




A Hardware- and Accuracy-Efficient Approximate Multiplier with Error Compensation for Neural Network and Image Processing Applications

Sudeh Shirkavand Saleh Abad¹ · Mohammad Hossein Moaiyeri¹ 

Received: 22 July 2021 / Revised: 1 July 2022 / Accepted: 1 July 2022 /

Published online: 22 July 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Approximate computing is a promising method for reducing energy dissipation and design complexity in various applications, where high accuracy is not a significant need. This study proposes an efficient approximate multiplier using a full adder as an approximate 4:2 compressor. This simplification reduces power and hardware overheads. While this technique reduces the accuracy to some extent, the multiplier is still more accurate than necessary in real applications like neural networks and image processing. Meanwhile, an efficient error compensation module is presented to promote the accuracy of the proposed approximate multiplier. Accordingly, our design provides an effective compromise between accuracy and hardware metrics. The hardware simulations are conducted using HSPICE with the 7 nm tri-gate FinFET model. Furthermore, the accuracy and quality of the proposed approximate multiplier are evaluated using MATLAB. According to the results, the proposed design provides far better trade-offs between the performance characteristics and preciseness than its counterparts. The proposed design improves the power-delay product, energy-delay product, and figure of merit considering both energy and quality metrics, on average, by 33%, 44%, and 34%. At the same time, it offers comparable accuracy metrics in error-resilient applications when compared to the other high-accuracy approximate multipliers with error recovery modules.

Keywords Approximate computing · Multiplier · Error compensation · Neural network · Image processing

✉ Mohammad Hossein Moaiyeri
h_moaiyeri@sbu.ac.ir

Sudeh Shirkavand Saleh Abad
s.shirkavand@mail.sbu.ir

¹ Faculty of Electrical Engineering, Shahid Beheshti University, Tehran, Iran

1 Introduction

Energy dissipation has turned into a critical issue in integrated circuits due to the wide application of computing blocks, especially in neural networks. To achieve a revolution in deep learning and artificial intelligence, limiting the design and implementation costs and energy dissipation is necessary. Deep learning generally consists of the training and inference phases. In the training phase, a neural network learns to develop classification abilities through gradient-based backpropagation algorithms. These algorithms include critical floating-point calculations and require much more accuracy than the inference phase [9, 18]. It has been demonstrated that it would be reasonable to perform the calculations of the inference step approximately after the exact calculations of the training step in a neural network. All of the layers in a convolutional neural network (CNN) retain their essential characteristics despite having a certain amount of error in their multiplications. They are resilient to relatively small arithmetic errors [19]. Although the training phase has more calculations, they can be performed offline in advance, and the inference can be performed using approximate computing on the trained network data [30].

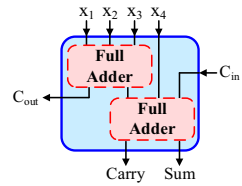
The error-resilient applications like image processing, data mining, computer vision, pattern recognition, and mobile multimedia do not necessarily require precise results [15, 16, 24]. As a result, approximate computing can be used effectively in these applications due to the significant reduction in the design and implementation costs and energy dissipation. In contrast, the quality of the results is still acceptable [33]. Accordingly, accurate complex calculations are not reasonable in many applications, considering the increased complexity of integrated circuits and the importance of reducing power dissipation, especially in battery-operated electronic devices [25].

The scaling of the planar CMOS has faced severe problems such as reduced gate control, higher power density, and drain-induced barrier lowering (DIBL) [31]. The FinFET technology has replaced the planar CMOS because of its 3D multi-gate structure that provides higher gate control, lower short channel effects, and a higher $I_{\text{on}}/I_{\text{off}}$ ratio. Moreover, its intrinsic body eliminates random dopant fluctuations [13]. Although self-heating in FinFET can increase power density, it can be compensated by saving power consumption using approximate computing [26].

One of the most widely used arithmetic blocks is multiplication. As multipliers are often located on the critical path of digital systems, an efficient approximate multiplier design can improve digital systems' performance and energy efficiency [26]. Multiplication usually has three steps: partial products (PPs) generation using an array of AND gates, partial products reduction, and calculating the final output using a ripple carry adder [8]. Considering these three stages, the second one is the most critical and power-hungry stage. Thus, reducing the complexity of this step can significantly enhance the performance and energy efficiency of a multiplier [2, 7]. Usually, a computing module, known as a compressor, is used in the reduction phase [8]. Two full adders form the structure of a conventional 4:2 exact compressor, as shown in Fig. 1. In [2, 14, 24, 26, 32, 37, 42], various simplified structures for a 4:2 compressor were proposed.

In this paper, we present an efficient approximate multiplier. In our design, a full adder is used as an approximate 4:2 compressor, reducing circuit complexity, saving

Fig. 1 Structure of the exact 4:2 compressor



energy and area, and speeding up the circuit. Moreover, the proposed approximate multiplier uses an error recovery module to enhance accuracy. Accordingly, the proposed approximate multiplier effectively balances hardware efficiency and accuracy for neural networks and image processing applications.

The article is continued as follows: Sect. 2 provides a brief review of the background of the research. In Sect. 3, the proposed approach is described in detail. Section 4 presents the comprehensive simulation results and discussions, and finally, Sect. 5 concludes the paper.

2 Backgrounds

2.1 Previous Work

In a multiplication operation, partial products are generated using AND gates. Assuming the inputs of a multiplier are distributed uniformly in general [14, 42], the probability that a partial product (an input bit to a compressor), generated by a two-input AND gate, equals ‘0’ (‘1’) is $\frac{3}{4}$ ($\frac{1}{4}$). Using 4:2 compressors is one of the most efficient methods for compressing partial products. The conventional design of an exact 4:2 compressor is shown in Fig. 1 [8].

This compressor is a 5:3 counter with five inputs (x_1 , x_2 , x_3 , x_4 , and C_{in}) and three outputs (Sum, Carry, and C_{out}). In the exact 4:2 compressor, C_{out} (followed by the C_{in} input) is only ‘1’, when $x_4x_3x_2x_1 = “1111”$, whose probability is $\frac{1}{4} \times \frac{1}{4} \times \frac{1}{4} \times \frac{1}{4} = \frac{1}{256}$ (0.39%). Therefore, the C_{in} and C_{out} signals are ignored with an insignificant accuracy loss in most approximate compressors [32].

Several approximate multipliers with error compensation capability based on approximate 4:2 compressors have already been suggested in the literature. In [42], by manipulating the truth table, an approximate 4:2 compressor was designed. In this design, when $x_3x_4 = “11”$, the Carry and Sum outputs are set to ‘1’ and XNOR (x_1, x_2), respectively. This schema provides an approximate 4:2 compressor with a more straightforward structure than its exact counterpart and a probability of $\frac{16}{256}$ (6.25%) non-directional errors.

In [14], simpler compressors with the same accuracy were presented by modifying the approximate compressor suggested in [42]. The last four rows of the truth table of this compressor all have an error distance (ED) of -1 (see Table 1). In the structure of the approximate multiplier proposed in [14], by adding a positive bit generated by an error recovery module to the exact compressor of the first accurate column, the

Table 1 The truth table of the approximate 4–2 compressors with error recovery modules

x_4-x_1	Ha [14]			Strollo_a [37]			Strollo_b [37]			Pei_1 [32]		
	Carry	Sum	ED	C	S	ED	C	S	ED	C	S	ED
0000	0	0	0	0	0	0	0	0	0	0	0	0
0001	0	1	0	0	1	0	0	1	0	0	0	-1
0010	0	1	0	0	1	0	0	1	0	0	0	-1
0011	1	0	0	1	0	0	1	0	0	0	1	-1
0100	0	1	0	0	1	0	0	1	0	0	0	-1
0101	1	0	0	1	0	0	1	0	0	0	1	-1
0110	1	0	0	1	0	0	1	0	0	0	1	-1
0111	1	1	0	1	0	-1	1	0	-1	0	1	-2
1000	0	1	0	0	1	0	0	1	0	0	0	-1
1001	1	0	0	1	0	0	1	0	0	0	1	-1
1010	1	0	0	1	0	0	1	0	0	0	1	-1
1011	1	1	0	1	1	0	1	1	0	0	1	-2
1100	0	1	-1	1	0	0	1	0	0	0	1	-1
1101	1	0	-1	1	1	0	1	1	0	0	1	-2
1110	1	0	-1	1	1	0	1	1	0	0	1	-2
1111	1	1	-1	1	1	-1	1	1	-1	0	1	-3

x_4-x_1	Pei_2 [32]			Pei_3 [32]			Pei_4 [32]			Kumar [20]		
	C	S	ED	C	S	ED	C	S	ED	C	S	ED
0000	0	0	0	0	0	0	0	0	0	0	0	0
0001	0	0	-1	0	0	-1	0	1	0	1	0	0
0010	0	0	-1	0	0	-1	0	0	-1	1	0	0
0011	0	1	-1	0	0	-2	0	1	-1	1	0	-1
0100	0	0	-1	0	0	-1	0	1	0	1	0	0
0101	0	1	-1	0	1	-1	0	1	-1	0	1	0
0110	0	1	-1	0	1	-1	0	1	-1	0	1	0
0111	0	1	-2	0	1	-2	0	1	-2	0	1	-1
1000	0	0	-1	0	0	-1	0	0	-1	1	0	0
1001	0	1	-1	0	1	-1	0	1	-1	0	1	0
1010	0	1	-1	0	1	-1	0	0	-2	0	1	0
1011	0	1	-2	0	1	-2	0	1	-2	0	1	-1
1100	0	1	-1	0	0	-2	0	1	-1	0	1	0
1101	0	1	-2	0	1	-2	0	1	-2	1	1	0
1110	0	1	-2	0	1	-2	0	1	-2	1	1	0
1111	0	1	-3	0	1	-3	0	1	-3	1	1	-1

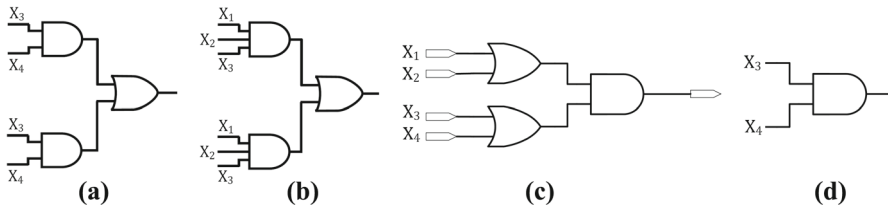


Fig. 2 The schematics of the error recovery modules shown in the previous papers **a** [14] **b** [37] **c** [32] **d** [20]

errors of two compressors are corrected, which enhances the output accuracy. The error recovery module suggested in [14] is shown in Fig. 2a.

In [37], approximate multipliers with error recovery capability were proposed based on the approximate compressor presented in [38]. As shown in Table 1, in this approximate compressor, errors occur in the cases that $x_1x_2x_3 = "111"$. An error recovery module is used to detect and correct the results of two compressors of this multiplier (see Fig. 2b).

Three 4:2 compressors with only one output (Sum) were suggested in [32]. The truth tables of these compressors are shown in Table 1. In the approximate multiplier presented in [32], several compressors are considered together as a compressor chain (CC). To reduce the error of CC, a positive bit is inserted into the exact compressor of the first accurate column. The error recovery module, which can be used in this multiplier, prevents the generation of this positive bit if there is no error in the case of the "0000" input. Otherwise, in this case, it causes an additional error that is most likely to occur (32%). The error recovery module presented in [32] is shown in Fig. 2c.

In [20], a multiplier architecture using a new 4:2 approximate compressor with 34 transistors was proposed. In this multiplier, the errors caused by approximation are compensated using three two-input AND gates as the error recovery module (see Fig. 2d) in the last column of the approximate region.

2.2 Accuracy Metrics

One of the most important accuracy parameters in designing approximate circuits is error distance (ED), the arithmetic distance between an erroneous output and its correct value. The normalized mean error distance (NMED), mean relative error distance (MRED), and Number of Effective bits (NoEB) are three critical metrics for quantifying accuracy. These metrics, giving an illustrative view of the accuracy regardless of the application, are calculated as [11, 17, 23]:

$$\text{MRED} = \frac{1}{2^{2N}} \sum_{i=1}^{2^{2N}} \frac{|\text{ED}_i|}{M_i} \quad (1)$$

$$\text{NMED} = \frac{1}{(2^N - 1)^2} \sum_{i=1}^{2^{2N}} \frac{|\text{ED}_i|}{2^{2N}} \quad (2)$$

$$\text{NoEB} = 2N - \log_2(1 - \text{ERMS}) \quad (3)$$

where N is the bit length of a multiplier, M_i is the accurate output for each input, ED_i is the error distance between each input's exact and inexact results, and ERMS is the root-mean-square of the errors produced by the multiplier.

The peak signal-to-noise ratio (PSNR) and the mean structural similarity index (MSSIM) are two important application-dependent metrics that evaluate the quality of the images generated by approximate multipliers [23].

$$\text{PSNR} = 10 \log_{10} \left(m \times p \times \text{MAX}_I^2 / \sum_{i=0}^{m-1} \sum_{j=0}^{p-1} [I(i, j) - K(i, j)]^2 \right) \quad (4)$$

where $I(i, j)$ and $K(i, j)$ are the exact and inexact values for each pixel, m and p are the image dimensions, and MAX_I is the maximum possible pixel value in the image.

The MSSIM parameter calculates the structural similarity of images resulting from exact and inexact multiplications based on the ability of the human visual system to extract image structure information. This metric is calculated as follows:

$$\text{MSSIM}(X, Y) = \frac{1}{M} \sum_{i=1}^M \frac{(2\mu_x \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}. \quad (5)$$

More details regarding this metric can be found in [11] and [40].

3 Proposed Error-Resilient Approximate Multiplier

This section proposes an innovative approach for designing approximate multipliers, using a full adder as an approximate 4:2 compressor and an efficient error recovery module. Our proposed approach offers an effective trade-off between hardware efficiency and accuracy for error-resilient applications such as neural networks and image processing. This simplification reduces the approximate multipliers' power consumption and overall delay while offering excellent accuracy and quality standards. It is worth mentioning that various full adder cells with various performance metrics have been presented so far, making the proposed approach flexible and designer-friendly. Our approximate structure uses the standard CMOS full adder [41]. Accordingly, in our suggested approximate 4:2 compressor, the C_{in} and C_{out} signals are ignored, and the Carry and Sum outputs are generated as

$$\text{Carry} = x_2(x_3 + x_4) + x_3x_4 \quad (6)$$

$$\text{Sum} = x_2 \oplus x_3 \oplus x_4. \quad (7)$$

As shown in Eqs. 6 and 7, the term x_1 does not exist in the compressor equations, and also in the truth table of this 4:2 compressor, shown in Table 2, when x_1 takes the

Table 2 The truth table of the suggested approximate 4:2 compressor

x_4-x_1	Carry	Sum	Probability	ED
0000	0	0	81/256	0
0001	0	0	27/256	− 1
0010	0	1	27/256	0
0011	0	1	9/256	− 1
0100	0	1	27/256	0
0101	0	1	9/256	− 1
0110	1	0	9/256	0
0111	1	0	3/256	− 1
1000	0	1	27/256	0
1001	0	1	9/256	− 1
1010	1	0	9/256	0
1011	1	0	3/256	− 1
1100	1	0	9/256	0
1101	1	0	3/256	− 1
1110	1	1	3/256	0
1111	1	1	1/256	− 1
Sum of the probabilities of the erroneous cases			64/256 (25.01%)	

logic value ‘1’, it is ignored. Therefore, in the suggested approximate 4:2 compressor, only three inputs are considered. Furthermore, the functionality of the suggested approximate 4:2 compressor makes it very suitable for efficient error recovery.

In addition to the error distance, the occurrence probability of each input is a critical factor in determining the error rate of an approximate compressor [11]. Assuming the multiplier input bits are uniformly distributed in general (considering an equal probability for the input bits to be ‘0’ or ‘1’) [11, 14, 32, 39, 42], the probability that a partial product generated by an AND gate (a compressor input) equals to ‘1’ (‘0’) is $\frac{1}{4}$ ($\frac{3}{4}$). Table 2 shows the occurrence probability for each input combination.

As shown in Table 2, although in eight cases, the error distance is -1 , the Sum of the occurrence probabilities of these erroneous cases is only 64/256 (25%). This is because the erroneous cases mainly occur for the inputs with more 1’s, which are less likely to occur at the output of the partial product generation part. In particular, the correctness of the output for the input “0000” (with the highest probability of occurrence) and the fact that one of the errors occurs in the “1111” input combination (with the lowest probability of occurrence) have a significant impact on the accuracy of the proposed design. Moreover, according to Table 2, the error distance in all eight erroneous cases is -1 s, which can be effectively compensated by feeding a bit $+1$ by an error recovery module.

As shown in Table 2, in eight cases, the error distance is -1 , which can be compensated by entering a bit $+1$ by an error recovery module. Since the errors occur only when $x_1 = '1'$, error detection and recovery are much more straightforward, and more compressors are recovered as compared to the previous designs performing error

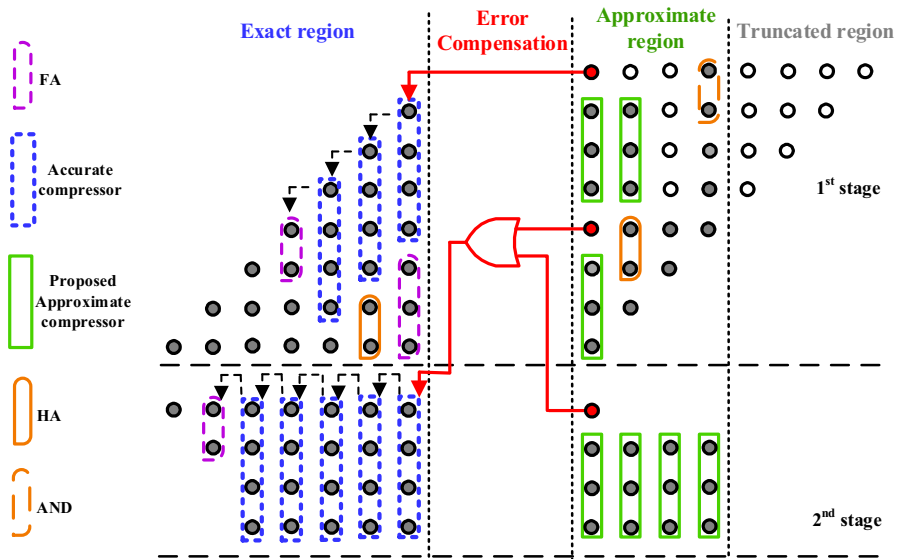


Fig. 3 Partial product reduction stages of the proposed approach

recovery [14, 20, 32, 37]. Unlike the approach presented in [32], the error correction bit in our design will not be generated when the output is correct.

The partial product reduction stage of the proposed approximate multiplier is shown in Fig. 3.

As low-bit multipliers are widely used in machine learning and neural network applications [1], we concentrate on an 8-bit Dadda multiplier without losing generality.

Moreover, according to the comprehensive analysis performed in [38], approximate compressors are well suited for implementing unsigned multipliers, and using them in signed multipliers can significantly degrade precision. For this reason and also to have fair comparisons with the previous approximate multipliers with the error compensation module [14, 20, 32, 37], we focus on unsigned approximate multipliers.

Similar to the other approximate multiplier with error recovery [18, 32, 42], the proposed multiplier has three parts:

1. The 4-bit truncated region (four least-significant columns)
2. The 4-bit approximate region, in which we use the suggested approximate compressor. This leads to a significant reduction in hardware and energy consumption while not having much effect on the accuracy of the circuit.
3. The accurate area, including the seven most significant columns containing much information, uses exact compressors [6] to maintain accuracy and compensate for the approximate area errors.

We can effectively trade-off between accuracy and energy efficiency using a combination of exact and approximate compressors. As we use a full adder as an approximate 4–2 compressor, the input x_1 is ignored. Therefore, the partial products being fed to the x_1 inputs of the approximate compressors are not produced (indicated with hollow

circles in the approximate region). In the first stage of the first column of the approximate region, three PPs are passed to the second stage, where they are added using the proposed approximate compressor. The remaining two PPs should be added using a half adder (HA). However, as the Sum output of the HA is not used in the second stage, the HA is replaced with one AND gate generating the Carry output. In the second stage of the second column, two PPs from the first stage and the Carry output generated by the AND gate in the first column are added using the proposed compressor. In the first stage of the third column, three PPs are added using the proposed compressors, and two other PPs are added using a HA. The Sum outputs of the compressor and HA circuits and one of the PPs are added in the second stage using the proposed approximate compressor. In the first stage of the fourth column, two sets of three PPs are added using two proposed approximate compressors. The Sum outputs of these compressors and the Carry output of the previous stage's compressor are added using the proposed compressors in the second stage. Two PPs from the first stage of the fourth column of the approximate region and the Carry output of the HA of the third column (x_1 input of the compressors) are used for error compensation. Accordingly, besides the more hardware-efficient structure of the approximate compressor, it also offers a considerable simplification of the structure of the approximate multiplier.

As shown in Fig. 3, the error recovery bit is generated only in the most significant column of the approximate region to have the greatest impact on accuracy while minimizing the overall hardware overhead. This column includes three approximate compressors, and the errors of all compressors in this column should be compensated. As indicated in Table 2, the error distance of the suggested approximate 4:2 compressor is -1 in the eight erroneous cases. Accordingly, the error can be compensated by adding a bit $+1$. Since the errors occur only when $x_1 = '1'$, the errors can be simply detected and compensated using an OR logic, as shown in Fig. 3. Two of the x_1 inputs, ignored in the approximate compressors, are fed to a two-input OR gate, and the output of the OR gate is used as one of the inputs of an exact 4:2 compressor in the first column of the exact region. Moreover, the other x_1 input of the last column of the approximate region is given to another exact 4:2 compressor in the first column of the exact region. Accordingly, when an error is detected in the last column of the approximate region and a bit $+1$ is entered into an exact compressor with a double significance, first, the error distance of the last approximate column is compensated, and secondly, some of the errors created by the previous compressors are also compensated.

In [14, 20, 37, 38], the total occurrence probabilities of the erroneous cases are 6.25%, 1.56%, and 6.25, respectively. This probability is 25% in our proposed approach, more than the designs suggested in [14, 20, 37, 38]. Therefore, feeding a $+1$ to the column with a double significance has a more considerable impact on the error compensation in our design than its previous counterparts.

Using only a two-input CMOS OR gate with six transistors in the error compensation part, the proposed multiplier can feed the error compensation bits into three approximate compressors of the exact region. Accordingly, the proposed error compensation approach is more efficient compared to its counterparts [14, 20, 32, 37].

4 Simulation and Comparisons

In this section, the proposed approximate design is simulated and evaluated in various aspects of accuracy and performance compared with its contemporary counterpart.

4.1 Hardware Analysis

The proposed approximate multiplier and the previous approximate multipliers with error recovery capability have been simulated using the HSPICE tool and the 7 nm tri-gate FinFET technology, one of the current industrial technologies. The critical parameters of the FinFET model used [10] are presented in Table 3. The simulations have been performed with 0.7 V supply voltage and 2 GHz frequency.

For a fair comparison, we have simulated all of the previous multipliers with error recovery capability according to the details provided in the corresponding references using 7 nm FinFET technology. Due to fin quantization, the number of Fins should increase to enhance the driving current of a FinFET. However, because of the 3-dimensional structure of FinFET, it significantly increases the switching capacitance and heat dissipation. Furthermore, it enlarges the area due to the minimum fin pitch. Therefore, as the power and area are two critical parameters in approximate circuits, and as the electrons and holes can have near mobilities in FinFET [13], single-fin FinFETs have been used. It is also worth mentioning that the efficient exact 4:2 compressor proposed in [6] has been used in the exact part for all of the approximate multipliers to make a fair comparison.

The simulation results are given in Table 4. It is worth mentioning that the delay of each multiplier is the worst-case critical path delay. Moreover, a long stream of random input combinations, considering switching activity, have been generated and used in simulations to estimate the power dissipation of the multipliers. Power-delay product (PDP) and energy-delay product (EDP) have also been calculated to make a trade-off between power and delay and have an illustrative evaluation regarding the energy efficiency of the multipliers. In addition, the area of each multiplier has

Table 3 The critical parameters of the 7 nm FinFET model [10]

Parameters	<i>n</i> -type	<i>p</i> -type
Physical fin thickness	7 nm	7 nm
Fin height	32 nm	32 nm
Gate length	21 nm	21 nm
Fin pitch	27 nm	27 nm
Equivalent oxide thickness	1.25 nm	1.25 nm
Body doping	10^{16} cm^{-3}	10^{16} cm^{-3}
Source/drain doping	$2 \times 10^{20} \text{ cm}^{-3}$	$2 \times 10^{20} \text{ cm}^{-3}$
Low field mobility (μ_0)	$260 \text{ cm}^2/\text{V s}$	$220 \text{ cm}^2/\text{V s}$
Gate work function (Φ_M)	4.372 eV	4.812 eV

Table 4 Simulation results of the 8×8 multipliers

Dadda multipliers with error correction	Critical path delay (ps)	Power (μ W)	PDP (fJ)	EDP (J ps)	Transistors	Area (μm^2)
Proposed	214	12.51	2.68	573	1212	10.09
Ha [14]	247	13.41	3.31	818	1302	10.84
Strollo_a [37]	280	16.52	4.63	1295	1452	12.08
Strollo_b [37]	280	17.18	4.81	1347	1534	12.77
Pei_1 [32]	255	15.34	3.91	997	1364	11.35
Pei_2 [32]	255	15.62	3.98	1016	1392	11.59
Pei_3 [32]	255	15.28	3.88	994	1360	11.32
Pei_4 [32]	255	15.23	3.88	990	1344	11.19
Kumar [20]	250	15.32	3.83	958	1366	11.37
Exact (base on [6])	253	18.80	4.76	1203	1530	12.74

been estimated as the area occupied by its transistors based on the FinFET geometries described in detail in [35] ($\lambda = 7$ nm) and Table 3.

According to the simulation results, the proposed design improves performance metrics significantly compared to the exact multiplier. Compared to the exact multiplier, the proposed multiplier reduces PDP, EDP, and area by 43.8%, 52.4%, and 21%.

Most of the other approximate multipliers listed in Table 4 have longer critical path delays than the exact multiplier designed based on the efficient exact compressor presented in [6]. The proposed multiplier improves the delay, power, and area compared to other approximate multipliers with error recovery modules. According to the simulation results, the proposed approximate multiplier improves the PDP, EDP, and area, on average, by 33%, 44.4%, and 12.5%, compared to the other approximate counterparts with the error recovery module. This is due to the lower complexity of the proposed multiplier and its straightforward error compensation module.

4.2 Accuracy Metrics

The NMED, MRED, and NoEB metrics have been calculated for the approximate multipliers according to Eqs. 1–3 using MATLAB, considering all 65,536 input combinations to evaluate the accuracy of the approximate multipliers. The simulated accuracy metrics are tabulated in Table 5.

Although the Strollo_a and Strollo_b multipliers [37] apply the same approximate compressors and error correction module, the first design has much better accuracy metrics due to using a higher number of exact columns, leading to higher energy consumption and hardware overhead. The multipliers proposed in [14] and [38] have superior accuracy metrics mainly due to their low error distances, which have been

Table 5 Accuracy metrics of the approximate multipliers

Multipliers with error correction	NMED ($\times 10^{-4}$)	MRED ($\times 10^{-2}$)	NoEB
Proposed	9.3	2.06	9.7
Ha [14]	4.8	0.90	10.3
Strollo_a [37]	4.0	0.74	10.4
Strollo_b [37]	15.0	1.10	8.1
Pei_1 [32]	8.0	4.61	9.9
Pei_2 [32]	5.0	1.41	10.5
Pei_3 [32]	5.8	1.54	10.3
Pei_4 [32]	7.2	1.83	10.0
Kumar [20]	4.2	0.70	10.4

achieved with the cost of more hardware overhead. As high-accuracy multipliers with error recovery modules have been compared, achieving more energy efficiency without considerable accuracy loss is the main challenge. As shown in Table 5, the proposed design has acceptable error metrics. The MRED of the proposed multiplier is less than Pei_1, and its NMED is less than Strollo_b, which are among the high-accuracy multipliers. This is mainly due to the efficient error correction approach of the proposed multiplier, which effectively compensates for the negative error distances. In addition to the reduced hardware overhead and energy dissipation, the proposed multiplier also has very low NMED and MRED values (compared to the mid-accuracy approximate multipliers such as [12, 27, 34, 39]).

Regarding the NoEB metric, the multipliers have almost comparable performance, except Strollo_b [37], which has a relatively lower NoEB than the other approximate multipliers mainly due to the higher number of approximate columns.

The proposed approximate multiplier's high accuracy leads to a satisfactory quality in image processing and neural network applications, as will be discussed in the following sub-section.

4.3 Applications

4.3.1 Neural Networks

This section evaluates the performance of each of the approximate multipliers in neural network applications. Neural networks are outstanding solutions for many machine learning tasks, such as image classification [36]. As the neural networks are inherently error-resilient and use multiplication in their structure, approximate multiplication can be considered a promising solution for significantly reducing energy consumption [5].

A multilayer perceptron (MLP) and a convolutional neural network (CNN) are considered to assess the performance of the approximate multipliers in classifications of the MNIST (Mixed National Institute of Standards and Technology) and SVHN (Street View House Numbers) datasets, respectively.

MNIST is a dataset of handwritten numbers, including 60,000 images for training and 10,000 for inference [22]. We use an MLP network to classify this dataset. This NN consists of 784 input neurons (each input image contains 28×28 pixels), 50 neurons in the hidden layer, and ten output neurons. The outputs are considered the probability of each of the classifications into ten classes (digits 0 to 9) [5]. This MLP uses the ReLU activation function, which is a simple $\max(0, x)$ function. SVHN is a real-world image dataset that houses numbers in Google Street View images. It consists of 73,257 images for training and 26,032 images for testing the network [29]. We use the LeNet-5 network to classify this dataset [21]. This network has seven layers, including three convolutional layers, two pooling layers, and two fully connected layers. We also use the ReLU activation function in this case. We have converted the original 32×32 RGB images to 32×32 grayscale images using the “Luma” mapping, reducing complexity [29].

These two neural networks have been simulated and analyzed in MATLAB. In the training process, accurate multipliers have been employed to calculate the neuron weights, and the approximate multipliers have been used for the classification process.

It is worth mentioning that both negative and positive neuron weights are generated in the training process. Accordingly, we need signed approximate multipliers. Accordingly, after the training process, we converted the neuron weights into sign-magnitude representation, and then the multiplications were performed using the discussed approximate multipliers. However, as we only convert the neuron weights once after the training process, it does not result in power and delay overheads in the classification process, where we use approximate circuits. Suppose the outputs of the neural network should be converted to 2's complement. In that case, its energy overhead is also negligible compared to the whole neural network (in the neural networks under study, there are only ten 8-bit outputs).

The classification accuracy for each of the multipliers is shown in Fig. 4. For the MNIST dataset, the 8-bit exact multiplier [6] has an accuracy of 96.5%, and the approximate multipliers show accuracies between 95 and 95.2% (at most 1.5% less than the exact multiplier), except the Pei_1 multiplier. To compensate for the error caused by an approximate compressor chain in the Pei_1 multiplier, a bit + 1 is fed to the exact compressor of the first column in the exact region. A detailed analysis has revealed that when using the MNIST dataset, one of the input operands is zero in more than 80% of cases [28]. However, the error correction bit provides a nonzero output value, and this error is summed. As shown in Fig. 4, the proposed multiplier with an accuracy of 95.2% (equal to Ha) is the best approximate multiplier regarding the classification accuracy.

For the SVHN dataset, the accuracy of the exact multiplier is 81.5%. In this case, the accuracies of the approximate multipliers are between 81.4 and 81.6%. This demonstrates that almost all approximate compressors with error recovery modules provide classification accuracies comparable to the exact design.

The classification accuracy results demonstrate that our proposed approximate multiplier can be considered an efficient alternative for the exact multiplier in neural network applications, as it offers significantly lower hardware and energy overheads while providing a comparable accuracy.

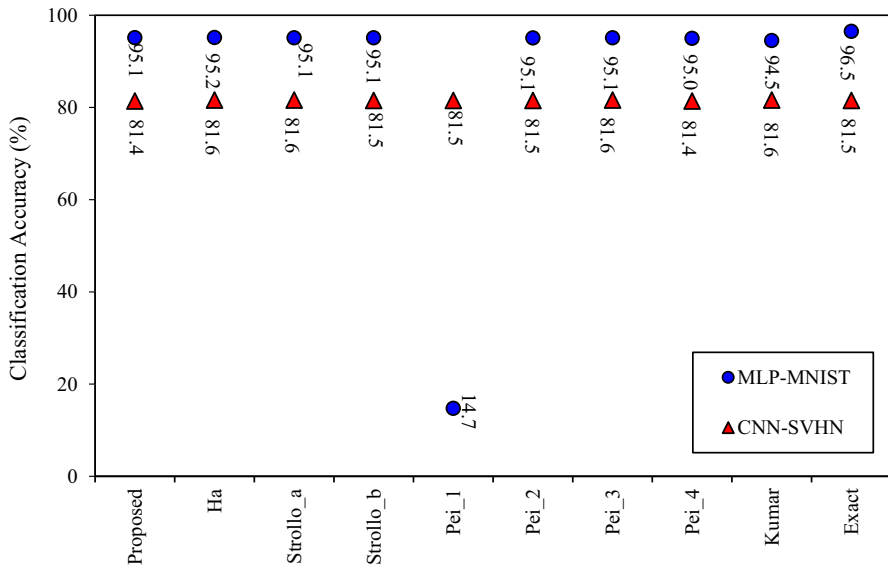


Fig. 4 Classification accuracies of the MNIST and SVHN datasets using the multipliers under study

4.3.2 Image Processing

As a widely used algorithm in image processing, image multiplication is considered to evaluate the quality of the approximate multipliers in real-world applications. The output image is calculated for the multiplication algorithm by multiplying two images pixel-by-pixel in MATLAB. The PSNR and MSSIM metrics have been calculated to compare the approximate output quality multipliers. Tables 6 and 7 present the PSNR and MSSIM values as two application-dependent metrics for the image multiplication.

Table 6 The PSNRs for the image multiplications

Multipliers	House \times sky	Lake \times lake gray	Sky \times Lena	Wheel \times Jetplane_gray
Proposed	54.49	54.73	54.41	54.77
Ha [14]	56.86	57.09	57.00	56.77
Strollo_a [37]	57.83	58.43	58.29	57.44
Strollo_b [37]	49.08	48.54	49.83	45.47
Pei_1 [32]	55.31	55.22	55.09	55.66
Pei_2 [32]	57.24	57.13	57.15	56.96
Pei_3 [32]	56.43	56.54	56.47	56.49
Pei_4 [32]	55.46	55.39	55.51	55.14
Kumar [20]	57.29	57.61	57.63	57.48

Table 7 The MSSIMs for the image multiplications

Multipliers	House \times sky	Lake \times lake gray	Sky \times Lena	Wheel \times Jetplane_gray
Proposed	0.9997	0.9994	0.9996	0.9992
Ha [14]	0.9998	0.9996	0.9998	0.9994
Strollo_a [37]	0.9998	0.9997	0.9999	0.9995
Strollo_b [37]	0.9990	0.9990	0.9990	0.9991
Pei_1 [32]	0.9997	0.9994	0.9997	0.9993
Pei_2 [32]	0.9998	0.9996	0.9998	0.9994
Pei_3 [32]	0.9998	0.9996	0.9998	0.9994
Pei_4 [32]	0.9997	0.9994	0.9997	0.9992
Kumar [20]	0.9998	0.9997	0.9998	0.9995

According to the results, the proposed multiplier offers PSNR greater than 50 dB and MSSIM greater than 0.999, which are more than enough for error-resilient applications where quality is also vital. The proposed multiplier has 11.7% higher PSNR than Strollo_b, while this design has, on average, 38.3% more hardware overhead than the proposed design. The PSNR and MSSIM of the proposed approach are, on average, 3.6% and 0.015% less than the other seven multipliers, which are negligible. In contrast, the proposed design offers considerably lower hardware overheads (see Table 4).

To provide a visual evaluation regarding the impact of the proposed approximate multiplier on image quality, an example of image multiplication based on the exact and our proposed multipliers is illustrated in Fig. 5. The results conclude that no difference is visible between the outputs of the exact and our proposed approximate multipliers.

In order to have a more comprehensive evaluation and a fairer judgment, the figure of merit (FOM) given in Eq. 8 is considered. Many figures of merit, considering hardware and quality parameters, have been introduced in the literature [2–4, 34, 42]. However, each of these FOMs puts more importance on some parameters. The FOMs given in Eqs. (8)–(10) reasonably consider various aspects of performance and accuracy for an approximate multiplier.

$$\text{FOM}_1 = \frac{\text{EDP}}{\text{PSNR} \times \text{MSSIM}} \quad (8)$$

$$\text{FOM}_2 = \frac{\text{PDP}}{\text{PSNR} \times \text{MSSIM}} \quad (9)$$

$$\text{FOM}_3 = \frac{\text{ADP}}{\text{PSNR} \times \text{MSSIM}} \quad (10)$$

The FOMs of the approximate multipliers are shown in Fig. 6. According to the results, the proposed design improves FOM1, FOM2, and FOM3, on average, by 43%, 31%, and 26%, respectively, compared to the other high-accuracy approximate multipliers. Accordingly, our proposed design offers a more effective trade-off between



Fig. 5 An example of image multiplication

hardware efficiency and accuracy and is a promising approach for approximate computing applications.

5 Conclusion

In this paper, we have proposed an energy-efficient approximate multiplier in which a full adder has been used as an approximate 4:2 compressor. Besides simplifying the multiplier structure, it leads to a straightforward error recovery module consisting of only a two-input OR gate. The proposed multiplier and the other multipliers with error correction modules have been simulated using the 7 nm tri-gate FinFET model. Moreover, they have been investigated in neural networks and image processing real-life applications. While the proposed design significantly improves the energy consumption and number of transistors compared to its counterparts, it is categorized as a high-accuracy approximate multiplier with a PSNR greater than 50 dB. Moreover, the classification accuracy results reveal that our proposed approximate multiplier can be an efficient alternative for the exact multipliers in neural network applications. Our results conclude that the proposed design offers an effective trade-off between hardware efficiency and accuracy and is a promising design for error-resilient applications.

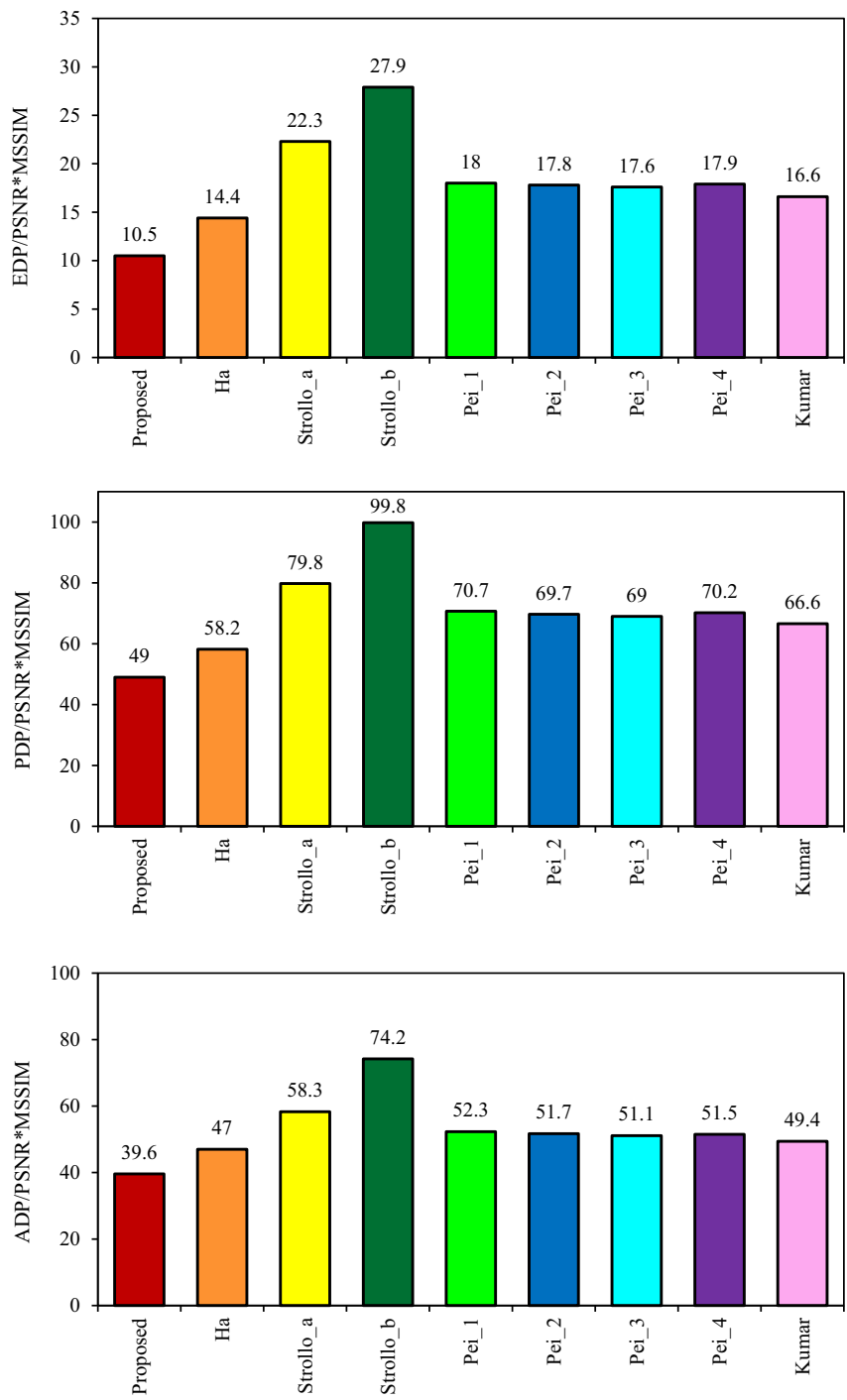


Fig. 6 The FOMs for the approximate multipliers

Data Availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

References

1. H. Afzali-Kusha, M. Vaeztourshizi, M. Kamal, M. Pedram, Design exploration of energy-efficient accuracy-configurable Dadda multipliers with improved lifetime based on voltage overscaling. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **28**, 1207–1220 (2020). <https://doi.org/10.1109/tvlsi.2020.2978874>
2. M. Ahmadinejad, M.H. Moaiyeri, F. Sabetzadeh, Energy and area efficient imprecise compressors for approximate multiplication at nanoscale. *AEU Int. J. Electron. Commun.* (2019). <https://doi.org/10.1016/j.aeue.2019.152859>
3. O. Akbari, M. Kamal, A. Afzali-Kusha, M. Pedram, Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **25**, 1352–1361 (2017). <https://doi.org/10.1109/tvlsi.2016.2643003>
4. M.S. Ansari, H. Jiang, B.F. Cockburn, J. Han, Low-power approximate multipliers using encoded partial products and approximate compressors. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **8**, 404–416 (2018). <https://doi.org/10.1109/jetcas.2018.2832204>
5. M.S. Ansari, V. Mrazek, B.F. Cockburn, L. Sekanina, Z. Vasicek, J. Han, Improving the accuracy and hardware efficiency of neural networks using approximate multipliers. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **28**, 317–328 (2020). <https://doi.org/10.1109/tvlsi.2019.2940943>
6. A. Arasteh, M. Hossein Moaiyeri, M. Taheri, K. Navi, N. Bagherzadeh, An energy and area efficient 4:2 compressor based on FinFETs. *Integration* **60**, 224–231 (2018). <https://doi.org/10.1016/j.vlsi.2017.09.010>
7. D. Baran, M. Aktan, V.G. Oklobdzija, Energy efficient implementation of parallel CMOS multipliers with improved compressors, in *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics And Design—ISLPED'10* (2010). <https://doi.org/10.1145/1840845.1840876>
8. C.H. Chang, J. Gu, M. Zhang, Ultra low-voltage low-power CMOS 4–2 and 5–2 compressors for fast arithmetic circuits. *IEEE Trans. Circuits Syst. I Regul. Pap.* **51**, 1985–1997 (2004). <https://doi.org/10.1109/tcsi.2004.835683>
9. Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun and O. Temam, DaDi-anNao: a machine-learning supercomputer, in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture* (2014), pp. 609–622. <https://doi.org/10.1109/MICRO.2014.58>
10. L.T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, G. Yeric, ASAP7: a 7-nm finFET predictive process design kit. *Microelectron. J.* **53**, 105–115 (2016). <https://doi.org/10.1016/j.mejo.2016.04.006>
11. D. Esposito, A.G.M. Strollo, E. Napoli, D. De Caro, N. Petra, Approximate multipliers based on new approximate compressors. *IEEE Trans. Circuits Syst. I Regul. Pap.* **65**, 4169–4182 (2018). <https://doi.org/10.1109/TCSI.2018.2839266>
12. A. Gorantla, Design of approximate compressors for multiplication. *ACM J. Emerg. Technol. Comput. Syst.* **13**, 1–17 (2017). <https://doi.org/10.1145/3007649>
13. S.K. Gupta, K. Roy, Low power robust FinFET-based SRAM design in scaled technologies, in *Circuit Design for Reliability* (2015), pp. 223–253. <https://doi.org/10.1109/les.2017.2746084>
14. M. Ha, S. Lee, Multipliers with approximate 4–2 compressors and error recovery modules. *IEEE Embed. Syst. Lett.* **10**, 6–9 (2018). <https://doi.org/10.1109/LES.2017.2746084>
15. J. Han, M. Orshansky, Approximate computing: An emerging paradigm for energy-efficient design, in *2013 18th IEEE European Test Symposium (Ets)* (2013), pp. 1–6. <https://doi.org/10.1109/ETS.2013.6569370>
16. H. Jiang, S. Angizi, D. Fan, J. Han, L. Liu, Non-volatile approximate arithmetic circuits using scalable hybrid spin-CMOS majority gates. *IEEE Trans. Circuits Syst. I Regul. Pap.* **68**, 1217–1230 (2021). <https://doi.org/10.1109/tcsi.2020.3044728>
17. L. Jinghang, H. Jie, F. Lombardi, New metrics for the reliability of approximate and probabilistic adders. *IEEE Trans. Comput.* **62**, 1760–1771 (2013). <https://doi.org/10.1109/tc.2012.146>

18. M.S. Kim, A.A.D. Barrio, L.T. Oliveira, R. Hermida, N. Bagherzadeh, Efficient Mitchell's approximate log multipliers for convolutional neural networks. *IEEE Trans. Comput.* **68**, 660–675 (2019). <https://doi.org/10.1109/tc.2018.2880742>
19. M.S. Kim, A.A. Del Barrio Garcia, H. Kim, N. Bagherzadeh, The effects of approximate multiplication on convolutional neural networks. *IEEE Trans. Emerging Top. Comput.* (2021). <https://doi.org/10.1109/tetc.2021.3050989>
20. U.A. Kumar, S.K. Chatterjee, S.E. Ahmed, Low-power compressor-based approximate multipliers with error correcting module. *IEEE Embed. Syst. Lett.* (2021). <https://doi.org/10.1109/les.2021.3113005>
21. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998). <https://doi.org/10.1109/5.726791>
22. Y.J.H.Y.L.C.E.M. Lecun, *The MNIST Database of Handwritten Digits* (1998).
23. V. Leon, G. Zervakis, D. Soudris, K. Pekmetzi, Approximate hybrid high radix encoding for energy-efficient inexact multipliers. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **26**, 421–430 (2018). <https://doi.org/10.1109/tvlsi.2017.2767858>
24. C. Liu, *Design and Analysis of Approximate Adders and Multipliers* (University of Alberta, 2014). <https://doi.org/10.7939/R3M38H>
25. W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, F. Lombardi, Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications. *IEEE Trans. Circuits Syst. I Regul. Pap.* **65**, 2856–2868 (2018). <https://doi.org/10.1109/tcsi.2018.2792902>
26. M.H. Moaiyeri, F. Sabetzadeh, S. Angizi, An efficient majority-based compressor for approximate computing in the nano era. *Microsyst. Technol.* **24**, 1589–1601 (2017). <https://doi.org/10.1007/s00542-017-3587-2>
27. A. Momeni, J. Han, P. Montuschi, F. Lombardi, Design and analysis of approximate compressors for multiplication. *IEEE Trans. Comput.* **64**, 984–994 (2015)
28. V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, K. Roy, Design of power-efficient approximate multipliers for approximate artificial neural networks, in *Proceedings of the 35th International Conference on Computer-Aided Design* (2016), pp. 1–7. <https://doi.org/10.1145/2966986.2967021>
29. Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, *Reading Digits in Natural Images with Unsupervised Feature Learning* (2011)
30. P.E. Novac, G. Boukli Hacene, A. Pegatoquet, B. Miramond, V. Gripon, Quantization and deployment of deep neural networks on microcontrollers. *Sensors (Basel)* (2021). <https://doi.org/10.3390/s21092984>
31. S. Panchanan, R. Maity, S. Baishya, N. PratapMaity, A surface potential model for tri-gate metal oxide semiconductor field effect transistor: analysis below 10 nm channel length. *Eng. Sci. Technol. Int. J.* **24**, 879–889 (2021). <https://doi.org/10.1016/j.jestech.2020.12.020>
32. H. Pei, X. Yi, H. Zhou, Y. He, Design of Ultra-low power consumption approximate 4–2 compressors based on the compensation characteristic. *IEEE Trans. Circuits Syst. II Express Briefs* **68**, 461–465 (2021). <https://doi.org/10.1109/tcsii.2020.3004929>
33. K. Roy, A. Raghunathan, Approximate computing: an energy-efficient computing technique for error resilient applications. *IEEE Comput. Soc. Annu. Symp. VLSI* **2015**, 473–475 (2015). <https://doi.org/10.1109/ISVLSI.2015.130>
34. F. Sabetzadeh, M.H. Moaiyeri, M. Ahmadinejad, A majority-based imprecise multiplier for ultra-efficient approximate image multiplication. *IEEE Trans. Circuits Syst. I Regul. Pap.* **66**, 4200–4208 (2019). <https://doi.org/10.1109/tcsi.2019.2918241>
35. S. M. Salahuddin, J. Hailong, V. Kursun, A novel 6T SRAM cell with asymmetrically gate underlap engineered FinFETs for enhanced read data stability and write ability. in *International Symposium on Quality Electronic Design (ISQED)* (2013), pp. 353–358. <https://doi.org/10.1109/ISQED.2013.6523634>
36. J. Schmidhuber, Deep learning in neural networks: an overview. *Neural Netw.* **61**, 85–117 (2015). <https://doi.org/10.1016/j.neunet.2014.09.003>
37. A.G.M. Strollo, D. De Caro, E. Napoli, N. Petra, G. Di Meo, Low-power approximate multiplier with error recovery using a new approximate 4–2 compressor, in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)* (2020), pp. 1–4. <https://doi.org/10.1109/ISCAS45731.2020.9180767>
38. A.G.M. Strollo, E. Napoli, D. De Caro, N. Petra, G. Di Meo, Comparison and extension of approximate 4–2 compressors for low-power approximate multipliers. *IEEE Trans. Circuits Syst. I Regul. Pap.* **67**, 3021–3034 (2020). <https://doi.org/10.1109/TCSI.2020.2988353>

39. S. Venkatachalam, S.-B. Ko, Design of power and area efficient approximate multipliers. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **25**, 1782–1786 (2017). <https://doi.org/10.1109/tvlsi.2016.2643639>
40. Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.* **13**, 600–612 (2004). <https://doi.org/10.1109/tip.2003.819861>
41. N.H.E. Weste, D.F. Harris, D.M. Harris, P.E.D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective* (Pearson/Addison-Wesley, 2005)
42. Z. Yang, J. Han, F. Lombardi, Approximate compressors for error-resilient multiplier design. *IEEE Int. Symp. Defect Fault Toler. VLSI Nanotechnol. Syst. (DFTS)* **2015**, 183–186 (2015). <https://doi.org/10.1109/DFT.2015.7315159>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.