Original scientific paper
https://doi.org/10.33180/InfMIDEM2026.204

Informacije MIDEM
Journal of Microelectronics, Electronic Components and Materials
Vol. 56, No. 2(2026), XX – XX

HYBRID APPROXIMATE ARITHMETIC CIRCUITS WITH MERITS BALANCE FOR CNN INFERENCE

Balasaraswathi Subramanian*, Padmapriya Subramanian**
Department of ECE, PSG Institute of Technology and Applied Research, Coimbatore, Tamil Nadu, India.
*Corresponding Author E-mail: balasaraswathi@psgitech.ac.in
** E-mail: padmapriya@psgitech.ac.in

Abstract – Multiply-Accumulate (MAC) units dominate computation and energy use in deep neural network accelerators, comprising over 98% of operations in modern convolutional workloads. Exploiting the error resilience of convolutional neural network inference, this paper proposes hybrid approximate MAC architectures. These selectively approximate key datapath modules—partial product generation, reduction, and accumulation—for balanced trade-offs in power, delay, area, and accuracy. A compact radix-8 Booth array arrangement is introduced that adapts a single-bit MSB sign/zero-extension technique to support both signed × signed and unsigned × signed operations, simplifying sign handling and reducing compressor requirements. Approximation is further applied by eliminating complex ±3X Booth terms, introducing energy-efficient approximate half adders, full adders and 4:2 compressors guided by rigorous error-metric analysis, and employing a hybrid accumulator that combines an exact Han–Carlson prefix adder in the MSBs with a lightweight Lower-OR Adder in the LSBs. Four configurable AxMAC variants are proposed to explore fine-grained accuracy–efficiency trade-offs. Synthesized in 45 nm CMOS technology, the proposed designs achieve up to 43.5% delay reduction, 73.6% area savings, and 46.5% power reduction compared to an exact radix-8 Booth MAC, while maintaining controlled error characteristics. CNN inference is evaluated on pre-trained INT8-quantized LeNet-5, AlexNet, VGG-13, ResNet-18, and GoogLeNet models across MNIST, CIFAR-10, and ImageNet, using selected-layer approximation without retraining or fine-tuning.

Keywords – Approximate computing, Low-power arithmetic circuits, Hybrid parallel prefix adder, Radix-8 Booth multiplier, Deep Neural Network inference

Accepted Article

HIBRIDNA ARITMETIČNA VEZJA S PRIBLIŽNIM IZRAČUNOM IN URAVNOTEŽENIMI PREDNOSTMI ZA SKLEPANJE V NEVRONSKIH MREŽAH (CNN)

Izvleček – Enote za množenje in akumulacijo (MAC) prevladujejo pri računalniških operacijah in porabi energije v pospeševalnikih globokih nevronskih mrež, saj predstavljajo več kot 98 % operacij v sodobnih konvolucijskih delovnih obremenitvah. Ta članek, ki izkorišča odpornost konvolucijskih nevronskih mrež na napake pri sklepanju, predlaga hibridne arhitekture približnega MAC-a. Te selektivno približujejo ključne module podatkovne poti – generiranje delnih produktov, redukcijo in akumulacijo – za uravnotežene kompromise med porabo energije, zakasnitvijo, površino in natančnostjo. Predstavljena je kompaktna razporeditev Boothovega niza z osembitno osnovo, ki prilagaja enobitno tehniko razširitve znaka/ničle MSB za podporo tako operacijam s × brez predznak kot tudi brez × s predznakom, s čimer poenostavlja obravnavo predznakaznaka in zmanjšuje zahteve glede kompresorja. Približevanje se dodatno uporabi z odpravo kompleksnih ±3X Boothovih členov, uvedbo energetsko učinkovitih približnih polovičnih seštevalnikov, polnih seštevalnikov in kompresorjev 4:2, ki temeljijo na strogi analizi meril napak, ter z uporabo hibridnega akumulatorja, ki združuje natančen seštevalnik s predpono Han–Carlson v MSB-jih z lahkim seštevalnikom Lower-OR

v LSB-jih. Predlagane so štiri nastavljive različice AxMAC za raziskovanje podrobnih kompromisov med natančnostjo in učinkovitostjo. Predlagane zasnove, sintetizirane v 45-nm CMOS-tehnologiji, v primerjavi z natančnim MAC-om Booth z osnovo 8 dosežejo do 43,5 % zmanjšanje zakasnitve, 73,6 % prihranek prostora in 46,5 % zmanjšanje porabe energije, hkrati pa ohranjajo nadzorovane značilnosti napak. Sklepanje v nevronskih mrežah (CNN) je ocenjeno na vnaprej usposobljenih modelih LeNet-5, AlexNet, VGG-13, ResNet-18 in GoogLeNet s kvantizacijo INT8 na podatkovnih naborih MNIST, CIFAR-10 in ImageNet, pri čemer se uporablja aproksimacija izbranih plasti brez ponovnega usposabljanja ali natančnega prilagajanja.

Ključne besede – Približno računanje, aritmetična vezja z nizko porabo energije, hibridni vzporedni predponski seštevalnik, Boothov množitelj s številsko osnovo 8, sklepanje v globokih nevronskih mrežah

I. INTRODUCTION

The rapid development of Artificial Intelligence (AI) and Machine Learning (ML) has positioned Deep Neural Networks (DNNs) as the foundation of modern intelligent systems, enabling breakthroughs in image classification, object detection, and speech recognition. These applications depend on extensive matrix multiplications distributed across multiple neural layers, demanding billions of Multiply-Accumulate (MAC) operations during both training and inference phases [1]. Consequently, matrix multiplication has emerged as the most time- and energy-intensive operation in AI computation, making the efficiency of MAC units a critical determinant of overall system performance. To meet this demand at scale, modern AI accelerators such as Google TPU v4 [2] employ large systolic arrays tightly optimized for dense General Matrix-Matrix Multiplication (GEMM), which dominates DNN workloads as illustrated in Figure 1. As summarized in Table 1, MAC operations constitute more than 98% of the reported computational utilization across the considered neural-network workloads [1], highlighting the importance of MAC-unit efficiency in DNN accelerator design.

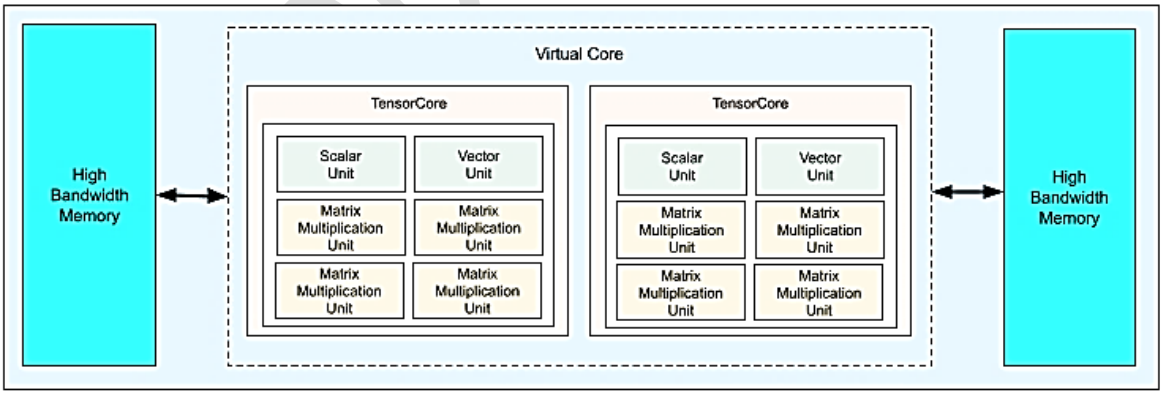


Figure 1. Google TPU v4 System Architecture [2]

Reducing hardware cost and power consumption while maintaining high throughput remains a key challenge in accelerator design. Approximate computing can exploit the error resilience of neural-network inference, where appropriately controlled computational inaccuracies may result in limited degradation in application-level accuracy [3], [4]. Techniques such as quantization and bit truncation further support efficient deployment. This work focuses on INT8 Convolutional Neural-Network (CNN) inference accelerators, where 8-bit operands and higher-precision accumulation are commonly employed to balance computational efficiency and numerical accuracy. Such mixed-precision

configurations are supported by widely used inference frameworks, including TensorRT and TensorFlow Lite.

Table 1. MAC Count Comparison for various Neural Network Workloads [1]

Utiliz. Of MAC (%)	LeNet-5	AlexNet	VGG-13	ResNet-18	GoogLeNet
	98.1	99.8	99.4	99.89	99

Approximate computing techniques can be implemented at the circuit, architectural, and software levels. At the circuit level, the MAC unit consists of key components such as compressors [5], [6], multipliers [7], [8], and adders [9], [10], where approximation can significantly reduce area, delay, and power consumption. Previous studies mainly focused on approximating individual submodules independently [11], [12]. However, combined optimization of the partial product generation, compression stage and accumulation stage has received limited attention. To address this limitation, this work proposes an error-aware approximate MAC design methodology that identifies an efficient combination of approximate components to improve hardware efficiency while preserving acceptable computational accuracy. The major contributions of this work are as follows:

1. **Extended radix-8 Booth Encoding:** An extended radix-8 Booth encoding scheme is developed using a single-bit MSB sign/zero-extension technique to support both *signed* × *signed* and *unsigned* × *signed* multiplication while simplifying sign handling.
2. **Approximate Partial Products (AxPP):** Two mappings, AxPP1 ($\pm 3X \rightarrow \pm 2X$) and AxPP2 ($\pm 3X \rightarrow \pm 4X$), are investigated. AxPP1 is selected for the proposed design due to its simpler hardware, while AxPP2 is used for comparison.
3. **Approximate Adders and Compressors:** Proposed approximate Half-Adder (HA), Full-Adder (FA) and 4:2 compressor designs are developed using an error-metric–guided method, achieving 54%–82% energy savings in the reported circuit-level comparisons with controlled error characteristics.
4. **Hybrid Accumulator Design:** A proposed hybrid accumulator combines an exact Han–Carlson Parallel Prefix Adder (HC-PPA) for the Most Significant Bits (MSBs) with a lightweight Lower-OR Adder (LOA) structure for the Least Significant Bits (LSBs), providing an accuracy–hardware-efficiency trade-off.
5. **Configurable AxMAC Variants:** Four configurable AxMAC variants are developed — AxMAC-Mul1, AxMAC-Mul2, AxMAC-ACC, and AxMAC-Mul+ACC – achieving Power-Delay Product (PDP) savings of 15.9%–69.7% and Area-Delay Product (ADP) savings of 41.5%–85.1% compared with the exact radix-8 Booth MAC.
6. **Deep Learning Validation:** The proposed AxMAC variants are evaluated using pre-trained CNN models, including LeNet-5, AlexNet, VGG-13, ResNet-18, and GoogLeNet, across MNIST, CIFAR-10, and ImageNet. The evaluation uses selected-layer approximation without retraining or fine-tuning to assess the impact of approximate computation on CNN inference accuracy.

This paper is organized as follows. Section II outlines the foundational concepts and related research progress in this area. Section III introduces the proposed designs and compares the approximate MAC submodules. Section IV shows the proposed variants of approximate MAC unit. Section V presents the synthesis results, accuracy evaluation, and comparative analysis. Section VI concludes the work and outlines directions for future research.

II. RELATED WORKS

Recent research on efficient MAC architectures can be broadly classified into architectural and arithmetic optimizations, approximate computing approaches, and the resulting research gaps.

A. Architectural and Arithmetic Optimizations

Accepted Article

Sarnala and Pillutla [13] proposed a Hybrid Parallel Prefix Adder (HPPA) for FPGA-based CNN acceleration, achieving a 16.78% improvement in computational speed through optimized addition logic within the MAC datapath. Tang and Han [14] introduced a dual-mode Truncation Error Compensation (TEC) method for Booth-based MAC units, yielding over 40% improvement in area–delay–error performance relative to conventional truncated designs. Tung and Huang [15] proposed a high-performance MAC architecture that integrates addition and accumulation within the Partial Product Reduction (PPR) stage, reducing carry propagation delay and improving computational throughput. Cheng et al. [16] developed a sparse CNN accelerator based on a radix-4 Booth multiplier, achieving 14.37 TOPS/W efficiency at high sparsity levels, demonstrating the effectiveness of hardware-aware sparsity in improving MAC utilization.

B. Approximate Computing for MAC Optimization

Approximate computing has emerged as an effective design strategy for MAC units in error-tolerant CNN and edge-AI workloads. Nojehdeh et al. [17] demonstrated that substituting exact adders and multipliers with approximate counterparts reduces energy consumption by 64% and silicon area by 43% with negligible accuracy loss. Li et al. [18] developed an approximate Processing Element (PE) employing stochastic rounding, achieving a 29% reduction in Power–Delay Product (PDP) while preserving inference accuracy. Di Meo et al. [19] proposed a segmented approximate MAC design with over 60% power savings, and Zhang et al. [20] introduced a 4:2 approximate compressor that improved the Power–Delay–Area Product (PDAP) by 66%. Chhajed et al. [21] presented BitMAC, a bit-serial MAC architecture for low-power edge devices, achieving 25.9% area reduction, 35.6% lower power, and 14.3% lower latency in 45 nm CMOS compared to conventional designs.

At the arithmetic submodule level, Napoli et al. [22] conducted a comprehensive analysis of approximate full adder designs using 14 nm FinFET technology, enumerating all truth-table variants of a standard full adder and identifying Pareto-optimal designs that offer the best trade-offs among power, area, and accuracy. Rosa et al. [23] proposed Approximate Parallel Prefix Adders (AxPPA) by introducing approximations in the prefix-operator logic, reporting energy and area reductions with limited output-quality degradation. Tang et al. [24] designed Approximate radix-4 Booth Encoders (AR4BE1 and AR4BE2) for signed multipliers, reducing area, delay, and power by up to 32% while maintaining acceptable accuracy for image processing applications. Balasaraswathi and Padmapriya [25] introduced Hybrid Parallel Prefix Adders (HyPPA), combining approximate lower-bit adders — including LOA, Copy Adder, and ETA-1 — with exact prefix logic in the upper bits, achieving up to 57% power and 65% energy savings with minimal accuracy loss. While HyPPA demonstrates the effectiveness of hybrid exact–approximate prefix addition, the present work extends this concept to the accumulation stage of an approximate MAC architecture. Khurshid [26] proposed CORDIC-based approximate MAC architectures with adaptive computation stages and reported improved error characteristics for 16-bit operations. Nojehdeh et al. [27] further explored approximate arithmetic for time-multiplexed MAC blocks in Artificial Neural Networks (ANNs), reducing hardware area by up to 64% and energy by 43% through selective replacement of exact units, with their LEBZAM multiplier demonstrating controlled precision trade-offs via bit-level truncation.

C. Research Gap

Despite these advances, the interaction among approximation techniques across different MAC stages remains less explored. This work develops four configurable AxMAC variants with selective approximation in partial product generation, reduction, and accumulation, and evaluates their hardware, error, and CNN inference characteristics.

III. PROPOSED METHODOLOGY FOR STAGE-WISE APPROXIMATE MAC DESIGN

Archive Article

The proposed approximate MAC (AxMAC) architecture pursues hardware efficiency through a *structured, stage-wise approximation strategy* applied independently to the three canonical phases of MAC computation: Partial Product Generation (PPG), Partial Product Reduction (PPR), and Accumulation (ACC). The MAC operation is expressed as:

$$y = w \cdot x + b \tag{1}$$

where $w \in \mathbb{Z}^8$ denotes the signed 8-bit CNN weight (multiplier), $x \in \mathbb{N}^8$ the unsigned 8-bit input activation (multiplicand), and b the bias accumulation operand. The product $w \cdot x$ is a 16-bit integer result, which is subsequently accumulated by a 32-bit adder to produce y.

A. Exact radix-8 Modified Booth Encoding and PPG:

Assume that two N-bit signed binary values, X and Y, stand for multiplicand and multiplier, respectively. X and Y can be expressed in 2's complement representation with the following equations (2) and (3).

$$X = -x_{N-1}2^{N-1} + \sum_{i=0}^{N-2} x_i\, 2^i \tag{2}$$

$$Y = -y_{N-1}2^{N-1} + \sum_{i=0}^{N-2} y_i\, 2^i \tag{3}$$

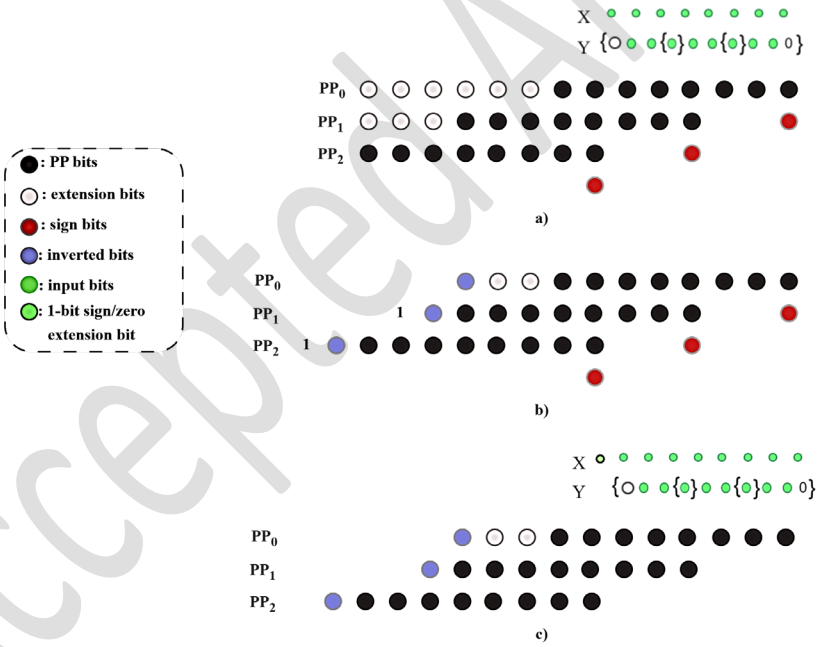


Figure 2. Partial product array of the radix-8 Modified Booth Encoding (MBE) for an 8-bit multiplier, producing three partial product rows PP_0, PP_1, and PP_2 (j = 0, 1, 2): (a) Conventional array with full sign-extension (b) Modified array after sign-bit compression using two's complement correction (c) Proposed approximate array

To achieve the product of X and Y , the radix-8 Modified Booth Encoding (MBE) [8], [24], [28], arranges four bits of Y into a set $\{y3_{j+2}, y3_{j+1}, y3_j, y3_{j-1}\}$ with one bit overlapping as shown in Figure 2. While grouping, the y_{-1} bit is assumed to be logic-0 and the sign bit y_{N-1} can be extended to higher bits [29]. The Booth encoded digit for the j-th group, M_j, is computed as (4).

$$M_j = -4y3_{j+2} + 2y3_{j+1} + y3_j + y3_{j-1} \qquad for \quad 0 \le j \le \lceil N/3 \rceil - 1 \tag{4}$$

The encoded bit maps each four-bit group to a signed digit drawn from the set $\{0, \pm 1X, \pm 2X, \pm 3X, \pm 4X\}$, where X denotes the multiplicand. Consequently, an N-bit multiplier

Accepted Article

generates $\lceil N/3 \rceil$ partial product rows, compared to $\lceil N/2 \rceil$ rows for radix-4 encoding and N rows for non-encoded multiplication. The partial product (PP) rows of multiplier are obtained in (5).

$$PP_{ij} = M_j \cdot X_i \quad for \quad 0 \le j \le \lceil N/3 \rceil - 1 \tag{5}$$

In Eq. (5), j is the row index (PP_0, PP_1, PP_2 in Figure 2) and i is the bit position within that row. When sign bit of a Booth encoded digit $y3_{j+2}$= 0, the PP_{ij} takes only positive values. When $y3_{j+2}$=1, a negative partial product row is obtained by 2's complement of the corresponding positive partial product row. Therefore, it is achieved by adding logic-1 ($y3_{j+2}$) at the LSB and complementing each individual bit of the positive partial product row. The conventional and modified partial product array of radix-8 Booth algorithm is explained through Figure 2(a) and (b)

i) Proposed radix-8 Booth PP array:

The proposed radix-8 Booth PP array for an 8-bit multiplier is shown in Figure 2(c). It uses a single-bit MSB sign/zero-extension of the multiplicand to support both signed and unsigned operands. For unsigned inputs, the extension bit is set to logic-0, whereas for signed inputs it is set equal to the original MSB. Thus, the extended multiplicand is expressed as

$$X' = \{s, X_{N-1}, X_{N-2}, .., X_0\} \tag{6}$$

where $s = 0$ for unsigned inputs and $s = X_{N-1}$ for signed inputs. In the INT8 CNN configuration, X represents the unsigned activation and Y represents the signed weight. The single bit extension enables the same radix-8 Booth datapath to handle both signed and unsigned multiplicands without a separate unsigned multiplication datapath. The proposed scheme also folds the two's-complement sign-correction bits into a single partial product row, reducing the PP matrix height compared with conventional sign-extension approaches. For the 8-bit radix-8 configuration, three partial product rows, PP_0–PP_2, are generated.

ii) Proposed Approximate PP terms:

The $\pm 3X$ multiple in radix-8 Booth encoding increases hardware complexity because it requires additional addition logic. To reduce this overhead, two approximate partial product (AxPP) designs, AxPP1 and AxPP2, are introduced. AxPP1 replaces $\pm 3X$ with $\pm 2X$, while AxPP2 replaces $\pm 3X$ with $\pm 4X$. Both methods eliminate the dedicated $\pm 3X$ generation circuitry and simplify the Booth selector by limiting the required multiples to $\{0, \pm X, \pm 2X, \pm 4X\}$. AxPP1 is selected for the proposed design because $\pm 2X$ is already available in the standard radix-8 Booth multiple set, requiring only remapping of the $\pm 3X$ decode states. Its positive and negative mappings also produce opposite-signed approximation errors, which provides the potential for partial error cancellation, when such terms occur with appropriate operand distributions. Validation of this behavior using layer-wise CNN operand distributions is left for future work. AxPP2 is used as a comparative mapping because it replaces $\pm 3X$ with $\pm 4X$, increasing the magnitude of the approximated terms.

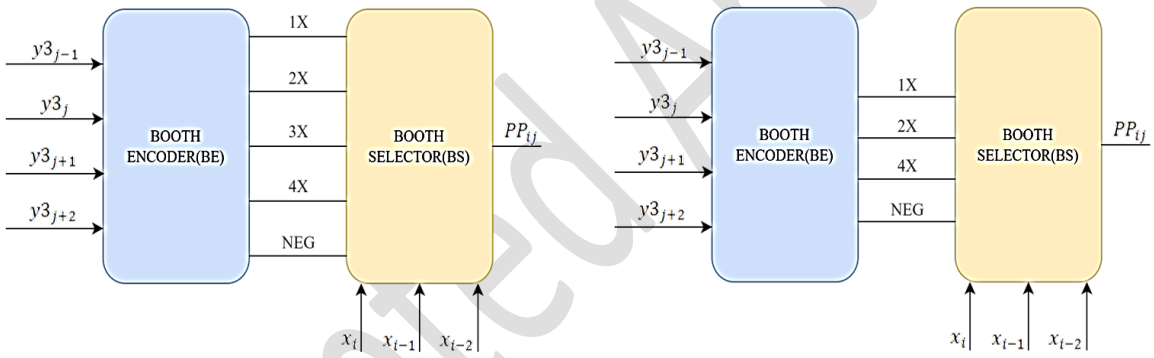
Table 2. Error Distance Comparison of exact and approximate radix-8 Booth PPG.

Booth bits $\{y3_{j+2}, y3_{j+1}, y3_j, y3_{j-1}\}$	M_j	PP_{ij}	**AxPP1**	**ED**	**AxPP2**	**ED**
0000	0	0	0	0	0	0
0001	+1	+1X	+1X	0	+1X	0
0010	+1	+1X	+1X	0	+1X	0
0011	+2	+2X	+2X	0	+2X	0
0100	+2	+2X	+2X	0	+2X	0
0101	+3	+3X	+2X	-1	+4X	1

Accepted Article

0110	+3	+3X	+2X	-1	+4X	1
0111	+4	+4X	+4X	0	+4X	0
1000	−4	−4X	−4X	0	−4X	0
1001	−3	−3X	−2X	1	−4X	-1
1010	−3	−3X	−2X	1	−4X	-1
1011	−2	−2X	−2X	0	−2X	0
1100	−2	−2X	−2X	0	−2X	0
1101	−1	−1X	−1X	0	−1X	0
1110	−1	−1X	−1X	0	−1X	0
1111	0	0	0	0	0	0

Accordingly, AxPP1 ($\pm 3X \rightarrow \pm 2X$) is adopted for the approximate PPG stage of AxMAC-Mul1 and AxMAC-Mul+ACC. AxMAC-Mul2 and AxMAC-ACC retain the exact PPG because their approximation is introduced in the PPR and accumulation stages, respectively. Figure 3 illustrates the conventional and proposed radix-8 Booth PPG architectures. In the conventional design, the Booth selector generates the required multiple from the full set $\{0, \pm X, \pm 2X, \pm 3X, \pm 4X\}$. In the proposed approximate design, the $\pm 3X$ multiples are eliminated and the selector uses the reduced control signals $\{1X, 2X, 4X, \text{NEG}\}$. This reduces multiple-generation and selector complexity, thereby reducing switching activity and partial product generation power.



a) Conventional PP generation b) Approximated PP generation

Figure 3. Booth Encoder–Selector Architecture

B. Partial Product Reduction (PPR):

In high-performance multipliers, the PPR stage, typically implemented using Wallace or Dadda trees, contributes significantly to switching activity and energy consumption [30]. Exact adders and compressors provide accurate reduction but increase hardware complexity and critical-path delay. Approximate compressors relax selected logic operations to reduce area, delay, and power while exploiting the error tolerance of CNN inference [20].

i) Proposed Half Adders

Half adders (HAs) are basic components of multiplier reduction trees. In this work, three approximate HA variants are evaluated against the exact HA to reduce hardware cost, delay, and energy with bounded error. The functional expressions and error characteristics are summarized in Tables 3 and 4. Among the evaluated variants, the XOR-less HA provides a suitable hardware–error trade-off with a maximum ED of 1 and is therefore selected for the proposed PPR stage. The carry-free HA offers lower delay but has a maximum ED of 2, while the LOA-style HA provides lower hardware cost with higher error.

Table 3. Functional Comparison of exact and approximate HA variants.

Accepted Article

Variant	Approximate Sum	Approximate Carry	Hardware Cost	Error Cases	Max ED
Exact HA	$S = A \oplus B$	$C = A \cdot B$	High (XOR+AND)	-	-
XOR-less HA	$S = A \vee B$	$C = A \cdot B$	Low (OR + AND)	Only (1,1)	1
Carry-Free HA	$S = A \oplus B$	$C = 0$	Very Low (XOR only)	Only (1,1)	2
LOA-Style HA	$S = A \vee B$	$C = B$	Ultra-Low	(0,1), (1,1)	3

*The selected design is highlighted.

Table 4. Truth Table of exact and approximate half-adder variants.

A	B	Exact S	Exact C	XOR-less S	XOR-less C	ED	Carry-free S	Carry-free C	ED	LOA-S	LOA-C	ED
0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	1	0	0	1	1	2
1	0	1	0	1	0	0	1	0	0	1	0	0
1	1	0	1	1	1	1	0	0	2	1	1	1

Among the evaluated approximate half adders, the XOR-less HA provides a suitable trade-off between error and hardware cost, with a maximum ED of 1, making it suitable for reduction stages where moderate error constraints are required. The carry-free HA achieves the lowest delay by completely eliminating the carry path, making it attractive for deep compressor trees, although its maximum ED = 2 limits use to lower-significance positions. The LOA-style HA provides lower hardware cost but higher error, making it more suitable for LSB-dominant or highly error-tolerant applications. The XOR-less HA is chosen as the preferred building block for the proposed MAC architecture based on its hardware-error trade-off.

ii) Proposed Full Adders

Full adders (FA) serve as the core arithmetic components in the PPR phase in multipliers. In this work, four approximate FA architectures are evaluated against the exact FA to reduce hardware cost, delay, and energy with bounded error [22]. Their functional expressions and error characteristics are summarized in Tables 5 and 6. Among the evaluated variants, AxFA provides a suitable accuracy–hardware trade-off, with a maximum ED of 1, compared with an ED of 2 observed in CFFA, M-LOA, and HSFA. AxFA also preserves the exact carry equation while simplifying the sum-generation logic. Therefore, AxFA is selected for integration into the PPR stage of the proposed MAC architecture.

Table 5. Functional Comparison of exact and approximate FA variants.

Variant	Sum Expression	Carry Expression	Hardware Cost	Error Cases	Max ED
Exact Full Adder	$S = A \oplus B \oplus C_{in}$	$C = (A.B) \vee (A.C_{in}) \vee (BC_{in})$	High	–	0
AxFA (XOR-less FA)	$S = A \vee B \vee C_{in}$	$C = (A.B) \vee (A.C_{in}) \vee (BC_{in})$	Low	Few	1
CFFA (Carry-Free FA)	$S = A \oplus B \oplus C_{in}$	$C = 0$	Very Low	Many	2
M-LOA FA	$S = A \vee B$	$C = B \vee C_{in}$	Ultra-Low	Moderate	2
HSFA (Hybrid Simple-Logic FA)	$S = A \oplus B$	$C = A \vee B$	Low	Few	2

*The selected design is highlighted.

Accepted Article

Table 6. Truth Table of exact and approximate FA variants.

A	B	Cin	Exact (S,C)	AxFA (S,C)	ED	CFFA (S,C)	ED	M-LOA (S,C)	ED	HSFA (S,C)	ED
0	0	0	0,0	0,0	0	0,0	0	0,0	0	0,0	0
0	0	1	1,0	1,0	0	1,0	0	1,1	2	1,0	0
0	1	0	1,0	1,0	0	1,0	0	1,1	2	1,1	2
0	1	1	0,1	1,1	1	0,0	2	1,1	1	1,1	1
1	0	0	1,0	1,0	0	1,0	0	1,1	2	1,1	2
1	0	1	0,1	1,1	1	0,0	2	1,1	1	1,1	1
1	1	0	0,1	1,1	1	0,0	2	1,1	1	0,1	0
1	1	1	1,1	1,1	0	1,0	2	1,1	0	1,1	0

iii) Proposed 4:2 Compressors

4:2 compressors are key components of the partial product reduction tree and strongly influence its area, delay, and switching activity [15]. In this work, three approximate 4:2 compressors are evaluated against the exact compressor to reduce hardware cost while maintaining controlled error [31]. The functional expressions are summarized in Table 7.

Table 7. Functional Comparison of exact and approximate 4:2 variants.

Compressor	Sum output S	Carry output C
Exact 4:2	$S_{\text{exact}} = x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus C_{in}$	$C_{\text{exact}} = ((x_2 \oplus x_1) \cdot (x_4 \oplus x_3)) \vee C_{in}((x_2 \oplus x_1) \oplus (x_4 \oplus x_3))$
AxCOMP1	$S_{\text{Ax1}} = x_4 \oplus x_3 \oplus x_2 \oplus x_1$	$CAx1 = (x_1.x_2) \vee (x_3.x_4)$
AxCOMP2	$S_{\text{Ax2}} = x_3 \oplus x_2 \oplus x_1$	$CAx2 = (x_1 \oplus x_2) \vee (x_3.x_4)$
AxCOMP3	$S_{\text{Ax3}} = (x_4 + x_3) \oplus (x_2 \oplus x_1)$	$CAx3 = (x_1.x_2) \vee (x_3.x_4)$

*The selected design is highlighted.

Table 8. Truth Table of exact and approximate 4:2 compressor variants.

x_4	x_3	x_2	x_1	S_{exact}	C_{exact}	S_{Ax1}	C_{Ax1}	ED1	S_{Ax2}	C_{Ax2}	ED2	S_{Ax3}	C_{Ax3}	ED3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	0	1	0	1	1	0	1	0	0	1
0	0	1	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	0	1	0	1	0	0	1	0	0	0	0	1
0	1	0	1	0	1	0	1	0	1	1	1	1	1	1
0	1	1	0	0	1	0	1	0	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	0	1	1
1	0	0	0	1	0	0	0	1	1	0	0	1	0	0
1	0	0	1	0	1	0	1	0	1	1	1	1	1	1
1	0	1	0	0	1	0	1	0	1	1	1	1	1	1
1	0	1	1	1	1	1	1	0	1	1	0	1	1	0
1	1	0	0	0	1	0	1	0	0	1	0	1	0	1
1	1	0	1	1	1	0	1	1	0	1	1	1	1	0
1	1	1	0	1	1	0	1	1	1	1	0	1	1	0
1	1	1	1	0	1	0	1	0	1	1	1	1	1	1

Table 8 lists the complete truth table of the proposed 4:2 compressor, showing the mapping from the four input bits $(x_4, x_3, x_2, x_1)$to the exact sum and carry outputs $(S_{ex}, C_{ex})$and three approximate realizations (S_1, C_1), (S_2, C_2), and (S_3, C_3). The last three columns, ED1, ED2, and ED3, report the corresponding error distance for each approximation, defined as the absolute difference

Accepted Article

between the exact and approximate output values. Table 8 shows that the approximate compressors produce ED values of 0 or 1 for the evaluated input combinations.

Table 9 compares the synthesis results and error probabilities of the exact compressor and six approximate variants – Design [5], Design [31], Design [32], AxCOMP1, AxCOMP2, and AxCOMP3. All designs are synthesized using the 45 nm CMOS standard-cell library under identical synthesis constraints mentioned in Section V. Error probability is reported over 256 weighted input occurrences, following the probability model used in the referenced approximate-compressor studies. The results illustrate different trade-offs between hardware cost and error probability. The PDP–error and ADP–error plots illustrate the hardware–error trade-offs among the exact and approximate compressors in Figure 4.

Table 9. Hardware Performance and error comparison of exact and approximate 4:2 compressors.

Design	Area (µm²)	Power (nW)	Delay (ps)	Error Probability	ADP (µm²·ps)	PDP (fJ)
Exact	35.84	734.38	402	–	14407.68	0.2952
Design [5]	14.65	243.87	199	175/256	2915.35	0.04853
Design [31]	26.10	775.92	451	16/256	11771.10	0.34994
Design [32]	25.87	503.47	436	4/256	11279.32	0.21951
AxCOMP1	23.28	440.31	203	16/256	4725.84	0.08938
AxCOMP2	15.34	416.86	184	37/256	2822.56	0.07670
AxCOMP3	13.06	334.75	179	175/256	2455.28	0.06293

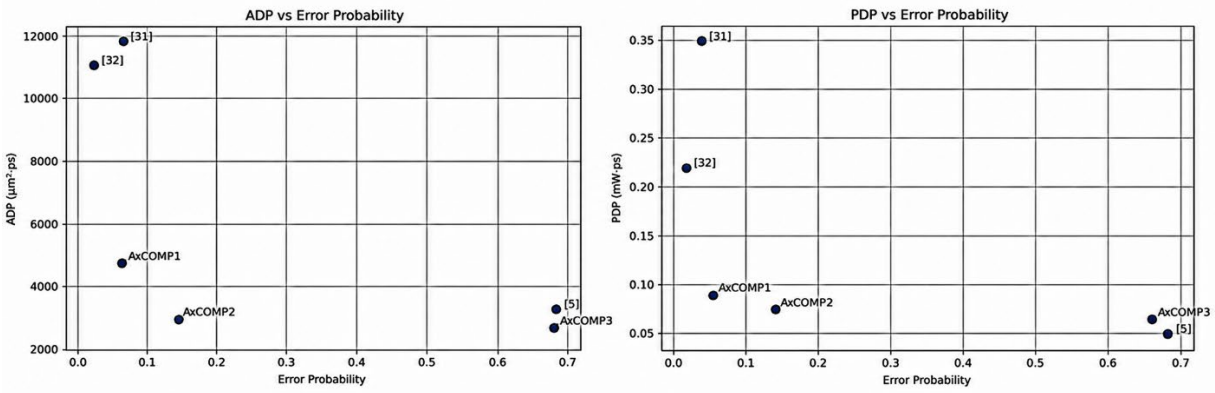


Figure 4. Area–Delay and Power–Delay Trade-Offs of Approximate Compressors

Among the existing approximate designs, Design [5] provides substantial hardware reduction but has a high error probability of 175/256. Design [32] has a low error probability of 4/256 but higher delay and hardware cost, while Design [31] provides an intermediate error probability of 16/256 with higher delay and power. Among the proposed compressors, AxCOMP1 provides a 16/256 error probability with reduced area, power, and delay compared with the exact compressor. AxCOMP2 and AxCOMP3 provide further hardware reductions but exhibit higher error probabilities of 37/256 and 175/256, respectively. Therefore, AxCOMP1 is selected for the PPR stage based on its hardware–error trade-off.

C. Proposed Hybrid Approximate Accumulator:

In a MAC unit, the generated products are accumulated using an adder tree or carry-propagate structure. Prefix adders generate carries in parallel and can reduce carry-propagation delay [22], [23], [25], [33]. In the proposed work, a Hybrid Han–Carlson Parallel Prefix Adder (HyPPA-HC) is employed in the accumulation stage, with approximation confined to the lower eight bits. The proposed HyPPA-HC divides the accumulator into three regions: (i) an exact Han–Carlson prefix network for bits 31-8, (ii) a semi-exact local-carry section for bits 7-4, and (iii) an approximate Lower–OR Adder

(LOA) for bits 3-0. Thus, the 32-bit accumulator retains exact computation in the upper 24 bits while approximation is restricted to the lower 8 bits. Figure 5 shows the block-level structure of the proposed 32-bit HyPPA-HC with these three regions. For all bit positions, the fundamental Propagate (p_i) and Generate (g_i) signals are defined as

$$p_i = a_i \oplus b_i, \qquad g_i = a_i \cdot b_i \tag{7}$$

and the prefix operator used in the exact Han–Carlson stages follows the canonical associative formulation.

$$(G, P) = (G_k, P_k) \circ (g_j, p_j) = (G_k \vee P_k g_j, P_k p_j) \tag{8}$$

The LOA region (bits 0-3) suppresses all carry dependencies and replaces the exact sum function with the OR formulation in (9)

$$s_i^{\mathrm{LOA}} = a_i \vee b_i \qquad 0 \le i \le 3, \tag{9}$$

thereby collapsing the carry network depth to $O(1)$ and eliminating both the local full-adder cells and

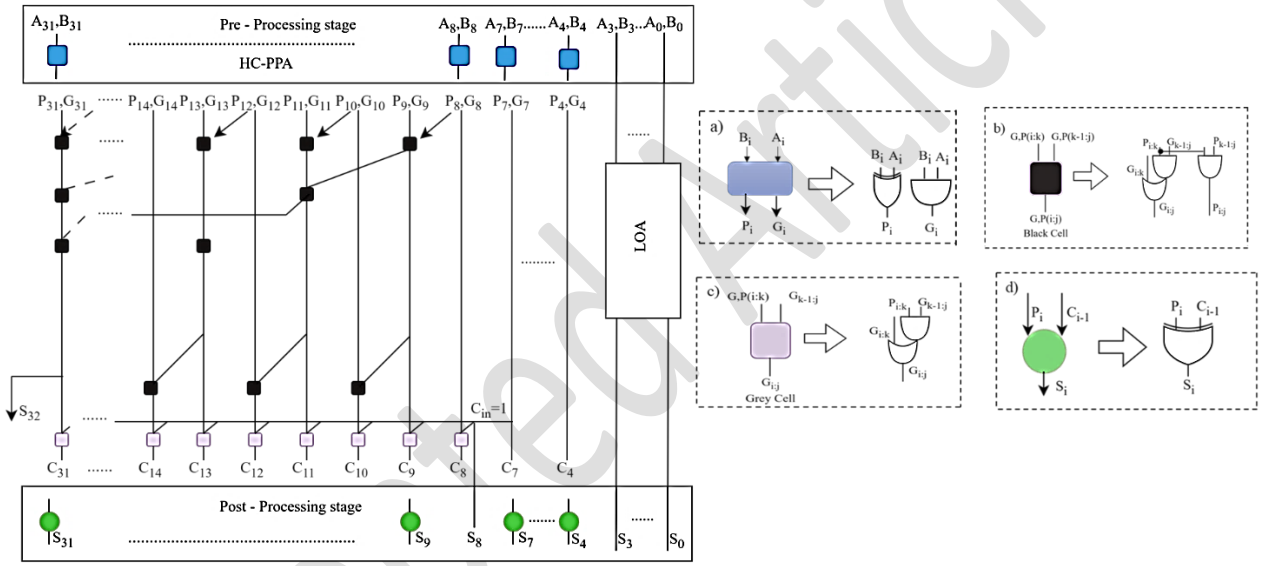


Figure 5. Proposed Block-level architecture of HyPPA-HC.

the forward carry arcs. The upper LSB part (bits 4-7) retains the propagate–generate semantics but deliberately omits prefix expansion; instead, carries are resolved through a constrained intra-nibble recursive relation.

$$C_{i+1} = g_i \vee (p_i . C_i) \qquad 4 \le i \le 7, \tag{10}$$

with the LOA boundary condition $C_4 = 0$. This yields an exact local carry-out $C_8 = G_{7:4} \vee P_{7:4} C_4$, where $G_{7:4}, P_{7:4}$ denote the group-generate and group-propagate terms of the nibble. The MSB region consumes C_8 as the injected initial carry and implements a 24-bit Han–Carlson prefix topology for bits 31-8, which combines sparse levels (as Brent–Kung) and dense levels (as Kogge–Stone) to realize logarithmic-depth carry computation with bounded fan-out. The exact MSB sum is finally produced as

$$s_i = p_i \oplus C_i \qquad 8 \le i \le 31, \tag{11}$$

The obtained hybridization drastically reduces the total prefix-cell count by eliminating the entire lower 8-bit prefix tree and collapsing the lowest nibble into pure combinational OR logic.

IV. DESIGN OF APPROXIMATE MAC

Accepted Article

This section presents four approximate MAC variants obtained by selectively introducing approximation into the Partial Product Generation (PPG), Partial Product Reduction (PPR), and Accumulation (ACC) stages [34], [35].

A. The **AxMAC-Mul1** design introduces approximation only in the PPG stage, simplifying Booth encoding by replacing complex ±3*X* terms with ±2*X* following the AxPP1 scheme defined in Section III.A.ii. The PPR, partial-sum CPA, and accumulation stages use exact arithmetic as discussed in Section III A & B, as shown in Figure 6.

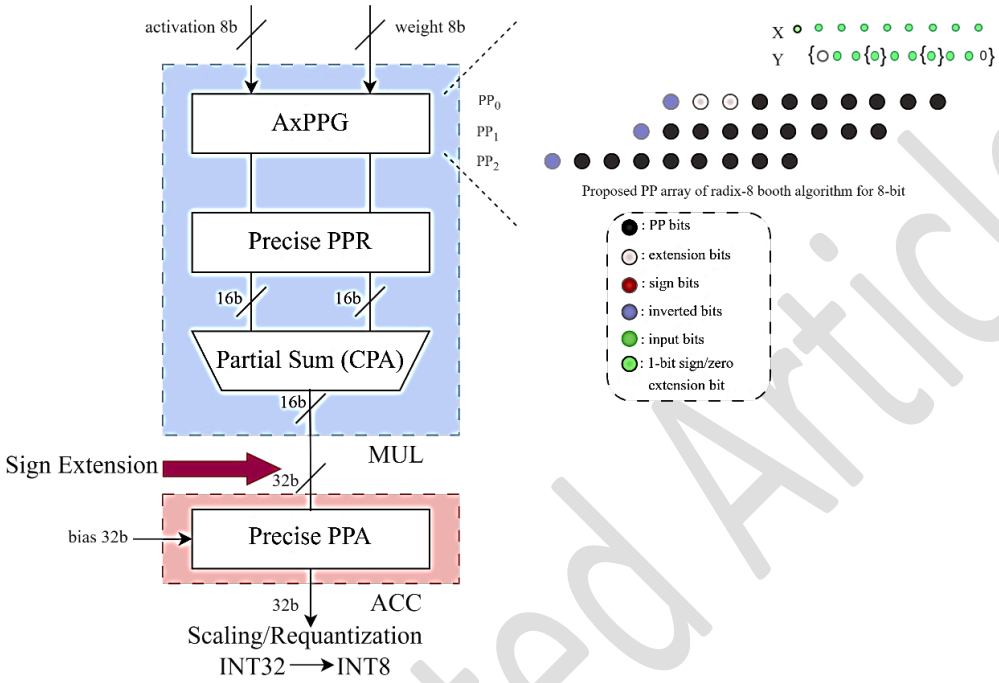


Figure 6. Block-level architecture of the proposed AxMAC-Mul1

B. The **AxMAC-Mul2** design uses a selective-approximation approach, applying approximation only in the PPR stage while the PPG, partial-sum CPA, and accumulation stages remain exact. In the PPR stage, the selected XOR-less HA, AxFA, and AxCOMP1 are assigned to the lower-significance columns, while exact adders and compressors are retained in the higher-significance columns according to Algorithm 1.

Algorithm 1 specifying Approximate Designs for Different Columns of AxMAC-Mul2

Input: N → Operand length
Assumption: Initially, all compressors are exact.
Definitions:
i → Reduction stage index
j → Column index
compressor(i,j) → Compressor assigned at stage i and column j
Output:
Assignment of approximate compressors in LSB columns
and exact compressors in MSB columns

```
------------------------------------------------------------
-- Assigning approximate compressors in LSB region --
------------------------------------------------------------
1:  for i in [stage1, stage2, ..., stage(log2(N/3) − 1)]
2:      for j in [column 0 to column (N/2)]
3:          if number_of_bits(j) == 2 then
```

Accepted Article

```
4:          compressor(i,j) ← XOR-less HA
5:        else if number_of_bits(j) == 3 then
6:          compressor(i,j) ← AxFA
7:        else if number_of_bits(j) >= 4 then
8:          compressor(i,j) ← AxCOMP1
9:        end if
10:     end for
11: end for
-------------------------------------------------------------
-- Assigning exact compressors in MSB region --
-------------------------------------------------------------
12: for i in [stage1, stage2, ..., stage(log2(N/3) − 1)]
13:     for j in [column (N/2 + 1) to column (2N − 3)]
14:       if number_of_bits(j) == 2 then
15:         compressor(i,j) ← Exact HA
16:       else if number_of_bits(j) == 3 then
17:         compressor(i,j) ← Exact FA
18:       else if number_of_bits(j) >= 4 then
19:         compressor(i,j) ← Exact 4:2 Compressor
20:       end if
21:     end for
22: end for
```

C. The **AxMAC-ACC** design introduces approximation only in the accumulation (ACC) stage, while all multiplier components (PPG, PPR, partial-sum CPA) remain exact. The 32-bit accumulator uses the HyPPA-HC structure described in Section III.C, with an exact Han–Carlson region for bits 31-8, a semi-exact local-carry region for bits 7-4, and an approximate LOA region for bits 3-0. Thus, approximation is confined to the lower eight bits of the 32-bit accumulator, while the upper 24 bits remain exact.

D. The **AxMAC-Mul+ACC** design applies approximation across all three MAC stages: the AxPP1 scheme ($\pm 3X \rightarrow \pm 2X$) for PPG, the XOR-less HA, AxFA, and AxCOMP1 for PPR, and the HyPPA-HC adder for 32-bit accumulation with exact MSBs and approximate LSBs. The partial-sum CPA remains exact.

V. SYNTHESIS AND ACCURACY EVALUATION RESULTS

This section evaluates the proposed AxMAC variants in terms of circuit delay, occupied area, and power dissipation, arithmetic error and CNN inference accuracy. The following evaluations were conducted to validate the design performance:

- **Performance Evaluation**: The proposed AxMAC variants and all reference designs are described in Verilog HDL and functionally verified by RTL simulation. Logic synthesis is performed using Cadence Genus with GPDK045 45 nm CMOS standard-cell library characterized at 1.08 V and 125 °C under the worst-case (SS) process corner. Power is estimated from a Switching Activity Interchange Format (SAIF) file captured during RTL simulation for all evaluated designs. Since the AxMAC variants are purely combinational, no timing constraint is applied to their input-to-output paths; consequently, setup/hold timing slack is not defined for these unconstrained paths. The critical-path delay reported in Table 10 is the data-arrival time of the longest combinational path from the primary inputs to the primary outputs, extracted using the *report_timing -unconstrained* command. To validate physical implementability, the proposed AxMAC-Mul+ACC design is taken through the placement and routing flow using Cadence Innovus; the resulting layout is shown in Figure 7.

Accepted Article

- **Error–Hardware Trade-off Analysis**: Python-based error analysis is used to evaluate the proposed approximate MACs. AxMAC-Mul1 and AxMAC-Mul2 are evaluated exhaustively over 2^{16} combinations of 8-bit operands, while AxMAC-ACC and AxMAC-Mul+ACC are evaluated using 100 000 uniformly random generated full-MAC Monte Carlo samples.
- **Inference Accuracy Validation**: PyTorch-based software emulation models each AxMAC variant as an approximate convolution operator. Approximation is applied only to the selected convolutional layer, while all remaining operations are kept exact. Pre-trained INT8-quantized LeNet-5, AlexNet, VGG-13, ResNet-18, and GoogLeNet models are evaluated in inference-only mode without retraining or fine-tuning across MNIST, CIFAR-10, and ImageNet.

A. Delay, Power and Area Results

Table 10 summarizes the synthesis results of the proposed AxMAC variants. Compared with the exact radix-8+WT+RC MAC, AxMAC-Mul1, AxMAC-Mul2, AxMAC-ACC, and AxMAC-Mul+ACC reduce delay by 6.8%, 19.9%, 36.3%, and 43.5%, respectively. The corresponding area reductions are 37.2%, 56.9%, 67.6%, and 73.6%, while the PDP reductions are 15.9%, 40.3%, 47.0%, and 69.7%, respectively. The proposed AxMAC variants and reference designs show different hardware characteristics due to their respective approximation strategies. To assess physical implementability, the AxMAC-Mul+ACC design is taken through the placement-and-routing flow using Cadence Innovus as in Figure 7. The normalized area–delay and power–delay trade-offs among the evaluated designs are represented in Figure 8.

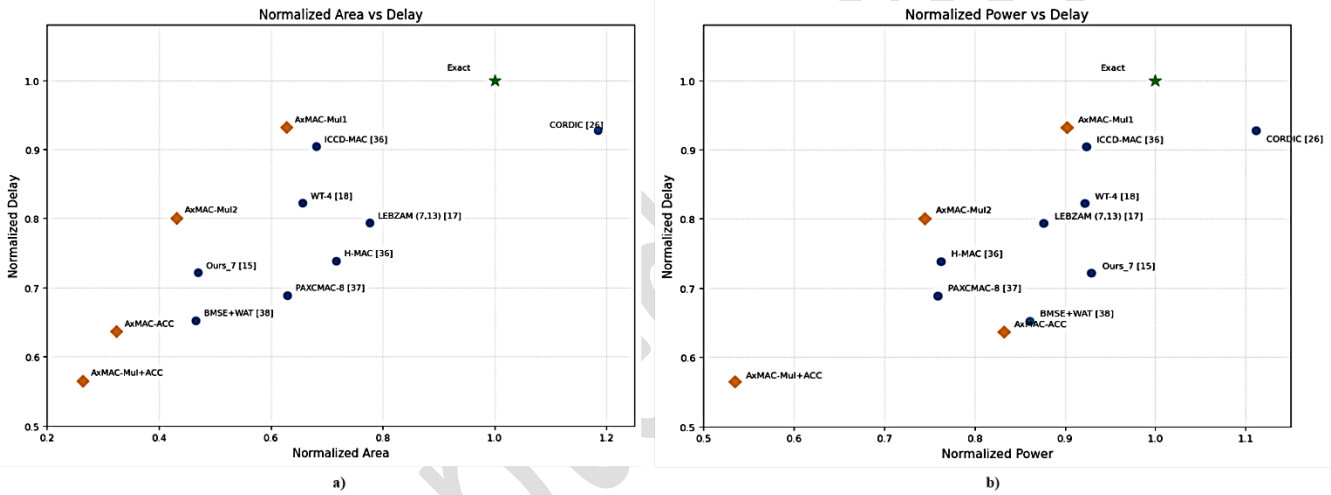
Table 10. Hardware Performance for Proposed MAC variants and Reference Designs (delay = unconstrained critical-path delay)

Works	Delay (ns)	Area (μm²)	Power (μW)	ADP (μm²·ns)	PDP (pJ)	PDAP (×10⁷)	Norm. PDAP
Exact (radix-8+WT+RC)†	13.07	3878.04	1065.71	50,685.983	13.92	5.399	1.00
AxMAC-Mul1	12.18	2435.21	961.504	29,660.858	11.711	2.859	0.530
AxMAC-Mul2	10.47	1671.88	793.89	17,504.584	8.312	1.390	0.257
AxMAC-ACC	8.32	1256.14	887.047	10,451.085	7.38	0.927	0.172
AxMAC-Mul+ACC	7.39	1023.84	570.221	7,566.178	4.213	0.431	0.080
ICCD-MAC[36]	11.82	2643.24	984.375	31,243.097	11.635	3.069	0.568
H-MAC[36]	9.66	2778.57	812.809	26,840.986	7.85	2.175	0.403
LEBZAM (7,13)[17]	10.37	3011.05	934.078	31,224.588	9.686	2.913	0.539
P_{AXC}MAC-8[37]	9.01	2441.79	808.974	22,000.528	7.288	1.781	0.330
BMSE+WAT [38]	8.53	1807.68	917.203	15,419.510	7.823	1.409	0.261
CORDIC[26]	12.13	4591.5	1184.069	55,694.895	14.362	6.605	1.224
Ours_7[15]	9.43	1824.07	990.465	17,200.980	9.34	1.704	0.316

Accepted Article

WT-4[18]	10.75	2547.26	982.301	27,383.045	10.559	2.695	0.499

†WT: Wallace Tree; RC: Ripple Carry adder.



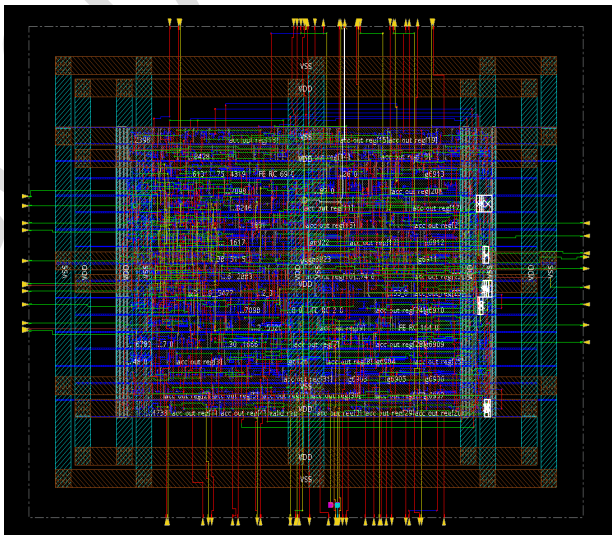


Figure 7. Placement-and-routing layout of the proposed AxMAC-Mul+ACC datapath.

Figure 8. Normalized (a) area-delay and (b) power-delay trade-off plots.

Figure 8 compares the normalized area-delay and power-delay trade-offs of the evaluated designs. The proposed AxMAC designs, particularly AxMAC-Mul+ACC, achieve the best trade-off by

simultaneously minimizing hardware cost and critical-path delay compared with the exact MAC and prior state-of-the-art MACs.

B. Error Analysis

To evaluate the arithmetic accuracy of the proposed AxMAC variants, Error Distance (ED), Mean Error Distance (MED), Normalized Mean Error Distance (NMED), and Mean Relative Error Distance (MRED) are calculated using Python-based simulation. AxMAC-Mul1 and AxMAC-Mul2 are exhaustively evaluated over all 2^{16} combinations of 8-bit operands, since their accumulation stage is exact. For AxMAC-ACC and AxMAC-Mul+ACC, due to the significantly larger input space, the error performance is evaluated using 100 000 uniformly random generated full-MAC input states. For each evaluated input combination, the error distance is defined as

$$ED = |\, Exact_{out} - Approx_{out} \,| \tag{12}$$

$$MED = \frac{1}{n}\sum_{i=1}^{n} ED_i \tag{13}$$

To calculate NMED, the Mean ED (MED) is first computed using (13). NMED is the average of the ED normalized with the highest error for all input combinations as (14), whereas MRED stands for the average of the relative errors for all input combinations as (15). The mathematical expressions for NMED and MRED are

$$NMED = \frac{1}{n}\sum_{i=1}^{n} \frac{ED_i}{ED_{max}} \tag{14}$$

$$MRED = \frac{1}{n}\sum_{i=1}^{n} \frac{ED_i}{Exact_{out\,i}} \tag{15}$$

Table 11. Error Comparison for Proposed MAC variants and State-of-the-Art Designs

Works	**MED**	**NMED**	**MRED**	**NoEB**
Exact (radix-8+WT+RC)†	-	-	-	16
AxMAC-Mul1	36.024	0.0028	0.0037	14.05
AxMAC-Mul2	57.38	0.0034	1.227	6.35
ICCD-MAC[36]	15331.91	0.1664	39.903	1.32
H-MAC[36]	98.34	0.080	0.0114	13.11
LEBZAM (7,13)[17]	209.403	0.3893	22.512	2.15
P_{AXC}MAC-8[37]	167.91	0.39	0.19	8.71
BMSE+WAT [38]	47.09	0.161	124.962	0.38
CORDIC[26]	271	0.0118	0.0025	15.29
Ours_7[15]	81.32	0.0531	87.304	0.19
WT-4[18]	97.65	0.912	0.478	7.70
Error Comparison using Monte Carlo				
Works	**MED**	**NMED**	**MRED**	**NoEB**

Accepted Article

AxMAC-ACC	42.495	0.0021	0.8944	7.39
AxMAC-Mul+ACC	91.624	0.0133	2.1824	4.68

†WT: Wallace Tree; RC: Ripple Carry adder.

Table 11 summarizes the MED, NMED, MRED, and NoEB of the proposed AxMAC variants and reference designs. AxMAC-Mul1 exhibits the lowest error among the proposed variants as approximation is confined to PPG. AxMAC-Mul2 and AxMAC-ACC introduce approximation in PPR and ACC, respectively, resulting in higher error metrics. AxMAC-Mul+ACC combines these approximation mechanisms and exhibits the highest error among the proposed variants. The reference designs, including ICCD-MAC, H-MAC, LEBZAM (7,13), P_{AXC}MAC-8, BMSE+WAT, CORDIC, Ours_7, and WT-4, exhibit different error characteristics according to their respective approximation mechanisms. Future work will extend the evaluation to actual CNN operand distributions across convolutional layers to further investigate layer-dependent error propagation. Figure 9 further illustrates the error–hardware relationships using NMED/MRED against PDP/ADP, highlighting the trade-offs among arithmetic accuracy, power, delay, and area across the evaluated designs.

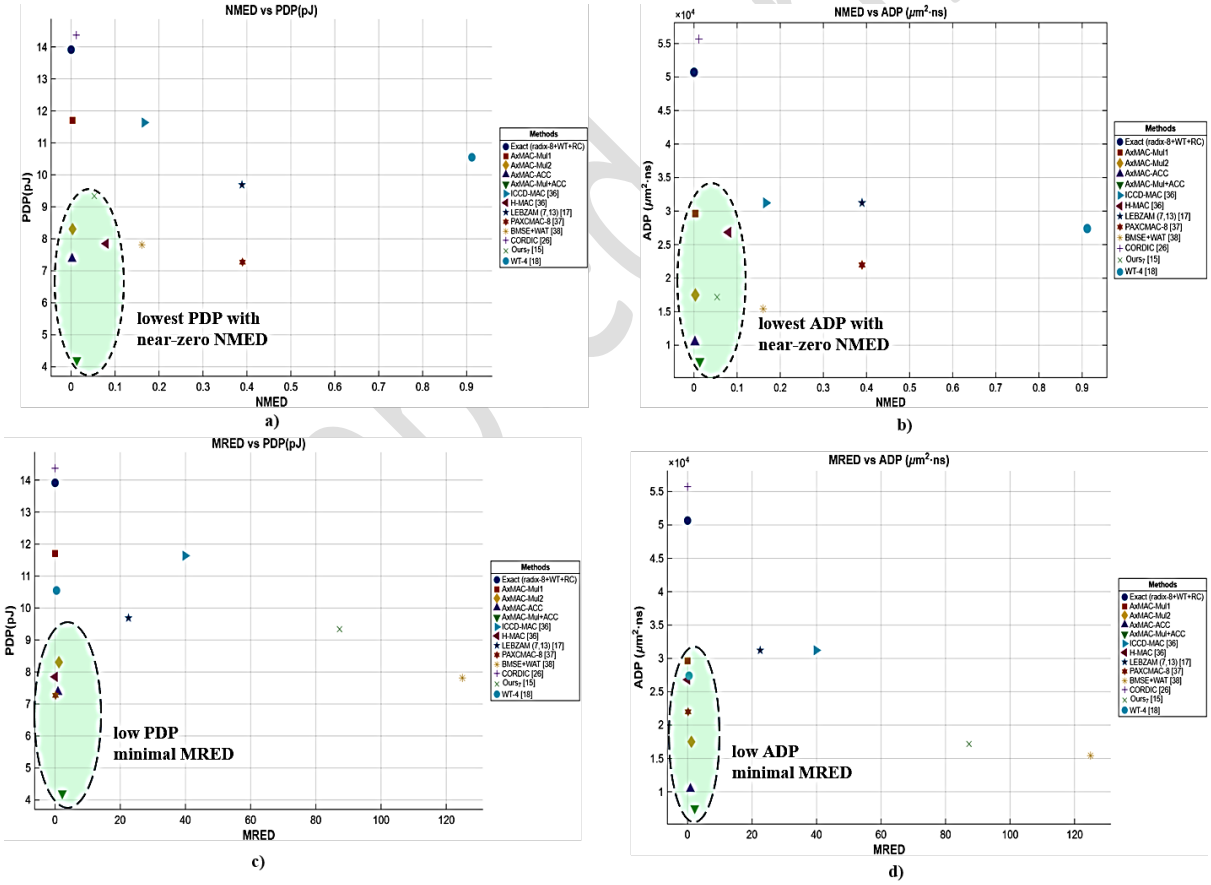


Figure 9. NMED-PDP, NMED-ADP, MRED-PDP, and MRED-ADP trade-off analysis of the evaluated MAC designs.

C. CNN Accuracy Evaluation

The proposed AxMAC variants are evaluated using pre-trained CNN models across MNIST, CIFAR-10, and ImageNet, including LeNet-5, AlexNet, VGG-13, ResNet-18, and GoogLeNet. The pre-trained models are used as the baseline networks, and no retraining or fine-tuning is performed after introducing the approximate MAC operations. To emulate the actual operating environment of a

hardware accelerator, both weights and activations are quantized into 8-bit fixed-point values for computation. To assess the effect of approximation on classification performance, each AxMAC variant is introduced at the selected convolutional layer indicated in Table 12, while the remaining network operations are retained as exact [39].

Table 12. CNN evaluation of the proposed AxMAC variants and prior approximate MAC designs.

Works	CNN models	Datasets	Chosen Layer	INT8 baseline	Accuracy (%)
AxMAC-Mul1	LeNet-5	MNIST	conv2	99.5	98.55
	AlexNet	ImageNet	features.6	69.95	68.91
	VGG-13	ImageNet	conv3_2	65.47	62.58
	ResNet-18	CIFAR-10	layer2.1.conv1	89.9	88.72
	GoogLeNet	CIFAR-10	inception3b.branch2.1	85.42	84.57
AxMAC-Mul2	LeNet-5	MNIST	conv2	99.5	98.22
	AlexNet	ImageNet	features.6	69.95	68.54
	VGG-13	ImageNet	conv3_2	65.47	62.2
	ResNet-18	CIFAR-10	layer2.1.conv1	89.9	88.46
	GoogLeNet	CIFAR-10	inception3b.branch2.1	85.42	83.61
AxMAC-ACC	LeNet-5	MNIST	conv2	99.5	97.82
	AlexNet	ImageNet	features.6	69.95	67.96
	VGG-13	ImageNet	conv3_2	65.47	61.95
	ResNet-18	CIFAR-10	layer2.1.conv1	89.9	88.61
	GoogLeNet	CIFAR-10	inception3b.branch2.1	85.42	83.85
AxMAC-Mul+ACC	LeNet-5	MNIST	conv2	99.5	96.69
	AlexNet	ImageNet	features.6	69.95	66.88
	VGG-13	ImageNet	conv3_2	65.47	59.82
	ResNet-18	CIFAR-10	layer2.1.conv1	89.9	87.05
	GoogLeNet	CIFAR-10	inception3b.branch2.1	85.42	82.43
Works	**CNN models**	**Datasets**	**Chosen Layer**	**INT8 baseline**	**Accuracy (%)**
ICCD-MAC [36]	AlexNet	ImageNet	NA	NA	66.61
	ResNet-18	ImageNet	NA	NA	61.51
H-MAC [36]	AlexNet	ImageNet	NA	NA	85.55
	ResNet-18	ImageNet	NA	NA	89.37

As Table 12 summarizes, the accuracy varies with the CNN model, selected layer, and approximation configuration. Prior works, including ICCD-MAC and H-MAC, report accuracy from convolutional-layer evaluation. The present evaluation considers selected-layer approximation rather than network-wide MAC replacement. Full network-wide deployment across all convolutional layers without retraining or fine-tuning is left for future work. The worst-case operating point in Table 12 is AxMAC-Mul+ACC applied to VGG-13 on ImageNet. Its accuracy is 59.82%, compared with 65.47% for the exact-arithmetic baseline evaluated under identical settings, which corresponds to a degradation of 5.65 percentage points (8.6% relative to the baseline). This is the largest accuracy degradation observed in the evaluation. The result shows that approximating both multiplication and accumulation can reduce CNN accuracy. Therefore, AxMAC-Mul+ACC may not be suitable for accuracy-critical applications, but it can be considered for error-tolerant applications where hardware savings are more important than the loss in accuracy. For large-scale classification tasks, AxMAC-Mul1, which limits the degradation on the same benchmark to 2.89 percentage points, is the recommended configuration.

VI. CONCLUSION

Accepted Article

This work presents a circuit-level optimization of multiply–accumulate (MAC) units for energy-efficient CNN inference by selectively introducing approximation in partial product generation, reduction, and accumulation stages. The proposed AxMAC architectures employ a radix-8 Booth scheme with AxPP1-based $\pm 3X$ elimination ($\pm 3X \rightarrow \pm 2X$), error-metric-guided approximate compressors, and a hybrid Han–Carlson parallel prefix accumulator to reduce logic complexity and switching activity while maintaining controlled accuracy. Across the four proposed variants, the designs achieve up to 43.5% delay reduction, 73.6% area savings, and 46.5% power reduction, yielding PDP and ADP improvements of up to 69.7% and 85.1% respectively, against an exact radix-8 Booth MAC. Under selected-layer approximation, AxMAC-Mul1 achieves inference accuracies between 84.57% and 98.55% across the five evaluated (model, dataset) pairs. The results show model- and layer-dependent accuracy degradation.

Future Work

Future work will proceed along four directions. First, reconfigurable and adaptive AxMAC architectures that adjust approximation and precision according to application- and layer-sensitivity. Second, network-wide replacement of the exact MAC across all convolutional layers of each evaluated CNN, including retraining- or calibration-based recovery of end-to-end accuracy. Third, empirical characterization of AxPP1 error cancellation on captured INT8 weight and activation tensors from each convolutional layer, to identify layers with residual bias from asymmetric weight distributions. Fourth, integration of the proposed AxMAC into systolic-array processing-element arrays of several sizes, with a fully documented synthesis flow, a component-level area breakdown, post-layout static timing analysis and parasitic-extracted power sign-off, together with evaluation on advanced technology nodes and Gemmini-class accelerator frameworks.

Acknowledgments

The authors would like to express their sincere gratitude to the Department of ECE, PSG Institute of Technology and Applied Research, for providing the necessary facilities and support to carry out this research work.

Author Contributions:

Balasaraswathi Subramanian: Conceptualization, methodology, design and implementation, simulations, result analysis, writing – original draft. **Padmapriya Subramanian:** Supervision, technical guidance, validation, manuscript review and editing.

Funding source: This research did not receive any grants from funding agencies in the public, commercial or non-profit sectors.

Conflict of interest

The authors declare that they have no conflict of interest.

Data Availability

Data will be made available on request.

REFERENCES

[1] H. Sharma *et al.*, “Bit fusion: Bit-Level dynamically composable architecture for accelerating deep neural networks,” *Proc. - Int. Symp. Comput. Archit.*, pp. 764–775, 2018, doi: 10.1109/ISCA.2018.00069.

[2] “TPU v4 | Google Cloud Documentation.” Accessed: Dec. 18, 2025. [Online]. Available:

Accepted Article

https://docs.cloud.google.com/tpu/docs/v4

[3] A. M. Dalloo, A. J. Humaidi, A. K. Al Mhdawi, and H. Al-Raweshidy, "Approximate Computing: Concepts, Architectures, Challenges, Applications, and Future Directions," *IEEE Access*, vol. 12, no. October, pp. 146022–146088, 2024, doi: 10.1109/ACCESS.2024.3467375.

[4] M. Traiola, A. Savino, and S. Di Carlo, "Probabilistic estimation of the application-level impact of precision scaling in approximate computing applications," *Microelectron. Reliab.*, vol. 102, no. June, p. 113309, 2019, doi: 10.1016/j.microrel.2019.06.002.

[5] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra, and G. DI Meo, "Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 67, no. 9, pp. 3021–3034, 2020, doi: 10.1109/TCSI.2020.2988353.

[6] B. G Gowda, P. H C, M. V N, and M. Rao, "A probabilistic approach to design inexact compressors for approximate booth multipliers," *Analog Integr. Circuits Signal Process.*, vol. 122, no. 2, pp. 1–23, 2025, doi: 10.1007/s10470-025-02327-x.

[7] A. Kumari and R. P. Palathinkal, "Design and Analysis of Energy Efficient Approximate Multipliers for Image Processing and Deep Neural Network," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 72, no. 2, pp. 854–867, 2025, doi: 10.1109/TCSI.2024.3479279.

[8] Y. Zhu, W. Liu, P. Yin, T. Cao, J. Han, and F. Lombardi, "Design, evaluation and application of approximate-truncated Booth multipliers," *IET Circuits, Devices Syst.*, vol. 14, no. 8, pp. 1305–1317, 2020, doi: 10.1049/iet-cds.2019.0398.

[9] J. Lee, H. Seo, H. Seok, and Y. Kim, "A Novel Approximate Adder Design Using Error Reduced Carry Prediction and Constant Truncation," *IEEE Access*, vol. 9, pp. 119939–119953, 2021, doi: 10.1109/ACCESS.2021.3108443.

[10] M. Pashaeifar, M. Kamal, A. Afzali-Kusha, and M. Pedram, "A Theoretical Framework for Quality Estimation and Optimization of DSP Applications Using Low-Power Approximate Adders," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 66, no. 1, pp. 327–340, 2019, doi: 10.1109/TCSI.2018.2856757.

[11] G. Raut, J. Mukala, V. Sharma, and S. K. Vishvakarma, "Designing a Performance-Centric MAC Unit with Pipelined Architecture for DNN Accelerators," *Circuits, Syst. Signal Process.*, vol. 42, no. 10, pp. 6089–6115, 2023, doi: 10.1007/s00034-023-02387-2.

[12] P. Thejaswini, G. Suresh, V. Chiraag, and S. Nandi, "Approximate CNN Hardware Accelerators for Resource Constrained Devices," *IEEE Access*, vol. 13, no. December 2024, pp. 12542–12553, 2025, doi: 10.1109/ACCESS.2025.3529668.

[13] B. B. Sarnala and S. R. Pillutla, "High-speed convolutional neural networks using parallel prefix adders," *AEU - Int. J. Electron. Commun.*, vol. 196, no. December 2024, p. 155791, 2025, doi: 10.1016/j.aeue.2025.155791.

[14] S. N. Tang and Y. S. Han, "A High-Accuracy Hardware-Efficient Multiply-Accumulate (MAC) Unit Based on Dual-Mode Truncation Error Compensation for CNNs," *IEEE Access*, vol. 8, pp. 214716–214731, 2020, doi: 10.1109/ACCESS.2020.3040366.

[15] C. W. Tung and S. H. Huang, "A High-Performance Multiply-Accumulate Unit by Integrating Additions and Accumulations into Partial Product Reduction Process," *IEEE Access*, vol. 8, pp. 87367–87377, 2020, doi: 10.1109/ACCESS.2020.2992286.

Accepted Article

[16] Q. Cheng *et al.*, "A Low-Power Sparse Convolutional Neural Network Accelerator With Pre-Encoding Radix-4 Booth Multiplier," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 70, no. 6, pp. 2246–2250, 2023, doi: 10.1109/TCSII.2022.3231361.

[17] M. E. Nojehdeh, L. Aksoy, and M. Altun, "Efficient hardware implementation of artificial neural networks using approximate multiply-accumulate blocks," *Proc. IEEE Comput. Soc. Annu. Symp. VLSI, ISVLSI*, vol. 2020-July, pp. 96–101, 2020, doi: 10.1109/ISVLSI49217.2020.00027.

[18] T. Li, H. L. Jiang, H. Mo, J. Han, L. B. Liu, and Z. G. Mao, "Approximate Processing Element Design and Analysis for the Implementation of CNN Accelerators," *J. Comput. Sci. Technol.*, vol. 38, no. 2, pp. 309–327, 2023, doi: 10.1007/s11390-023-2548-8.

[19] G. Di Meo, G. Saggese, A. G. M. Strollo, and D. De Caro, "Approximate MAC Unit Using Static Segmentation," *IEEE Trans. Emerg. Top. Comput.*, vol. 12, no. 4, pp. 968–979, 2024, doi: 10.1109/TETC.2023.3315301.

[20] M. Zhang, S. Nishizawa, and S. Kimura, "Area Efficient Approximate 4-2 Compressor and Probability-Based Error Adjustment for Approximate Multiplier," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 70, no. 5, pp. 1714–1718, 2023, doi: 10.1109/TCSII.2023.3257852.

[21] H. Chhajed, G. Raut, N. Dhakad, S. Vishwakarma, and S. K. Vishvakarma, "BitMAC : Bit-Serial Computation-Based Efficient Multiply-Accumulate Unit for DNN Accelerator," *Circuits, Syst. Signal Process.*, vol. 41, no. 4, pp. 2045–2060, 2022, doi: 10.1007/s00034-021-01873-9.

[22] E. Napoli, E. Zacharelos, A. G. M. Strollo, and G. Di Meo, "Approximate Full-Adders: A Comprehensive Analysis," *IEEE Access*, vol. 12, no. August, pp. 136054–136072, 2024, doi: 10.1109/ACCESS.2024.3463182.

[23] M. M. E. A. Da Rosa, G. Paim, P. U. L. Da Costa, E. A. C. Da Costa, R. I. Soares, and S. Bampi, "AxPPA: Approximate Parallel Prefix Adders," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 31, no. 1, pp. 17–28, 2023, doi: 10.1109/TVLSI.2022.3218021.

[24] X. Tang, Y. Li, W. Liu, and D. Shang, "Two Novel Approximate Radix-4 Booth Encoders for Efficient Signed Approximate Booth Multipliers," *IEEE Embed. Syst. Lett.*, vol. 18, no. 1, pp. 73–76, 2025, doi: 10.1109/LES.2025.3567482.

[25] S. Balasaraswathi and S. Padmapriya, "Hybrid Parallel Prefix Adders for Error-Tolerant Applications: Design and Implementation," in *2025 Second International Conference on Intelligent Technologies for Sustainable Electric and Communications Systems (iTech SECOM)*, Coimbatore, India: IEEE, 2025. doi: 10.1109/iTechSECOM64750.2025.11307240.

[26] B. Khurshid, "High-performance CORDIC-based approximate MAC architectures for FPGA platforms," *Integration*, vol. 101, no. December 2024, p. 102338, 2025, doi: 10.1016/j.vlsi.2024.102338.

[27] M. Esmali Nojehdeh and M. Altun, "Energy-Efficient Hardware Implementation of Fully Connected Artificial Neural Networks Using Approximate Arithmetic Blocks," *Circuits, Syst. Signal Process.*, vol. 42, no. 9, pp. 5428–5452, 2023, doi: 10.1007/s00034-023-02363-w.

[28] S. Immareddy, A. Sundaramoorthy, and A. Alagarsamy, "Design and implementation of hybrid (radix-8 Booth and TRAM) approximate multiplier using 15-4 approximate compressors for image processing application," *J. Real-Time Image Process.*, vol. 21, no. 2, pp. 1–11, 2024, doi: 10.1007/s11554-024-01427-7.

[29] A. Rafiq and M. Jenihhin, "An optimized design of delay-and energy-efficient Booth

Accepted Article

multiplier," *e-Prime - Adv. Electr. Eng. Electron. Energy*, vol. 9, no. July, p. 100698, 2024, doi: 10.1016/j.prime.2024.100698.

[30] L. Sayadi, S. Timarchi, and A. Sheikh-Akbari, "Two Efficient Approximate Unsigned Multipliers by Developing New Configuration for Approximate 4:2 Compressors," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 70, no. 4, pp. 1649–1659, 2023, doi: 10.1109/TCSI.2023.3242558.

[31] M. Ha and S. Lee, "Multipliers With Approximate 4-2 Compressors and Error Recovery Modules," *IEEE Embed. Syst. Lett.*, vol. 10, no. 1, pp. 6–9, 2018, doi: 10.1109/LES.2017.2746084.

[32] Z. G. Tasoulas, G. Zervakis, I. Anagnostopoulos, H. Amrouch, and J. Henkel, "Weight-Oriented Approximation for Energy-Efficient Neural Network Inference Accelerators," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 67, no. 12, pp. 4670–4683, 2020, doi: 10.1109/TCSI.2020.3019460.

[33] V. Trivedi, K. Lalwani, G. Raut, A. Khomane, N. Ashar, and S. K. Vishvakarma, "Hybrid ADDer: A Viable Solution for Efficient Design of MAC in DNNs," *Circuits, Syst. Signal Process.*, vol. 42, no. 12, pp. 7596–7614, 2023, doi: 10.1007/s00034-023-02469-1.

[34] S. Ullah, H. Schmidl, S. S. Sahoo, S. Rehman, and A. Kumar, "Area-Optimized Accurate and Approximate Softcore Signed Multiplier Architectures," *IEEE Trans. Comput.*, vol. 70, no. 3, pp. 384–392, 2021, doi: 10.1109/TC.2020.2988404.

[35] N. Ashar, G. Raut, V. Trivedi, S. K. Vishvakarma, and A. Kumar, "QuantMAC: Enhancing Hardware Performance in DNNs with Quantize Enabled Multiply-Accumulate Unit," *IEEE Access*, vol. 12, no. February, pp. 43600–43614, 2024, doi: 10.1109/ACCESS.2024.3379906.

[36] H. Xiao, H. Xu, X. Chen, Y. Wang, and Y. Han, "Fast and High-Accuracy Approximate MAC Unit Design for CNN Computing," *IEEE Embed. Syst. Lett.*, vol. 14, no. 3, pp. 155–158, 2022, doi: 10.1109/LES.2021.3137335.

[37] S. Garg, K. Monga, N. Chaturvedi, and S. Gurunarayanan, "Tunable Energy-Efficient Approximate Circuits for Self-Powered AI and Autonomous Edge Computing Systems," *IEEE Access*, vol. 13, no. February, pp. 43607–43630, 2025, doi: 10.1109/ACCESS.2025.3548458.

[38] F. U. D. Farrukh, C. Zhang, Y. Jiang, Z. Zhang, Z. Wang, Z. Wang, and H. Jiang, "Power Efficient Tiny Yolo CNN Using Reduced Hardware Resources Based on Booth Multiplier and WALLACE Tree Adders," *IEEE Open J. Circuits Syst.*, vol. 1, pp. 76–87, 2020, doi: 10.1109/OJCAS.2020.3007334.

[39] G. Zervakis, F. Frustaci, O. Spantidi, I. Anagnostopoulos, H. Amrouch, and J. Henkel, "Leveraging Highly Approximated Multipliers in DNN Inference," *IEEE Access*, vol. 13, pp. 47897–47911, 2025, doi: 10.1109/ACCESS.2025.3550520.

Copyright © 2026 by the Authors. This is an open access article distributed under the Creative Commons Attribution (CC BY) License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Arrived: 30.01.2026
Accepted: 09.10.2026

Archive Article