electronics

Article

INVCAM: An Inverted Compressor-Based Approximate Multiplier

Kimia Darabi [1], Sahand Divsalar [1], Shaghayegh Vahdat [1,*], Nima Amirafshar [2] and Nima TaheriNejad [2]

[1] School of Electrical and Computer Engineering, University of Tehran, Tehran P.O. Box 14395-515, Iran; kimia.darabi@ut.ac.ir (K.D.); s.divsalar96@ut.ac.ir (S.D.)
[2] Institute of Computer Engineering (ZITI), Heidelberg University, 69120 Heidelberg, Germany; nima.amirafshar@ziti.uni-heidelberg.de (N.A.); nima.taherinejad@ziti.uni-heidelberg.de (N.T.)
* Correspondence: vahdat_s@ut.ac.ir

Abstract

In this paper, a novel 8-bit approximate multiplier, called INVCAM, is proposed in which the inverted partial products (PPs) are summed using approximate 4:2 compressors. This design allows for flexibility in applying approximations, enabling the multiplier to be tuned to the specific accuracy requirements of different applications. By adjusting the number of approximated bits, the multiplier can operate with a better balance between desirable hardware characteristics and acceptable levels of error. Our approach ensures that INVCAM is customizable for a wide range of applications. The results indicate that INVCAM reduces delay, power, and area by up to 21.5%, 70.0%, and 57.6%, respectively, compared to the state-of-the-art (SoTA) approximate multipliers within its mean relative error distance (MRED) range, and by 42.4%, 80.1%, and 68%, compared to an exact multiplier. The efficacy of INVCAM is evaluated in image processing and deep neural network (DNN) applications. The images processed by different configurations of INVCAM have PSNR and SSIM values greater than 28.9 dB and 0.81, respectively, which manifests the acceptable quality of the processed approximate images. In the DNN application, the classification accuracy of the models implemented using INVCAM(7) is within 0.6% of the original model accuracy. When the number of approximate bits is increased to nine, less than 5% accuracy reduction is observed compared to an exact model, while the power-delay-area product of the multiplier improves by 46%.

Keywords: approximate multiplier; 4:2 compressor; inverted partial products; image processing; deep neural networks

Academic Editor: Paolo Stefano Crovetti

Received: 8 December 2025
Revised: 27 December 2025
Accepted: 31 December 2025
Published: 2 January 2026

1. Introduction

Due to the widespread use of portable electronic devices with limited power budgets, such as cell phones, tablet PCs, different internet of things (IoT) devices, and wearable devices, reducing power consumption has become a remarkable challenge for hardware designers. Several approaches, such as power gating [1], clock gating [2], and dynamic voltage-frequency scaling (DVFS) [3], have been introduced to address this issue. Compression is another method that can be used to reduce the number of required arithmetic operations, which directly affects the consumed power. For instance, Refs. [4,5] have investigated classification based on compressed representations of images. In these approaches, the original image is first projected into a lower-dimensional domain using matrix multiplications, and the classification is then performed directly on the compressed result without

https://doi.org/10.3390/electronics15010216

full image reconstruction. This reduces data movement and computational complexity, making such methods attractive for hardware-efficient implementations.

Some of the common applications, such as image processing and deep neural networks (DNNs), that run on the aforementioned devices are intrinsically error tolerant. It means that they can maintain an acceptable performance even when their arithmetic operations are performed with a certain amount of error [6]. The ubiquity of these applications, which include a large number of multiplication and addition operations, has introduced approximate computing as a popular research topic in the field of low-power hardware design [7]. In other words, exact arithmetic units can be replaced with approximate ones, which results in significant improvements in hardware design parameters (i.e., delay, power, and area) without considerable loss in the overall performance of the system. Approximation can be applied to different abstraction levels of hardware design, including the transistor level [8,9], circuit level [10], gate level [11], architecture level [12], and algorithmic level [13].

Amongst all arithmetic operations used in the error-tolerant applications, multiplication is the best candidate for introducing approximation, because multipliers are great in number, consume significant power, and have a longer critical path compared to adders. A conventional multiplier consists of three stages: partial product generation (PPG), partial product reduction (PPR), and accumulation (ACC). In the PPG stage, the PPs are commonly generated using a simple AND operation, whereas in the PPR stage, they are reduced to two rows using adders, and the results are summed in the ACC stage to obtain the final output.

Approximation can be applied to any one of these stages or a combination of them. For instance, in approximate Booth multipliers, the PPs are commonly generated in an approximate fashion [14]. To introduce approximation in the PPR stage, generated PPs can be approximately added using approximate full-adders (FAs) [15] or 4:2 compressors [16]. The final accumulation stage can also be implemented using approximate adders in which the less significant columns are approximately calculated while the more significant columns are added exactly to keep the introduced error in an acceptable range [17]. Amongst the mentioned strategies for designing approximate multipliers, utilizing approximate compressors is more common due to the complexity of these components, which can be significantly reduced by applying approximation [18].

In this paper, we propose a novel approximate multiplier that uses approximate 4:2 compressors for reducing the tree consisting of inverted PPs. The main contributions of this paper can be summarized as follows:

- Integrating NAND-based PPG with a zero-count-driven PPR stage to simplify arithmetic operations and reduce hardware complexity.
- Designing lightweight approximate inverted adder structures (inverted 4:2 compressor (AIC), half-adder (AIHA), and full-adder (AIFA)) specifically optimized for the zero-count PPR, providing reduced gate count and delay.
- Employing a carry-free accumulation approach tailored to the proposed architecture and enhancing it with an error correction module (ECM) to efficiently mitigate errors from truncated carries.
- Presenting a flexible approximate multiplier architecture with tunable approximation levels, which achieves up to 23%, 48%, and 38% improvements in delay, power, and area compared to prior designs with comparable accuracy.

The rest of this paper is organized as follows. A selection of relevant approximate multipliers from the literature is reviewed in Section 2. The proposed approximate multiplier, named INVCAM, and its hardware implementation are presented in Section 3. The hardware design parameters, as well as error metrics of INVCAM, are compared with those of the state-of-the-art (SoTA) designs in Section 4. The efficacy of INVCAM in image processing and DNN applications is also assessed in Section 4. Finally, this paper is concluded in Section 5.

2. Related Work

In this section, a brief review of some of the previous efforts in designing approximate multipliers is provided. One major approach is proposing approximate 4:2 compressors and utilizing them in the PPR stage of an approximate multiplier (e.g., [1]). Since the conventional exact 4:2 compressor comprises two exact FAs, a straightforward method for designing an approximate 4:2 compressor is employing approximate FAs in the structure of the compressor [15]. As another method, one may approximate the truth table of the compressor with the aim of reducing its hardware implementation complexity (e.g., [17–20]). These structures can lead to significant improvements in hardware parameters (i.e., delay, power, and area). For instance, the compressors proposed in [17] can reduce the energy-delay product (EDP) by up to 99% when compared to an exact compressor, and the one proposed in [21] uses only four logic gates (i.e., NOR, XOR, and OR gates) to implement a compressor with an error rate of 70/256.

It is worth noting that in most compressor designs, the produced error depends on the order in which the PPs are applied to the compressor input terminals. Therefore, in Refs. [22,23], an input reordering module was employed to reduce the error probability caused by different input connections. Additionally, the probability of the multiplier inputs in different applications may not follow a uniform distribution. This may significantly affect the performance of the multiplier in the considered application. Therefore, one may consider the input probabilities when designing approximate compressors [16]. For instance, an approximate multiplier-accumulator (MAC) unit for convolutional neural network (CNN) accelerators was proposed in [24], incorporating hybrid compressors designed based on the statistical distribution of CNN data. This design achieves a superior tradeoff between accuracy, latency, and power, with 23% higher accuracy compared to the prior designs.

While employing simple 4:2 compressors may lead to significant improvements in hardware design parameters of multipliers, it can also inflict large errors on their outputs. To improve the accuracy while imposing small hardware overheads, one may utilize ECMs in specific bit positions of the multiplier [17,25,26]. Using ECMs is not limited to approximate compressor-based multipliers. In [27], low-power approximate 2-bit multipliers are utilized in the structure of a larger multiplier, called AxRMs, and ECMs are used to reduce their error.

Furthermore, the compressors can be designed such that they compensate for each other′s error. For instance, in [6], Sayadi et al. proposed a fast and efficient approximate multiplier with two-stage error-compensating compressors, one with a negative mean error (i.e., -0.42) and the other with a positive mean error (i.e., 0.48), to find a balance between efficiency and accuracy. This design achieved an 82% improvement in power–delay product (PDP) compared to an exact 8-bit multiplier while maintaining acceptable accuracy.

To enable runtime switching between exact and approximate operating modes in accuracy-configurable multipliers, dual-quality 4:2 compressors with different levels of accuracy can also be used. For instance, the dual-quality compressors proposed in [1] achieved up to 93.8% and 5.7% reductions in EDP in approximate and exact modes compared to an exact Dadda multiplier, respectively.

To reduce the number of PPs and improve the hardware parameters with a small effect on the average multiplier error, truncating the less significant columns of PPs is also a common method [26,28]. However, this method can lead to large relative errors when the input operands of the multiplier have small values. To handle this issue, one may truncate the input operands based on the position of their leading one bit (LOB) and perform the arithmetic operations on the small bit-width truncated operands. TOSAM [29], DR-ALM [30], and AMCAL [31] are examples of LOB-based approximate multipliers. Furthermore, Nambi et al. [32] developed a decoder-based multiplier (DeBAM) in which a 2-bit decoder was used to reduce the number of generated PPs, leading to lower hardware complexity and power consumption. This multiplier achieved a 41% power reduction compared to exact multipliers while maintaining low error rates.

Employing Booth encoding can also reduce the number of PPs significantly. However, the encoder implementation becomes complex for high radixes and imposes large hardware overheads. In these structures, approximate encoders can be utilized to improve hardware design parameters. For instance, Mohanty [14] proposed hybrid designs for radix-64 and radix-256 Booth encoders. In Booth multipliers, the PPs can also be accumulated approximately using approximate compressors and counters [33].

3. Proposed INVCAM and Its Hardware Implementation

In this section, we propose an approximate multiplier, named INVCAM, which utilizes less complex approximate modules in the PPR stage to reduce hardware overhead. Inverter, NAND, NOR, and XOR/XNOR gates are the basic logic cells, which can implement any Boolean function. The aim of this paper is to design simplified approximate adders (i.e., HA, FA, and 4:2 compressor) using a smaller number of basic logic cells. Reducing gate count is one of the conventional methods used for assessing the effectiveness of approximate arithmetic units, which has been used in [16,21,23].

In a conventional multiplier, PPs are generated using AND gates [34]. INVCAM, however, applies an efficient logic-level optimization to reduce the transistor count by generating PPs using NAND gates, similar to what has been carried out in [34]. Thus, in the PPR stage of INVCAM, the number of zeros needs to be counted as opposed to the number of ones. Therefore, all the arithmetic blocks used in the PPR stage of INVCAM should be designed such that they generate inverted outputs based on their inverted inputs. In other words, the approximate HA, FA, and compressors are constructed with the understanding that their inputs represent inverted values. As a result, these components are designed to produce inverted approximate outputs, which can be directly combined with the remaining PPs in subsequent reduction stages. They may be built from various basic logic cells such as NOR, NAND, and XOR, but we try to use a minimal number of cells while ensuring compatibility with the inverted inputs and outputs. We also propose new HA and carry look-ahead adder (CLA) configurations for use in the ACC stage to handle inverted inputs and produce regular (not inverted) outputs.

The dot diagram of an 8-bit INVCAM with eight approximate columns, denoted by INVCAM(8), is depicted in Figure 1. As shown in the figure, this multiplier requires three stages to generate the final multiplication result. The dots in the first stage show the inverted PPs generated using NAND gates. In the PPR and ACC stages, the conventional full adder (FA) and 4:2 compressor (COM), as well as the proposed approximate inverted 4:2 compressor (AICOM) and approximate inverted HA (AIHA), are used, the structures of which will be explained in detail in this section.

It is worth noting that the INVCAM design shown in Figure 1 demonstrates the use of approximate adders in the eight rightmost columns (the least significant ones). However, this design can be adjusted to accommodate different numbers of approximate bits. Therefore, different configurations of INVCAM can be obtained by considering a trade-off between accuracy and hardware design parameters (i.e., delay, power, and area). By increasing the number of columns in which approximate adders are used, hardware design parameters can be improved while the accuracy deteriorates. To switch amongst different numbers of approximate bits and configurations, we can slide the vertical dotted line (see Figure 1) in the horizontal direction through the inverted PPs. The results concerning the trade-off between the accuracy and hardware design parameters for different configurations of INVCAM (with 7 to 12 approximate bits) will be provided in Section 4.

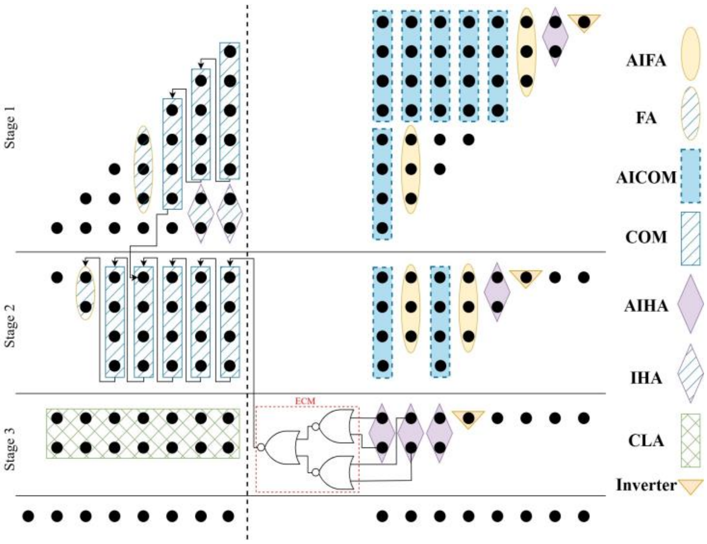


Figure 1. Dot diagram and reduction levels of an 8-bit INVCAM with eight approximate columns (i.e., INVCAM(8)). Dots denote partial products, and arrows indicate carries propagated to the next stage.

As shown in Figure 1, the proposed approximate multiplier consists of several adder blocks, including exact and approximate FA, HA, and 4:2 compressors, as well as their inverted structures, which generate the inverted outputs based on the applied inverted inputs. In the rest of this section, these adder blocks and their hardware implementations will be explained in detail.

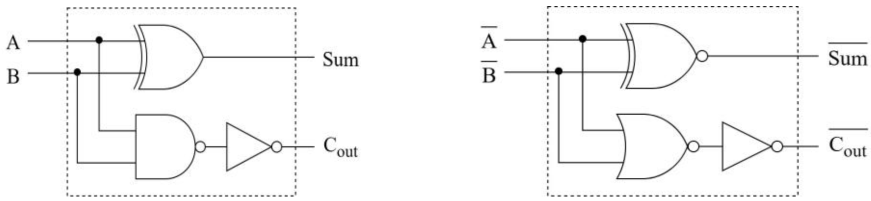
3.1. Exact Inverted Adder Block

Since all inputs of the adder blocks are inverted, the outputs of these blocks should also be inverted for further utilization in the subsequent reduction levels. Assuming that the inputs and outputs of a conventional HA are denoted by (A, B) and $(Carry, Sum)$, the inputs and outputs of the inverted HA (IHA) become $(\overline{A}, \overline{B})$ and $(\overline{Carry}, \overline{Sum})$, respectively, which can be calculated using the following equations:

$$\overline{Sum} = \overline{\left(A \bigoplus B\right)} = \overline{\overline{A} \bigoplus \overline{B}}, \tag{1}$$

$$\overline{Carry} = \overline{(A \cdot B)} = \overline{A} + \overline{B}. \tag{2}$$

The gate-level structures of the conventional HA and the IHA are illustrated in Figure 2.



(**a**) (**b**)

Figure 2. Gate-level structure of (**a**) a conventional HA and (**b**) the proposed IHA.

In the case of FAs, the inverted outputs are automatically generated since the outputs of FAs are self-dual functions [35]. In other words, the function of complementary inputs is the complement of the function. As a result, the structures of the conventional FA and the inverted FA are identical.

A conventional exact 4:2 compressor consists of two FAs connected to each other, as shown in Figure 3a. The internal output carry signal of a compressor (i.e., C_{out}) is applied to the next compressor as its input carry (i.e., C_{in}). The structure of an exact inverted 4:2 compressor (EIC) is illustrated in Figure 3b, which receives inverted inputs and produces inverted outputs. The figure shows that the internal structure of the conventional 4:2 compressor and its inverting version are the same, and their difference originates from their inputs, which lead to inverted outputs.

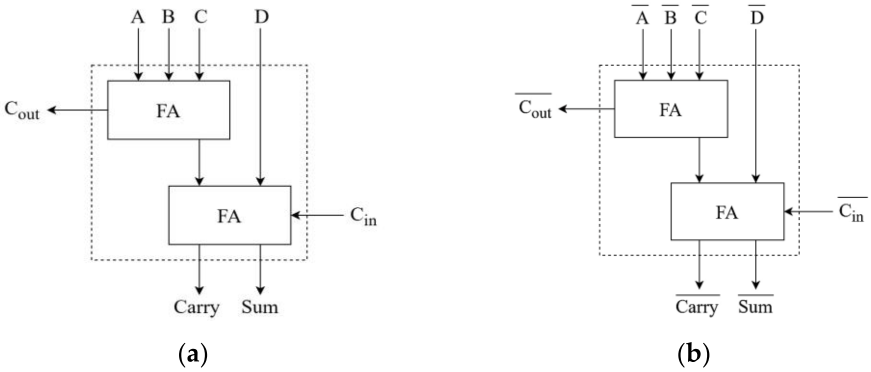


Figure 3. (**a**) Conventional and (**b**) inverted exact 4:2 compressor. Signals A–D and Cin denote the compressor inputs, while Sum denotes the output sum signal and C_{out} and Carry represent the output carry signals.

3.2. Proposed Approximate Inverted 4:2 Compressor (AICOM)

In most approximate 4:2 compressors, no internal carry signal (i.e., C_{out}) is propagated between adjacent compressors. Therefore, they have only four inputs (no C_{in}) and two outputs (i.e., $Carry, Sum$). The truth table of the proposed AICOM is provided in Table 1, in which Out_{APX} and $\overline{Out}_{APX}$ represent $(Carry, Sum)$ and $(\overline{Carry}, \overline{Sum})$ combinations in the approximate mode, whereas Out shows the exact result. Furthermore, $Prob$ shows the occurrence probability of each case, assuming that the inputs of the compressor (i.e., $\overline{A}, \overline{B}, \overline{C}, \overline{D}$) are the inverted PPs of the multiplier. Assuming the multiplier input bits are completely random and independent, the inverted PPs become "1" ("0") with the probability of 3/4 (1/4). Therefore, the probability that the '1111' input is applied to an AICOM equals $(3/4)^4$. Experimental results for practical applications also attest to the reliability of these probabilities. Deriving all NAND-based (i.e., inverted) PPs generated by multiplication operations in the first layer of the VGG-16 CNN model used for inference on the CIFAR-10 test dataset (including 10,000 images) demonstrates the appearance of '1' ('0') bits with a probability of 79.5% (20.5%), which has an acceptable proximity to 75% (25%). The error (Err) can also be defined as

$$Err = Out - Out_{APX}. \tag{3}$$

As presented in Table 1, AICOM is designed such that the error becomes zero for highly probable input combinations. Furthermore, the mean error $(MErr)$ of the proposed AICOM is small and can be found as

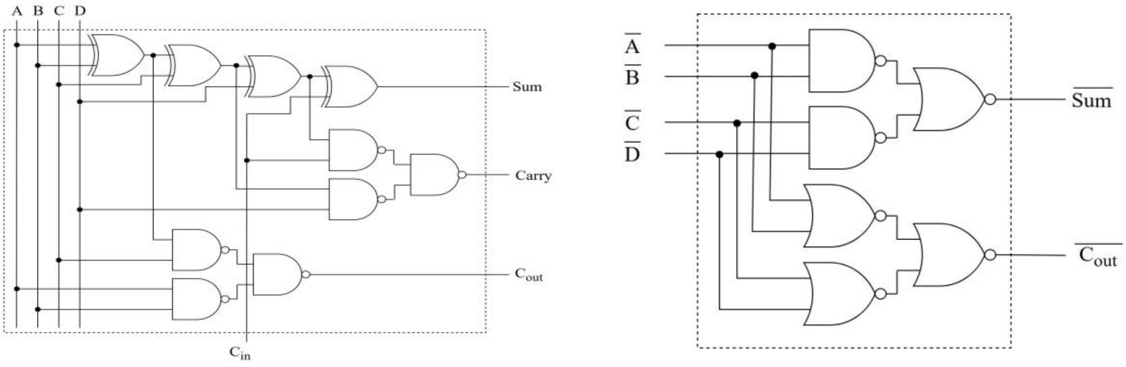
$$MErr = \sum_{i=1}^{16} Err_i \times Prob_i = \frac{-9+9+9+9+9-9+1}{256} = 0.074, \tag{4}$$

assuming completely random and independent input operands. However, the *MErr* for the mentioned practical case (first layer of VGG-16) can be calculated as 0.056, which is even less than the baseline mean error obtained in (4).

Table 1. Truth table of AICOM.

$\overline{A}$	$\overline{B}$	$\overline{C}$	$\overline{D}$	Out	Out_{APX}	$\overline{Out}_{APX}$	Err	$Prob$
1	1	1	1	00	00	11	0	81/256
1	1	1	0	01	01	10	0	27/256
1	1	0	1	01	01	10	0	27/256
1	1	0	0	10	11	00	−1	9/256
1	0	1	1	01	01	10	0	27/256
1	0	1	0	10	01	10	+1	9/256
1	0	0	1	10	01	10	+1	9/256
1	0	0	0	11	11	00	0	3/256
0	1	1	1	01	01	10	0	27/256
0	1	1	0	10	01	10	+1	9/256
0	1	0	1	10	01	10	+1	9/256
0	1	0	0	11	11	00	0	3/256
0	0	1	1	10	11	00	−1	9/256
0	0	1	0	11	11	00	0	3/256
0	0	0	1	11	11	00	0	3/256
0	0	0	0	100	11	00	+1	1/256

The overall structure can be implemented using a small number of basic logic cells to reduce power and area. The gate-level implementation of AICOM and that of a conventional 4:2 compressor are depicted in Figure 4. As shown in the figure, the number of NAND and XOR gates has reduced from 6 and 4 to 2 and 0, respectively, in AICOM compared to an exact compressor, while the number of NOR gates has increased to 4.



(**a**) (**b**)

Figure 4. Gate-level structure of (**a**) conventional 4:2 compressor and (**b**) the proposed AICOM.

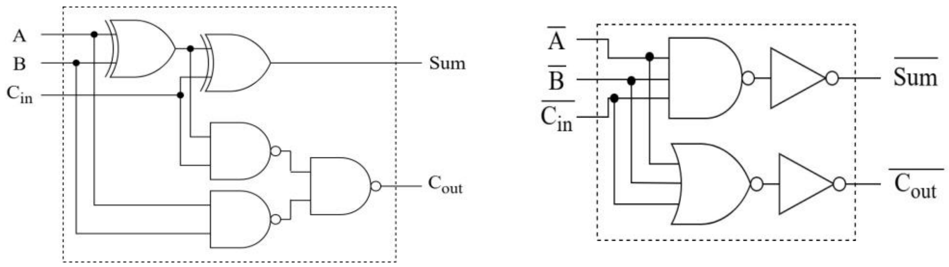
3.3. Proposed Approximate Inverted Full Adder (AIFA)

AIFA is proposed to be utilized in the least significant bit positions of the multiplier to reduce the delay and complexity of the structure. The outputs of this module are

$$\overline{Sum} = \overline{A} \cdot \overline{B} \cdot \overline{C}, \tag{5}$$

$$\overline{Carry} = \overline{A} + \overline{B} + \overline{C}. \tag{6}$$

Figure 5 displays the gate-level structures of a conventional FA alongside the proposed AIFA, which shows how the hardware of the FA is simplified in AIFA compared to the exact design. We see in the figure that the number of NAND and XOR gates has reduced from 3 and 2 to 1 and 0, respectively, in AIFA compared to an exact FA, while the number of inverter and NOR gates has increased to 2 and 1, respectively.



(**a**) (**b**)

Figure 5. Gate-level structure of (**a**) conventional FA and (**b**) the proposed AIFA.

3.4. Proposed Approximate Adder for ACC Stage

As mentioned previously, all of the adder blocks used in the PPR stage are designed such that they generate inverted outputs when inverted inputs are applied to them. Therefore, the inputs of the final addition step, i.e., the ACC stage, are inverted as well. However, the adder used in this stage should produce the multiplication result (not its inverted form). Since two m-bit operands are applied to the adder used in the ACC stage, the carry signal must propagate between different columns, which degrades the computation speed. Therefore, another level of approximation is introduced at this stage by employing approximate inverted half adders (AIHAs) for the LSBs, which truncate the carry propagation chain and generate the approximate non-inverted *sum* signal (i.e., sum_{APX}) using the inverted inputs as follows:

$$Sum_{APX} = \overline{\overline{A}\cdot\overline{B}} \tag{7}$$

To maintain acceptable accuracy, an exact CLA is used for calculation of the MSBs of the product. This CLA accepts inverted inputs and produces regular outputs. In this CLA, all parts remain unchanged except for the part producing the generate (G) signal for each column, which is originally computed using AND gates and is now implemented using NOR gates to support inverted inputs. Finally, in order to mitigate the error caused by the approximate computation of the lower significant bits and truncation of the carry chain, we generate an error correction (EC) signal by the ECM and apply it as the input carry of the rightmost exact compressor of Stage 2, as shown in Figure 1. The gate-level structure of the ECM is depicted in Figure 6, in which $\overline{X_1}$ and $\overline{X_2}$ show the elements in the $(n-1)^{th}$ column of the final stage of INVCAM(n), while $\overline{X_3}$ and $\overline{X_4}$ represent the elements in the n^{th} column. For simplicity, the idea behind designing ECM is explained based on the original values of the signals (i.e., X_1, X_2, X_3, and X_4). When both elements in a column are equal to 1 (e.g., $X_1 = X_2 = 1$ or $X_3 = X_4 = 1$), the generated carry for the corresponding column is deterministically equal to "1". To mitigate the error introduced by the approximate HAs in this stage, where the output is underestimated due to carry chain truncation, ECM strategically overestimates the input carry for the $(n+1)^{th}$ column, ensuring a carry is generated whenever either the $(n-1)^{th}$ or the n^{th} column generates a carry. Since the carry of the $(n-1)^{th}$ and n^{th} columns can be found as $X_1 \cdot X_2$ and $X_3 \cdot X_4$, respectively, the compensation term can be found as $(X_1 \cdot X_2) + (X_3 \cdot X_4)$. However, since all internal signals in INVCAM structures are generated in inverted form, the compensation term of ECM should be inverted as well. Therefore, the output of ECM can be found as $\overline{(X_1 \cdot X_2) + (X_3 \cdot X_4)}$, which is equivalent to $\overline{\left(\overline{\overline{X_1}+\overline{X_2}}\right)+\left(\overline{\overline{X_3}+\overline{X_4}}\right)}$ and can be implemented using three two-input NOR gates, as is shown in Figures 1 and 6.

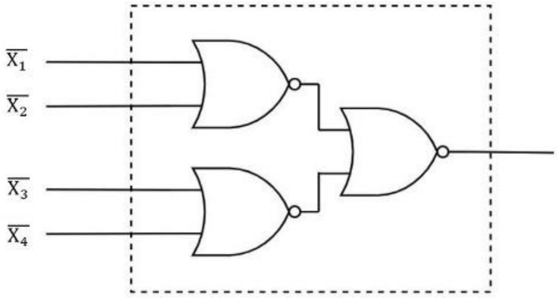


Figure 6. Gate-level structure of ECM.

3.5. 16-Bit Multiplier

A 16-bit multiplier can be constructed by partitioning each operand into an 8-bit high part (A_H) and an 8-bit low part (A_L):

$$A \times B = 2^{16}A_H B_H + 2^8(A_L B_H + A_H B_L) + A_L B_L \tag{8}$$

A 16-bit multiplier can be built using four 8-bit multipliers. As illustrated in Figure 7, the most significant product term ($O_{HH} \approx A_H B_H$) is generated using the proposed 8-bit approximate multiplier (i.e., INVCAM(m)), ensuring that the most significant portion of the result benefits from the highest accuracy.

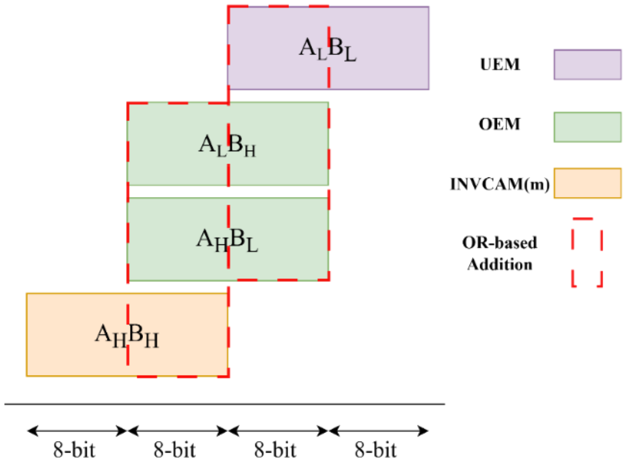


Figure 7. Overall structure of INVCAM16(m).

To further improve hardware efficiency, the terms ($A_L B_H$, $A_H B_L$, and $A_L B_L$) are computed using two complementary approximate multipliers:

- An overestimating multiplier (OEM), constructed solely from overestimating adders.
- An underestimating multiplier (UEM), constructed solely from underestimating adders.

OEM (UEM) outputs are, on average, greater than (less than) those of an exact multiplier. OEM and UEM both use NAND operations for the addition of inverted partial products along each column (equivalent to performing OR operations on regular and non-inverted partial products), but they differ in how they assign outputs. In UEM, the result of the NAND operation on the partial products of the i^{th} column ($2 \leq i \leq 15$) is directly assigned to the i^{th} output bit of the multiplier, whereas in OEM, this result is assigned to the $(i+1)^{th}$ output bit. The OEM is used to generate $O_{LH} = A_L B_H$ and $O_{HL} = A_H B_L$, which can partially compensate for the underestimation introduced by 8-bit INVCAM. The least significant product term ($O_{LL} \approx A_L B_L$) is generated using UEM, as this term has the smallest impact on the final accuracy and can compensate for the overestimation error caused by the OEMs for small input values.

To obtain the final 32-bit result, we must compute $2^{16}O_{HH} + 2^{8}(O_{LH} + O_{HL}) + O_{LL}$. To avoid the area overhead of wide 32-bit adders, we employ a lightweight column-wise reduction method using OR gates, which significantly simplifies accumulation while maintaining acceptable accuracy. It is worth noting that addition using OR gates underestimates the final output, which can also compensate for the overestimation error caused by OEMs. Higher bit-width multipliers (e.g., N-bit) can be recursively implemented using lower bit-width structures (e.g., $N/2$-bit). In this paper, the 8-bit INVCAM architecture with m approximate bits is referred to as INVCAM(m). The corresponding 16-bit multiplier, which utilizes INVCAM(m) for the multiplication of the most significant segments of the input operands (i.e., generating O_{HH}), is denoted as INVCAM16(m).

4. Results and Discussion

In this section, first, the results for some of the common error metrics are provided for different configurations of INVCAM and other SoTA approximate multipliers. Next, their hardware characteristics (i.e., delay, power consumption, and area) are compared with those of the other SoTA approximate and exact multipliers. Finally, the efficacy of INVCAM is assessed by using it in two different image-processing applications (i.e., image multiplication and Sobel edge detection) and well-known DNN models used for image classification applications.

4.1. Error Metrics Evaluation

To evaluate the accuracy of the 8-bit INVCAM, common error metrics are computed and compared with those of the other SoTA approximate multipliers. The considered error metrics are defined below for an N-bit multiplier.

- Error Rate (-ER) is the ratio of the number of erroneous results to the total number of cases.

$$ER = \frac{Number\ of\ erroneous\ results}{2^{2N}} \tag{9}$$

- Error distance (ED) is the absolute difference between the exact (P) and approximate (P^*) results.

$$ED = |P - P^*| \tag{10}$$

- ED_{max} is the maximum ED across all input combinations.

$$EDmax = \max\left\{ ED(i) \middle| 1 \le i \le 2^{2N} \right\} \tag{11}$$

- Mean error distance (MED) is the average of the EDs over all input combinations.

$$MED = \frac{1}{2^{2N}} \sum_{i=1}^{2^{2N}} |ED_i| \tag{12}$$

- Relative error distance (RED) normalizes the error distance with respect to the exact value.

$$RED = \frac{|P - P^*|}{P} \tag{13}$$

- Mean relative error distance ($MRED$) is the average of the REDs over all input combinations.

$$MRED = \frac{1}{2^{2N}} \sum_{i=1}^{2^{2N}} RED_i \tag{14}$$

- Normalized mean error distance ($NMED$) normalizes MED with respect to the maximum output value (i.e., $\left(2^N - 1\right)^2$). $NMED$ allows us to compare approximate multipliers with different bit widths.

$$NMED = \frac{MED}{\left(2^N - 1\right)^2} \tag{15}$$

In the rest of this paper, INVCAM(m) represents an 8-bit INVCAM in which approximate adders are utilized in the "m" least significant columns. Figure 8 illustrates the trend of $MRED$ and $NMED$ changes according to the variations of m. The highest acceptable value for m is 13, as 4:2 compressors are not used in the remaining more significant columns (three leftmost columns), according to Figure 1. As expected, $MRED$ and $NMED$ increase with an increase in the number of approximate bits. However, the rate of $MRED$ and $NMED$ changes decreases for larger m values. For instance, when m is increased from 7 to 8, the value of $MRED$ increases by 94%, while only a 17% increase is observed in $MRED$ when m changes from 12 to 13.

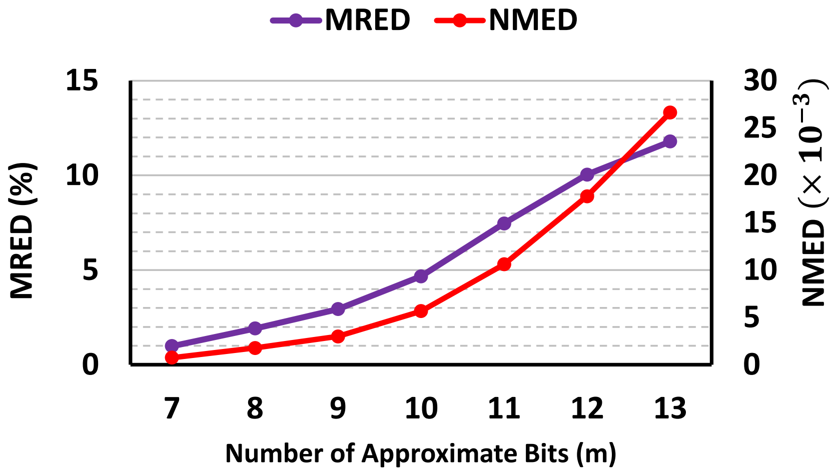


Figure 8. $MRED$ and $NMED$ of INVCAM(m) according to the number of approximate bits (m).

Table 2 presents the ER, $MRED$, and $NMED$ of different configurations of 8- and 16-bit INVCAM, while these parameters for other SoTA approximate 8-bit multipliers introduced in Sections 1 and 2 are presented in Table 3. The considered SoTA approximate multipliers are denoted as Mi-j from here onward, where i is the reference number and j is the number of the design presented in each previous work, if more than one design is presented. As is seen in Tables 2 and 3, increasing the number of approximate bits from 7 to 12 in INVCAM leads to configurations with lower $MRED$ than 15, 13, 8, 6, 3, and 1 of the 20 investigated SoTA approximate multipliers, respectively. This can be used as a criterion to decide which configuration of INVCAM is better for use in a certain application, based on the trade-off between error metrics and hardware characteristics, which will be discussed later.

Table 2. Error metrics of 8- and 16-bit configurations of INVCAM.

Architecture	***ER*** **(%)**	***MRED*** **(%)**	***NMED*** **($\times 10^{-3}$)**
INVCAM(7)	70.2	0.99	0.75
INVCAM16(7)	~100	2.64	1.87
INVCAM(8)	78.9	1.92	1.78
INVCAM16(8)	~100	3.56	3.07
INVCAM(9)	82.9	2.94	2.99
INVCAM16(9)	~100	4.52	4.02
INVCAM(10)	85.3	4.67	5.66
INVCAM16(10)	~100	5.94	5.86
INVCAM(11)	86.5	7.46	10.61
INVCAM16(11)	~100	8.70	10.90
INVCAM(12)	87.1	10.61	19.07
INVCAM16(12)	~100	11.94	19.35

Table 3. Error metrics of different 8-bit approximate multipliers.

Architecture	***ER*** **(%)**	***MRED*** **(%)**	***NMED*** **($\times 10^{-3}$)**
M20	3.6	0.05	0.09
M23–4	29.9	0.40	0.54
M23–3	42.4	0.88	0.71
M22–2	42.1	0.99	0.72
MUL4 [17]	76.3	1.52	0.77
D3 [26]	93.4	1.65	0.70
D2 [26]	93.4	2.15	0.80
MUL2 [17]	90.9	2.44	1.07
M21	99.1	2.61	1.14
MUL3 [17]	90.7	2.62	1.19
D1 [26]	93.4	2.92	1.35
M23–2	47.3	3.24	11.97
M18	85.2	3.88	3.26
MUL1 [17]	98.5	4.79	0.77
CDM8_95 [12]	70.9	5.18	6.79
M23–1	62.0	6.53	13.34
M22–1	61.8	7.59	13.04
CDM8_a6 [12]	73.6	7.66	11.95
TOSAM(0,3) [29]	99.1	7.66	20.78
CDM8_a7 [12]	75.4	10.84	20.32

4.2. Hardware Characteristics

In order to obtain the hardware characteristics of different 8-bit approximate multipliers as well as an exact Wallace multiplier, all designs were described using Verilog HDL and synthesized using the 45 nm Nangate technology [36]. The supply voltage was set to 1.1 V for all simulations. Table 4 shows the results for delay, power, area, PDP, power–delay–area product (PDA), EDP, and MRED for different configurations of INVCAM and other SoTA approximate 8-bit multipliers.

It can be inferred from Table 4 that all six configurations of INVCAM have lower power and area than 15 out of 20 considered approximate multipliers, while the other 5 multipliers (i.e., M21, TOSAM(0,3) [29], and all versions of CDM [12]) have lower power consumption than INVCAM(7) and INVCAM(8) by up to 37% and 32%, respectively, and also occupy less area. However, based on Table 3, both INVCAM(7) and INVCAM(8), on average, have lower MREDs than all five of these multipliers, by 82% and 64%, respectively. Also, 12 out of 20 considered approximate multipliers have equal or higher delays than all six configurations of INVCAM. Furthermore, INVCAM improves delay, power, and area by up to 42.4%, 76.1%, and 61.0%, respectively, compared to the SoTA approximate multipliers.

Table 4. Hardware characteristics and MRED of different approximate 8-bit multipliers.

Architecture	**Delay (ns)**	**Power (µW)**	**Area (µm^2)**	**PDP (fJ)**	**PDA (pJ × µm^2)**	**EDP (fJ × ns)**	**MRED (%)**
Exact (Wallace)	0.85	357	406	303	123.2	258	-
M20	0.85	297	333	252	84.1	215	0.05
M23–4	0.85	300	364	255	92.8	217	0.40
M23–3	0.85	282	342	240	82.0	204	0.88
M22–2	0.85	290	348	247	85.8	210	0.99
MUL4 [17]	0.83	249	308	207	63.7	172	1.52
D3 [26]	0.85	243	294	207	60.7	176	1.65
D2 [26]	0.80	261	325	209	67.9	167	2.15
MUL2 [17]	0.81	257	321	208	66.8	169	2.44
M21	0.80	192	251	154	38.6	123	2.61
MUL3 [17]	0.79	228	299	180	53.9	142	2.62
D1 [26]	0.85	255	326	217	70.7	184	2.92
M23–2	0.68	278	340	189	64.3	129	3.24
M18	0.83	281	354	233	82.6	194	3.88
MUL1 [17]	0.83	248	320	206	65.9	171	4.79
CDM8_95 [12]	0.77	191	241	147	35.4	113	5.18
M23–1	0.67	269	333	180	60.0	121	6.53
M22–1	0.65	287	361	187	67.3	121	7.59
CDM8_a6 [12]	0.63	190	246	120	29.4	76	7.66
TOSAM(0,3) [29]	0.68	144	198	98	19.3	67	7.66
CDM8_a7 [12]	0.56	176	223	99	21.9	55	10.8
INVCAM(7)	0.80	227	290	182	52.7	145	0.99
INVCAM(8)	0.77	213	284	164	46.6	126	1.92
INVCAM(9)	0.74	164	236	121	28.6	90	2.94
INVCAM(10)	0.64	130	198	83	16.5	53	4.67
INVCAM(11)	0.51	86	153	44	6.7	22	7.46
INVCAM(12)	0.49	71	130	35	4.5	17	10.61

The hardware characteristics of 16-bit INVCAM configurations are reported in Table 5. To further demonstrate the scalability of the proposed multiplier, the hardware characteristics' improvements over the exact multiplier for 8- and 16-bit multipliers are illustrated in Figure 9. The figure shows that the improvements in hardware characteristics of the proposed multiplier amplify as the multiplier bit-width increases.

Table 5. Hardware characteristics of 16-bit INVCAM configurations compared to the exact counterpart.

Architecture	**Delay (ns)**	**Power (µW)**	**Area (µm^2)**	**PDP (fJ)**	**PDA (pJ × µm^2)**	**EDP (fJ × ns)**	**MRED (%)**
Exact (Wallace)	1.22	2080	1785	2538	4529.6	3096	-
INVCAM16(7)	0.83	310	466	257	119.9	214	2.64
INVCAM16(8)	0.80	295	455	236	107.4	189	3.56
INVCAM16(9)	0.77	239	397	184	73.1	142	4.52
INVCAM16(10)	0.64	221	388	141	54.9	91	5.94
INVCAM16(11)	0.51	155	312	79	24.7	40	8.70
INVCAM16(12)	0.49	142	296	70	20.6	34	11.94

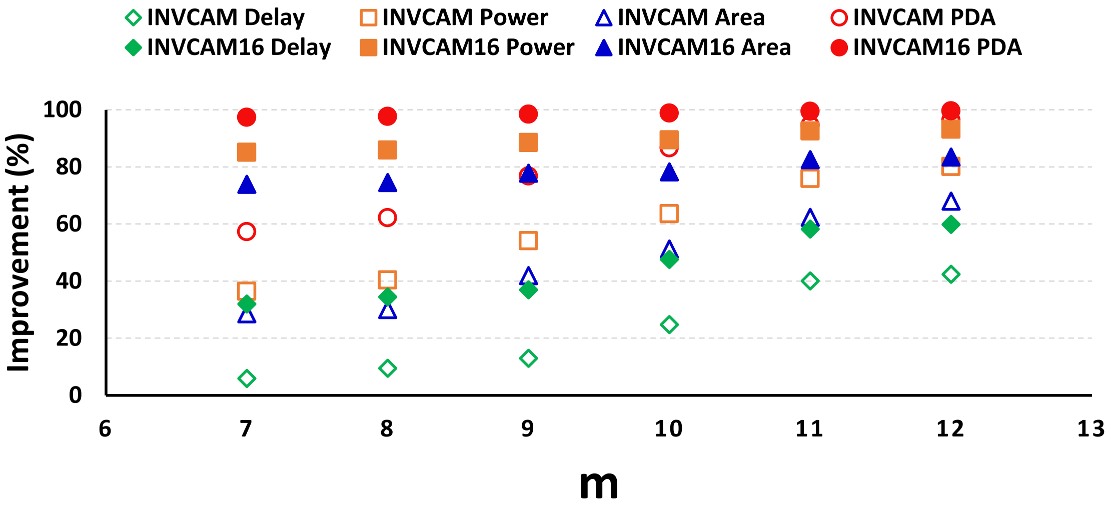


Figure 9. Hardware characteristic improvements of different configurations of 8- and 16-bit INVCAM(m) with respect to exact multipliers.

4.3. Applications

In this section, the efficacy of INVCAM is assessed and compared with other SoTA approximate multipliers in image processing and DNN applications, both of which are popular applications for the employment of approximate multipliers due to their intrinsic resiliency against error.

4.3.1. Image Processing

The approximate multipliers are used in two different image-processing applications: image multiplication and Sobel edge detection. For this purpose, each multiplier was mathematically modeled using MATLAB 2022a and its efficacy was investigated in the mentioned applications.

- *Image Multiplication:*

In this application, each pixel in the first image is multiplied with its corresponding pixel in the second image to generate the output. The peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) of the output images generated using different configurations of INVCAM were measured with respect to the images generated by an exact multiplier. The results are provided in Table 6, which shows that all approximate images produced using different configurations of INVCAM have SSIM and PSNR values greater than 0.81 and 28.9 dB, respectively.

Table 6. Comparison of output image quality in image multiplication and Sobel edge detection using different 8-bit multipliers.

	Image Multiplication								**Sobel Edge Detection**							
Architecture	**Airplane × Clock**		**Cameraman × Moon**		**Clock × Moon**		**Airplane × Sailboat**		**Airplane**		**Clock**		**Cameraman**		**Boat**	
	PSNR	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**
INVCAM(7)	52.8	~1	53.9	~1	53.3	~1	52.7	~1	52.9	~1	50.1	~1	48.5	~1	46.6	0.99
INVCAM(8)	47.6	~1	48.4	~1	47.7	~1	48.0	~1	51.3	~1	48.2	~1	46.6	~1	45.1	0.99
INVCAM(9)	44.3	0.99	45.5	0.99	44.9	0.99	45.4	0.99	50.4	~1	47.0	~1	45.7	~1	44.2	0.99
INVCAM(10)	41.3	0.98	42.6	0.98	41.4	0.98	41.9	0.98	47.5	~1	43.8	~1	42.8	0.99	41.7	0.99
INVCAM(11)	35.6	0.93	37.4	0.94	35.9	0.93	36.0	0.94	45.9	~1	42.1	0.99	41.5	0.99	40.7	0.99
INVCAM(12)	28.9	0.81	31.2	0.83	30.3	0.82	29.8	0.84	40.9	~1	38.1	0.99	37.2	0.99	38.4	0.99

- *Sobel Edge Detection:*

This algorithm finds the edges in an image, using two 3 × 3 kernels defined as

$$K_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}, \tag{16}$$

$$K_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \tag{17}$$

which are convolved with the input image to generate G_x and G_y, respectively. Next, the output image is calculated as

$$Out = \sqrt{{G_x}^2 + {G_y}^2}. \tag{18}$$

The PSNR and SSIM of the output images for the Sobel edge detection application are provided in Table 6. Again, the approximate images produced using different configurations of INVCAM have high similarity to the exact images due to the high PSNR and SSIM values reported in the table. For example, INVCAM(12), which has the lowest accuracy, yields an average PSNR of 30.05 dB and 38.65 dB for the image multiplication and Sobel edge detection tasks, respectively. This has occurred while INVCAM has the most efficient hardware characteristics among other approximate designs with similar MRED ranges and also reduces delay, power, and area, on average, by 23%, 58%, and 47%, with respect to an exact multiplier, according to Table 4.

4.3.2. Neural Networks

DNN algorithms represent another application that benefits from relative resiliency against error. In order to compare the effectiveness of INVCAM in DNN applications, the images of the MNIST [37] and CIFAR-10 [38] datasets are classified using five different DNN models. These DNN models are briefly introduced below.

FC-3 is a model consisting of three fully-connected (FC) layers with 512, 256, and 10 neurons, respectively. The reason that the last layers of FC-3 and all other models consist of 10 neurons is that the number of output classes in both the MNIST and CIFAR-10 datasets is equal to 10.

LeNet-5 and VGG-11 are well-known convolutional models with 5 and 11 layers, respectively. Like the other CNN models, these models start with convolutional layers and terminate with a number of FC layers and one Softmax layer. The Softmax layer consists of 10 neurons, the outputs of which demonstrate the probabilities of each of the output classes. Also, batch normalization layers are used between each two consecutive convolutional layers in the VGG-11 model. ResNet-18 is another deep CNN model with 18 layers. However, the architecture of ResNet models is different from other CNN models in the sense that they use residual blocks with a feedforward connection to improve performance. Finally, DenseNet-121 is a CNN model with four dense blocks (blocks in which each layer is connected to all other layers) with 6, 12, 24, and 16 layers, respectively. The ReLU activation function has been used for all FC and convolutional layers of all the aforementioned models.

It is worth noting that since NN weights can be positive or negative, signed multiplication is required. To support signed operands, the absolute values of the inputs and the signed output are approximated using the one's complement method, implemented via a simple XOR operation between each operand and its sign bit. This results in a zero MSB in the approximate absolute values, leading to inefficient utilization of INVCAM, whose most significant bits are computed using exact adders. To improve precision, both operands are left-shifted by one bit before multiplication, and the resulting product is right-shifted by two bits to restore the correct magnitude. Finally, the correct sign of the output is recovered based on the original operand signs.

Table 7 shows the classification accuracies of the five mentioned DNN models. The results were obtained using the "AdaPT" framework presented in [39]. Also, pretrained models, provided in [40], were used for the CIFAR-10 dataset, whereas the FC-3 and LeNet-5 models were locally trained on the MNIST dataset with training accuracies of 99.36% and 99.34%, respectively. No retraining was performed on the approximate models. As expected, increasing the number of approximate bits in INVCAM leads to higher classification accuracy degradation. For instance, the classification accuracy of the LeNet-5 (DenseNet-121) model on the MNIST (CIFAR-10) dataset decreases by 0.49% (0.45%) and 2.73% (52.27%), with respect to the exact model, when INVCAM(7) and INVCAM(12) are used. Therefore, INVCAM(12), which uses the most aggressive level of approximation amongst the proposed designs, is not appropriate for DNN models with high sensitivity against the errors caused by approximation. Amongst all considered approximate multipliers, M20 achieves the highest classification accuracy due to its lowest MRED of only 0.05%. INVCAM(7) shows the second-highest accuracy (after M20) for the FC-3 model on MNIST. Also, INVCAM(7) shows the third-highest classification accuracy (after M20 and M23-4) for VGG-11 and ResNet-18 models on CIFAR-10, while it consumes 24% less power than both M20 and M23-4, as shown in Table 4. For the LeNet-5 model on MNIST, M23-4 and MUL4 [17] lead to slightly higher accuracies compared to INVCAM(7), while their PDAs are also 76% and 21% higher than INVCAM(7), respectively.

Table 7. Comparison of the performance of different approximate and exact multipliers in DNNs' application.

Dataset	**MNIST**		**CIFAR-10**		
Model	**FC-3**	**LeNet-5**	**VGG-11**	**ResNet-18**	**DenseNet-121**
Exact (Wallace)	97.82	97.63	92.34	92.92	93.99
M20	97.79	97.55	91.81	92.78	93.90
M23–4	97.68	97.19	91.78	92.59	93.76
M23–3	97.64	97.11	91.61	92.43	93.63
M22–2	97.62	97.08	91.66	92.52	93.56
MUL4 [17]	97.64	97.16	91.53	92.31	93.32
D3 [26]	97.57	97.06	91.42	92.26	93.37
D2 [26]	97.52	97.14	91.18	92.02	93.03
MUL2 [17]	97.50	97.03	91.06	91.87	92.88
M21	97.49	96.92	90.99	91.76	92.77
MUL3 [17]	97.52	97.05	91.03	91.72	92.70
D1 [26]	97.50	97.12	90.87	91.54	92.58
M23–2	97.48	96.75	90.73	91.42	92.43
M18	97.53	96.88	90.61	91.12	92.13
MUL1 [17]	97.41	96.76	90.58	90.87	91.38
CDM8_95 [12]	97.66	96.93	90.90	90.75	91.16
M23–1	97.27	96.63	90.34	90.28	89.41
M22–1	97.23	96.66	88.17	89.76	88.69
CDM8_a6 [12]	97.56	95.19	87.31	89.81	88.62
TOSAM(0,3) [29]	97.16	95.21	86.47	90.02	88.47
CDM8_a7 [12]	96.70	94.66	85.51	67.99	33.43
INVCAM(7)	97.77	97.14	91.74	92.53	93.54
INVCAM(8)	97.74	97.03	91.27	92.11	93.12
INVCAM(9)	97.72	96.97	90.79	91.57	92.57
INVCAM(10)	97.70	96.74	90.59	90.92	91.09
INVCAM(11)	97.43	96.41	88.23	90.08	88.71
INVCAM(12)	97.16	94.90	85.47	73.15	41.72

4.4. Discussion

Figure 10 demonstrates the efficacy of each approximate multiplier in improving the hardware metrics, including delay, power, area, and PDA, with respect to an exact multiplier. As illustrated in the figure, INVCAM(7) shows superior PDA, power, and area gains amongst the designs with *MRED*s below 1% (i.e., INVCAM(7), M20, M23-3, M23-4, and M22-2). INVCAM(8) and INVCAM(9) achieve the best PDA improvements while also excelling in power and area in the subsequent *MRED* ranges (i.e., [1%, 2%) and [2%, 4%)). Also, it can be observed that INVCAM(12) provides the greatest improvements amongst all approximate multipliers across all four metrics.

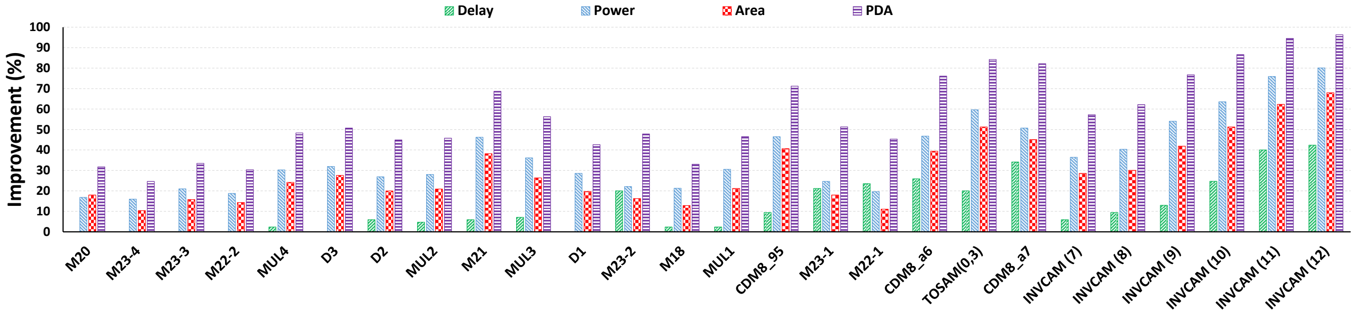


Figure 10. Hardware characteristics improvement of different approximate 8-bit multipliers with respect to an exact multiplier.

To compare the efficiency of different approximate multipliers considering both their accuracy and hardware characteristics, the PDA of different designs versus their *MRED* values are plotted in Figure 11. We see that different configurations of INVCAM are located on the Pareto-optimal curve, plotted by a dashed line (close to the southwestern corner of the diagram), which is an attestation to the superiority of INVCAM from the viewpoint of both accuracy and hardware characteristics. It is important to note that INVCAM does not achieve top performance in each individual parameter on its own. For example, M20 has the least *MRED*, yet its PDA is higher than INVCAM(7) by almost 60%. Amongst the investigated SoTA multipliers, D3 [26] and M21 are the closest to the Pareto-optimal curve. D3 [26] has 67% higher *MRED* and 15% higher PDA compared to INVCAM(7), while M21 has 11% less *MRED* and a 7% higher PDA compared to INVCAM(9).

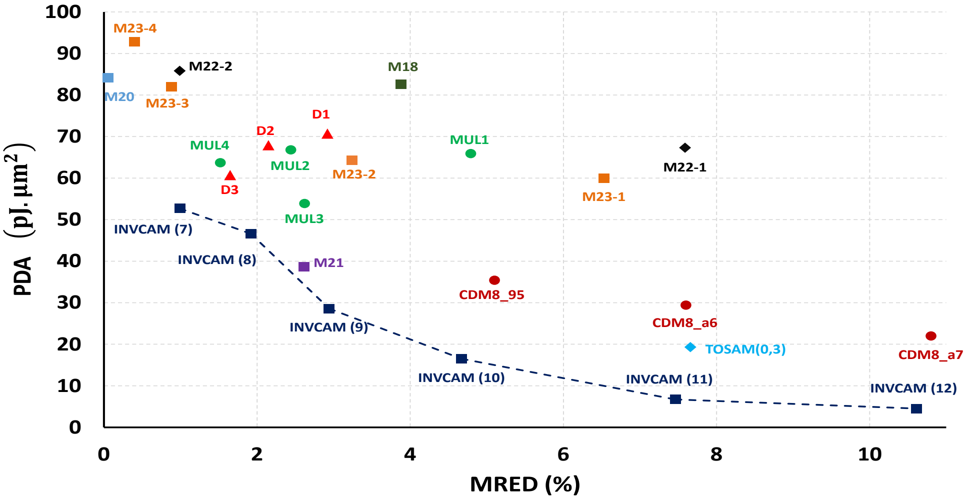


Figure 11. PDA vs. *MRED* of different approximate 8-bit multipliers.

In order to assess the performance of INVCAM in image-processing applications, it must be noted that output images with PSNR values greater than 30 dB are very similar to the exact images, and their slight difference cannot be detected by the human eye [16]. To illustrate it better, the produced approximate images along with the exact ones for both image-processing applications are provided in Figure 12. The results reveal that by using INVCAM(m) with $7 \leq m \leq 11$, one can be certain about achieving a high-quality image. Some of the images produced using INVCAM(12), such as Airplane × Clock and Airplane × Sailboat, have PSNR values below 30 dB, and their slight differences with the exact image can be detected by the human eye (see Figure 12).

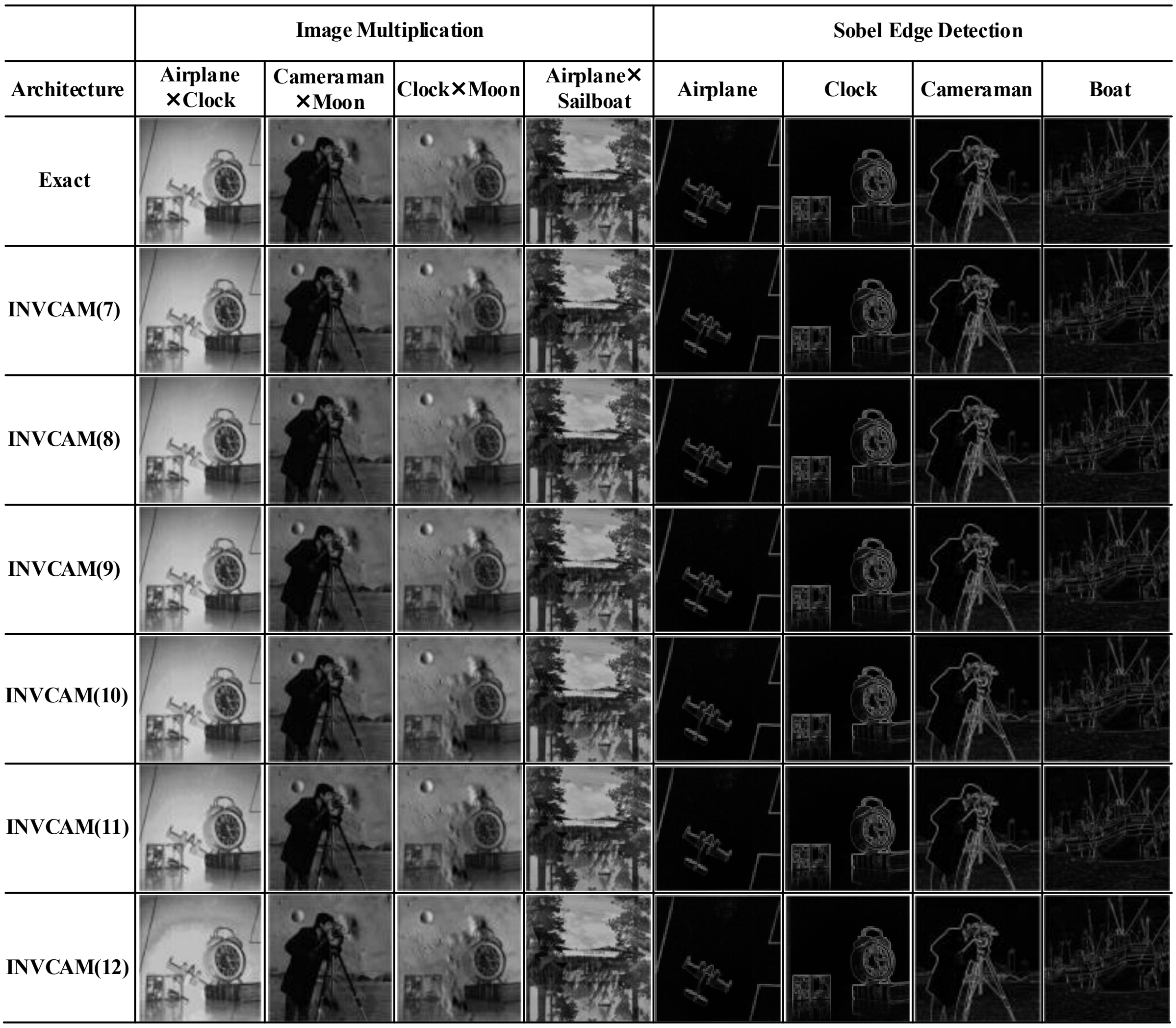


Figure 12. Output images in image multiplication and Sobel edge detection applications.

Different variations of INVCAM also show good performance in DNN applications, according to Table 7. Namely, INVCAM(7) shows the best performance compared to the other INVCAM configurations, and the maximum accuracy drop of the models using INVCAM(7) compared to their corresponding exact models is 0.6% for classification of the CIFAR-10 dataset using the VGG-11 model. It has comparable accuracy with respect to the exact models while consuming 39% and 28% less energy and area, respectively, compared to the exact multiplier. Amongst SoTA approximate multipliers, M20 achieves the highest accuracy in all considered benchmarks (which is expected based on its exceptionally low *MRED*), while based on Table 4, it consumes 31% more power compared to INVCAM(7). Also, increasing the error of approximate multipliers, or increasing the number of approximated bits in INVCAM, does not result in huge accuracy drops for FC-3 and LeNet-5 models, which are two relatively small models. However, this is not the case for the other three investigated models, which are larger than FC-3 and LeNet-5. For instance, increasing INVCAM's number of approximate bits to 12 causes an accuracy drop of up to 52% for the DenseNet-121 model with respect to the exact model.

To provide a more application-based comparison, Figure 13 shows the average accuracy degradation of different approximate multipliers versus their PDA. Again, different configurations of INVCAM are located on the Pareto-optimal curve (denoted by the dashed line), which shows their desirable trade-off between hardware efficiency and accuracy in DNN applications.

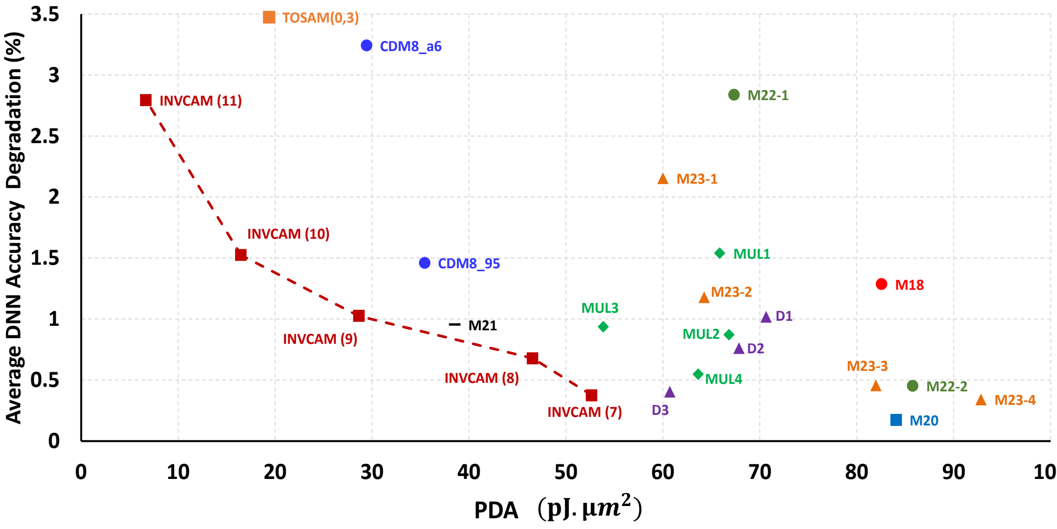


Figure 13. Average accuracy degradation of approximate DNN models vs. the PDA of the used approximate 8-bit multipliers.

In summary, the proposed INVCAM is designed to improve hardware efficiency for error-tolerant applications such as image processing and neural network inference, where exact arithmetic is often unnecessarily expensive. Compared to an exact 8-bit multiplier, INVCAM introduces several structural optimizations that significantly reduce power consumption, delay, and area. PPs are generated using NAND gates, eliminating explicit inverters and reducing logic complexity. In the PPR stages, novel approximate inverting adders are employed, which require fewer basic logic gates and completely avoid XOR/XNOR structures, leading to notable reductions in area and propagation delay. In addition, the accumulation stage avoids carry generation and propagation, substantially shortening the critical path. To mitigate the resulting approximation error, an error compensation module (ECM) is incorporated with only a small area overhead. While INVCAM introduces bounded approximation error and supports design-time configurability, the achieved reductions in PDA demonstrate that INVCAM is an efficient and well-motivated alternative to exact and existing approximate 8-bit multipliers.

5. Conclusions

In this paper, a novel approximate multiplier design, named INVCAM, was proposed in which inverted partial products were generated using logic-level optimization. Additionally, an error correction module was proposed to compensate for the error generated by all approximate components. INVCAM has a configurable number of approximate bits, which can be adjusted at design time to allow the design to be employed in various applications with different levels of error tolerance and hardware constraints.

INVCAM achieved significant improvements in power consumption, delay, and area compared to both the exact and other SoTA approximate multipliers, due to its simplified structure implemented using a smaller number of basic logic gates. INVCAM(12), which is the most efficient design from a hardware viewpoint, improved the delay, power, area, and energy by 12.5%, 60.0%, 41.7%, and 64.6%, respectively, compared to the SoTA approximate multipliers with a similar MRED range, and 42%, 80%, 68%, and 88%, compared to an exact multiplier. Additionally, the efficacy of INVCAM was investigated in image-processing and DNN applications. The results revealed the great performance of INVCAM in both applications. The six configurations of INVCAM achieved an average PSNR of 41.4 dB and 44.9 dB for image multiplication and Sobel edge detection applications, respectively. Moreover, employing different configurations of INVCAM in DNN models resulted in accuracies comparable to those of the exact models, with a maximum reduction of just 0.6% for INVCAM(7) and 2.9% for INVCAM(10). However, this slight accuracy degradation was accompanied by significant gains, reducing energy consumption by 40% and 72% for INVCAM(7) and INVCAM(10), respectively.

Author Contributions: Conceptualization, K.D. and S.D.; methodology, K.D.; software, K.D., S.D., and N.A.; validation, K.D., S.D., and N.A.; writing—original draft preparation, K.D. and S.D.; writing—review and editing, K.D., S.D., S.V., and N.T.; supervision, S.V.; project administration, S.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The evaluation data presented in this study are available on request from the corresponding author.

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

Glossary of Abbreviations

Abbreviation	**Full Term**
PPG	Partial product generation
PPR	Partial product reduction
ACC	Accumulation
AIHA	Approximate inverted half adder
IHA	Inverted half adder
AIFA	Approximate inverted full adder
FA	Full adder
AICOM	Approximate inverted compressor
COM	Compressor
CLA	Carry look-ahead adder
ER	Error Rate
ED	Error distance
MED	Mean error distance
RED	Relative error distance
MRED	Mean relative error distance
NMED	Normalized mean error distance

References

1. Akbari, O.; Kamal, M.; Afzali-Kusha, A.; Pedram, M. Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *25*, 1352–1361. [CrossRef]
2. Wu, Q.; Pedram, M.; Wu, X. Clock-gating and its application to low power design of sequential circuits. *IEEE Trans. Circuits Syst. I Fundam. Theory Appl.* **2000**, *47*, 415–420.
3. Höppner, S.; Vogginger, B.; Yan, Y.; Dixius, A.; Scholze, S.; Partzsch, J.; Neumärker, F.; Hartmann, S.; Schiefer, S.; Ellguth, G.; et al. Dynamic Power Management for Neuromorphic Many-Core Systems. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2019**, *66*, 2973–2986. [CrossRef]

4. Paolino, C.; Antolini, A.; Pareschi, F.; Mangia, M.; Rovatti, R.; Scarselli, E.F.; Gnudi, A.; Setti, G.; Canegallo, R.; Carissimi, M.; et al. Compressed Sensing by Phase Change Memories: Coping with Encoder non-Linearities. In Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Republic of Korea, 23–26 May 2021; pp. 1–5.
5. Balsa, J. Comparison of Image Compressions: Analog Transformations. *Proceedings* **2020**, *54*, 37. [CrossRef]
6. Sayadi, L.; Amirany, A.; Moaiyeri, M.H.; Timarchi, S. Balancing precision and efficiency: An approximate multiplier with built-in error compensation for error-resilient applications. *J. Supercomput.* **2024**, *81*, 109. [CrossRef]
7. Salehi Sheikhali Kelayeh, M.S.; Divsalar, S.; Vahdat, S.; TaheriNejad, N. ARTS: An Approximate Reduced Tree and Segmentation-based Multiplier. *Future Gener. Comput. Syst.* **2026**, *175*, 108098. [CrossRef]
8. Jiang, H.; Angizi, S.; Fan, D.; Han, J.; Liu, L. Non-Volatile Approximate Arithmetic Circuits Using Scalable Hybrid Spin-CMOS Majority Gates. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2021**, *68*, 1217–1230. [CrossRef]
9. Kim, S.; Kang, Y.; Baek, S.; Choi, Y.; Kang, S. Low-Power Ternary Multiplication Using Approximate Computing. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 2947–2951. [CrossRef]
10. Amanollahi, S.; Kamal, M.; Afzali-Kusha, A.; Pedram, M. Circuit-Level Techniques for Logic and Memory Blocks in Approximate Computing Systemsx. *Proc. IEEE* **2020**, *108*, 2150–2177. [CrossRef]
11. Esposito, D.; Strollo, A.G.M.; Napoli, E.; De Caro, D.; Petra, N. Approximate Multipliers Based on New Approximate Compressors. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2018**, *65*, 4169–4182. [CrossRef]
12. 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]
13. Chen, K.; Liu, W.; Han, J.; Lombardi, F. Profile-Based Output Error Compensation for Approximate Arithmetic Circuits. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**, *67*, 4707–4718. [CrossRef]
14. Mohanty, B.K. Efficient Approximate Multiplier Design Based on Hybrid Higher Radix Booth Encoding. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2023**, *13*, 165–174. [CrossRef]
15. Naresh, K.; Sai, Y.P.; Majumdar, S. Design of 8-bit Dadda Multiplier using Gate Level Approximate 4:2 Compressor. In Proceedings of the 35th International Conference on VLSI Design and 2022 21st International Conference on Embedded Systems (VLSID), Bangalore, India, 26 February–2 March 2022; pp. 269–274.
16. 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]
17. Pei, H.; Yi, X.; Zhou, H.; He, Y. Design of Ultra-Low Power Consumption Approximate 4–2 Compressors Based on the Compensation Characteristic. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 461–465. [CrossRef]
18. Zhang, Y.; Chen, X.; Guo, P.; Xie, G. Design and Analysis of Approximate Multiplier of Majority-Based Imprecise 4–2 Compressor for Image Processing. In Proceedings of the 2023 IEEE 23rd International Conference on Nanotechnology (NANO), Jeju City, Republic of Korea, 2–5 July 2023; pp. 1–5.
19. Teja, T.S.A.; Teja, G.S.; Ravindra, J.; Maddisetti, L. High Speed Multiplier using Embedded Approximate 4-2 Compressor for Image Multiplication. In Proceedings of the First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR), Hyderabad, India, 10–12 March 2022; pp. 1–5.
20. Koshe, S.S.; Rajgure, Y.; Sridevi, S.A. Novel Implementation of Low Power and High Performance 4-2 Compressors for Approximate Multipliers. In Proceedings of the 2022 International Conference on Futuristic Technologies (INCOFT), Belgaum, India, 24–26 November 2022; pp. 1–5.
21. Zhang, M.; Nishizawa, S.; Kimura, S. Area Efficient Approximate 4–2 Compressor and Probability-Based Error Adjustment for Approximate Multiplier. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *70*, 1714–1718. [CrossRef]
22. Krishna, L.H.; Rao, J.B.; Ayesha, S.; Veeramachaneni, S.; Mahammad, S.N. Energy Efficient Approximate Multiplier Design for Image/Video Processing Applications. In Proceedings of the 2021 IEEE International Symposium on Smart Electronic Systems (iSES), Jaipur, India, 18–22 December 2021; pp. 210–215.
23. Krishna, L.H.; Sk, A.; Rao, J.B.; Veeramachaneni, S.; Sk, N.M. Energy-Efficient Approximate Multiplier Design With Lesser Error Rate Using the Probability-Based Approximate 4:2 Compressor. *IEEE Embed. Syst. Lett.* **2024**, *16*, 134–137. [CrossRef]
24. Xiao, H.; Xu, H.; Chen, X.; Wang, Y.; Han, Y. Fast and High-Accuracy Approximate MAC Unit Design for CNN Computing. *IEEE Embed. Syst. Lett.* **2022**, *14*, 155–158. [CrossRef]
25. Gu, F.-Y.; Lin, I.-C.; Lin, J.-W. A Low-Power and High-Accuracy Approximate Multiplier with Reconfigurable Truncation. *IEEE Access* **2022**, *10*, 60447–60458. [CrossRef]
26. Kumar, U.A.; Bikki, P.; Veeramachaneni, S.; Ahmed, S.E. Power Efficient Approximate Multiplier Architectures for Error Resilient Applications. In Proceedings of the 2022 IEEE 19th India Council International Conference (INDICON), Kochi, India, 24–26 November 2022; pp. 1–5.
27. Waris, H.; Wang, C.; Xu, C.; Liu, W. AxRMs: Approximate recursive multipliers using high performance building blocks. *IEEE Trans. Emerg. Top. Comput.* **2022**, *10*, 1229–1235. [CrossRef]

28. Sabetzadeh, F.; Moaiyeri, M.H.; Ahmadinejad, M. An Ultra-Efficient Approximate Multiplier with Error Compensation for Error-Resilient Applications. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *70*, 776–780. [CrossRef]
29. Vahdat, S.; Kamal, M.; Afzali-Kusha, A.; Pedram, M. TOSAM: An energy-efficient truncation- and rounding-based scalable approximate multiplier. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *27*, 1161–1173. [CrossRef]
30. 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.* **2021**, *6*, 612–625. [CrossRef]
31. Zendegani, R.; Safari, S. AMCAL: Approximate multiplier with the configurable accuracy levels for image processing and convolutional neural network. *IEEE Access* **2024**, *12*, 94135–94151. [CrossRef]
32. Nambi, S.; Kumar, U.A.; Radhakrishnan, K.; Venkatesan, M.; Ahmed, S.E. DeBAM: Decoder-based approximate multiplier for low power applications. *IEEE Embed. Syst. Lett.* **2021**, *13*, 174–177. [CrossRef]
33. Shankar, R.G.; Ananthi, D.R. Approximate Booth Multipliers using Compressors and Counter. In Proceedings of the International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal, 26–28 April 2023; pp. 1658–1662.
34. 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]
35. Weste, N.H.E.; Harris, D.M. *CMOS VLSI Design*, 4th ed.; Addison-Wesley: Boston, MA, USA, 2011.
36. Nangate. 45 nm Open Cell Library. 2012. Available online: http://www.nangate.com (accessed on 5 May 2012).
37. Deng, L. The MNIST database of handwritten digit images for machine learning research [Best of the Web]. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [CrossRef]
38. Krizhevsky, A.; Nair, V.; Hinton, G. Learning Multiple Layers of Features from Tiny Images. 2009. Available online: http://cs.utoronto.ca/ kriz/learning-features-2009-TR.pdf (accessed on 15 December 2025).
39. 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]
40. Phan, H. *huyvnphan/PyTorch_CIFAR10*; v3.0.1; Zenodo: Geneva, Switzerland, 2021. [CrossRef]