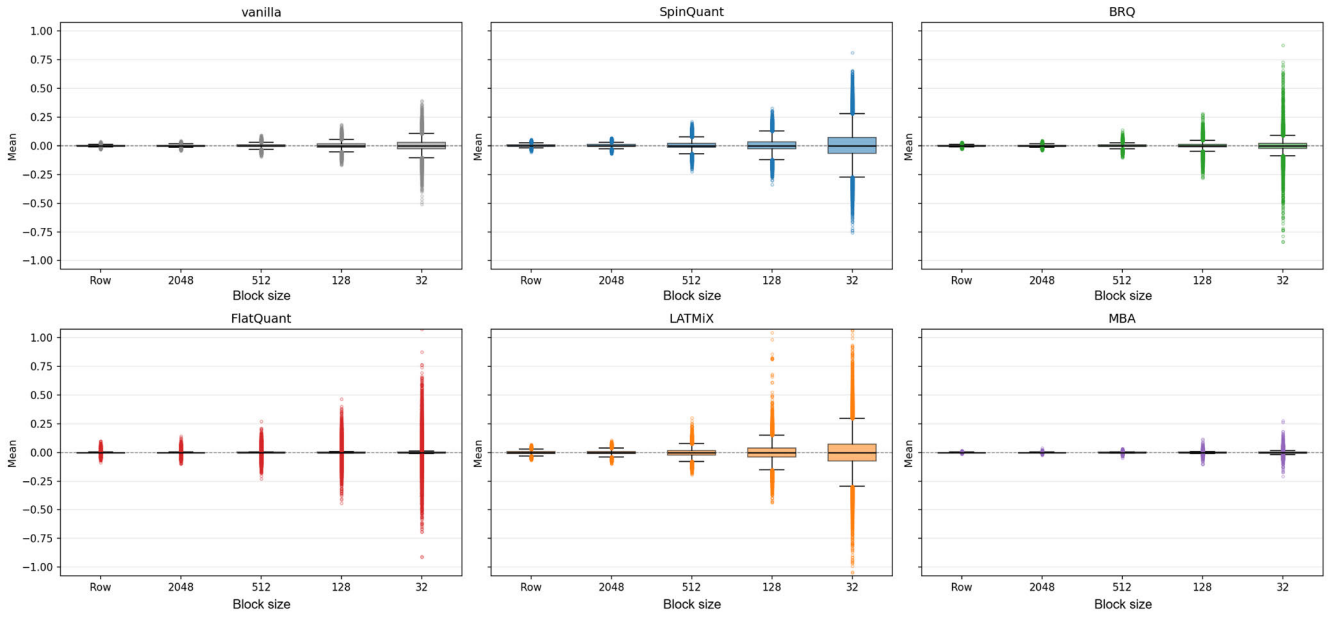


FIGURE 10. Block-wise kurtosis distributions across block sizes for MBA and other methods on LLaMA3-8B.

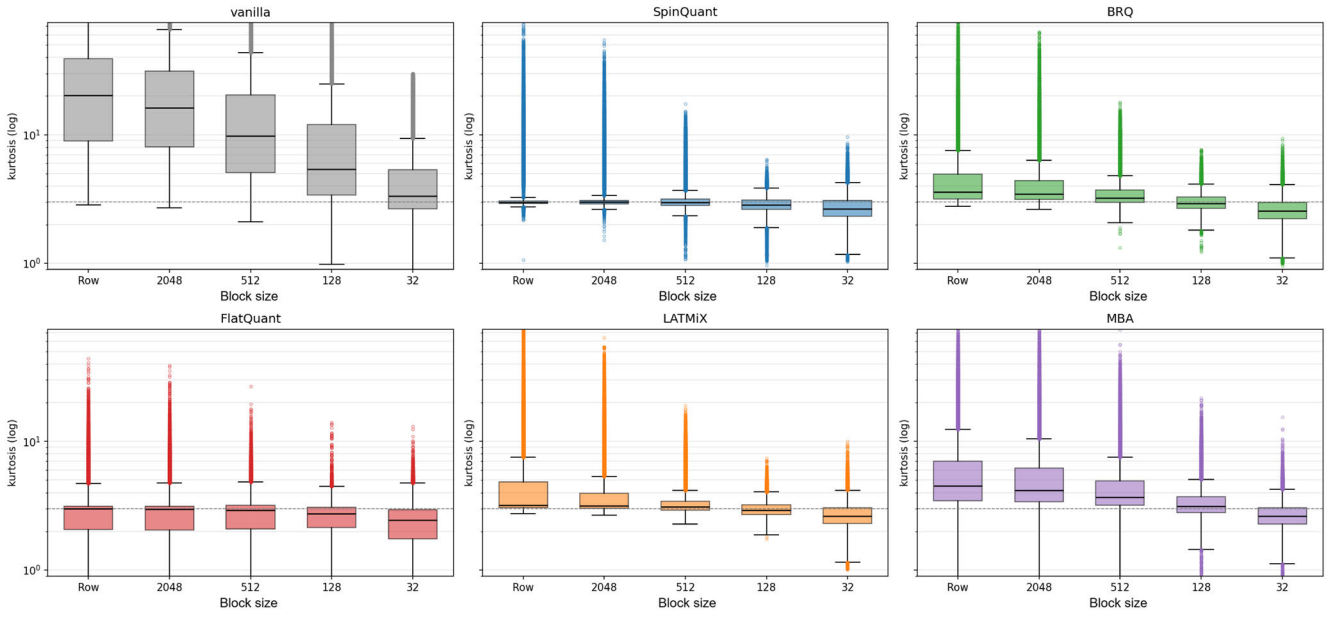


FIGURE 11. Block-wise mean distributions across block sizes for MBA and other methods on LLaMA3-8B.

approximately 45% of latency at a context length of 16384 [31]. Since MBA supports KV cache quantization via R_k and R_v, the memory traffic of attention can be effectively reduced under our method. Therefore, extending our method to efficiently support attention computation with minimal accuracy loss by performing $P = QK^T$, $Softmax(P)$, and $O = PV$ in MXFP4 precision is an important direction for future work. Recent work, such as SageAttention3 [36], has proposed techniques to mitigate the accuracy degradation of attention under low-precision quantization, which can be potentially integrated with our method to achieve efficient MXFP4 quantization for attention computation.

VII. CONCLUSION

In this paper, we presented MBA, a software-hardware co-design framework for efficient MXFP4 quantization of large language models. We proposed a learnable block-wise affine transformation that is explicitly aligned with the

MX block structure, effectively mitigating outliers and adapting the distribution to be MXFP4-friendly. We also designed a dedicated hardware unit to efficiently support the proposed transformation with minimal hardware overhead. Under W4A4KV4 MXFP4 quantization, MBA achieves the lowest WikiText-2 perplexity across all evaluated LLaMA models and the highest average zero-shot accuracy on all evaluated models except LLaMA2-7B. Furthermore, for the evaluated linear-layer MM workload, including the online affine transformation, MBA delivers up to 1.21× speedup, 1.4× energy reduction, 2.66× higher TOPS/W, and 1.9× higher TOPS/mm^2, demonstrating the hardware efficiency of the proposed accelerator for the evaluated workload.

ACKNOWLEDGMENT

(Sangki Park and Chanhoon Kim contributed equally to this work.)

REFERENCES

[1] B. D. Rouhani, N. Garegrat, T. Savell, A. More, K.-N. Han, R. Zhao, M. Hall, J. Klar, E. Chung, and Y. Yu, "OCP microscaling formats (MX) specification," *Open Compute Project*, 2023. [Online]. Available: https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf

[2] NVIDIA Corporation. (2024). *NVIDIA Blackwell Architecture Technical Brief*. Accessed: Apr. 8, 2026. [Online]. Available: https://www.nvidia.com/en-us/data-center/technologies/blackwell-architecture/

[3] Advanced Micro Devices, Inc. (2025). *AMD Instinct MI350X GPUs*. Accessed: Apr. 8, 2026. [Online]. Available: https://www.amd.com/en/products/accelerators/instinct/mi350/mi355x.html

[4] Microsoft Corporation. (2023). *Microsoft Azure Maia 100 AI Accelerator*. Accessed: Apr. 8, 2026. [Online]. Available: https://news.microsoft.com/source/features/ai/in-house-chips/

[5] S. Ashkboos, A. Mohtashami, M. Croci, B. Li, P. Cameron, M. Jaggi, D. Alistarh, T. Hoefler, and J. Hensman, "QuaRot: Outlier-free 4-bit inference in rotated LLMs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, pp. 100213–100240.

[6] Z. Liu, C. Zhao, I. Fedorov, B. Soran, D. Choudhary, R. Krishnamoorthi, V. Chandra, Y. Tian, and T. Blankevoort, "SpinQuant: LLM quantization with learned rotations," in *Proc. 13th Int. Conf. Learn. Represent.*, 2025, pp. 92009–92032. [Online]. Available: https://openreview.net/forum?id=ogO6DGE6FZ

[7] Y. Sun, R. Liu, H. Bai, H. Bao, K. Zhao, Y. Li, J. Hu, X. Yu, L. Hou, C. Yuan, X. Jiang, W. Liu, and J. Yao, "FlatQuant: Flatness matters for LLM quantization," 2024, *arXiv:2410.09426*.

[8] Y. Shao, P. Wang, Y. Chen, C. Xu, Z. Wei, and J. Cheng, "Block rotation is all you need for MXFP4 quantization," 2025, *arXiv:2511.04214*.

[9] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, J. K. Kim, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 764–775.

[10] S.-H. Noh, S. Kim, S. Kim, D. Kim, J. Kung, and Y. Kim, "Jack unit: An area- and energy-efficient multiply-accumulate (MAC) unit supporting diverse data formats," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2025, pp. 1–7.

[11] S. Ashkboos, M. L. Croci, M. G. D. Nascimento, T. Hoefler, and J. Hensman, "SliceGPT: Compress large language models by deleting rows and columns," in *Proc. 12th Int. Conf. Learn. Represent.*, 2024, pp. 11682–11701. [Online]. Available: https://openreview.net/forum?id=vXxardq6db

[12] O. Gordon, L. Dikstein, A. Netzer, I. Achituve, and H. Victor Habi, "LATMiX: Learnable affine transformations for microscaling quantization of LLMs," 2026, *arXiv:2602.17681*.

[13] G. Xiao, J. Lin, M. Seznec, W. Hao, J. Demouth, and S. H. Han, "SmoothQuant: Accurate and efficient post-training quantization for large language models," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 38087–38099.

[14] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," 2023, *arXiv:2307.09288*.

[15] AI@Meta. (2024). *Llama 3 Model Card*. [Online]. Available: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[16] S. Merity, C. Xiong, J. Bradbury, and R. Socher. (2017). *Pointer Sentinel Mixture Models*. [Online]. Available: https://openreview.net/forum?id=Byj72udxe

[17] Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi, "PIQA: Reasoning about physical commonsense in natural language," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 7432–7439.

[18] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "HellaSwag: Can a machine really finish your sentence?" in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 4791–4800.

[19] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. (2019). *An Adversarial Winograd Schema Challenge at Scale*. [Online]. Available: https://api.semanticscholar.org/CorpusID:199370376

[20] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? Try ARC, the AI2 reasoning challenge," 2018, *arXiv:1803.05457*.

[21] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, "Can a suit of armor conduct electricity? A new dataset for open book question answering," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 2381–2391.

[22] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, "BoolQ: Exploring the surprising difficulty of natural yes/no questions," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol. (Long Short Papers)*, 2019, pp. 2924–2936.

[23] M. Sap, H. Rashkin, D. Chen, R. LeBras, and Y. Choi, "SocialIQA: Commonsense reasoning about social interactions," 2019, *arXiv:1904.09728*.

[24] D. Paperno, G. Kruszewski, A. Lazaridou, N. Q. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernandez, "The LAMBADA dataset: Word prediction requiring a broad discourse context," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1525–1534. [Online]. Available: http://www.aclweb.org/anthology/P16-1144

[25] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate post-training quantization for generative pre-trained transformers," 2022, *arXiv:2210.17323*.

[26] H. Sharma, J. Park, E. Amaro, B. Thwaites, P. Kotha, A. Gupta, J. K. Kim, A. Mishra, and H. Esmaeilzadeh, "DNNWEAVER: From high-level deep network models to fpga acceleration," in *Proc. Workshop Cognit. Archit.*, 2016, pp. 1–12.

[27] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0," in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2007, pp. 3–14.

[28] C. Guo, C. Zhang, J. Leng, Z. Liu, F. Yang, Y. Liu, M. Guo, and Y. Zhu, "ANT: Exploiting adaptive numerical data type for low-bit deep neural network quantization," in *Proc. 55th IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2022, pp. 1414–1433, doi: 10.1109/MICRO56248.2022.00095.

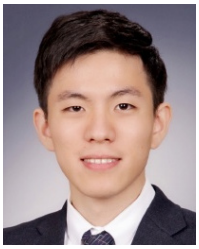
[29] C. Guo, J. Tang, W. Hu, J. Leng, C. Zhang, F. Yang, Y. Liu, M. Guo, and Y. Zhu, "OliVe: Accelerating large language models via hardware-friendly outlier-victim pair quantization," in *Proc. 50th Annu. Int. Symp. Comput. Archit.* New York, NY, USA: Association for Computing Machinery, Jun. 2023, pp. 1–15, doi: 10.1145/3579371.3589038.

[30] J. Zou, Y. Chen, X. Chen, C. Xu, and X. Chen, "AxCore: A quantization-aware approximate GEMM unit for LLM inference," in *Proc. 58th IEEE/ACM Int. Symp. Microarchitecture*. New York, NY, USA: Association for Computing Machinery, Oct. 2025, pp. 839–853, doi: 10.1145/3725843.3756094.

[31] W. Hu, Z. Zhang, H. Zhang, C. Zhang, C. Guo, Y. Feng, T. Hu, G. Li, G. Hu, J. Wang, and J. Leng, "M 2 XFP: A metadata-augmented microscaling data format for efficient low-bit quantization," in *Proc. 31st ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2026, pp. 1151–1167.

[32] W. Jang and T. Tambe, "BlockDialect: Block-wise fine-grained mixed format quantization for energy-efficient LLM inference," 2025, *arXiv:2501.01144*.

[33] J. Lee, J. Park, S. Cha, J. Cho, and J. Sim, "MX+: Pushing the limits of microscaling formats for efficient large language model serving," in *Proc. 58th IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2025, pp. 869–883.
[34] S. Kim, Y. Choi, J. Oh, B. Kim, and H.-J. Yoo, "LightRot: A light-weighted rotation scheme and architecture for accurate low-bit large language model inference," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 15, no. 2, pp. 231–243, Jun. 2025.
[35] S. Kim, Y. Choi, B. Kim, J. Oh, and H.-J. Yoo, "GyRot: Leveraging hidden synergy between rotation and fine-grained group quantization for low-bit LLM inference," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Jan. 2026, pp. 1–15.
[36] J. Zhang, J. Wei, P. Zhang, X. Xu, H. Huang, H. Wang, K. Jiang, J. Chen, and J. Zhu, "SageAttention3: Microscaling FP4 attention for inference and an exploration of 8-bit training," 2025, *arXiv:2505.11594*.



SANGKI PARK (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering from Hanyang University, Seoul, South Korea, in 2017, where he is currently pursuing the Ph.D. degree. His research interests include deep learning, model compression, and hardware acceleration.



CHANHOON KIM (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering from Dongguk University, Seoul, South Korea, in 2021. He is currently pursuing the Ph.D. degree in electronic engineering with Hanyang University, Seoul. His research interest includes hardware-accelerated large language model quantization.



SOOMIN RHO (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering from Hanyang University, ERICA Campus, Ansan, South Korea, in 2021. He is currently pursuing the Ph.D. degree in electronic engineering with Hanyang University, Seoul, South Korea. His research interests include model compression and hardware acceleration.



MINYOUNG LEE (Graduate Student Member, IEEE) received the B.S. degree in systems semiconductor engineering from Sangmyung University, Cheonan, South Korea, in 2021. He is currently pursuing the Ph.D. degree in electronic engineering with Hanyang University, Seoul, South Korea. His research interests include deep learning, model compression, and hardware acceleration.



KI-SEOK CHUNG (Member, IEEE) received the B.S. degree in computer engineering from Seoul National University, Seoul, South Korea, in 1989, and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, in 1998. He was a Senior Research and Development Engineer with Synopsys Inc., Mountain View, CA, from 1998 to 2000, and a Staff Engineer with Intel Corporation, Santa Clara, CA, from 2000 to 2001. He was also an Assistant Professor with Hongik University, Seoul, from 2001 to 2004. Since 2004, he has been a Professor with Hanyang University, Seoul. His research interests include low-power embedded systems design, multi-core architecture, image processing, reconfigurable processors and DSP design, SoC-platform-based verification, and systems software for MPSoCs.