An Approximate Carry Disregard Multiplier with Improved Mean Relative Error Distance and Probability of Correctness

N. Amirafshar*, A. S. Baroughi*, H. S. Shahhoseini*, and N. TaheriNejad†
* Iran University of Science and Technology, Tehran, Iran
E-mail: {nima_amirafshar, sadighbaroughi_a}@elec.iust.ac.ir, shahhoseini@iust.ac.ir
† TU Wien, Vienna, Austria
E-mail: nima.taherinejad@tuwien.ac.at

2022 25th Euromicro Conference on Digital System Design (DSD) | 978-1-6654-7404-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/DSD57027.2022.00016

***Abstract*—Nowadays, a wide range of applications can tolerate certain computational errors. Hence, approximate computing has become one of the most attractive topics in computer architecture. Reducing accuracy in computations in a premeditated and appropriate manner reduces architectural complexities, and as a result, performance, power consumption, and area can improve significantly. This paper proposes a novel approximate multiplier design. The proposed design has been implemented using 45 nm CMOS technology and has been extensively evaluated. Compared to existing approximate architectures, the proposed approximate multiplier has higher accuracy. It also achieves better results in critical path delay, power consumption, and area up to 47.54%, 75.24%, and 92.49%, respectively. Compared to the precise multipliers, our evaluations show that the critical path delay, power consumption, and area have been improved by 39%, 18%, and 6%, respectively.**

***Index Terms*—Approximate computing, multiplier, high performance, power efficient, area efficient.**

I. Introduction

Nowadays, there is a significant increase in the volume of data and operations in various applications, leading to reduced performance and increased energy consumption due to hardware limitations. Therefore, it is necessary to improve performance, power consumption, and energy efficiency using novel paradigms. One of the efficient methods in this regard is approximate computing. Approximate computing refers to calculations that compromise exact (fully correct) results to gain efficiency in speed, area, and power or energy consumption. Many different applications are inherently error-tolerant. E.g., in many applications, due to human perceptual limitations, the decline in the quality of results is not detectable. Hence, approximate computing can be used in image processing [1]–[3], multimedia [4], [5], machine learning [6], [7], and scientific computing applications [8], [9].

Multiplication is one of the most common arithmetic operations frequently used in various applications. Due to the critical path delay and significant power consumption of exact conventional multipliers, approximate multipliers have gained currency and have improved a sundry of approximable end-applications. Nowadays, various applications such as IoT, Neural Networks, Image Processing, and Deep Learning have turned to low-power processors to improve performance and reduce area, memory resources, and power consumption. One of the key methods proposed is using 8-bit or even smaller arithmetic units [10]–[13]. Given the crucial role that 8-bit multipliers play in such schemes, we focus this study on the design of 8-bit multipliers.

This paper proposes an approximate 8-bit multiplier based on carry disregard, significantly reducing critical path delay and power consumption compared to existing precise and approximate multipliers. Applications have different error tolerance levels, and non-compliance leads to unacceptable results. Therefore, the error level is one of the essential criteria we considered in our design, and the proposed multiplier has the highest accuracy among previous approximate multipliers. The main contributions of this paper are as follows:

- Design and develop an approximate multiplier based on an array architecture that takes advantage of carry disregard in calculating the sum of partial products.
- Evaluation of the proposed design compared to conventional precise multipliers and existing approximate multipliers in terms of accuracy, critical path delay, power consumption, and area. As we show, the proposed approximate multiplier strikes the highest accuracy.

The rest of this paper is organized as follows. Section II briefly reviews related works and analyzes the conventional precise multiplier structure and challenges. We propose our novel approximate multiplier in Section III. Section IV describes the experimental results and evaluations, and the article is concluded in section V.

II. Related Work

Approximate multipliers are suitable for energy-efficient computing in applications with inherent tolerance to inaccuracy. Many applications perform extensive computation on an enormous amount of approximable data. Approximation on such applications would be considered efficient. Since adders and multipliers are the fundamental computational units, approximate adders and multipliers have been explored lately [14]–[16]. Approximate adders are mostly carry-disregard at some stages or use carry-prediction scheme-based [16]–[18].

2771-2508/22/$31.00 ©2022 IEEE
DOI 10.1109/DSD57027.2022.00016

Recently, approximate circuits, e.g., adders and multipliers, are often explored to assure area, latency, and energy efficiency [19]–[21]. [15] describes the designs of approximate array multipliers bypassing parts of an accurate array multiplier. These bypasses are followed by different input and output assignments within the multiplier, exploring ways of approximating an accurate array multiplier. [17] proposed an approach to introduce errors into the truth table of an accurate adder and design Approximate Adder (AA). Then a complex circuit, namely an approximate Dadda multiplier (ADM), is designed using AAs. Authors conclude that their AAs and ADMs have a trade-off between area, power, delay, and accuracy. [22] proposed approximate array multiplier designs in which partial products of the 8×8 array multiplier are divided into Group A and Group B, with four partial products. Partial product groups are simultaneously generated and then accumulated with approximate full adders. Dividing the partial products into separate groups aims to shorten critical path delay.

The main three multiplier stages are partial product generation, partial product accumulation, and final addition. AND gates usually generate partial products. On the other hand, the carry-save adder (CSA) array, Wallace, and Dadda trees are common structures for accumulating partial products. Therefore, the general method for designing an approximate multiplier is to use an approximation in its three main stages: the approximations in the partial products generation, approximations in the partial products tree, and finally, approximations in adders or adders compressors [23]. Approximate Wallace and Dadda tree structures have a minor delay, which can be further reduced by using compressors [24]. In contrast, a conventional array multiplier has a smaller area and can be reconfigured due to its modular and homogeneous structure.

The CSA array is the conventional structure used for a precise multiplier in the form of partial products, which are then summed together in a specific way. Eventually, the final result is obtained. If an exact 8-bit multiplier is considered whose operands are a and b, then a general structure of this multiplier could be according to Figure 1. Except for the first row, which contains only AND gates, the other rows of the multiplier are arrays of Partial Product Units (PPU). Figure 2 shows that each PPU contains an AND gate and is used for single-bit multiplication of a_i and b_i. It also has a Full Adder that computes S_{out} and C_{out} outputs using the three inputs: S_{in}, C_{in}, and AND gate output.

The main challenges that a multiplier faces are critical path delays, power consumption, and area. Consider a precise n-bit multiplier with an array architecture; In general, the multiplier cells are divided into two groups of AND gates and PPUs, the number of each of which is n and n^2–n, respectively. Cells located in the critical path of this multiplier include AND gates and PPUs. For simplicity in calculating the critical path delay, the AND gate and OR gate delays are assumed to be equal to a 1 unit delay, and the XOR gate is considered to have a 2 unit delay. Hence, the PPU delay is equal to 5 unit delays. As a result, the critical path delay equals $15n - 19$. For example, in an 8-bit multiplier, the total numbers of AND gates and PPUs are 8 and 56, respectively. So their numbers in the critical path are 1 and 20. Figure 1 shows the critical path in red. The critical path delay in this example is equal to 101 unit delays.

The main factor in the critical path delay is the significant dependence among PPUs. On the other hand, since PPUs are the majority of multiplier cells, they significantly impact power consumption and area. As a result, if the number and dependence between PPUs can be somehow reduced, better results can be achieved in terms of critical path delay, power consumption, and area. In this paper, we aim to achieve low error because the degree of accuracy determines the quality of output in various applications.

III. Proposed Scheme

This section deals with the main idea of the proposed scheme. Our proposed approximate 8-bit multiplier is presented, which significantly improves critical path delay, power consumption, and area in contrast to the conventional multiplier.

A. Disregarding the Carry in the Computations

The main factor in the critical path delay is the dependence among the PPUs and the overall number of PPUs in the critical path. Therefore, reducing the overall number of PPUs and their dependencies are necessary to improve the delay. Each partial product is independent of the other and can be added in any order. Thus, it is possible to divide the partial products into two distinct groups; In such a case, they can work parallel. In each group, the partial products are computed and added together. Finally, the result of the two groups is added together by a carry look-ahead adder, and the final result is obtained.

Despite the shortening of the critical path due to the parallel performance of the groups, the dependence between PPUs in each group continues to cause delays. The core and emphasis of our proposed method are to reduce the number of PPUs and shorten the critical path in each of the two groups. As a result, the overall delay is reduced. Hence, to eliminate these dependencies, approximate computing can be exploited. The main factor of delay is the dependence of each PPU on the carry entered into it, which another PPU calculates in the previous column. Therefore, by disregarding the carry in the computations, PPU columns can be independent.

B. Approximate 8-bit Multiplier

In different applications, the error tolerance degree is not the same, and they are different from each other. One of the essential criteria in approximate computations is the degree of accuracy since it determines the quality of the output. Consequently, the error in the proposed design should be as small as possible so that it can be used in general in various applications. Therefore, in this paper, the main goal is to achieve low error and a more appropriate balance with critical path delay, power consumption, and area.

Figure 3 shows the proposed approximate 8-bit multiplier architecture. This multiplier has two groups, A and B, with

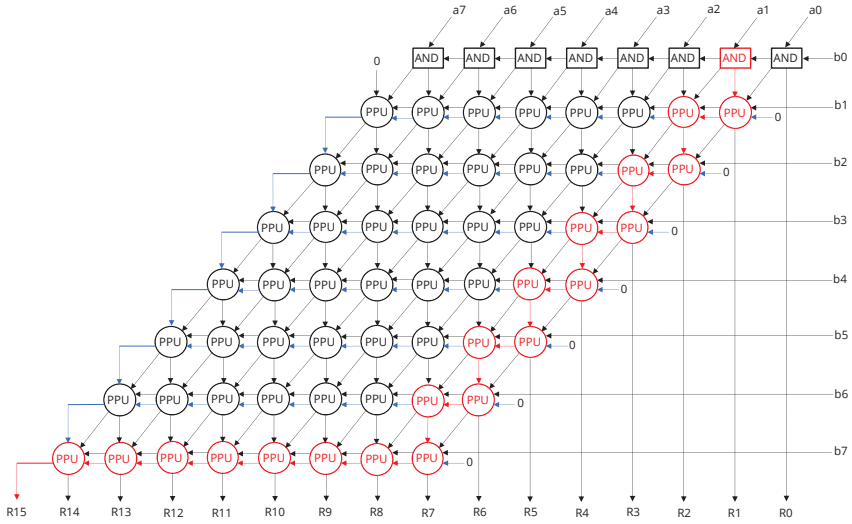

Fig. 1: Conventional 8-bit precise multiplier architecture

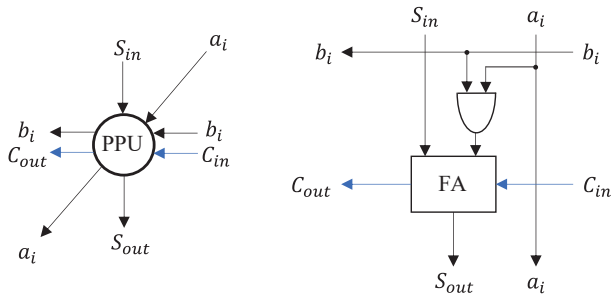

Fig. 2: Logic circuit and symbol of PPU

four partial products. In Group B, all PPUs compute the carry and transfer it to the PPU in the next column. I.e., no carries are disregarded in this group, and computations are performed accurately. In Group A, the carry is disregarded in all partial product units located in columns two, three, and four. In other words, these units do not have the carry input and output and are used as the Carry Disregarding Partial Product Unit (CDPPU). Carry disregarding makes columns two, three, four, and five independent of each other. The most significant advantage of this is their parallel and simultaneous operation, which leads to a shorter critical path and reduced delay.

Figure 4 shows the circuit of the CDPPU, Partial Product Unit with Half-adder (PPUH), and Partial Product Unit with Full-adder (PPUF) units. The CDPPU consists of only one AND gate for single-bit multiplication and one XOR gate for calculating the sum of the S_{in} input with the AND gate output. PPUH works like a conventional PPU, and it has no carry input. It also uses a Half-Adder to calculate the sum. PPUH, unlike CDPPU, calculates the carry output. PPUF is a combination of two conventional PPUs. This unit has two AND gates to perform two single-bit multiplications. It also has a Full Adder that computes S_{out} and C_{out} outputs using three S_{in} and two AND gate outputs. PPUF is used in column five of Group A.

Using PPUF instead of two partial product units reduces the carry output number to one output. As a result, the first partial product unit in column six of Group A has no carry input and is independent. In Group A, two PPUHs are used, and the remaining partial product units are conventional PPUs.

Figure 3 shows the critical path in red; This path starts from the fifth column of Group A and continues until the end of its computations. Due to the parallel performance of the two groups, with the completion of Group A computations, Group B proceeds simultaneously to the end of its seventh column. Therefore, the critical path continues from the eighth column to the last column of Group B. Finally, the Carry Look-ahead Adder (CLA) determines the multiplier output by calculating the sum of the results of Groups A and B; The critical path also ends with the end of the CLA calculations. Thus, the proposed approximate multiplier critical path has an AND gate, a PPUF, eleven PPUs, and a CLA. Considering the AND gate and OR gate delays are equal to a 1 unit delay, the XOR gate is considered to have a 2 unit delay. Therefore the delay of the PPUF, PPU, and CLA will be 5-, 5- and 6-unit delay, respectively. Therefore, the critical path delay in the proposed multiplier will be a 67 unit delay, which is significantly reduced compared to the conventional multiplier. On the other hand, disregarding the carry and using CDPPU, PPUH, and PPUF units instead of the conventional PPU has reduced power consumption and area due to their simpler circuit. Dividing partial products into two groups also removes a row of PPUs, another reason for improving power consumption and area.

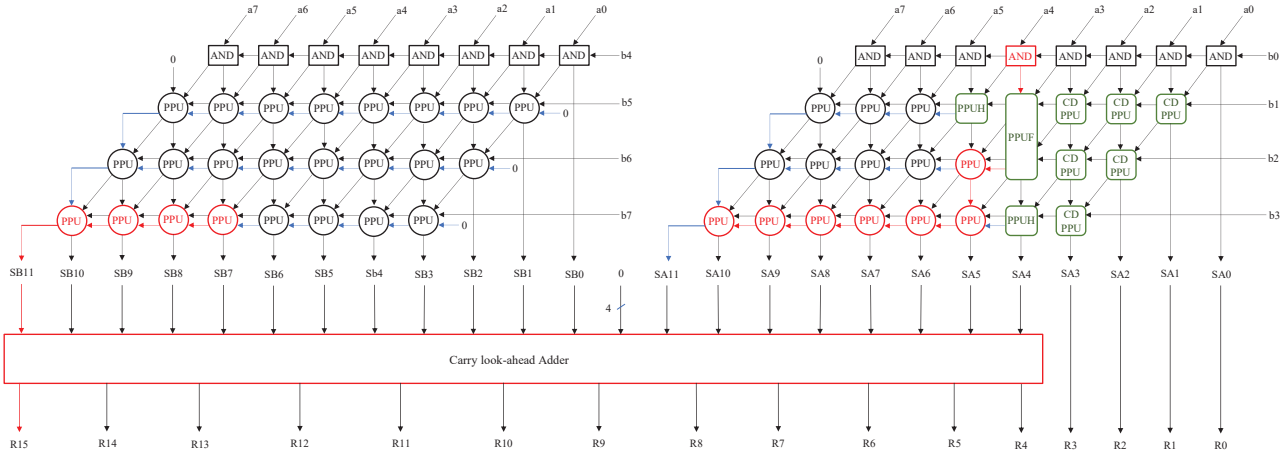

Fig. 3: Proposed approximate 8-bit multiplier architecture

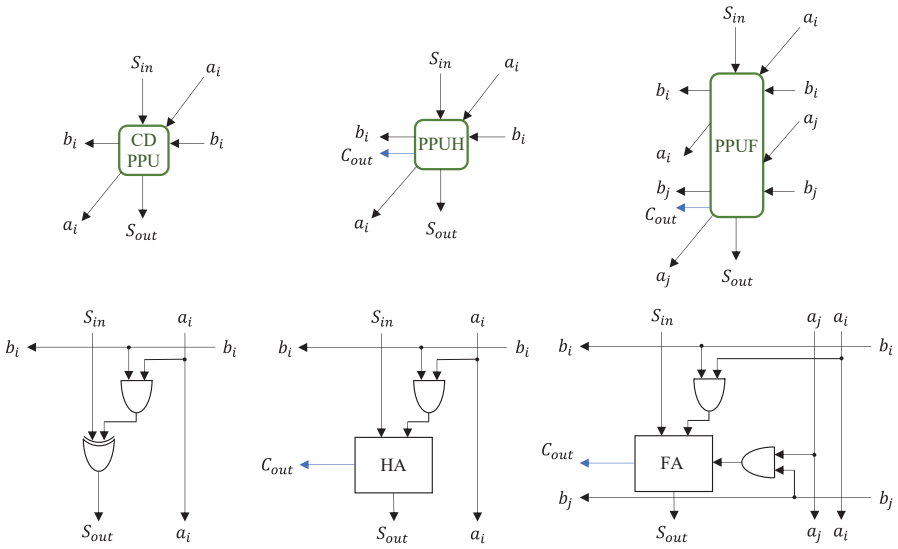

Fig. 4: Logic circuits and symbols of CDPPU, PPUH, and PPUF

IV. Experimental Results and Performance Evaluations

A. Hardware Efficiency Criteria

Critical path delay, power consumption, and area are the main criteria for evaluating hardware efficiency. We also consider Power Delay Product (PDP) and Power Area Delay Product (PADP) as additional criteria that combine the above single criteria to provide a more holistic assessment of various designs.

B. Accuracy Criteria

Accuracy criteria indicate the degree of accuracy and consequently the quality of the output results, which are determined based on the difference between the approximate and the corresponding exact results. The two criteria used in this paper are:

1) Probability of Correctness (PC): The PC refers to the ratio of the number of correct outputs over the total number of possible outputs. That is,

$$PC = \frac{\#OUT_c}{\#OUT_t}, \tag{1}$$

where $\#OUT_c$ is the number of correct outputs, and $\#OUT_t$ is the total number of possible outputs.

2) Mean Relative Error Distance (MRED): The MRED refers to the average of the difference between the exact multiplier output and the proposed approximate output divided by the corresponding exact output for all possible input combinations.

$$MRED = \frac{1}{2^{2N}} \sum_{k=1}^{2^{2N}} \frac{|OUT_{exact,k} - OUT_{approx,k}|}{OUT_{exact,k}} \tag{2}$$

We used both of these accuracy criteria for the proposed approximate 8-bit multiplier and calculated them for all possible input combinations (i.e., $2^{2*8} = 64K$). The PC indicates the number of correct results, whereas the MRED determines the amount of error.

C. Experiment Setups

We described the proposed approximate multiplier using Verilog HDL and verified it using the ISE Design Suite-Xilinx. Then, using Genus Synthesis Solution, we synthesized the proposed scheme with 45-nm NanGate technology and used Cadence to analyze the critical path delay, power consumption, and area. For error analyses, we used Python and determined MRED and PC for all possible input combinations.

D. Experimental Results

Table I reports the area, power consumption, delay, PDP, PADP, MRED, and PC of the proposed design and the base (accurate) design. Table I reports these numbers for other designs in the literature and how these numbers were obtained. Given the similarity and relevance of Ax1 [22] and Ax2 [22], we repeated the simulations for these designs. The key observations in the obtained results are the extremely low MRED of 0.0018 and high PC of 67.58%.

In this paper, One of the main goals is to achieve low error because the degree of accuracy determines the quality of output in various applications and is one of the crucial criteria in approximate calculations. Our understanding is that the error rate depends on two main factors, the number of carries that each column generates from the partial products, and the other is where the carries are generated. If we disregard carries of columns that have less probability of generating carries and are located among less significant bits, the output will have a smaller error. Hence, we propose to disregard the carry produced in the first four columns of Group A.

As Table I shows, the PC of the proposed scheme is 67.58%, which means our design has the correct output in most cases. The MRED of 0.0018 indicates that when the output results are not correct and have errors, their error is significantly small compared to the corresponding exact results. Regarding hardware efficiency criteria, critical path delay, power consumption, and area of the proposed design are 0.64 ns, 81.20 μW, and 278.8 μm^2, respectively. PDP is the product of power consumption and delay, indicating energy efficiency, which is 52.21 $(\mu W * ns)$ for the proposed design. PADP is the product of PDP and area, used to evaluate energy efficiency and area together. The PADP of the proposed design is 14556 $(\mu W * \mu m^2 * ns)$.

E. Comparison

Table I shows that the proposed approximate multiplier has the best PC among the designs reported and the best MRED compared to existing approximate architectures. PC of the proposed design is increased by an average of 58.8%, whereas MRED is improved by an average of 95.3%. Therefore, we see that the probability of correct output is significantly higher for the proposed approximate multiplier, and in cases where an error occurs at the output, the error is significantly smaller in value.

The proposed solution has a 38.5% improvement in critical path delay, 18% in power consumption, and 6% in the area compared to the exact multiplier with the same architecture. The improvement trend is present in PDP and PADP changes as well, decreasing by 49.4% and 52.4%, respectively. Compared to the exact multiplier, this significant reduction is due mainly to the proposed multiplier's high performance and low power consumption.

Regarding the critical path delay, power consumption, and PDP, Akbari et al. [24] (their Design 2) obtained the best result, followed by Edavoor et al. [25] and Momeni et al. [28] (Design 2). Akbari et al. [24] (Design 2) is better in delay, power consumption, and PDP 56.3%, 57.5%, and 81.4%, respectively, than the proposed multiplier. On the other hand, the proposed method is 99.4% and 66.63% better in MRED and PC, respectively, and 60.9% in the area. Compared to the proposed multiplier, Edavoor et al. [25] is better in delay, power consumption, and PDP, 48.4%, 37.2%, and 67.7%, respectively. However, the proposed design is 96.3% and 55.71% better in MRED and PC, respectively, and 88.7% in the area. As for Momeni et al. [28] (Design 2), compared to the proposed multiplier, their design is 42.1%, 26.5%, and 57.7% better in delay, power consumption, and PDP, whereas the proposed method is 99.9% and 66.82% better in MRED and PC, respectively, and 91% in the area. These three architectures achieved better results in delay, power consumption, and PDP due to a significant reduction in the accuracy of the multiplier. They are among the architectures with the highest error. In addition, all three of these architectures have a much larger area than the here proposed multiplier. Indeed, they are among the architectures with the largest area.

On the other hand, DRUM (3) [29], DSM (3) [30], and TOSAM (0,2) [21] architectures have the best area, in that order. At the same time, they are among the architectures with very low accuracy, and most of them, in terms of delay, power consumption, and PDP, have a much worse result than the proposed scheme. DRUM (3) [29], DSM (3) [30], and TOSAM (0,2) [21] in area are 48.7%, 34.7%, and 32.2% better than the proposed design, respectively. Nevertheless, the proposed multiplier in MRED is 98.6%, 98.7%, and 98.2%, better, which is a significant advantage for our design. Moreover, the proposed design is better in delay, power consumption, and PDP compared to DRUM (3) [29] by 8.6%, 21.9%, and 28.3% and better than DSM (3) [30] by 20%, 36.5%, and 49%. Whereas TOSAM (0,2) [21] is 9.3% better in delay, the proposed design is 32.3% and 25% better in power consumption and PDP.

In addition, PADP, the product of PDP and area, can be used to evaluate energy efficiency and area for all existing architectures simultaneously. The proposed approximate multiplier is, on average, 61.2% better than all other existing architectures in terms of PADP. The only exceptions are Akbari et al. [24] (Design 2), DRUM (3) [29], and TOSAM (0,2) [21],

TABLE I: Area, power consumption, delay, PDP, PADP, MRED, and PC of different unsigned 8-bit multipliers

Method	Designed by	Simulated by	Hardware efficiency criteria					Accuracy criteria	
			Area(μm^2)	Power(μW)	Delay(ns)	PDP	PADP	MRED	PC(%)
Edavoor et al.	[25]	[25]	2468	51.01	0.33	16.83	41536	0.0487	11.87
Ax1	[22]	Repeated by us	284.6	83.88	0.78	65.76	18715	0.0759	2.68
Ax2	[22]	Repeated by us	266.5	77.30	0.77	59.91	15966	0.1981	0.28
TOSAM (0,2)	[21]	[21]	186	120	0.58	69.60	12945	0.1010	NR*
TOSAM (1,5)	[21]	[21]	291	231	0.88	203.3	59160	0.0410	NR*
Ha and Lee	[26]	[25]	3624	79.24	0.50	39.62	143583	0.0326	24.20
Akbari et al. (Design 2)	[24]	[25]	713	**34.51**	**0.28**	**9.66**	**6887**	0.3135	0.95
Akbari et al. (Design 4)	[24]	[25]	3070	61.47	0.42	25.82	79267	0.0854	16.86
DQ4:2C4	[24]	[21]	281	186	0.57	106.02	29791	0.0810	NR*
Gorantla et al.	[27]	[25]	3715	80.19	0.49	39.29	145962	1.2000	13.06
Momeni et al. (Design 2)	[28]	[25]	3092	59.69	0.37	22.08	68271	4.2843	0.76
DRUM (3)	[29]	[21]	**143**	104	0.70	72.8	10410	0.1260	NR*
DRUM (4)	[29]	[21]	208	172	1	172	35776	0.0640	NR*
DSM (3)	[30]	[21]	182	128	0.80	102.4	18637	0.1440	NR*
DSM (5)	[30]	[21]	355	328	1.22	400.2	142071	0.0300	NR*
This work - Accurate			296.1	99.04	1.04	103.2	30557	0	100
This work - Proposed			278.8	81.20	0.64	52.21	14556	**0.0018**	**67.58**

* Not Reported.

all of which are among the architectures with the lowest accuracy. Akbari et al. [24] (Design 2), DRUM (3) [29], and TOSAM (0,2) [21] are 52.6%, 28.4%, and 11% better than the proposed design in PADP, respectively. However, the proposed multiplier is 99.4%, 98.5%, and 98.2%, better than those in MRED.

The proposed scheme shows a significant improvement in power consumption, critical path delay, and PDP compared to DSM (3) [30], that is, 75.24%, 47.54%, and 86.95%, respectively. It also has substantial advantages with regard to area and PADP compared to Gorantla et al. [27], i.e., 92.49% and 90.03%. On the other hand, PC has the largest improvement (67.3%) compared to Ax1 [22], and MRED has the most significant improvement (99.96%) compared to Momeni et al. [28] (Design 2). In conclusion, the proposed design has the best accuracy among all existing architectures.

V. Conclusion

This paper proposes a novel multiplier using approximate computations based on carry disregard. Disregarding carries shortens the critical path and reduces hardware complexity, thereby improving the critical path delay, power consumption, and area, which, compared to the accurate multiplier, are improved by 39%, 18%, and 6%, respectively. Unlike many approximate multipliers, this paper aims at reducing the accuracy optimally (i.e., minimally) while achieving better or acceptably good delay, power consumption, and area. Consequently, the proposed multiplier achieves the highest PC and the best MRED compared to the existing approximate architectures in the literature and improves them by an average of 58.8% and 95.3%, respectively. In addition, it has been able to achieve acceptably good delay, power consumption, and area.

The proposed design of this paper provides a basis and a starting point for future research. We believe that this method can be generalized to larger multipliers such as 16-, 32-, and 64-bit. Like the proposed 8-bit multiplier, we expect them to achieve high accuracy and better hardware efficiency, which is in our future works. Also, having many designs using compressor-based multi-operand adders in the literature gives us the idea of combining our proposed design with such methods. Since the two methods are compatible, we expect to achieve further improvement by doing so. This is an idea that we plan to examine in the future.

We also intend to extend carry disregard in the proposed scheme for the various existing scenarios and analyze the results obtained for accuracy, delay, power consumption, and area. Also, the ability to dynamically adjust the accuracy in the proposed design and use it in different applications with different error tolerances is another extension of this work, which we intend to pursue in the future.

References

[1] C. Ossimitz and N. TaheriNejad, "A fast line segment detector using approximate computing," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2021, pp. 1–5.

[2] S. E. Fatemieh, M. R. Reshadinezhad, and N. TaheriNejad, "Approximate in-memory computing using memristive imply logic and its application to image processing," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022, pp. 1–5.

[3] U. Lotrič, R. Pilipović, and P. Bulić, "A hybrid radix-4 and approximate logarithmic multiplier for energy efficient image processing," *Electronics*, vol. 10, no. 10, 2021.

[4] B. Garg and Y. Bisht, "A novel high performance reverse carry propagate adder for energy efficient multimedia applications," in *2019 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)*, 2019, pp. 296–299.

[5] F. Tu, S. Yin, P. Ouyang, L. Liu, and S. Wei, "Reconfigurable architecture for neural approximation in multimedia computing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 3, pp. 892–906, 2019.

[6] P. Yin, C. Wang, H. Waris, W. Liu, Y. Han, and F. Lombardi, "Design and analysis of energy-efficient dynamic range approximate logarithmic multipliers for machine learning," *IEEE Transactions on Sustainable Computing*, vol. 6, no. 4, pp. 612–625, 2021.

[7] X. Hu, T. Chen, H. Huang, Z. Liu, X. Li, and X. Xiong, "Efficient field-programmable gate array-based reconfigurable accelerator for deep convolution neural network," *Electronics Letters*, vol. 57, no. 6, pp. 238–240, 2021.

[8] A. Pothen, S. M. Ferdous, and F. Manne, "Approximation algorithms in combinatorial scientific computing," *Acta Numerica*, vol. 28, p. 541–633, 2019.

[9] B. Grigorian and G. Reinman, "Accelerating divergent applications on SIMD architectures using neural networks," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 12, no. 1, pp. 1–23, 2015.

[10] A. Garofalo, G. Tagliavini, F. Conti, L. Benini, and D. Rossi, "XpulpNN: Enabling energy efficient and flexible inference of quantized neural networks on RISC-V based IoT end nodes," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1489–1505, 2021.

[11] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *Journal of Machine Learning Research*, vol. 18, no. 187, pp. 1–30, 2018. [Online]. Available: http://jmlr.org/papers/v18/16-456.html

[12] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[13] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8604–8612.

[14] J. Lee, H. Seo, H. Seok, and Y. Kim, "A novel approximate adder design using error reduced carry prediction and constant truncation," *IEEE Access*, vol. 9, pp. 119 939–119 953, 2021.

[15] P. Balasubramanian, R. Nayar, and D. Maskell, "Approximate Array Multipliers," *Electronics*, vol. 10, p. 630, 2021.

[16] W. Choi, M. Shim, H. Seok, and Y. Kim, "DCPA: Approximate adder design exploiting dual carry prediction," *IEICE Electronics Express*, vol. 18, no. 23, pp. 20 210 431–20 210 431, 2021.

[17] G. Anusha and P. Deepa, "Design of approximate adders and multipliers for error tolerant image processing," *Microprocessors and Microsystems*, vol. 72, p. 102940, 2020.

[18] H. Seok, H. Seo, J. Lee, and Y. Kim, "COREA: Delay- and energy-efficient approximate adder using effective carry speculation," *Electronics*, vol. 10, no. 18, 2021.

[19] S. Ullah, S. Rehman, B. S. Prabakaran, F. Kriebel, M. A. Hanif, M. Shafique, and A. Kumar, "Area-optimized low-latency approximate multipliers for FPGA-based hardware accelerators," in *Proceedings of the 55th Annual Design Automation Conference*, ser. DAC '18. New York, NY, USA: Association for Computing Machinery, 2018.

[20] R. Nayar, P. Balasubramanian, and D. L. Maskell, "Hardware optimized approximate adder with normal error distribution," in *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. Los Alamitos, CA, USA: IEEE Computer Society, jul 2020, pp. 84–89.

[21] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "TOSAM: An energy-efficient truncation- and rounding-based scalable approximate multiplier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 5, pp. 1161–1173, 2019.

[22] H. Waris, C. Wang, W. Liu, and F. Lombardi, "AxSA: On the design of high-performance and power-efficient approximate systolic arrays for matrix multiplication," *Journal of Signal Processing Systems*, vol. 93, no. 6, pp. 605–615, 2021.

[23] H. Jiang, F. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proceedings of the IEEE*, vol. PP, pp. 1–28, 08 2020.

[24] 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, 2017.

[25] P. J. Edavoor, S. Raveendran, and A. D. Rahulkar, "Approximate multiplier design using novel dual-stage 4:2 compressors," *IEEE Access*, vol. 8, pp. 48 337–48 351, 2020.

[26] M. Ha and S. Lee, "Multipliers with approximate 4–2 compressors and error recovery modules," *IEEE Embedded Systems Letters*, vol. 10, no. 1, pp. 6–9, 2018.

[27] A. Gorantla and D. P, "Design of approximate compressors for multiplication," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, april 2017.

[28] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 984–994, 2015.

[29] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 418–425.

[30] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 6, pp. 1180–1184, 2015.