PRIM: Hybrid Array-Compressor Multipliers with Carry Disregard and OR-based Approximation

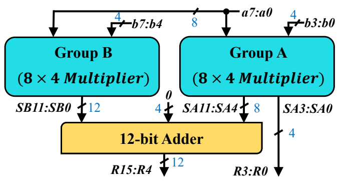
Nima Amirafshar[1], Gulafshan Gulafshan[1], Hadi Shahriar Shahhoseini[2], and Nima TaheriNejad[1,3]
[1]Institute of Computer Engineering (ZITI), Heidelberg University, Germany,
[2]School of Electrical Engineering, Iran University of Science and Technology, Iran,
[3]Institute of Computer Technology (ICT), Technische Universität Wien (TU Wien), Austria
Email: {nima.amirafshar, gulafshan, nima.taherinejad}@ziti.uni-heidelberg.de, shahhoseini@iust.ac.ir

***Abstract*—This paper introduces an efficient new 4:1 compressor that uses carry disregard and OR-based approximation, leading to the development of 13 approximate unsigned multipliers. The proposed multipliers, 8-bit array-comPressor oR-based carry dIsregard Multipliers (PRIM8s) demonstrate significant improvements in area, power, delay, and Power-Delay-Product (PDP) by an average of 29%, 31%, 25%, and 47%, compared to the exact multiplier. In the approximate multiplier literature, with our hardware, we establish new Pareto fronts for most criteria. The effectiveness of the proposed multipliers for noise reduction is demonstrated in an image-processing application using a low-pass Gaussian filter. On average, PRIM8s reduce power consumption and improve speed by 32.36% and 19.01% compared to the exact multiplier, while also enhancing image quality, as indicated by a 0.14% increase in Structural Similarity Index Measure (SSIM).**

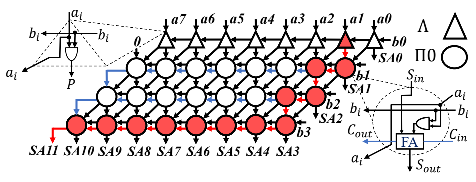
***Index Terms*—Approximate multipliers, OR-based carry disregard compressor, energy efficiency, image processing.**

I. Introduction

In recent years, the inherent error tolerance in applications like machine learning, scientific computing, and signal processing has allowed for the use of approximate arithmetic, particularly in fundamental operations like multiplication [1]–[4]. This trend is further supported by the limitations of human perception in areas such as image processing and multimedia [5]. Approximate multipliers can enhance performance and reduce hardware complexity while maintaining acceptable accuracy levels. Multiplication involves generating Partial Products (PPs), accumulating them, and performing a final addition. PP accumulation is typically the most hardware-intensive stage. Various approximation techniques have been proposed for unsigned multipliers, including logarithmic designs, operand truncation, and tree-based architectures [6], [7]. While tree structures offer speed, they consume more power and area. In contrast, array multipliers typically apply approximation techniques by using simplified Half Adders (HAs) or Full Adders (FAs), and by disregarding carries in specific columns of PPs [1], [8], [9]. In this paper, we introduce a new class of carry disregard unsigned multipliers that combine compressor and array architectures. These designs disregard carries and replace XOR gates with OR gates for PP summation, which simplifies the hardware and naturally balances the error introduced, eliminating the need for a compensator unit to correct errors. We present 13 approximate multipliers



(a) Exact unsigned 8-bit multiplier



(b) Exact 8×4 multiplier and the circuit of Partial Product Unit (Π0)

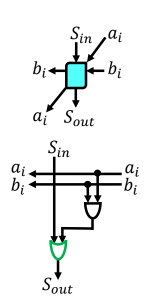
Fig. 1: Exact 8-bit multiplier using two exact 8×4 multipliers.

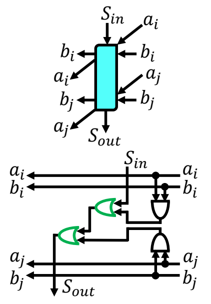
with varying error levels, demonstrating an improved trade-off between accuracy and efficiency.

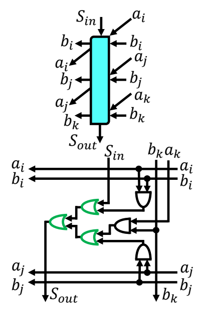
The key contributions in this paper are: (1) The design of a new approximate 4:1 compressor and partial product units, employing carry disregard and OR-based partial product accumulation, (2) Proposing a new methodology for designing approximate hybrid array-compressor multipliers, (3) Developing highly efficient approximate 8-bit unsigned multipliers, with most positioned on the Pareto fronts of hardware criteria, outperforming many recent approximate multipliers in the literature, (4) Analyzing the impact of our designs in an image processing application. The remainder of this paper is structured as follows: Section II reviews the preliminaries. In Section III, we introduce our 8-bit approximate unsigned multipliers. Section IV presents the experimental results and a comprehensive comparison with recent approximate multipliers. Section V explores the application of our designs in image processing, and the paper concludes in Section VI.

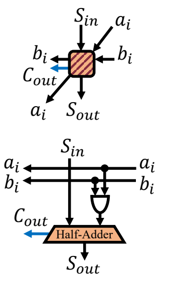
II. Preliminaries

The conventional design of multipliers typically uses an array architecture, where partial product units perform bit-wise multiplication and accumulate PPs. An 8-bit unsigned array multiplier consists of 8 rows of PPs, with each row containing 8 partial product units. While array multipliers feature a uniform and modular structure that reduces power


(a) Π1_1


(b) Π1_2


(c) Π1_3


(d) Π2

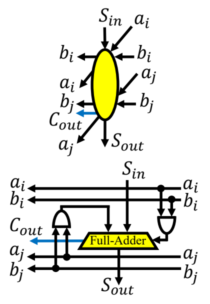
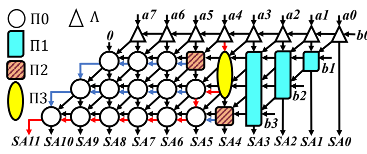
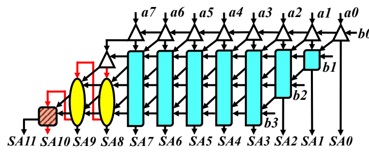

(e) Π3

Fig. 2: Circuits and representations of optimized partial product units.



(a) G_4


(b) G_8

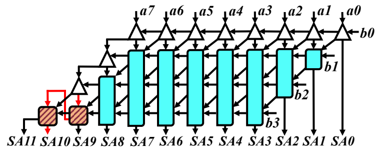

(c) G_9

Fig. 3: Proposed approximate 8 × 4 multipliers.

consumption and area, they suffer from increased critical path delay due to carry propagation. This delay results from the high interdependence among partial product units, requiring each unit to wait for its adjacent unit to generate a carry output. To address this issue, the structure illustrated in Figure 1a, which presents an 8-bit unsigned array multiplier using two exact 8×4 multipliers organized into groups A and B, reduces critical path delay for each group (highlighted in red in Figure 1b) and allows for parallel operation. The final multiplication result is achieved using a 12-bit adder. Each 8×4 multiplier comprises Partial Product Units (Π0s), with each Π0 containing an AND Gate (Λ) for single-bit multiplication and one FA for PPs summation. Although this architecture improves delay without sacrificing accuracy, the dependency between Π0s within each group remains. To address this challenge, we propose several approximate multipliers based on this structure, incorporating effective approximation techniques to significantly reduce hardware complexity while maintaining acceptable accuracy.

III. Proposed Approximate Unsigned Multipliers

In this paper, we propose 8-bit array-comPressor oR-based carry dIsregard Multipliers (PRIM8s) based on the structure illustrated in Figure 1. By disregarding carries from Column 1 up to a specified column in each 8×4 multiplier, the approximate columns can operate in parallel, significantly improving critical path delay. This approach also allows for simpler partial product units, resulting in reduced power consumption and area. Figure 2 shows the optimized partial product units. Our OR-based Carry Disregard Partial Product Units (Π1s) are designed without carry inputs or outputs in columns where carry propagation is disregarded. Π1_1 and Π1_2 (Figures 2a and 2b) use Λs for bit-wise multiplication and OR gates for summing the PPs. Figure 2c presents Π1_3, our approximate 4:1 compressor that accumulates four PPs using three Λs and three OR gates while disregarding all carries. By incorporating OR gates into the Π1s, we reduce hardware complexity and improve the error balance from carry disregard. For example, when Π1_3 computes the sum of four bits, an odd count of 1s results in $S_{out} = 1$ in both OR-based and XOR-based units, while an even count (not zero) yields $S_{out} = 0$ in the XOR-based unit but 1 in the OR-based unit, compensating for the error from ignored carries. Π2 and Π3 (Figures 2d and 2e) are further optimized units that operate without carry inputs, as they are used after the carry disregard columns. Π2 uses a single HA for summing PPs, while Π3 combines two Π0s to compute the sum of three PPs.

We developed all 10 possible configurations for the 8×4 multiplier based on carry disregard, labeled G_x, where x indicates the last column up to which carries are disregarded. In this paper, we focus on approximations applied only to Group A, as it occupies the least significant part of the 8-bit multiplier in Figure 1a, thus introducing minimal error. Figure 3 illustrates several of the proposed approximate 8×4 unsigned multipliers. For Group A, we selected configurations G_4 through G_a (where "a" represents the hexadecimal value 10). G_1 is an exact multiplier, as Column 1 does not generate any carry, while G_2 and G_3 offer minor hardware improvements. Configuration G_4 (Figure 3a) disregards carries up to Column 4, incorporating Π1_1, Π1_2, and an approximate 4:1 compressor (Π1_3) in those columns. The two subsequent columns use one Π3 and two Π2 units, as these units do not require carry inputs. Configurations G_5 through G_7 mirror G_4 but increase the number of Π1_3 compressors to 2, 3, and 4, respectively, with G_7 replacing the last partial product unit in the second row with an Λ since it handles only a single PP, similar to G_8 and G_9 in Figures 3b and 3c. In G_8 and G_9, we further streamline the number of units by employing additional Π3 and Π2 units. Among all proposed configurations, G_a has the simplest structure, disregarding all carries across columns and consisting solely of Π1s. For the 12-bit adder in Figure 1a, we selected a Ripple Carry

TABLE I: Hardware and accuracy results of proposed PRIM8s.

Multipliers	**Area** (μm^2)	**Power** (μW)	**Delay** (nS)	**PDP** (fJ)	**MRED**	**MED**	**NMED**	**NoEB**	**PC** (%)
Exact8 *	301	85.6	0.760	65.1	0	0	0	16	100
PRIM8_41R12	270	75.5	0.622	47.0	**0.0010**	**3.3**	**0.00005**	**13.03**	**68.3**
PRIM8_51R12	257	70.2	0.628	44.1	0.0022	8.4	0.00012	11.91	59.6
PRIM8_61R12	241	64.7	0.613	39.7	0.0041	18.5	0.00028	10.86	53.3
PRIM8_71R12	227	60.5	0.631	38.2	0.0071	38.7	0.00059	9.83	48.7
PRIM8_81R12	210	56.8	0.608	34.5	0.0115	79.2	0.00121	8.82	44.9
PRIM8_91R12	201	54.9	0.608	33.4	0.0153	123.2	0.00189	8.12	43.4
PRIM8_a1R12	196	53.5	0.602	32.2	0.0173	155.2	0.00238	7.67	42.9
PRIM8_51R11	252	68.6	0.628	43.1	0.0028	11.1	0.00017	11.64	53.6
PRIM8_61R10	233	62.3	0.616	38.4	0.0062	29.4	0.00045	10.42	41.1
PRIM8_71R9	216	56.1	0.636	35.7	0.0120	68.8	0.00105	9.30	31.5
PRIM8_81R8	195	50.1	0.610	30.6	0.0210	150.3	0.00231	8.23	24.3
PRIM8_91R7	182	45.5	0.610	27.8	0.0311	263.7	0.00405	7.39	21.1
PRIM8_a1R6	**175**	**41.9**	**0.590**	**24.7**	0.0407	400.7	0.00616	6.74	19.8

* Exact unsigned 8-bit multiplier (using two exact 8×4 multipliers).

Adder (RCA) due to its lower circuit complexity compared to other types of adders. We introduced two classes of PRIM8s, depending on whether an exact or approximate 12-bit RCA is used. The first class, PRIM8_x1R12, employs an exact 12-bit RCA (R12). In this naming scheme, the parameter x represents the type of approximate 8×4 multiplier (G_x) used in Group A, while the digit "1" indicates that Group B uses an exact multiplier (G_1). The second class, PRIM8_x1R$(16-x)$, for $5 \leq x \leq 10$, applies our approximation method to the 12-bit RCA. In this class, carries in the 12-bit adder are disregarded, and OR gates are used for addition from Bit 0 up to a specific bit that corresponds to the last carry-disregard column in Group A. The remaining bits are processed using a $(16-x)$-bit exact RCA. For example, in PRIM8_51R11 ($x = 5$), G_5 is used in Group A, which disregards carries up to Column 5, corresponding to Bit 0 of the 12-bit adder. Therefore, we used an OR gate and disregarded the carry for Bit 0 of the adder, while the remaining 11 bits are processed using an exact 11-bit RCA (R11). In summary, this paper presents 13 approximate 8-bit unsigned multipliers with varying levels of approximation. Our experiments demonstrate that these multipliers provide significant improvements in hardware efficiency compared to exact and recent approximate multipliers found in the literature.

IV. Experiments and Results

A. Experimental Setup & Results

The proposed approximate multipliers were designed in Verilog HDL and synthesized with Cadence Genus v2018, using 45nm NanGate technology, to assess their critical path delay, power consumption, and area. Table I presents the experimental results of PRIM8s. Mean Relative Error Distance (MRED), Mean Error Distance (MED), Normalized Mean Error Distance (NMED), Number of Effective Bits (NoEB), and Probability of Correctness (PC) are accuracy metrics calculated across all possible input configurations (i.e., 65,536 for 8-bit multipliers) [8]. Compared to the exact multiplier (Exact8 in Table I), PRIM8s improved area, power, delay, and Power-Delay-Product (PDP) by an average of 27%, 32%, 19%, and 45%, with an average MRED of 0.0133. For the first class of multipliers (PRIM8_R12), there are average improvements of 24%, 27%, 19%, and 41% in area, power, delay, and PDP, respectively, with an average MRED of 0.0084. The most hardware-efficient multiplier in this class is PRIM8_a1R12, which reduces area, power, delay, and PDP by 35%, 37%, 21%, and 50%, compared to Exact8. In the second class, where approximate 12-bit RCAs are employed, the multipliers show greater hardware efficiency than those in the first class, though with a slight reduction in accuracy, as indicated by an average MRED of 0.019. These designs, compared to Exact8, improved area, power, delay, and PDP by 31%, 37%, 19%, and 49%. Notably, PRIM8_a1R6 is the most efficient multiplier in the second class and among all PRIM8s, delivering substantial gains of 42%, 51%, 22%, and 62% in area, power, delay, and PDP, compared to Exact8.

B. Comparison

For a fair evaluation of the proposed PRIM8s, we compared them against 86 other recently developed approximate 8-bit unsigned multipliers, all synthesized using 45 nm NanGate technology, as reported in the reference papers. These multipliers encompass various architectures, which we have grouped into four main categories: compressor-based, array, logarithmic, and operand truncation. Figure 4 shows the power, delay, area, PDP, and Power-Delay-Area-Product (PDAP) of these multipliers plotted against their MRED values, with the Pareto front highlighting the most efficient designs. Figure 4 illustrates that all PRIM8s, except for PRIM8_91R7, are positioned on the Pareto front in terms of power consumption. For delay, while only PRIM8_41R12 is on the Pareto front, the remaining PRIM8s still demonstrate competitive speed. Although compressor-based multipliers lead the Pareto front in delay, they perform poorly in terms of power, area, PDP, and PDAP. In contrast, for area, PDP, and PDAP (key indicators of overall hardware efficiency) nearly all PRIM8s, except for PRIM8_71R9, are on the Pareto front. Overall, PRIM8s rank among the top designs, achieving a more favorable balance between hardware efficiency and accuracy compared to other approximate designs in the literature.

V. Case Study: Image Processing

A low-pass Gaussian filter smooths images by reducing high-frequency noise while preserving low-frequency details [8], [10]. This is achieved by convolving the image with a 3×3 Gaussian kernel, as defined in Equation (1). Since multiplication is fundamental to the convolution process, the proposed PRIM8s replace exact multipliers to enhance computational efficiency. This case study uses an 8-bit color image (Figure 5a), initially converted to grayscale to retain essential details, as shown in Figure 5b. Gaussian noise with zero mean and 1% variance is then added to the grayscale image, depicted in Figure 5c. After that, the noisy image is processed using the Gaussian filter implemented with exact multiplier and PRIM8s as shown in Figures 5d to 5f.

$$f_G = \frac{1}{1023}\begin{bmatrix} 97 & 121 & 97 \\ 121 & 151 & 121 \\ 97 & 121 & 97 \end{bmatrix} \quad (1)$$

Table II presents key performance metrics, including Maximum Error Distance (Max ED), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM), for filtered images generated by Exact8 and PRIM8s, along with comparisons to the original grayscale image. In Table II,

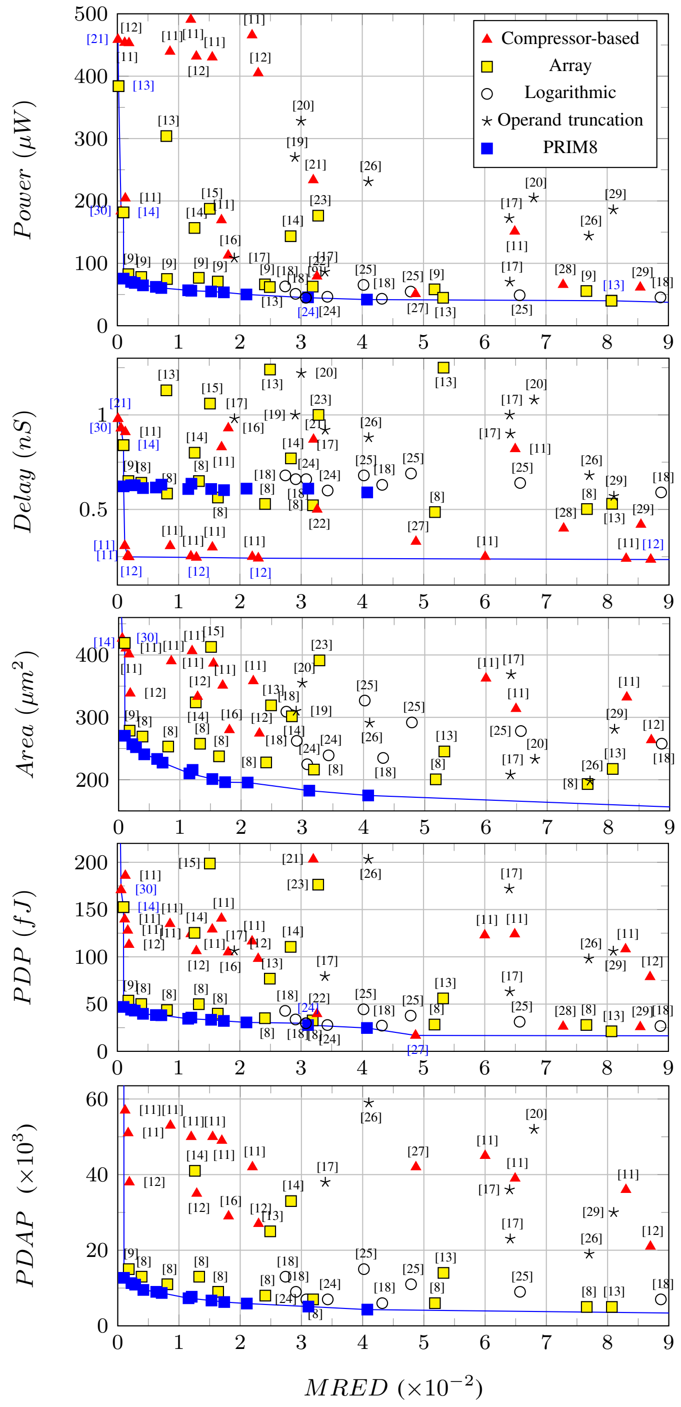


Fig. 4: New Pareto fronts for hardware criteria versus MRED.

SSIM(Ex, Apx) indicates that the structural differences between filtered images using Exact8 and PRIM8s are insignificant, with a maximum of 0.23%. Figures 5e and 5f highlight the best and worst filtered image qualities among PRIM8s, showing minimal visual differences and only a 0.22% structural difference, SSIM(Ex, Apx), thereby justifying the omission of intermediate cases. Table II demonstrates that PRIM8s reduce power consumption and improve speed by an average of 32.36% and 19.01% compared to Exact8 while enhancing image quality, with a 0.14% increase in SSIM(Y, Ex/Apx). This improvement in image quality is due to filtering applied

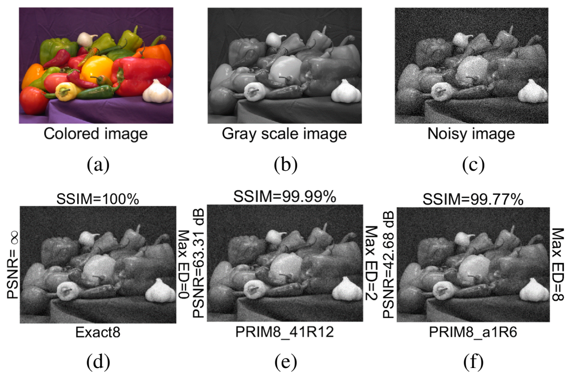


Fig. 5: Image Processing Examples; (a)-(c) before Gaussian filtering, (d)-(e) after filtering.

TABLE II: Performance metrics of Gaussian low pass filter design using exact and approximate multiplier

Multipliers	**Max ED** Ex, Apx	**PSNR (dB)** Ex, Apx	**SSIM (%)** Ex, Apx	**PSNR (dB)** Y, Ex/Apx	**SSIM (%)** Y, Ex/Apx	**Power** (W)	**Delay** (μS)
Exact8	0	∞	100	28.91	64.07	102.03	149.42
PRIM8_41R12	2	63.31	99.99	28.93	64.12	89.06	122.29
PRIM8_51R12	2	60.82	99.97	28.89	64.01	82.81	123.46
PRIM8_61R12	2	57.82	99.95	28.98	64.42	76.32	120.52
PRIM8_71R12	3	55.22	99.93	29.95	64.21	71.36	124.05
PRIM8_81R12	4	53.58	99.92	28.91	64.16	67.00	119.53
PRIM8_91R12	4	53.25	99.91	28.88	63.98	64.76	119.53
PRIM8_a1R12	4	53.18	99.91	28.91	64.21	63.11	118.35
PRIM8_51R11	2	58.83	99.96	28.95	64.29	80.92	123.46
PRIM8_61R10	3	54.49	99.91	28.92	64.33	73.49	121.11
PRIM8_71R9	3	51.07	99.87	28.91	64.05	66.17	125.04
PRIM8_81R8	5	48.06	99.87	28.86	64.11	59.10	119.93
PRIM8_91R7	6	45.74	99.85	28.91	64.34	53.67	119.93
PRIM8_a1R6	8	42.68	99.77	28.82	64.47	49.42	115.99

Ex: Filtered image with exact multiplier, **Apx:** Filtered image with approximate multiplier, **Y:** Grayscale image

to a noisy image (Figure 5c) rather than to the noise-free grayscale image (Figure 5b). Due to the presence of random noise, it is unpredictable whether filtering with exact hardware would perform better than approximate hardware. Notably, the approximation technique can enhance image quality compared to the exact hardware, depending on the noise intensity, type, and the specific approximate multiplier used.

VI. Conclusion

In this paper, we introduced a new methodology for designing approximate hybrid array-compressor multipliers (PRIM8s). Our experiments demonstrate that by appropriately integrating the advantages of both architectures, we can create more efficient multipliers. We also showed that employing two different approximation methods, which introduce errors in opposite directions (positive and negative), not only enables further hardware optimizations but also controls the error level without requiring an additional compensator. Accordingly, we developed a new 4:1 compressor that disregards all carries and uses OR-based approximation for its sum output. Consequently, most of PRIM8s rank among the Pareto front designs compared to other approximate multipliers in the literature. Moreover, the Gaussian filter designed with the proposed multipliers is efficient in terms of power, delay, and noise filtering compared to the exact counterpart. Since our methodology offers over hundreds of configurations for PRIM8s, we plan to use evolutionary algorithms in future work to explore this design space further and identify even more efficient approximate multipliers.

References

[1] S. Shakibhamedan *et al.* Ace-cnn: Approximate carry disregard multipliers for energy-efficient cnn-based image classification. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(5):2280–2293, 2024.

[2] H. J. Damsgaard *et al.* Adaptive approximate computing in edge ai and iot applications: A review. *Journal of Systems Architecture*, 150:103114, 2024.

[3] S. Shakibhamedan *et al.* An analytical approach to enhancing dnn efficiency and accuracy using approximate multiplication. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ ICML 2024)*.

[4] G. Armeniakos *et al.* Hardware approximate techniques for deep neural network accelerators: A survey. *ACM Comput. Surv.*, 55(4), nov 2022.

[5] F. Seiler and N. TaheriNejad. Accelerated image processing through imply-based nocarry approximated adders. *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–14, 2024.

[6] Y. Wu *et al.* A survey on approximate multiplier designs for energy efficiency: From algorithms to circuits. *ACM Trans. Des. Autom. Electron. Syst.*, 29(1), jan 2024.

[7] H. Jiang *et al.* Approximate arithmetic circuits: A survey, characterization, and recent applications. *Proceedings of the IEEE*, 108(12):2108–2135, 2020.

[8] N. Amirafshar *et al.* Carry disregard approximate multipliers. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 70(12):4840–4853, 2023.

[9] N. Amirafshar *et al.* An approximate carry disregard multiplier with improved mean relative error distance and probability of correctness. In *Euromicro Conference on Digital Systems Design 2022 (DSD2022)*, pp. 1–7, 2022.

[10] G. Gulafshan *et al.* Power efficient image processing with tmr tunable hybrid approximate adders. In *2023 IEEE 23rd International Conference on Nanotechnology (NANO)*, pp. 556–561, 2023.

[11] M. S. Ansari *et al.* Low-power approximate multipliers using encoded partial products and approximate compressors. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(3):404–416, 2018.

[12] S. D. S. and N. M. Sk. Low power, high speed approximate multiplier for error resilient applications. *Integration*, 84:37–46, 2022.

[13] V. Mrazek *et al.* Evoapprox8b: library of approximate adders and multipliers for circuit design and benchmarking of approximation methods. In *DATE*, pp. 258–261, 2017.

[14] H. Waris *et al.* Hybrid partial product-based high-performance approximate recursive multipliers. *IEEE Trans. Emerg. Topics Comput.*, 10(1):507–513, 2022.

[15] S. Rehman *et al.* Architectural-space exploration of approximate multipliers. In *IEEE/ACM ICCAD*, pp. 1–8, 2016.

[16] Y. Guo *et al.* Design of power and area efficient lower-part-or approximate multiplier. In *TENCON-IEEE Region 10 Conference*, 2018.

[17] S. Hashemi *et al.* Drum: A dynamic range unbiased multiplier for approximate applications. In *IEEE/ACM ICCAD*, pp. 418–425, 2015.

[18] P. Yin *et al.* Design and analysis of energy-efficient dynamic range approximate logarithmic multipliers for machine learning. *IEEE Transactions on Sustainable Computing*, 6(4):612–625, 2021.

[19] S. Vahdat *et al.* Letam: A low energy truncation-based approximate multiplier. *Computers & Electrical Engineering*, 63:1–17, 2017.

[20] S. Narayanamoorthy *et al.* Energy-efficient approximate multiplication for digital signal processing and classification applications. *IEEE TVLSI*, 23(6):1180–1184, 2015.

[21] S. Venkatachalam and S.-B. Ko. Design of power and area efficient approximate multipliers. *IEEE TVLSI*, 25(5):1782–1786, 2017.

[22] M. Ha and S. Lee. Multipliers with approximate 4–2 compressors and error recovery modules. *IEEE Embedded Systems Letters*, pp. 6–9, 2018.

[23] P. Kulkarni *et al.* Trading accuracy for power with an underdesigned multiplier architecture. In *24th Int. Conf. on VLSI Design*, 2011.

[24] W. Liu *et al.* Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(9):2856–2868, 2018.

[25] M. S. Kim *et al.* Efficient mitchell's approximate log multipliers for convolutional neural networks. *IEEE Transactions on Computers*, 68(5):660–675, 2019.

[26] S. Vahdat *et al.* Tosam: An energy-efficient truncation- and rounding-based scalable approximate multiplier. *IEEE TVLSI*, 27(5), 2019.

[27] P. J. Edaoovr *et al.* Approximate multiplier design using novel dual-stage 4:2 compressors. *IEEE Access*, 8:48337–48351, 2020.

[28] K. Manikantta Reddy *et al.* Design and analysis of multiplier using approximate 4-2 compressor. *AEU - International Journal of Electronics and Communications*, 107:89–97, 2019.

[29] O. Akbari *et al.* Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers. *IEEE TVLSI*, 25(4):1352–1361, 2017.

[30] C.-H. Lin and I.-C. Lin. High accuracy approximate multiplier with error correction. In *IEEE ICCD*, pp. 33–38, 2013.