

Article

Energy-Efficient Neural Network Acceleration Using Most Significant Bit-Guided Approximate Multiplier

Pengfei Huang , Bin Gong, Ke Chen * and Chenghua Wang

College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, No. 29 Jiangjun Avenue, Jiangning District, Nanjing 210016, China; pffhuang@nuaa.edu.cn (P.H.); bingong@nuaa.edu.cn (B.G.); chwang@nuaa.edu.cn (C.W.)

* Correspondence: chen.ke@nuaa.edu.cn

Abstract: The escalating computational demands of deep learning and large-scale models have led to a significant increase in energy consumption, highlighting the urgent need for more energy-efficient hardware designs. This study presents a novel weight approximation strategy specifically designed for quantized neural networks (NNs), resulting in the development of an efficient approximate multiplier leveraging most significant one (MSO) shifting. Compared to both energy-efficient logarithmic approximate multipliers and accuracy-prioritized non-logarithmic approximate multipliers, our proposed logarithmic-like design achieves an unparalleled balance between accuracy and hardware costs. When compared with the baseline exact multiplier, our innovative design exhibits remarkable reductions, encompassing a decrease of up to 28.31% in area, a notable 57.84% reduction in power consumption, and a diminution of 11.86% in delay. Experimental outcomes reveal that the proposed multiplier, when applied in neural networks, can conserve approximately 60% of energy without compromising task accuracy. Concurrently, experiments focused on the transformer accelerator and image processing illustrate the substantial energy savings that can be obtained for Large Language Models (LLMs) and image processing tasks through the implementation of our proposed design, further validating its efficacy and practicality.

Keywords: approximate computing; large language model; coefficient multiplier; logarithmic multiplier



Citation: Huang, P.; Gong, B.; Chen, K.; Wang, C. Energy-Efficient Neural Network Acceleration Using Most Significant Bit-Guided Approximate Multiplier. *Electronics* **2024**, *13*, 3034. <https://doi.org/10.3390/electronics13153034>

Academic Editors: Ke Xie, Shoue Chen and Xiaolong Wang

Received: 5 July 2024
Revised: 30 July 2024
Accepted: 31 July 2024
Published: 1 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The field of artificial intelligence has experienced significant progress, notably with the emergence of deep neural networks (DNNs) and Large Language Models (LLMs). However, these sophisticated models have introduced considerable computational and memory challenges, stemming from their ever-growing sizes, as thoroughly analyzed in [1]. These challenges are even more acute for edge devices, which are constrained by inherent limitations associated with their computational power. To enable the practical deployment of neural networks (NNs) on edge platforms, diverse software-based model compression techniques, such as quantization and pruning, have been presented as feasible solutions. Additionally, recent efforts leveraging the approximate computing (AxC) paradigm at the circuit level have persuasively demonstrated its capacity to improve system performance [2]. This improvement results from the reduction in computational complexity and memory demands, attained through replacing precise arithmetic elements with their less accurate equivalents. The key operations of arithmetic processors—namely, multiplication and addition—significantly impact their overall energy consumption. Notably, multiplication consumes more energy than addition. The core arithmetic computation performed within NNs is the multiply–accumulate (MAC) operation. As millions of multiplication operations are performed by an NN accelerator, approximate multipliers (AMs) have garnered extensive research attention due to their considerable significance in terms of improving the efficiency and accuracy of neural network applications [3–5].

Previous research on approximate multipliers customized for DNNs has delivered encouraging results regarding efficiency [6]. In the case of deploying NNs on resource-constrained edge devices, the employment of approximate multipliers provides an appealing means to strike a balance between accuracy and energy cost. This accomplishment relies on alleviating and relaxing the stringent requirements for perfectly precise multiplication outcomes, thus facilitating the creation of approximate hardware designs that are more resource- and energy-efficient. The study in [7] demonstrated the efficiency of AxC in DNNs through presenting a new AxC DNN layer that incorporates approximate multipliers into the DNN design process. In [8], three hybrid approximate multipliers were proposed by employing NAND gates instead of AND gates and applying different combinations of the compressors to provide diverse trade-offs between accuracy and hardware efficiency. An open-source adaptable approximate multiplier design, driven by input distribution and polarity, was suggested in [9] to generate optimized approximate multipliers. The purpose is to minimize the average square of the absolute error of an approximate multiplier based on the probability distributions of operands extracted from the target application, with the input polarity taken into consideration. In [10], a methodology for designing approximate array multipliers leveraging the concept of carry disregard was introduced. Through strategically selecting the positions for carry disregard, it becomes feasible to disregard a small number of carries, thereby achieving enhanced speed and a reduced area while simultaneously maintaining superior overall accuracy and ensuring suitability for the intended application.

Approximate logarithmic multipliers provide a simpler design but lead to a notably higher accuracy loss, whereas approximate non-logarithmic multipliers demonstrate a lower computational error at the cost of an increased design complexity [11]. To address the need for reduced energy consumption while maintaining an acceptable accuracy loss, this study proposes an energy-efficient most significant one-driven shifting multiplier (MSAM). The MSAM can obtain a range of highly efficient multiplier designs by dynamically configuring the approximation factor (k) and the precision factor (m). By leveraging the approximation factor (k), the multiplier is strategically divided into exact and approximate sub-multipliers. Within the approximation region, the re-configurable constant coefficient multiplier (RCCM) principle [12] is applied, substituting conventional multiplication operations with shift, addition, and subtraction operations. This journal article builds upon and extends our previous conference paper [13], offering several key differences and novel contributions. A summary of these advancements is provided below:

- We devised an alternative and more aggressive approximation strategy for the multiplier approximation process, pushing the boundaries of accuracy–efficiency trade-offs.
- We further conducted a comprehensive error analysis and thoroughly discuss the trade-offs between the accuracy and energy consumption.
- In addition to analyzing popular data sets, we also explore image processing using different AxC schemas.
- The advantages of our proposed approximate multiplier designs are substantiated through a rigorous theoretical analysis and extensive experimental demonstrations.

The remainder of this paper is structured as follows: In Section 2, a concise review of the background and related work is offered. Section 3 details our innovative methodology for energy-efficient neural network acceleration, incorporating a most significant bit-guided dynamic approximate multiplier to significantly enhance performance. In Section 4, an error analysis and simulation results are provided. Finally, Section 5 concludes the paper.

2. Preliminaries

A logarithmic multiplier (LM) converts numbers in binary format into logarithmic format, facilitating the transformation from multiplication to addition. Mitchell's algorithm was initially proposed in [14]. Given two binary numbers A and B , where $A = 2^{k_1}(1 + x_1)$

and $B = 2^{k_2}(1 + x_2)$, the logarithmic multiplier initially converts them into their logarithmic forms as follows:

$$\log_2(A \cdot B) = \log_2(2^{k_1+k_2}(1 + x_1)(1 + x_2)) = (k_1 + k_2) + \log_2(1 + x_1) + \log_2(1 + x_2), \quad (1)$$

where k_1 and k_2 denote the positions of the most significant bit (with a value of 1) in operands A and B , respectively, and x_1 and x_2 represent the fractional parts of A and B ($x_1, x_2 \in [0, 1]$). In Mitchell's algorithm, when $0 \leq x < 1$, the expression $\log_2(1 + x)$ is approximately equal to x . Therefore, the logarithm of the product can be expressed as follows:

$$\log_2(A \cdot B) \approx k_1 + k_2 + x_1 + x_2. \quad (2)$$

The logarithmic multiplier encompasses several crucial components: leading-one detectors (LODs), logarithmic converters, an adder, and an anti-logarithmic converter. The LODs are tasked with identifying the most significant 1 bit within the input binary numbers. Subsequently, the logarithmic converters generate the logarithmic representations of these binary numbers. These logarithmic values are added together to obtain the product, which is then decoded back into a binary number using the anti-logarithmic converter.

3. The Proposed Most Significant One-Driven Shifting Approximate Multiplier

In this study, we introduce a most significant one-driven shifting approximate multiplier that employs a logarithmic-like approach. This novel multiplier achieves improved accuracy while maintaining hardware costs comparable to, or even lower than, those of traditional approximate logarithmic multipliers.

3.1. The Architecture of the Design

The fundamental operations in neural networks (NNs) involve multiplications, specifically the multiplication of weights w by activations x . The architecture for an NN utilizing the proposed most significant one-driven shifting approximate multiplier (MSAM) is illustrated in Figure 1. The MSAM primarily comprises two core components: an approximate sub-multiplier (ASM) and an exact sub-multiplier (ESM), enhanced with auxiliary shifters and adders for a straightforward and coherent structural design. Specifically, the weight w is divided into two parts based on the significance of its bits: the most significant bits form w_H , processed by the ESM, while the less significant bits are manipulated to form w'_L , processed by the ASM. Subsequently, the result from the ESM is shifted, and an addition operation is performed to generate the final result r .

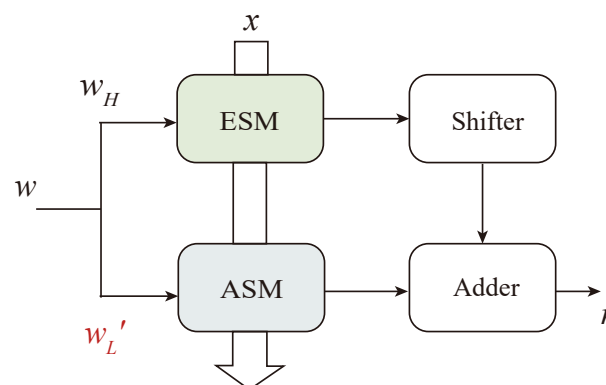


Figure 1. The architecture of the most significant one-driven shifting approximate multiplier (MSAM) block design.

A sign converter similar to that in [15] is employed to extend the MSAM to support signed numbers, as illustrated in the left part of Figure 2. The internal feature of the MSAM block is also presented in Figure 2. We introduce two pivotal parameter factors that confer the attribute of dynamic configurability to the multiplier, catering to a diverse

range of precision requirements that might fluctuate across various application contexts. The first is the approximation factor, denoted as k , which is utilized for the partitioning of the multiplier. This partitioning operation separates the n -bit multiplier into two distinct segments: k lower-order bits and $(n - k)$ higher-order bits. This partitioning scheme effectively decomposes a solitary multiplication operation into two discrete multiplicative processes involving the multiplicand and two multipliers with smaller bit widths. Second, the precision factor, denoted as m , is incorporated within the approximation strategy governing the ASM, and plays a crucial role in facilitating configurable precision settings within the scope of the sub-multiplier.

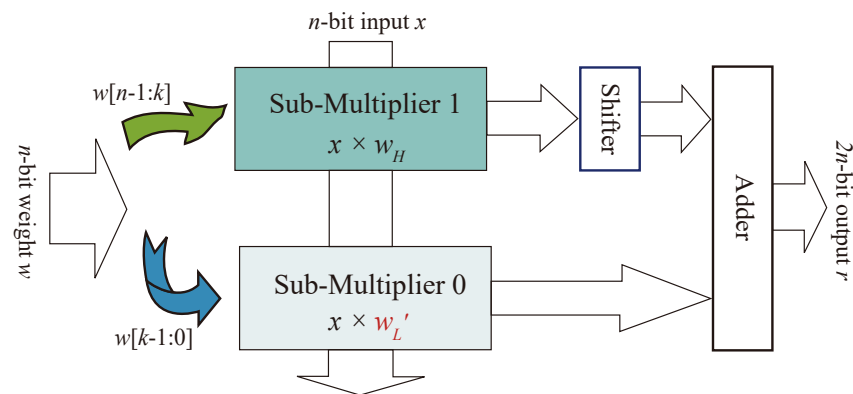


Figure 2. Extension of the proposed approach to support signed numbers, including internal feature of MSAM block.

3.2. The Proposed Dynamic Weight Approximation Strategies

To achieve a balance between computational efficiency and accuracy in the MSAM design, we introduce multiple weight approximation strategies tailored for the ASM component, based on the fundamental principles of the logarithmic multiplier (LM) and re-configurable constant coefficient multiplier (RCCM) design. These strategies aim to accommodate various application requirements and optimize performance under different conditions, encompassing the transformation of multiplication operations into simpler arithmetic operations such as shifts, addition, or subtraction. Meanwhile, the ESM component follows conventional, precise multiplication techniques to ensure accurate computation. To ensure accurate cumulative addition, the output generated by the ESM undergoes a sequence of shifting operations to align it with the output from the ASM. This alignment is crucial for the cumulative addition process, which ultimately determines the final output of the MSAM block. By combining these distinct strategies in the ASM and ESM components, we aim to optimize the overall performance of the MSAM design.

The activation value x serves as the multiplicand, while the weight w assumes the role of the multiplier in the multiplication operation within a neural network (NN). To optimize the approximate sub-multiplier (ASM) for neural networks, we propose a method that partitions the model's weights into distinct data ranges and approximates the weights within these ranges as constant values. This collection of approximate constant values, denoted as W' , effectively corresponds to the constant coefficient set C employed in the RCCM framework [12]. A fundamental difference is that the proposed set W' requires the determination of only a sole initial constant, w'_0 , while the remaining constants in the set are deduced with respect to this initial constant. Furthermore, this innovative weight approximation strategy enables the dynamic selection of appropriate approximate constant values based on the most significant one (MSO) in the input weight data.

3.2.1. One-Dominating Approximation Strategy (OAS)

The process of single-multiplier (w) approximation using the proposed method is depicted in Figure 3. Leading-one detection (LOD) is utilized to detect the most significant one (MSO) of the multiplier w . Once the MSO is identified, the subsequent $(m - 1)$ bits

are set to 1, while the remaining bits are set to 0. If the position of the MSO is at the mso -th bit (counting from 0, where mso is the index of the MSO bit), all numbers within the range of $w \in [2^{mso}, 2^{mso+1} - 1]$ are approximated as a constant value of one, according to Equation (3).

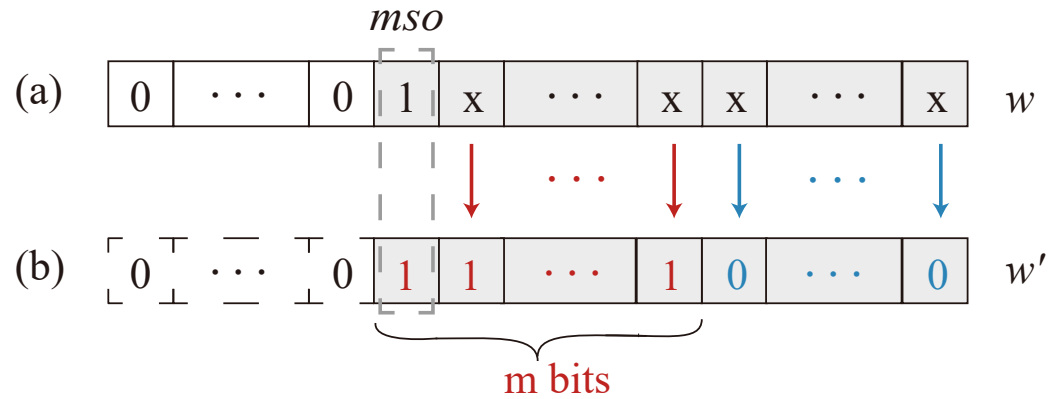


Figure 3. Example of the multiplier one-dominating approximation process: (a) original number; (b) approximated number.

$$w' = \begin{cases} w'_0, & mso < m \\ \sum_{i=mso-m+1}^{mso} 2^i, & mso \geq m. \end{cases} \quad (3)$$

The value of w'_0 is determined by the factor m . For instance, when $m = 1$, w'_0 is set to $1b$; meanwhile, when $m = 2$, w'_0 is set to $11b$. In this approximation method, the multiplier is dynamically split into separate data range intervals based on the position of the MSO in the input data. These intervals are then approximated using a set of constant coefficients. We refer to m as the precision factor, as it controls the number of approximation intervals.

The selection of m consecutive bits starting from the MSO and setting them to 1 is based on two principal rationales. First, this enables the establishment of a connection between this particular group of approximate values and the initial approximate constant coefficient w'_0 through a succession of shifting operations. As a result, through offering an initial approximate constant w'_0 and relying on the product of x and w'_0 , we can effectively reconstitute the multiplication outputs of the remaining coefficients in relation to the input x . This method significantly streamlines the hardware implementation and lowers complexity. Second, a sensible choice of the value of m allows for the strategic positioning of the approximate constants near the average value within each weight interval, thus maximizing the precision of the multiplier. In essence, it allows the approximated constants to gravitate toward the central value of the individual weight range, thereby enhancing the overall accuracy of the multiplier architecture.

3.2.2. Zero-Dominating Approximation Strategy (ZAS)

In the above one-dominating approximation strategy, all m bits starting from the MSO position are set to 1 for an important reason: this approximated m -bit number should closely resemble the average value of all k -bit numbers. However, in the extreme case where $m = k$, the approximated number becomes the largest, rather than approximating the mean value. To resolve this dilemma, we propose an alternative zero-dominating approximation strategy, as illustrated in Figure 4. Immediately after the position of the MSO, not all the remaining bits are set to 1. Instead, the first bit immediately following the MSO position is set to 0, with the rest of those bits being set to 1. This effectively avoids the issue of excessively large approximate numbers while ensuring proximity to the mean value. With a 0 acting as a separator, this also allows for reasonable application of carry discard techniques [10], thereby reducing latency and power consumption.

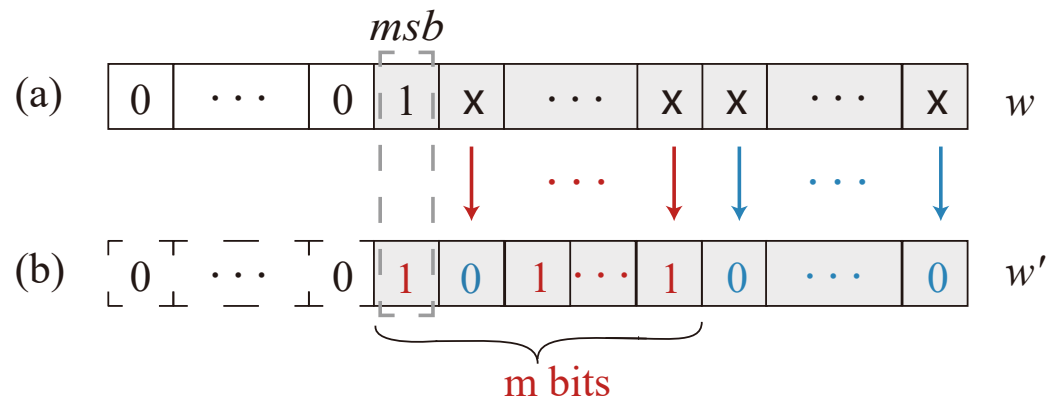


Figure 4. Example of the multiplier zero-dominating approximation process: (a) original number and (b) approximated number.

3.2.3. Compensation for Zero-Dominating Approximation Strategy

To better leverage the advantages of approximation, we typically do not choose $m = k$. Empirically, setting $m \leq \log_2(k) + 1$ tends to provide a favorable balance between energy consumption and accuracy benefits.

As shown in Table 1, in terms of the MSAM-42 approach (i.e., $k = 4$ and $m = 2$), the average result obtained with the OAS is slightly larger than the median, while the average result with the ZAS is significantly smaller than the median.

Table 1. Results obtained with MSAM-42¹ using various approximation strategies.

The k Bits (Input $w[3:0]$)	One-Dominating Approximation	Zero-Dominating Approximation	Compensation for Zero-Dominating
0000	0000	0000	0000
0001			
0010	0011	0010	0101
0011			
0100			
0101	0110	0100	0101
0110			
0111			
1001			
1010			
1011			
1100	1100	1000	1010
1101			
1110			
1111			

¹ MSAM-42 indicates that the approximate factor k is 4 and the corresponding precision factor m is 2.

Although this can better curtail energy consumption, it may also bring excessive errors. To achieve a better equilibrium in this regard, we introduce a compensation scheme to mitigate errors and guarantee that the overall accuracy is not notably compromised. As depicted in Figure 5, considering that the approximated number with ZOS is likely smaller than the actual one, through adding an extra compensation of 1 after the m -bit, we can effectively reduce the error distance while the additional energy consumption is essentially negligible.

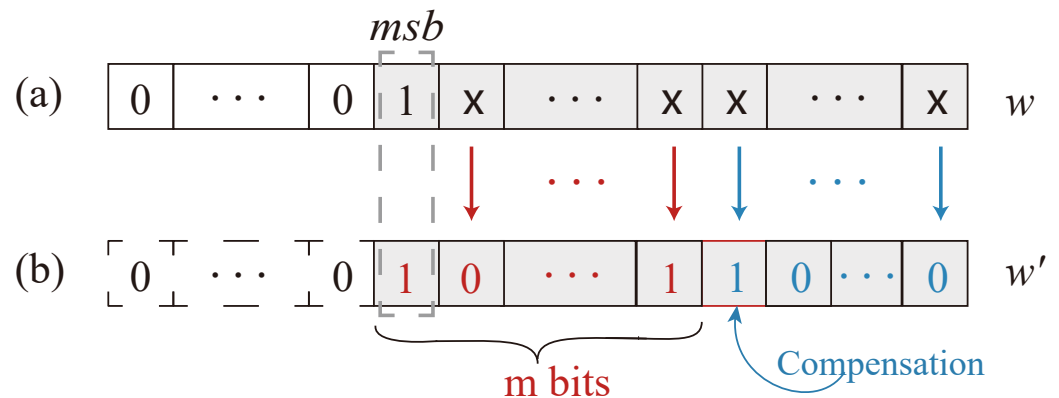


Figure 5. Example of the multiplier zero-dominating approximation with compensation process: (a) original number and (b) approximated number.

3.3. Implementation of the Hardware Design

The core of the MSAM design's excellence lies in its attentive management of the ASM, as refining its hardware directly and significantly affects the overall energy consumption of the multiplication unit.

As illustrated in Figure 6, the process commences with a series of shift and addition operations, jointly generating the product of the multiplicand x and the specific substitution value w'_0 for w_L in accordance with the factors k and m . The position (pos) of the most significant one bit within the input weight w_L , identified by the leading-one detector (LOD), establishes the control signal for the subsequent multiplexer (MUX). This signal governs the shift in $r0_{tmp}$. Subsequently, the multiplexer is utilized to generate the output $r0$ for the approximate sub-multiplier (ASM) module. Meanwhile, the ESM features an accurate approach to multiplication. Eventually, the output from the ESM undergoes a succession of shifting operations and is then merged with the outcome from the ASM, ultimately yielding the final output of the MSAM.

To accommodate signed operations, the MSAM architecture can be augmented through integrating sign conversion circuits both before and after unsigned operations, as depicted in Figure 2. These circuits employ the complement and execute an OR operation with a logic 1b in the least significant bit, serving as an approximation for negation when dealing with negative inputs.

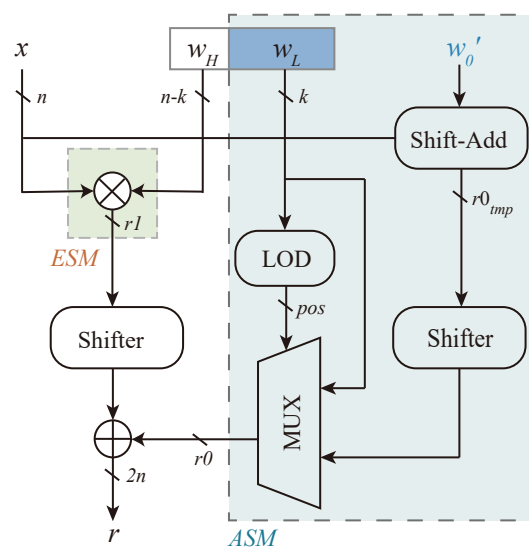


Figure 6. Hardware design implementation of MSAM.

4. Results and Discussion

Our proposed MSAM design is implemented using Verilog HDL, along with other advanced approximate multipliers. The designs were synthesized using the Synopsys Design Compiler (DC) with 28 nm CMOS technology, and the simulation was conducted using approximately 1,000,000 sets of random numbers, covering the entire input space of 8×8 multipliers (65,536 combinations) at a clock frequency of 200 MHz. The VCS tool was utilized for the simulation, generating back-annotated switching activity files to facilitate power analysis. The NMED metric was employed to quantify the error of the proposed designs, which are comprehensively evaluated across the entire input space using uniformly distributed data. The proposed unsigned MSAMs were compared with Mitch-w [14], DR-ALM-7 [15], DRUM6 [16], and AxRM2 [17], while the signed MSAMs were compared with Mitch-w [14], DR-ALM-7 [15], C-SSM ($m = 4$) [18], and Approx-Booth [19]. For clear and convenient comparisons, MSAM refers to the use of the OSA approximation strategy, MSAM^Z indicates the use of the ZSA approximation strategy, and MSAM^C denotes ZSA with additional compensation.

The abbreviations used in this manuscript are presented in Table 2.

Table 2. Abbreviations for approximate multiplier designs.

Multiplier	Techniques	Description
Exact	-	Exact multiplier.
MSAM- km	Logarithmic-like	The proposed most significant one-driven shifting approximate multiplier under varying approximate factors k and precision factors m .
Mitch-w [14]	Logarithmic	Mitchell's unsigned approximate logarithmic multiplier with w -bit operand.
Mitch-w* [14]	Logarithmic	Mitchell's signed approximate logarithmic multiplier with w -bit operand.
DR-ALM-7 [15]	Logarithmic	Approximate non-iterative unsigned logarithmic multiplier with L_i -bit operand ($L_i = 7$).
DR-ALM-7* [15]	Logarithmic	Approximate non-iterative signed logarithmic multiplier with L_i -bit operand ($L_i = 7$).
DRUM6 [16]	Logarithmic	Unbiased multiplier with $k(k = 6)$ bits in the operate from the leading one.
AxRM2 [17]	Non-logarithmic	Approximate recursive multipliers using high-performance building blocks.
C-SSM ($m = 4$) [18]	Non-logarithmic	Approximate static segmented multiplier with a contiguous segment of m bits ($m = 4$).
Approx-Booth [19]	Non-logarithmic	Low-cost approximate radix-4 Booth multiplier with simplified compressor and encoder designs.

4.1. Error Analysis

Deep learning accelerators that incorporate approximate multipliers and adders have been devised to mitigate energy consumption and delay, albeit with an acceptable trade-off in accuracy [20]. As a general principle, a higher degree of acceleration tends to result in a greater reduction in accuracy. The error characteristics of an approximate multiplier (AM) are typically ascertained through exhaustive simulations that encompass all conceivable input combinations or, alternatively, through Monte Carlo simulations that utilize random inputs sampled from a pre-determined distribution, such as the normal or uniform distribution.

Consider that $\mathcal{M}^*$ denotes the output of the approximate multiplier and $\mathcal{M}$ represents the exact output. The error distance (ED) provides a precise measurement of the absolute difference between these two outputs, which is formulated as $ED = |\mathcal{M} - \mathcal{M}^*|$. The relative error distance (RED) expresses the error as a percentage relative to the exact result. Furthermore, the mean error distance (MED) and the mean relative error distance (MRED) are widely used metrics for evaluating the accuracy of approximate designs. In addition

to MED and MRED, the normalized MED (NMED) and normalized RED (NRED) are stable metrics that are independent of the implementation size, making them valuable for assessing approximate designs of varying sizes. As demonstrated in [21], the NMED of an approximate multiplier (AM) has a significant impact on the output error of a neuron in a neural network, particularly when dealing with a large number of inputs.

For the specified MSAM, with the value of k remaining constant and ensuring the consistency of the operand bit-width for all operations, we employ MRED and NMED, as two effective and widely used metrics, in order to evaluate the error introduced by the ASM block. These metrics serve as indispensable means for the assessment of the errors inherent in the proposed design, with a focus on different configurations of the precision factor m . Figure 7 presents the MRED curves for k values of 3, 4, and 5, plotted against variations in the precision factor m , enabling the identification of distinct trends within the curves. For different values of k , the MRED reaches its minimum when m is 1 or 2. Contrary to other AMs, in the design of the MSAM, the precision factor m is not necessarily better when it is larger. In fact, when m exceeds $\log_2 k + 1$, an excessive number of bits are set to 1, leading to a significant deviation in the approximated value from the actual value. Additionally, a larger m results in more bits participating in the calculation, thereby consuming more energy. It is worth noting that the results obtained from the NMED metric closely align with these experimental findings. Consequently, in subsequent comparative analysis and detailed examinations, we prioritized m values of 1 or 2. This strategic choice was made with the aim of attaining an optimum balance in the design, effectively reconciling the precision requirements and overhead hardware cost considerations.

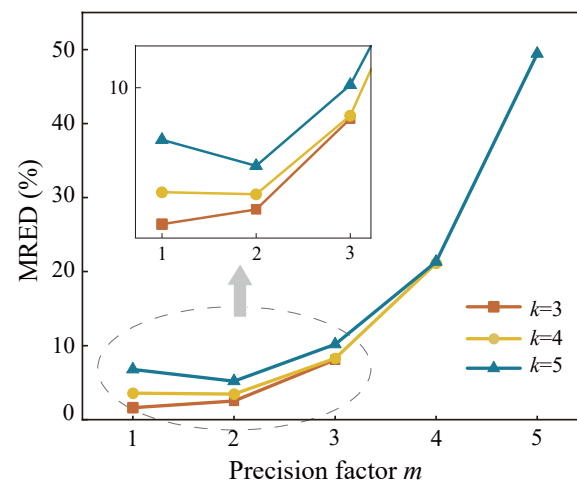


Figure 7. Mean relative error distance curves of MSAM under varying approximate factors k and precision factors m .

4.2. Synthesis and Simulation Results

Diverse MSAM- km multipliers, existing logarithmic multipliers, and advanced non-logarithmic multipliers were evaluated to assess their hardware performance and accuracy metrics.

In our approximate multiplier design, one operand is divided into two components: ESM and ASM, controlled by the approximation factor k , which determines the number of bits approximated. The precision factor m represents the degree of approximation. The level of approximation is primarily governed by k ; empirically, a larger k leads to a higher degree of approximation, simpler design, and lower energy consumption, while a smaller k results in less approximation and reduced energy savings. The parameter m decides how the k approximate bits are handled. A larger m retains more bits, but it is important to note that this does not mean retaining the original precise bits; instead, it uses m bits of all ones. Therefore, a larger m does not necessarily translate to higher precision. Statistically, an m closer to k tends to make k approximate bits all ones, leading to less precision. From

experimental results and statistical analysis, a reasonable value for m is around $\log_2(k)$, making m closer to the mean of all k -bit numbers. The introduction of parameters k and m enables the fine-tuning of the design to achieve the desired balance between accuracy and energy efficiency. However, these additional parameters might require meticulous consideration and optimization during the design phase to ensure optimal performance.

Logarithmic multipliers offer lower energy consumption but sacrifice accuracy. Our design mitigates this by providing a higher accuracy level while maintaining a lower power consumption than non-logarithmic multipliers. A comparison of NMED and the power delay product (PDP) for unsigned 8-bit multipliers is shown in Figure 8, where the proposed MSAMs are highlighted with a red square. While DR-ALM-7 [15] and Mitchell [14] exhibited lower PDP, they resulted in higher NMED. Conversely, DRUM6 [16] and MSAM-21 produced smaller NMED but demanded higher PDP. Notably, AxRM2 [17], MSAM-52, MSAM-41, and MSAM-42 achieved relatively lower PDP as well as smaller NMED. Among these, as indicated by the red circle, MSAM-41 and MSAM-42 emerge as the most appealing designs, achieving the optimal trade-off between PDP and NMED.

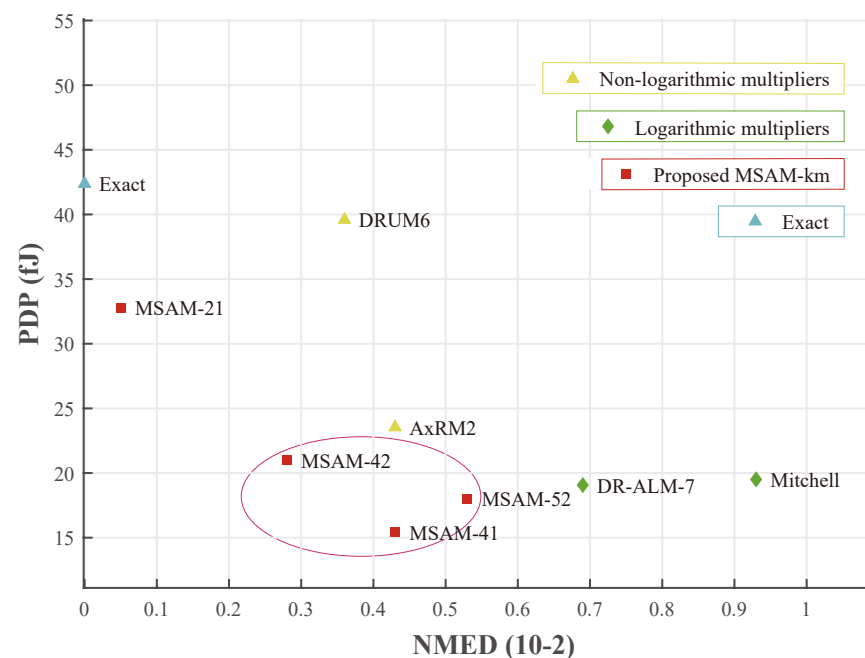


Figure 8. NMEDs and PDPs of 8-bit unsigned approximate multipliers.

The detailed results of this evaluation are summarized in Table 3. Through carefully configuring the parameters k and m , we attained a hardware performance comparable to (or, in some cases, exceeding) that of the Mitchell [14] and DR-ALM-7 [15] multipliers (for instance, with $km = 41$ or $km = 52$). When comparing MSAM-41 and MSAM-52 with DR-ALM-7, even in situations where the power delay product (PDP) performance surpasses the latter, remarkable improvements were still witnessed in terms of NMED, reaching 37.68% and 23.19%, respectively. In contrast, when compared to AxRM2 [17], MSAM-41 demonstrates significant enhancements, including a 32.11% reduction in area utilization, a 23.33% decrease in delay, and a 27.95% improvement in PDP, all while maintaining an equivalent NMED. These findings highlight the impressive hardware efficiency achieved with the MSAM design, when compared to state-of-the-art AM designs.

Although the ZAS offers a more simplified hardware design, resulting in superior energy consumption and area utilization, the NMED is increased compared to the corresponding OAS strategy. Our design provides multiple approximate strategies such as ZAS and OAS to achieve a more dynamic range of trade-offs, thereby meeting the demands of a wider range of applications.

Table 3. Performance comparison of 8-bit unsigned approximate multipliers.

Design	Area (um ²)	Power (uW)	Delay (ns)	PDP (fJ)	NMED (10 ^{−2})
Exact	154.48	76.75	0.59	45.28	0
MSAM-21	147.29	59.55	0.59	35.13 (22%)	0.05
MSAM ^Z -21	144.22	56.2	0.58	32.60 (28%)	0.07
MSAM-32	143.01	52.3	0.52	27.20 (40%)	0.17
MSAM ^Z -32	139.05	50.4	0.5	25.20 (44%)	0.19
MSAM-31	129.53	48.39	0.52	25.16 (44%)	0.17
MSAM ^Z -31	126.45	44.66	0.49	21.88 (52%)	0.19
MSAM-42	125.5	41.78	0.52	21.73 (52%)	0.28
MSAM ^Z -42	120.79	38.98	0.49	19.10 (58%)	0.34
MSAM-41	111.89	36.86	0.46	16.96 (63%)	0.43
MSAM ^Z -41	103.76	35.33	0.45	15.90 (65%)	0.47
MSAM-52	110.75	32.36	0.52	16.83 (63%)	0.53
MSAM ^Z -52	104.39	31.33	0.51	15.98 (65%)	0.59
Mitchell [14]	103.45	30.01	0.65	19.51 (−57%)	0.93
DR-ALM-7 [15]	98.28	28.46	0.67	19.07 (−58%)	0.69
DRUM6 [16]	192.53	56.10	0.77	43.20 (−5%)	0.36
AxRM2 [17]	164.81	39.23	0.60	23.54 (−48%)	0.43

The proposed designs are in the shaded section.

4.3. Evaluation, Simulation, and Analysis

4.3.1. Evaluation of Performance on Deep Neural Networks

For a more comprehensive evaluation of the multiplier’s performance when used in large models, we conducted experiments using the CIFAR-10 data set and employed three representative deep neural networks: VGG-19, known for its deep convolutional layers; ResNet-50, which effectively mitigates the vanishing gradient problem in deep networks through the use of residual connections; and DenseNet-121, which achieves efficient feature utilization through dense connectivity mechanisms. We conducted experiments using these complex networks, aiming to thoroughly assess the multiplier’s performance and applicability. Additionally, we expanded our experimental analysis to the more challenging ImageNet data set through incorporating SqueezeNet, a compact yet efficient deep neural network architecture designed for optimized performance on constrained devices.

In order to assess the performance of the approximate multipliers, we employed the Adapt framework [22] to facilitate the simulation and evaluation of various multiplier designs. As shown in Figure 9, the inputs consist of the pretrained NN model, quantization configurations, and the design file of the approximate multiplier (based on Verilog HDL). Initially, the layer graph is constructed using the parameters of the convolutional layer and the adjacent batch normalization layer. Subsequently, the design file of the approximate multiplier was analyzed. Based on the quantization configurations, the corresponding approximate layers are generated. These approximate layers then replace the layers in the original layer graph. Following this, the reconstructed layer graph is processed through the AdaPT framework for retraining and inference, adhering to the specified retraining strategy. It is noteworthy that for the training data set, only a representative subset is required, which can be approximately 10% of the original training set, for calibration purposes. Lastly, performance reports are provided, detailing the accuracy and energy consumption.

Model retraining and fine-tuning can be employed to alleviate the accuracy deterioration arising from certain approximations, such as pruning and approximate multiplication [23]. These approaches enable the model to adapt and become more resistant to the errors introduced by such approximations. During the inference phase, these techniques facilitate the AxC models to incur lower hardware cost while attaining significantly higher accuracy. To evaluate the performance of the approximate multipliers effectively, experiments were carried out on the CIFAR-10 data set through direct inference without any

retraining. This methodology permitted a rapid assessment of the multipliers' performance. Meanwhile, for the ImageNet data set, a more efficient retraining strategy was adopted by selecting only 10% of the training subset. This strategy was aimed at performing model tuning during the training phase, enabling accuracy improvement and substantial energy reduction during the actual inference phase. It showcases the efficacy of retraining and model tuning in mitigating errors and enhancing accuracy without the encumbrance of expensive hardware requirements.

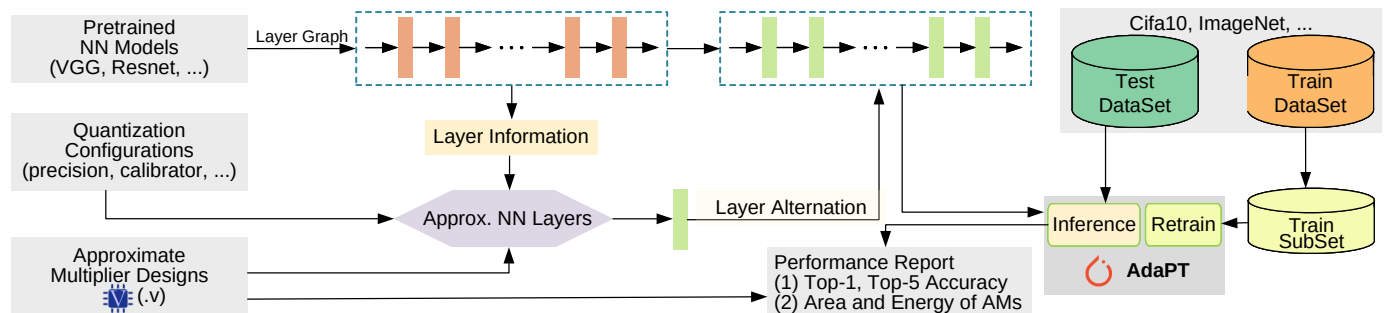


Figure 9. Evaluation flow based on AdaPT framework.

Several signed versions of the MSAM multiplier were implemented and thoroughly compared with the precise radix-4 Booth multiplier, alongside two logarithmic multipliers—namely, Mitchell [14] and DR-ALM-7 [15]—as well as two non-logarithmic approximate multipliers—C-SSM [18] and Approx-Booth [19]. To ensure a fair comparison, we replicated the designs of the 8-bit C-SSM (with $m = 4$) and Approx-Booth (with $w_G, w_A = 6, 6$). The evaluation of these signed multipliers strictly followed the previously outlined experimental setup. The results of this comprehensive comparison are presented in Table 4.

When considering the results on the CIFAR-10 data set, the MSAM approximate multiplier presented the best performance in terms of both accuracy and energy reduction. Specifically, the MSAM-31 multiplier not only outperformed the DR-ALM-7 in terms of network evaluation accuracy, but also achieved an additional 9% energy savings per single multiplier compared to DR-ALM-7, highlighting its efficiency. DR-ALM-7, on the other hand, demonstrated superior network performance when compared to the other approximate multipliers, with the exception of the MSAM variants.

Table 4. Comparison of the hardware costs and accuracies of 8-bit signed multipliers on the CIFAR10 and ImageNet data sets.

Multiplier	NMED (10^{-2})	Area (μm^2)	Power (μW)	Delay (ns)	Energy (fJ)	CIFAR10			ImageNet	
						VGG-19	ResNet-50	DenseNet-121	Top-1	Top-5
Exact radix-4	0	276.70	94.27	0.60	56.56	93.69%	93.57%	93.84%	57.37%	80.19%
Signed MSAM-21	0.34	173.88	52.82	0.62	32.75 (−42%)	93.66%	92.81%	93.58%	54.96%	78.21%
Signed MSAM-32	0.40	169.60	47.78	0.58	27.71 (−51%)	93.23%	91.34%	93.30%	52.02%	76.00%
Signed MSAM-31	0.61	156.11	42.82	0.55	23.55 (−58%)	93.60%	93.06%	93.64%	54.01%	77.60%
Signed MSAM-42	0.60	152.08	39.71	0.60	23.83 (−58%)	93.08%	91.71%	92.93%	50.37%	74.35%
Signed MSAM-41	1.20	138.47	34.10	0.49	16.71 (−70%)	91.24%	91.26%	91.46%	43.31%	68.00%
Mitchell* [14]	1.12	132.30	39.37	0.71	27.95 (−51%)	89.64%	92.52%	91.75%	50.19%	74.20%
DR-ALM-7* [15]	0.78	127.13	39.23	0.73	28.64 (−49%)	93.29%	92.24%	93.37%	53.92%	77.38%
C-SSM ($m = 4$) [18]	3.62	276.32	82.37	0.43	35.42 (−37%)	40.24%	61.17%	73.28%	2.22%	7.47%
Approx-Booth [19]	0.17	218.23	73.18	0.55	40.25 (−29%)	12.29%	10.65%	9.99%	17.44%	36.33%

The top three approximate multipliers that demonstrated the greatest adaptability to the ImageNet data set were MSAM-21, MSAM-31, and DR-ALM-7. These multipliers exhibited exceptional performance across the various network architectures, highlighting their versatility and robustness. For applications that prioritize high network accuracy,

MSAM-21 stood out as a notable performer, which achieved a 1.04% improvement in Top-1 accuracy and a 0.83% improvement in Top-5 accuracy, when compared to DR-ALM-7. These improvements are significant and can have a substantial impact on the overall performance of the network. The results presented in Table 4 provide a comprehensive comparison of the hardware costs and accuracies of the 8-bit signed multipliers on the CIFAR10 and ImageNet data sets. The MSAM and DR-ALM-7 multipliers consistently demonstrated strong performance, with the MSAM variants showing particular promise in terms of accuracy and energy efficiency. These findings have important implications for the design and optimization of multipliers for use in deep learning applications.

4.3.2. Performance Evaluation of Transformer Matrix Multiplication Accelerator

The resource and energy savings achieved with the proposed multiplier when deployed in LLMs on edge devices were verified. This verification was accomplished through the application of the multiplier to a transformer matrix multiplication accelerator, specifically designed to handle matrices of size 16×16 . When evaluating the hardware overhead, with exact multipliers serving as the baseline for comparison, we focused on assessing the power consumption of the accelerator associated with combinational logic. Ref. [24] highlights the possibility of performing inference in LLMs with up to 175B parameters without any performance degradation utilizing LLM.int8(), which enables the multiplication of over 99.9% of values in 8-bit. Ref. [25] demonstrates the efficiency of approximate multipliers in the context of vision transformer models. As depicted in Figure 10, our matrix multiplication accelerator involves two 8-bit integer matrices, X_{18} and W_{18} . To accelerate the multiplication process, we substituted the exact multiplication with an approximated method.

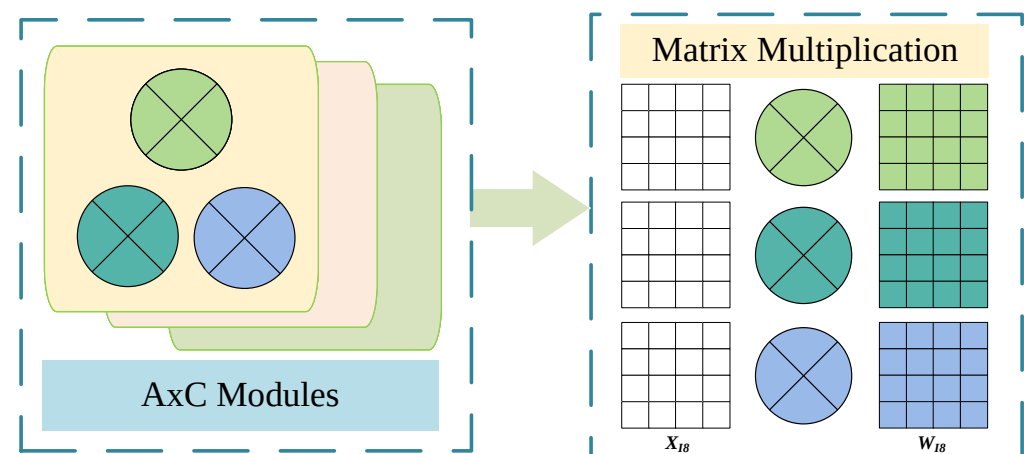


Figure 10. Matrix multiplication accelerator.

Table 5 presents the experimental results of the matrix multiplication module in the Transformer accelerator, highlighting the effective reduction in hardware overhead achieved with the proposed multiplier. The results demonstrate the multiplier's ability to optimize resource and energy usage when deploying LLMs on edge devices. Compared to the baseline result obtained using the exact multiplier, the MSAMs exhibit significant reductions in area, power consumption, and latency, with MSAM^Z-41 showing the most pronounced improvements. The employment of approximate computing has demonstrated the possibility of improving the efficiency of transformers by minimizing their computational complexity and memory demands [25]. Considering that large models have higher computational requirements, this suggests that more approximate modules can be included. Therefore, our future research should continue to explore and delve into a more appropriate and rational integration of these MSAMs, aiming to lower energy consumption while maintaining acceptable accuracy.

Table 5. Hardware performance of the proposed design with the transformer accelerator.

	Area (μm^2)	Comb. Power (mW)	Delay (ns)
Baseline	90,391.64	2.07	0.89
MSAM-41	67,618.91 (−25.19%)	1.82 (−12.08%)	0.79 (−11.24%)
MSAM ^Z -41	47,618.35 (−47.32%)	1.46 (−29.47%)	0.75 (−15.73%)
MSAM ^C -41	51,618.67 (−42.89%)	1.55 (−25.12%)	0.76 (−14.61%)

4.4. Case Study: Image Matrix Multiplication Accelerator

In our approach, we conduct a pixel-by-pixel matrix multiplication between an input image and a mask image, where the value of the corresponding pixel in the output image is determined according to the resulting product. Figure 11 shows the image multiplication results, where the image quality is positively correlated with the peak signal-to-noise ratio (PSNR); as such, the PSNR values for the generated images are included in their respective subtitles. The results obtained with most of our MSAMs indicate high quality, demonstrating a very good PSNR when compared to the exact multiplier in Figure 11c. Due to the relatively uniform distribution of the image data, as our design incorporates ESM, it has high utility and robustness in the context of image processing.

**Figure 11.** The results of multiplying (a) an input image with (b) a mask image using the selected multipliers.

4.5. Discussion on the Proposed MSAMs

4.5.1. Advantages and Limitations

The proposed MSAMs offer considerable energy savings while constraining the errors of approximate multipliers through the employment of most significant bit-guided approximation strategies. These strategies provide several primary advantages over other works. Firstly, MSAMs employ most significant bit-guided approximation techniques, facilitating multiplication to be carried out via shifting. This results in significant decreases in both area and energy consumption, making our approach highly efficient for neural network applications. Secondly, unlike logarithmic approaches, our approach guarantees that the approximate output is not one-sidedly biased. This feature enables compensation among different approximations, leading to enhanced overall accuracy and reliability. Furthermore, a key benefit of our approximation strategy is that it focuses on only one operand in the multiplication operation while fully preserving the precision of the other operand. This distinctive approach effectively ensures precision, providing a significant superiority over most other works in the field. The precise control over the approximation strategies and the ability to preserve precision in one operand contribute to the improved performance in neural network applications.

One major limitation of the proposed MSAMs is the relatively high error probability caused by the usage of pre-defined components to replace specific bits. This would render our multipliers as not applicable in scenarios where demanding precision is needed. Despite this limitation, the proposed MSAMs possess considerable practical worth, as the scale of the errors is not substantial, and given that the errors can be both positive and negative, the overall error is acceptable for batch operations.

4.5.2. Future Work

Matrix multiplication serves as the fundamental building element in the transformer architecture [25]. In the future, we aim to further explore the design of approximate multipliers with concentrated errors and establish an approximate multiplier library adapted for LLMs. Software-level approximations, like lower-complexity networks and pruning, facilitate the energy-efficient acceleration of DNNs [23]. While both software and hardware approximation techniques independently offer substantial energy benefits for a DNN, we intend to combine these techniques into a synergistic framework in future endeavors to maximize energy savings. We hope that this article will attract more researchers to conduct research in this aspect, allowing the application of MSAMs in more complex scenarios with tight energy budgets and strict accuracy requirements.

5. Conclusions

In this study, we proposed an energy-efficient most significant bit-guided dynamic approximate multiplier for neural network acceleration. Using a weight approximation strategy tailored for NNs and LLMs ultimately resulted in a high-efficiency approximate multiplier grounded in MSO shifting principles. The resultant MSAM design adeptly achieves a harmonious balance between precision and hardware cost. When applied to both LLMs and DNNs, our approach led to substantial reductions in hardware resource overheads. In comparison to the traditional exact multipliers used in LLMs and DNNs, our design achieved a remarkable 60% reduction in energy consumption without any significant adverse impacts on task accuracy. Furthermore, we conducted a case study on image merging, and the results demonstrate that our multipliers can be used to generate high-quality images. Compared with the exact multiplier, our approach performed very well when using the PSNR as an evaluation metric.

Author Contributions: Conceptualization, P.H.; Software, B.G.; Validation, B.G.; Writing—original draft, P.H.; Writing—review & editing, K.C.; Supervision, C.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by grants from the National Natural Science Foundation of China (62101252, and 62134002).

Data Availability Statement: Upon reasonable request, the corresponding author can provide the data supporting the findings of this study.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Raiaan, M.A.K.; Mukta, M.S.H.; Fatema, K.; Fahad, N.M.; Sakib, S.; Mim, M.M.J.; Ahmad, J.; Ali, M.E.; Azam, S. A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges. *IEEE Access* **2024**, *12*, 26839–26874. [\[CrossRef\]](#)
2. Rasch, M.J.; Mackin, C.; Gallo, M.L.; Chen, A.; Fasoli, A.; Odermatt, F.; Li, N.; Nandakumar, S.R.; Narayanan, P.; Tsai, H.; et al. Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators. *Nat. Commun.* **2023**, *14*, 5282. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Towhid, A.; Omid, R.; Mohammadi, K. On the Design of Iterative Approximate Floating-Point Multipliers. *IEEE Trans. Comput.* **2023**, *72*, 1623–1635. [\[CrossRef\]](#)
4. Zhang, H.; Ko, S.B. Efficient Approximate Posit Multipliers for Deep Learning Computation. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2023**, *13*, 201–211. [\[CrossRef\]](#)
5. Sayadi, L.; Timarchi, S.; Sheikh-Akbari, A. Two Efficient Approximate Unsigned Multipliers by Developing New Configuration for Approximate 4:2 Compressors. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2023**, *70*, 1649–1659. [\[CrossRef\]](#)
6. Schaefer, C.J.; Taheri, P.; Horeni, M.; Joshi, S. The Hardware Impact of Quantization and Pruning for Weights in Spiking Neural Networks. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *70*, 1789–1793. [\[CrossRef\]](#)
7. Pinos, M.; Mrazek, V.; Vaverka, F.; Vasicek, Z.; Sekanina, L. Acceleration Techniques for Automated Design of Approximate Convolutional Neural Networks. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2023**, *13*, 212–224. [\[CrossRef\]](#)
8. Ahmadinejad, M.; Moaiyeri, M.H. Energy- and Quality-Efficient Approximate Multipliers for Neural Network and Image Processing Applications. *IEEE Trans. Emerg. Top. Comput.* **2022**, *10*, 1105–1116. [\[CrossRef\]](#)
9. Li, Z.; Zheng, S.; Zhang, J.; Lu, Y.; Gao, J.; Tao, J.; Wang, L. Adaptable Approximate Multiplier Design Based on Input Distribution and Polarity. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2022**, *30*, 1813–1826. [\[CrossRef\]](#)
10. Amirafshar, N.; Baroughi, A.S.; Shahhoseini, H.S.; TaheriNejad, N. Carry Disregard Approximate Multipliers. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2023**, *70*, 4840–4853. [\[CrossRef\]](#)
11. Lotrič, U.; Pilipović, R.; Bulić, P. A Hybrid Radix-4 and Approximate Logarithmic Multiplier for Energy Efficient Image Processing. *Electronics* **2021**, *10*, 1175. [\[CrossRef\]](#)
12. Faraone, J.; Kumm, M.; Hardieck, M.; Zipf, P.; Liu, X.; Boland, D.; Leong, P.H. AddNet: Deep neural networks using FPGA-optimized multipliers. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *28*, 115–128. [\[CrossRef\]](#)
13. Gong, B.; Chen, K.; Huang, P.; Wu, B.; Liu, W. Most Significant One-Driven Shifting Dynamic Efficient Multipliers for Large Language Models. In Proceedings of the 2024 IEEE International Symposium on Circuits and Systems (ISCAS), Singapore, 19–22 May 2024; pp. 1–5.
14. Mitchell, J.N. Computer Multiplication and Division Using Binary Logarithms. *IRE Trans. Electron. Comput.* **1962**, *EC-11*, 512–517.
15. Yin, P.; Wang, C.; Waris, H.; Liu, W.; Han, Y.; Lombardi, F. Design and analysis of energy-efficient dynamic range approximate logarithmic multipliers for machine learning. *IEEE Trans. Sustain. Comput.* **2020**, *6*, 612–625. [\[CrossRef\]](#)
16. Hashemi, S.; Bahar, R.I.; Reda, S. DRUM: A dynamic range unbiased multiplier for approximate applications. In Proceedings of the 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, USA, 2–6 November 2015; pp. 418–425.
17. Waris, H.; Wang, C.; Xu, C.; Liu, W. AxRMs: Approximate recursive multipliers using high-performance building blocks. *IEEE Trans. Emerg. Top. Comput.* **2021**, *10*, 1229–1235. [\[CrossRef\]](#)
18. Strollo, A.G.M.; Napoli, E.; De Caro, D.; Petra, N.; Saggese, G.; Di Meo, G. Approximate multipliers using static segmentation: Error analysis and improvements. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2022**, *69*, 2449–2462. [\[CrossRef\]](#)
19. Park, G.; Kung, J.; Lee, Y. Simplified Compressor and Encoder Designs for Low-Cost Approximate Radix-4 Booth Multiplier. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *70*, 1154–1158. [\[CrossRef\]](#)
20. Jiang, H.; Santiago, F.J.H.; Mo, H.; Liu, L.; Han, J. Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications. *Proc. IEEE* **2020**, *108*, 2108–2135. [\[CrossRef\]](#)
21. Mo, H.; Wu, Y.; Jiang, H.; Ma, Z.; Lombardi, F.; Han, J.; Liu, L. Learning the Error Features of Approximate Multipliers for Neural Network Applications. *IEEE Trans. Comput.* **2024**, *73*, 842–856. [\[CrossRef\]](#)
22. Danopoulos, D.; Zervakis, G.; Siozios, K.; Soudris, D.; Henkel, J. AdaPT: Fast Emulation of Approximate DNN Accelerators in PyTorch. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2023**, *42*, 2074–2078. [\[CrossRef\]](#)
23. Sanyal, S.; Negi, S.; Raghunathan, A.; Roy, K. Approximate Computing for Machine Learning Workloads: A Circuits and Systems Perspective. In *Approximate Computing*; Liu, W., Lombardi, F., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 365–395.

24. Dettmers, T.; Lewis, M.; Belkada, Y.; Zettlemoyer, L. GPT3.int8(): 8-bit Matrix Multiplication for Transformers at Scale. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022.
25. Danopoulos, D.; Zervakis, G.; Soudris, D.; Henkel, J. TransAxx: Efficient Transformers with Approximate Computing. *arXiv* **2024**, arXiv:2402.07545.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.