Journal of Information Systems Engineering and Management
2025, 10(25s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Efficient Approximate Multiplier for Image Processing Application

Arham Dodal[1], Marimuthu R[2], Prof. Shweta Nishit Jain[3], S. Ravi[4], Dr. Sonali S. Patil[5]
[1]Department of Electrical and Electronics Engineering, Vellore Institute of Technology, Vellore, India. arhamdodal@gmail.com
[2]School of Electrical Engineering, Vellore Institute of Technology, Vellore, India. rmarimuthu@vit.ac.in
[3]Department of Electronics and Telecommunication, International Institute of Information Technology, Pune, India. shwetaj@isquareit.edu.in
[4]School of Electronics Engineering, Vellore Institute of Technology,Vellore, India. msravi@vit.ac.in
[5]Department of Information Technology, International Institute of Information Technology, Pune, India. sonalip@isquareit.edu.in

ARTICLE INFO

Received: 28 Dec 2024

Revised: 16 Feb 2025

Accepted: 28 Feb 2025

ABSTRACT

The field of approximate computing has garnered significant interest as a promising area for relatively error-resistant applications. At its core, approximate computing revolves around a trade-off between efficiency and accuracy. To optimize power consumption, delay, and area, a certain level of accuracy is sacrificed, provided it remains within acceptable limits. This research introduces a novel design for an approximate 4:2 compressor. Two distinct configurations for implementing this compressor are proposed and evaluated within the framework of an 8x8 Dadda multiplier. The study assesses both technology-dependent and technology-independent parameters, comparing them with the most recent approximate multipliers reported in recent literature. The performance of these multipliers is tested using a 45-nm standard CMOS technology node. Additionally, the quality and precision of the proposed approximate multiplier are evaluated using a range of statistical metrics. To demonstrate their practical utility, the proposed multipliers are applied to image processing tasks. The results indicate that the proposed designs outperform existing approximate multipliers in terms of efficiency for image processing applications.

Keywords: approximate computing, image quality metrics, VLSI, compressors.

INTRODUCTION

In the realm of computer arithmetic, digital logic circuits are widely employed due to their exceptional reliability and precision. However, in many applications, particularly those related to machine learning, multimedia and image processing, a certain level of computational error or imprecision is not only tolerated but can also produce useful results. In such cases, striving for the highest accuracy and precision in models and algorithms may not always be the most efficient or practical approach. To address the power and speed requirements of these applications, various methods have been proposed across different design abstraction levels. Approximate computing methods, in particular, are explicitly designed to meet these specifications by allowing for a controlled reduction in computational accuracy. This approach is well-suited for applications that do not require a single correct answer but instead need acceptable results that are sufficiently close to the actual solution.

Approximate computing has demonstrated significant success in numerous error-resilient applications. For instance, in multimedia data processing, minor inaccuracies are often imperceptible to human senses, making approximation an acceptable trade-off. Similarly, in large-scale data mining and pattern recognition, approximate results can still provide meaningful insights. In machine learning, particularly during the training phase, approximate computations can accelerate convergence while maintaining the overall effectiveness of the model [1], [2].

Multiplier design generally involves three key steps. Firstly, generating partial products which are intermediate products obtained using arrays of AND logic gates. Secondly, summing the intermediate results. And finally, computing the result (product) using intermediate adder circuits. Among these stages, the stage where the partial products are reduced is the most computationally demanding, consuming the most power and requiring the largest area. [3], [4]. This is quite evident as the partial product reduction stage has the most amount of raw calculation. Hence, optimizing this step is critical to increase the efficiency and energy use of the multiplier. Reducing partial products is done using computational modules called compressors. Full adders play a crucial role in both Dadda and Wallace multipliers. Nonetheless, more advanced compressor designs, like 4:2 or 5:2 structures, offer greater efficiency in the stage where intermediate results are summed facilitating the development of more efficient multiplier architectures [5].

In this study, we present and analyze a novel design for an approximate 4:2 compressor. Our findings show that this

optimized compressor achieves superior performance with respect to delay and power efficiency when contrasted with traditional 4:2 compressor designs referenced in prior research [5]. The proposed compressors are integrated into the Dadda multiplier through two unique inexact multiplication approaches. Additionally, we provide simulation results based on 45nm CMOS technology, focusing on key metrics including delay, chip area, and power efficiency. The accuracy of the multiplier is assessed, and its application in image processing tasks is explored. Findings suggest that these proposed architectures are highly effective for approximate computing, achieving an optimal balance between accuracy and computational efficiency.

This paper is divided into five sections. Section 2 presents an overview of the fundamental concepts that serve as the foundation for this research. Section 3 details the development of the proposed approximate 4:2 compressor and its integration into the Dadda multiplier. Section 4 focuses on the simulation results, assessing design accuracy and performance in image processing applications. Lastly, Section 5 summarizes the key findings and conclusions derived from this study.

BACKGROUND

In the approximate computing literature, one can find numerous techniques are utilized to enhance efficiency while preserving an acceptable degree of accuracy. Components like full-adders, half-adders, and compressors are often replaced with their approximate versions to improve efficiency. In some cases, basic logic gates are also employed to simplify the reduction of partial products. Moreover, since the least significant bits have very little affect over the final result of the multiplication, their results are often not calculated but are rather set to a constant (either 0 or 1).

In [7], Momeni et. al. have designed two approximate compressors. They are created by altering the truth table of a precise compressor. Two distinct approaches for implementing each of these proposed approximate compressors have been introduced (in total four multiplier designs are proposed in the paper). The initial compressor design yields 12 errors out of 32 possible results. On the other hand, the second compressor generates just 4 errors out of 16 possible outcomes, reducing the error rate to 25%.

Ahmadinejad et al. [8] presented a new method to develop a simplified logic function for an approximate 4:2 compressor. By evaluating the likelihood of inputs to the compressor, the researchers adjusted the truth table of the exact compressor. Their 4:2 approximate design computes the Sum and Carry outputs using only the four input bits from the partial products, disregarding Cin and Cout. This simplification results in seven errors out of 16 possible input states. The design can be implemented using just 16 transistors. Additionally, the authors developed a very compact approximate 5:2 compressor that also omits Cin and Cout, requiring only 20 transistors for implementation. The compressors are designed keeping in mind the balance between computational accuracy and hardware cost. Ahmadinejad et. al. have provided two multiplier designs each employing the two inexact compressors respectively. The two proposed multiplier designs ignore the last four columns of partial products, in total setting 10 partial products to ”0”.

In [9], a 4:2 compressor is proposed by fixing Cin and Cout to '0'. This design employs only six transistors to produce the Sum and Carry, aiming to minimize energy consumption. Notably, the input x2 is ignored in this approach. The multiplier presented by Ejtahed [9] is segmented into three regions. However, since one input bit is disregarded by the compressor, certain products are also excluded. Moreover, in this design, the Carry output from the in Stage 1, generated by the compressors functions as an alternative input (x2) for the subsequent compressors, effectively skipping the computation of Carry in three instances and reducing the number of partial products by 29.

Minaeifar et al. [6] introduced three multiplier architectures that discard the least significant bits (LSBs) of operands to simplify calculations. By predefining the LSBs as zero, the design significantly cuts down transistor usage by eliminating 28 AND operations necessary for generating partial products. Their proposed models integrate both exact and approximate 4:2 compressors along with different types of adders to streamline partial product reduction and facilitate final result computation.

One notable feature of these designs [6] is the use of an OR gate instead of the conventional XOR gate for generating Sum outputs in the approximate half-adder, leading to hardware efficiency. Additionally, the Carry output remains functional through an AND gate, requiring 12 transistors for proper operation. The approximate full adder design simplifies the Sum output assignment by directly linking it to the x1 input. Meanwhile, the Carry output is derived using a three-input OR gate, incorporating x1, x2, and Cin, enabling the full adder to function with just eight transistors.

PROPOSED DESIGN

This section presents a design of an approximate compressor and two schemes for 8-bit multiplier using the proposed approximate compressor are presented.

Our design incorporates approximate half-adders and full-adders, along with the proposed approximate compressor. In the approximate half-adder design, the Sum output is obtained by utilizing an OR gate instead of the traditional XOR gate. Meanwhile, the Carry output continues to function through an AND gate, necessitating the use of 12 transistors for its realization [6]. Likewise, in the approximate full adder, the XOR gate is replaced with an OR gate to simplify the Sum computation. Meanwhile, the *Carry* output is derived from the AND operation of the three input signals. Thus, the implementation of the given approximate full adder requires 20 transistors. To the best of our knowledge, the approximation of the *Carry* output has not been implemented in this manner. In contrast, the approximation of the *Sum* output, as described, is frequently seen in the approximate computing literature.

1.1 Approximate Compressor Design

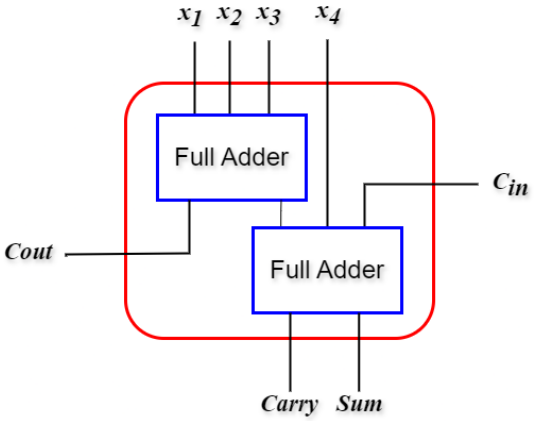


Figure 1: Schematic of a 4:2 compressor

Figure 1 illustrates a 4:2 compressor constructed using full adders. When implemented in CMOS technology, it typically uses 116 transistors. By replacing the conventional full-adder cells with their approximate counterparts, as discussed earlier, an accuracy level of 75% is achieved, calculated as the proportion of correct outputs relative to the total number of outputs. To further reduce transistor usage while maintaining this accuracy, we propose an innovative method. Our approach involves utilizing the same hardware configuration for generating both the Carry and Cout signals in the 4:2 compressor. Based on our review, no previous work has introduced a 4:2 compressor in the manner depicted in Figure 2. By reusing the hardware components, this design improves efficiency, maintaining an accuracy rate of 75% (24 correct outputs out of 32).

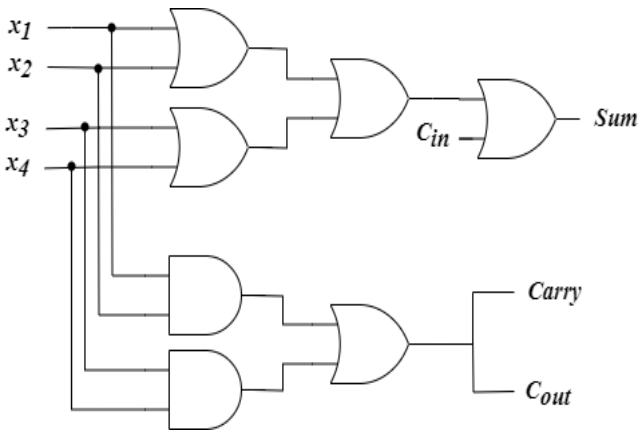


Figure 2: Schematic of 4:2 compressor

1.2 Multiplier Design

In this section, we integrate the proposed approximate compressor (figure 2) into an 8x8 Dadda multiplier. As discussed

earlier, the paper presents two unique designs for integrating these approximate components. Typically, parallel multipliers are organized into three stages. Firstly, partial products are generated using AND gates, while the second stage focuses on summing these partial products with the help of compressors and adders.

The third stage involves generating the final result, which is achieved through a ripple *Carry* adder. This stage completes the multiplication process by combining all the partial results into the final output. Approximation methods are typically applied in the initial two stages: generating partial products and reducing them. In the first stage, optimization can involve skipping the generation of lower-bit partial products entirely. In [8], the authors suggested truncating the lower four bits of the generated partial products, setting them to 0. In [6], a novel approach was introduced where the two most least significant bits (LSB) of the multiplier and multiplicand are ignored. Furthermore, the lower four bits of the product are set to '1'. Drawing inspiration from these methods, we truncated the lower five-bit and four-bit partial products in the first and second proposed multipliers, respectively, setting the corresponding product bits to 0. The approximate components, depicted in Figure 3 and Figure 4, are employed in the second stage. As discussed before, partial product reduction takes place in the second stage. Furthermore, in the final ripple *Carry* adder stage, we used approximate full adders and approximate half adders instead of the traditional full and half adders to compute the final output.

In Novel1, the C_{in} bit of the approximate compressor in the 8th and 6th column (8th/6th column from the LSB) is set to 0. Similarly, the x_4 bit of the compressor in the 9th column is set to 0 for both Novel1 and Novel2. It is important to note that in Novel2, the output of the OR operation in 5th column is not only the 5th output bit but also the C_{in} for the compressor in next column.

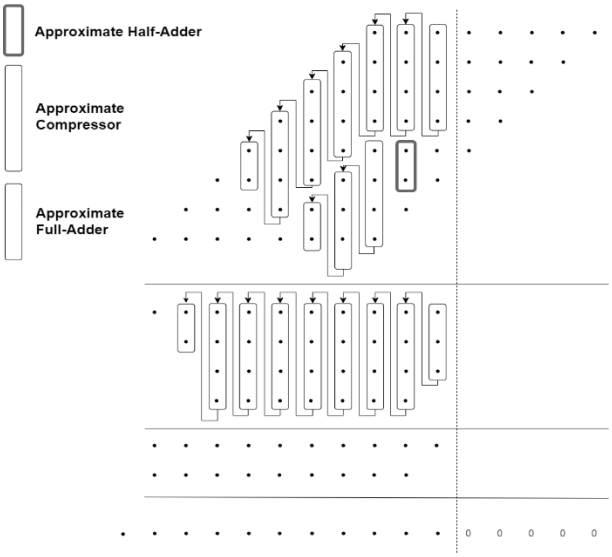


Figure 3: Design of NOVEL1 Multiplier

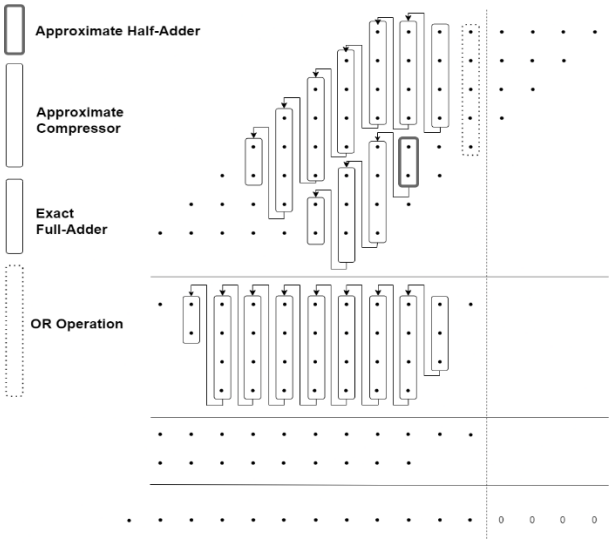


Figure 4: Design of NOVEL2 Multiplier

RESULTS

In this section, the proposed multipliers are scrutinized based on technology dependent and technology independent metrics. This section also describes the methods and criteria used to quantify the performance and accuracy of the multipliers in basic mathematical image processing problems, covering error evaluation metrics and comprehensive performance analysis. For a fair comparison, all approximate multipliers which are been used for comparison and along with the proposed designs are implemented using Python in the PyCharm IDE, following the guidelines outlined in the referenced articles. The performance of the two proposed multipliers is analyzed through transistor-level simulations using CMOS technology with a 45nm process node, conducted using Cadence's Genus and Innovus tools.

4.1 Error Evaluation Metrics

This section offers an overview of key aspects and widely recognized measures for evaluating the effectiveness of approximate multipliers. Typically, the discrepancy between two values, one being erroneous (a) and the other correct (b), is quantified using a metric called Error Distance. This is determined by computing the absolute difference between the two values, represented mathematically as E_{diff}.

$$E_{diff} = |a - b| \qquad (4)$$

The Normalized Mean Error Distance (NMED) and the Mean Relative Error Distance (MRED) are well-established metrics frequently used to quantify the accuracy of approximate circuits. These measures work by computing the average error distances across multiple test scenarios and then normalizing the obtained results [12].

$$NMED = \left(\frac{1}{2^N-1}\right)^2 \sum_{i=1}^{2^{2N}} \frac{E_d iff}{2^{2N}} \quad (5)$$

$$MRED = \frac{1}{2^{2N}} \sum_{i=1}^{2^{2N}} \frac{E_d iff}{M_i} \quad (6)$$

Where Mi is the maximum possible product.

Studies indicate that the Mean Relative Error Distance (MRED) exhibits exponential growth as the count of lower-order bits in an adder increases. While MRED serves as a valuable measure for evaluating adder performance, it may introduce bias when adders contain varying quantities of lower-order bits, leading to fluctuations in maximum error. To mitigate this issue, a refined version of MRED, referred to as the Normalized Error Distance (NMED), has been introduced as a standardized assessment metric [10].

Additionally, the concept of the Number of Effective Bits (NoEB) has been proposed, with ERMS denoting the root mean square error associated with approximate multipliers [13]. The NoEB value is determined through the equation provided below.

$$\text{NoEB} = 2N - \log_2(1 - \text{ERMS}) \quad (7)$$

1.3 Accuracy Analysis

A detailed list of accuracy metrics for the 8x8 approximate multipliers can be found in Table: 1. These values were obtained by assessing all 65,536 likely input combinations. The data indicate that proposed 'Novel1' and 'Novel2' multipliers do not significantly outperform the multipliers referenced form the articles in terms of the statistical metrics namely the NoEB, MRED, and NMED. This is partly due to the fact that the multipliers have their lower bits truncated.

Table 1: Multiplier Accuracy Analysis

Multiplier	MRED	NMED	NoEB
Minaeifar mul1 [6]	0.0685	0.0047	7.3
Minaeifar mul3 [6]	0.1208	0.0135	5.7
Momeni multiplier1 [7]	0.8	0.0601	3.9
Momeni multiplier2 [7]	0.59	0.082	3.5
Ejtahed [9]	0.0998	0.008	6.7
Ahmadinejad 4 2 [8]	0.0901	0.0195	4.9
Ahmadinejad 5 2 [8]	0.0875	0.019	4.9
NOVEL1	0.1622	0.0478	3.5
NOVEL2	0.1759	0.039	3.8

1.4 Image Processing Methods

This section evaluates the effectiveness of approximate multipliers in practical applications by utilizing them in three fundamental image processing tasks: image multiplication, sharpening, and smoothing.

For image multiplication, the multiplier processes two images on a pixel-by-pixel basis, combining them to generate a single resultant image. Figure 5 presents both the input images and the final processed output. Additionally, Equation 8 is used to compute the enhanced image in the sharpening process.

$$Y(x,y) = 2X(x,y) - 1 + \frac{1}{273}\sum_{m=-2}^{2}\sum_{n=-2}^{2}\text{Mask}_{sharpening}(m+3,n+3)\cdot X(x-m,y-n) \qquad (8)$$

Here, variables X and Y denote the input and output images, respectively, while the sharpening mask matrix is characterized as follows:

$$Mask_sharpening = \begin{bmatrix} 1,4,7,4,1; \\ 4,16,26,16,4; \\ 7,26,41,26,7; \\ 4,16,26,16,4; \\ 1,4,7,4,1 \end{bmatrix} \qquad (9)$$

Additionally, in the smoothing process, the final image is generated through the following approach:

$$Y(x,y) = \frac{1}{60}\sum_{m=-2}^{2}\sum_{n=-2}^{2} Mask_{smoothing}(m+3,n+3)\cdot X(x-m,y-n) \qquad (10)$$

Here the smoothing mask matrix used is

$$Mask_smoothing = \begin{bmatrix} 1,1,1,1,1; \\ 1,4,4,4,1; \\ 1,4,12,4,1; \\ 1,4,4,4,1; \\ 1,1,1,1,1 \end{bmatrix} \qquad (10)$$

It is crucial to emphasize that in both of these methods, fundamental computations like addition, subtraction, and division are expected to be performed with precision.

Within the realm of image processing, key metrics commonly aid in evaluating image quality: the Mean Structural Similarity Index Metric (MSSIM) and the Peak Signal-to-Noise Ratio (PSNR) [11]. PSNR plays a significant role in measuring image quality across different processing applications and is derived using the following equations:

$$MSE = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\big(x(i,j) - y(i,j)\big)^2 \qquad (12)$$

MSE refers to Mean Square Error, where M and N define the size of the image, while x(i,j) and y(i,j) denote the pixel values of the images under analysis.

$$PSNR = 10\cdot\log_{10}\left(\frac{MAX^2}{MSE}\right) \qquad (13)$$

The variable MAXI is the highest possible pixel value within an image.

Within digital image processing, the Mean Structural Similarity Index (MSSIM) serves as a widely used measure for comparing two images. Compared to conventional methods such as Mean Square Error (MSE) and Peak Signal-to-Noise Ratio (PSNR), MSSIM considers human visual perception rather than merely relying on pixel-level differences to provide a more precise assessment.

The Structural Similarity Index is calculated using the following equation: (14)

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \qquad (14)$$

Where,

- σx^2 and σy^2: Variance of images x and y respectively.
- μ_x and μ_y: Denotes the mean intensity values of images x and y.
- σ_{xy}: Indicates the covariance between images x and y.

- C_1 and C_2: Constants used to maintain stability in division operations.

The MSSIM is calculated by averaging the SSIM values from the three channels (15).

$$MSSIM(x,y) = \frac{1}{M}\sum_{j=1}^{M} SSIM(x_j, y_j) \qquad (15)$$

More information on MSSIM as given in eq. could be found in [14]

Table 2: PSNR and MSSIM Values for Multiplication Operation

Multiplier	**Lighthouse x Plane**		**Lena**	
	PSNR	**MSSIM**	**PSNR**	**MSSIM**
Novel1	29.985	0.760	31.172	0.852
Novel2	30.511	0.528	31.800	0.778
Minaeifar mul1 [6]	43.797	0.991	43.624	0.984
Minaeifar mul3 [6]	32.535	0.956	35.110	0.958
Momeni multiplier1 [7]	29.517	0.768	29.211	0.652
Momeni multiplier2 [7]	30.253	0.793	29.748	0.708
Ahmadinejad4 2 [8]	32.432	0.861	33.287	0.879
Ahmadinejad 5 2 [8]	32.249	0.870	34.255	0.889
Ejtahed [9]	40.830	0.984	41.006	0.971

(a) Grayscale Image

(b) RGB Image

(c) Exact output

Figure 5: (a) and (b) used for multiplication application.

(a) Novel1

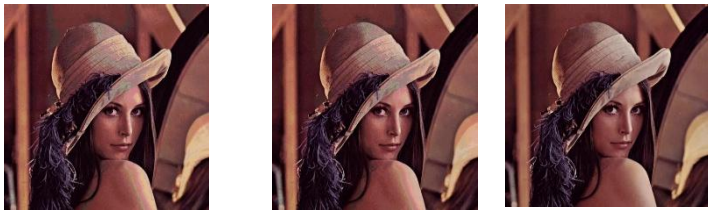
(b) Novel2

(c) Minaeifar mul1

(d) Minaeifar mul3

(e) Momeni mul1

(f) Momeni mul2



(h) Ahmadinejad 5:2 (i) Ejtahet (j) Ahmadinejad 4:2

Figure 6: Output images of multiplication operation using approximate multiplier.

Table 3: PSNR and MSSIM values for Image Sharpening Algorithm

Multiplier	**PSNR**				**MSSIM**			
	Plane	**Lighthouse**	**Boat**	**Lena**	**Plane**	**Lighthouse**	**Boat**	**Lena**
NOVEL1	26.509	25.261	24.919	25.342	0.938	0.912	0.953	0.926
NOVEL2	24.451	29.137	28.956	27.416	0.938	0.870	0.916	0.904
Momeni multiplier1 [7]	13.265	10.727	11.238	11.133	0.882	0.758	0.755	0.766
Momeni multiplier2 [7]	14.367	11.743	12.381	12.224	0.901	0.780	0.796	0.787
Minaeifar mul1 [6]	44.612	42.571	43.112	41.454	0.995	0.995	0.995	0.994
Minaeifar mul3 [6]	22.605	26.131	24.828	25.353	0.990	0.991	0.991	0.991
Ejtahed [9]	33.440	36.909	37.394	36.271	0.989	0.987	0.984	0.981
Ahmadinejad 4 2 [8]	32.543	35.225	32.128	33.364	0.975	0.972	0.973	0.970
Ahmadinejad 5 2 [8]	34.883	37.112	34.420	35.679	0.970	0.974	0.977	0.977

Table 4: PSNR and MSSIM values for Image Smoothing Algorithm

Multiplier	**PSNR**				**MSSIM**			
	Plane	**Lighthouse**	**Boat**	**Lena**	**Plane**	**Lighthouse**	**Boat**	**Lena**
NOVEL1	21.617	12.637	21.667	13.564	0.926	0.636	0.908	0.717
NOVEL2	30.790	12.938	24.866	13.849	0.918	0.637	0.874	0.722
Momeni multiplier1 [7]	11.048	28.920	7.706	25.003	0.760	0.897	0.661	0.875
Momeni multiplier2 [7]	11.474	25.339	8.122	22.105	0.783	0.841	0.694	0.854
Minaeifar mul1 [6]	31.782	13.470	35.221	14.230	0.991	0.647	0.992	0.735
Minaeifar mul3 [6]	28.634	13.255	30.780	14.084	0.996	0.646	0.997	0.732
Ejtahed [9]	33.123	13.882	33.842	14.431	0.983	0.646	0.973	0.742
Ahmadinejad 4 2 [8]	29.058	14.147	23.540	14.711	0.909	0.645	0.935	0.751
Ahmadinejad 5 2 [8]	31.071	14.135	24.829	14.721	0.944	0.645	0.948	0.751

1.5 Multiplier Performance

As observed from the data in table that the proposed multipliers outperform the multipliers referenced form different articles with respect to delay and chip area. We believe this occurs because the lower bits are truncated, and the

multipliers being almost entirely constructed out of approximate counterparts. 'Novel1', is 55% better regarding area and 70% better concerning power consumption 'Novel2' as 'Novel1'. This could be understood from the fact that 'Novel1' has one more bit truncated than 'Novel2' and employs approximate full-adder modules instead of the exact ones in its last stage.

Table 5: Comparison of Multiplier Performance

Multiplier	Area (μm^2)	Delay (ps)	Power (mW)	PDP	FOM1	FOM2
NOVEL1	507.61	1000	0.254	0.254	0.266	0.009
NOVEL2	790.07	750	0.432	0.324	0.337	0.013
Minaeifar mul1 [6]	850.61	1800	0.284	0.511	0.513	0.011
Minaeifar mul3 [6]	870.77	1400	0.269	0.377	0.382	0.011
Momeni multiplier1 [7]	1332.89	1400	0.584	0.818	0.871	0.046
Momeni multiplier2 [7]	1180.5	1600	0.439	0.703	0.766	0.034
Ejtahed [9]	788.52	1600	0.283	0.453	0.457	0.011
Ahmadinejad 4 2 [8]	992.19	1400	0.439	0.615	0.627	0.021

For a more effective evaluation and scrutiny of our multipliers, Figure 7 illustrates the equilibrium between the adjusted Power Delay Product (PDP) and the precision metric, specifically the Normalized Mean Error Distance (NMED) [6].

Two performance indicators (FoM) are utilized for a more comprehensive and fair analysis. Each FoM emphasizes a different factor: FoM1 underscores the relationship between energy consumption and computational reliability, whereas FoM2 offers a detailed assessment of the efficiency of approximate multipliers by examining their power utilization and impact in image analysis [15]. The formulas for FoM1 and FoM2 are presented below:

$$\mathrm{FOM1} = \mathrm{PDP}/(1 - NMED)$$

$$\mathrm{FOM2} = \mathrm{PDP}/(PSNR * MSSIM)$$

The Mean Structural Similarity Index Metric (MSSIM) and the Peak Signal-to-Noise Ratio (PSNR) values for the multiplication operation of the two images, as provided in the table, are utilized in this analysis. The two FOMs are plotted in Fig: 8 where the FOM2 values have been normalized.

CONCLUSION

This study presents, we have designed two innovative multipliers which demonstrate a notable reduction in area and delay while preserving a good level of accuracy. By employing exact and approximate circuits, including 4:2 compressors and various adder circuits, we have effectively enhanced the efficiency of our approximate multipliers. This approach underscores the growing importance of approximate computing in reducing energy consumption and improving performance in digital circuits.

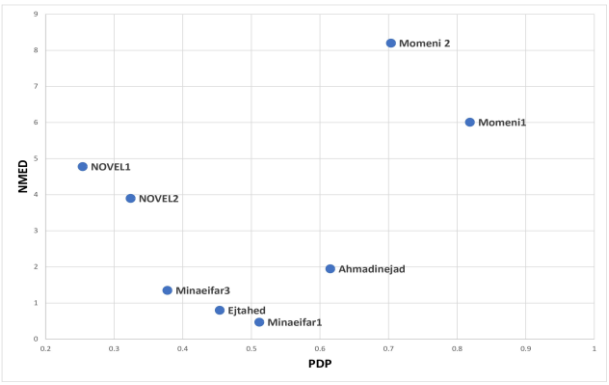


Figure 7: PDP vs Normalized NMED

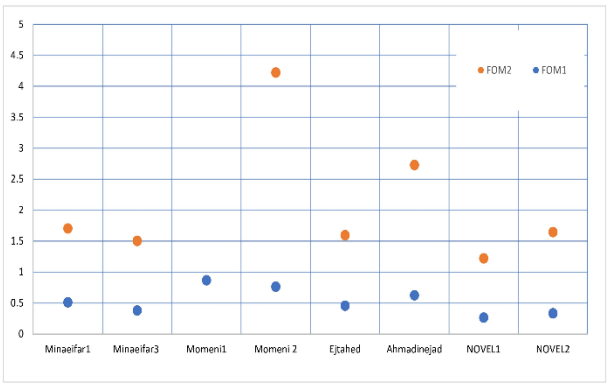


Figure 8: FOM1 and Normalized FOM2

The proposed multipliers, NOVEL1 and NOVEL2, exhibit impressive improvements over existing designs. NOVEL1 achieves a 61.9% reduction in area compared to the Momeni multiplier1 [7] and outperforms Minaeifar mul1 [6] with a 44.4% reduction in delay. Additionally, NOVEL1 also benefits from 10.7% lower power consumption than Minaeifar mul3 [6]. On the other hand, NOVEL2 stands out with the lowest delay, being 46.4% faster than Minaeifar mul1 [6], and offers a compact design with 33.4% less area compared to Ahmadinejad 4:2, providing an excellent balance between speed and power efficiency.

In summary, the design and implementation of our approximate multipliers showcase substantial improvements in both circuit-level performance and error metrics. Through the optimization of approximate compressor designs, we have developed multipliers that minimize area and delay while maintaining an effective balance between power consumption and computational speed. These achievements underscore the promise of approximate computing in enabling more efficient digital systems.

REFERENCES

[1] Q. Xu, T. Mytkowicz, and N. S. Kim, “Approximate Computing: A Survey,” IEEE Design Test, vol. 33, no. 1, pp. 8–22, Feb. 2016, doi: https://doi.org/10.1109/MDAT.2015.2505723.

[2] V. K. Chippa, Srimat Chakradhar, K. Roy, and A. Raghunathan, “Analysis and characterization of in- herent application resilience for approximate computing,” Design Automation Conference, May 2013, doi: https://doi.org/10.1145/2463209.2488873.

[3] Abdoreza Pishvaie, Ghassem Jaberipur, and A. Jahanian, “Improved CMOS (4;2) compressor designs for parallel multipliers,” Computers and Electrical Engineering, vol. 38, no. 6, pp. 1703–1716, Nov. 2012, doi: https://doi.org/10.1016/j.compeleceng.2012.07.015.

[4] D. Baran, Mustafa Aktan, and V. G. Oklobdzija, “Energy efficient implementation of parallel CMOS multipliers with improved compressors,” International Symposium on Low Power Electronics and Design, Aug. 2010, doi: https://doi.org/10.1145/1840845.1840876.

[5] C.-H. . Chang, J. Gu, and M. Zhang, “Ultra Low-Voltage Low-Power CMOS 4:2 and 5-2 Compressors for Fast Arithmetic Circuits,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 10, pp. 1985–1997, Oct. 2004, doi: https://doi.org/10.1109/tcsi.2004.835683.

[6] Atefeh Minaeifar, Ebrahim Abiri, Kourosh Hassanli, Mehrzad Karamimanesh, and F. Ahmadi, “Energy efficient approximate multipliers compatible with error-tolerant application,” Computers and electrical engineering, vol. 114, pp. 109064–109064, Mar. 2024, doi: https://doi.org/10.1016/j.compeleceng.2023.109064.10.1016/j.compelec eng.2023.109064.

[7] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, “Design and Analysis of Approximate Compres- sors for Multiplication,” IEEE Transactions on Computers, vol. 64, no. 4, pp. 984–994, Apr. 2015, doi: https://doi.org/10.1109/tc.2014.230821.

[8] M. Ahmadinejad, M. H. Moaiyeri, and F. Sabetzadeh, “Energy and area efficient imprecise compressors for approximate multiplication at nanoscale,” AEU - International Journal of Electronics and Communications, vol.

110, p. 152859, Oct. 2019, doi: https://doi.org/10.1016/j.aeue.2019.152859.

[9] S. A. H. Ejtahed and S. Timarchi, “Efficient Approximate Multiplier Based on a New 1-Gate Approxi- mate Compressor,” Circuits, Systems, and Signal Processing, vol. 41, no. 5, pp. 2699–2718, Jan. 2022, doi: https://doi.org/10.1007/s00034-021-01902-7.

[10] J. Liang, J. Han, and F. Lombardi, “New Metrics for the Reliability of Approximate and Probabilistic Adders,” vol. 62, no. 9, pp. 1760–1771, Sep. 2013, doi: https://doi.org/10.1109/tc.2012.146.

[11] A. Hor´e and D. Ziou, “Image Quality Metrics: PSNR vs. SSIM,” IEEE Xplore, Aug. 01, 2010. https://ieeexplore.ieee.org/document/5596999 (accessed Dec. 17, 2020).

[12] S. Shirkavand Saleh Abad and M. H. Moaiyeri, “A Hardware- and Accuracy-Efficient Approximate Multiplier with Error Compensation for Neural Network and Image Processing Applications,” Circuits, Systems, and Signal Processing, vol. 41, no. 12, pp. 7057–7076, Jul. 2022, doi: https://doi.org/10.1007/s00034-022-02110-7.

[13] Ferdos Salmanpour, Mohammad Hossein Moaiyeri, and Farnaz Sabetzadeh, “Ultra-Compact Imprecise 4:2 Compressor and Multiplier Circuits for Approximate Computing in Deep Nanoscale,” Circuits Systems and Signal Processing, vol. 40, no. 9, pp. 4633–4650, Mar. 2021, doi: https://doi.org/10.1007/s00034-021-01688-8.

[14] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image Quality Assessment: From Error Visibility to Structural Similarity,” IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, Apr. 2004, doi: https://doi.org/10.1109/tip.2003.819861.

[15] Farnaz Sabetzadeh, Mohammad Hossein Moaiyeri, and M. Ahmadinejad, “A Majority-Based Imprecise Mul- tiplier for Ultra-Efficient Approximate Image Multiplication,” IEEE Transactions on Circuits and Systems I- regular Papers, vol. 66, no. 11, pp. 4200–4208, Nov. 2019, doi: https://doi.org/10.1109/tcsi.2019.2918241.

[16] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, “Dual-Quality 4:2 Compressors for Utilizing in Dynamic Accuracy Configurable Multipliers,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 4, pp. 1352–1361, Apr. 2017, doi: https://doi.org/10.1109/tvlsi.2016.2643003.