Energy-Efficient Approximate Squaring Unit

Mahmoud Masadeh[1], Alain Aoun[2], and Sofiène Tahar[2]
mahmoud.s@yu.edu.jo, a_alain@ece.concordia.ca, tahar@ece.concordia.ca
[1]Computer Engineering Department, Yarmouk University, Irbid, Jordan
[2] Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada

***Abstract*—The squaring circuit is an essential computational element of Digital Signal Processing (DSP) designs that directly affect their area, speed and power consumption. Various DSP applications have noisy and redundant input data. Thus, implementing an approximate squaring function will cause minor quality degradation with a significant reduction in hardware costs. In this paper, we design various energy-efficient array-based approximate squaring units. The proposed designs are 8-bit, and have a reduced area, power, and delay compared to the exact array squarer, with an average of 14%, 6%, and 12%, respectively**

***Keywords*-Approximate Computing, Squaring Unit, Energy Efficiency**

I. Introduction

A large number of battery-powered Internet of Things (IoT) sensors perform data acquisition, which provides noisy input data for real-time digital signal processing (DSP) applications. Such battery-powered sensors should have a reduced footprint, small delay, and reduced power consumption where a degraded accuracy of the final results is acceptable [1]. DSP applications efficiently manipulate noisy inputs. Thus, exact results could be sacrificed for fast calculations with reduced power consumption.

Arithmetic modules define the performance of computing systems where the *squaring function* is a widely-used arithmetic operation in multimedia and DSP applications [2]. The parameters of the DSP module depend on the number and performance of various adders, multipliers, and squaring designs used. The squaring function is used extensively in various applications [3], e.g., object tracking, adaptive filtering, pattern recognition, and motion estimation.

DSP units with a parallel architecture are appropriate for high-performance applications. However, they have a large design area and power consumption. On the other hand, DSP units with a recursive architecture are suitable for small and low-power IoT nodes, which often have limited memory storage and operand size. This paper explains how an array-based exact squaring function can be approximated systematically. Then, we use a set of existing approximate full adders (FAs) to design an *array-based approximate squaring units*.

The rest of this article is organized as follows. Section II explains the design of an exact array multiplier and its simplification into exact array squarer. Section III introduces related work on designing approximate squaring circuits. Section IV explains the proposed squaring designs and discusses their accuracy and resource usage. The performance of the proposed squarers is analyzed with a real-time application in Section V and Section VI concludes the paper.

II. Accurate Array Multiplier & Squaring Circuit

This section describes an 8-bit exact array multiplier and an 8-bit unsigned exact squaring function where the design is based on array-based multiplication with two identical inputs. The approximation procedure is generic and can be applied to any N $\times$ N array multiplier.

A. Accurate Array Multiplier

An n-bit array multiplier is composed of n^2 AND gates for partial products (PP) generation, and $n-1$ n-bit adders for partial products accumulation [4]. An array multiplier has a simple design and a regular structure. Thus, it is suitable for multiplying small numbers, e.g., microcontrollers for IoT devices [5]. Due to its importance and applicability in various applications, there are multiple designs of approximate array multipliers, where they replace the exact FAs with approximate ones (e.g., [6], [7], [8]). They differ in the type of the used FAs, the amount of approximation and approximation structure, i.e., replacing arrows or columns in the partial product array.

As shown in Fig. 1, an 8-bit array multiplier includes 64 partial products generated by 64 AND gates. A8 to A1 represent the "*multiplicand*", B8 to B1 represent the "*multiplier*", B1A1 denotes the first partial product, B8A8 represents the last partial product, while P15 to P0 represent the final "*product*" [9]. As shown in Fig. 2, partial product accumulation requires 7 carry-ripple adders each of size 8-bit. Thus, 56 FAs are required, where eight FAs are replaced by half-adders (HAs) since the third input is zero. The HAs are highlighted in blue color in Fig. 2.

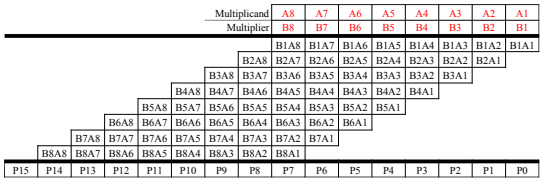

Fig. 1: Partial Products of an 8-bit Array Multiplier

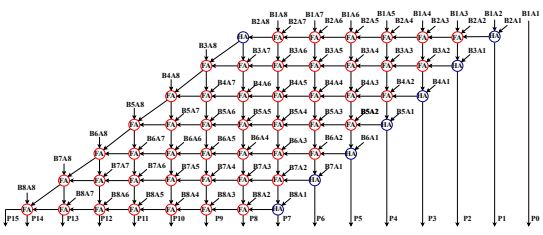

Fig. 2: Structure of Partial Product Accumulation and Summation for an 8-bit Array Multiplier

979-8-3503-7939-6/24/$31.00 ©2024 IEEE

B. Accurate Array Squaring Circuit

Fig. 1 shows that the partial products are symmetrical around the diagonal. Fig. 3 shows the simplified exact partial product tree of an *8-bit exact unsigned array squaring* circuit with 28 partial products generated by 28 AND gates which is greatly simplified compared to an 8-bit array multiplier. Moreover, Fig. 4 shows that 21 FAs and 7 HAs are required. As shown in Fig. 1, the height of the partial products array is N for an N-bit multiplier. However, the height of the squarer is reduced to $\lfloor N/2 \rfloor + 1$, which significantly decreases the hardware complexity and delay.

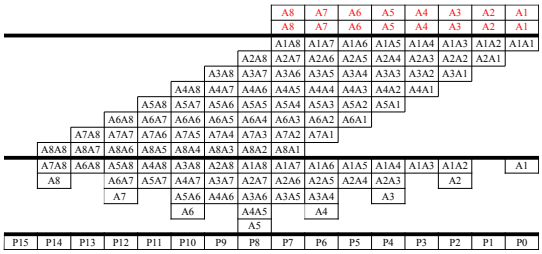

Fig. 3: Partial Products of an 8-bit Array Squaring Circuit

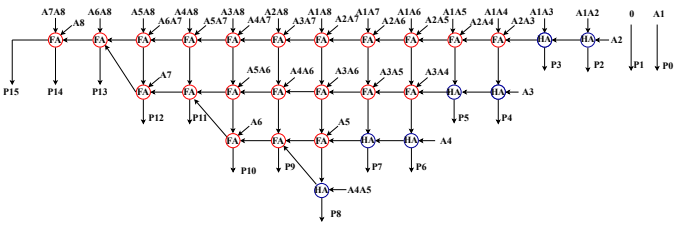

Fig. 4: Structure of Partial Product Accumulation and Summation for an 8-bit Array Squaring Circuit

C. Hardware Metrics

In order to show the benefits of simplifying the design of the squaring function compared to standard multiplication, we analyze various design metrics, i.e., area, power, and delay. We synthesized these designs using *Cadence Innovus* [10]. The synthesis uses a Cadence generic process design kit (GPDK) based on the 45nm CMOS technology node. Table I shows the 8-bit exact array multiplier characteristics and squaring designs. The squarer unit offers a reduction of 50.2%, 69.1%, and 30.2% in area, power, and delay respectively, compared to the array multiplier.

TABLE I: Area, Power, and Delay for the Exact Array Multiplier and Exact Squaring Function

Design	Area (μm^2)	Power (μW)	Delay (ps)
Array Multiplier	363.89	14.14	1,060
Squaring Unit	181.26	4.37	0.74

III. Related Work

Previous work on the approximation of the squaring function is based on various techniques including lookup table-based solutions [11], linear interpolation of the squaring function [12], simplification of the squaring algorithm [13], and simplified combinational logic for logarithmic design [14]. Another technique of developing approximate designs, while having full-bit width results, is utilizing approximate building blocks. Accordingly, different designs of approximate arithmetic units have been investigated, e.g., [15] [16] .

Traditional ROM-based look-up tables (LUT) are restricted to applications with small bit-size operands. For applications with large bit-size operands, the increase in the size of the look-up table is preventive [11]. In [12], the authors presented a linear approximation for the squaring function. They used shift, concatenation and addition operations instead of using multipliers and LUTs. Moreover, some compensation techniques were used to reduce the maximum relative errors.

The authors of [13] presented two simple combinational logic designs for bit-parallel approximate squarers, which are based on simplified combinational logic. The obtained hardware elements grow linearly with the width of input bits. However, the proposed designs are mainly suitable for LUT-based field-programmable gate array (FPGA) implementations only. The authors of [14] proposed an approximate squaring design utilizing simple algorithmic interpolation, where the proposed design is based on shift and subtract operations.

IV. Proposed Squaring Function

Squaring a number could be performed by multiplying the number by itself utilizing regular multipliers. Thus, approximate squaring could be evaluated by using approximate multipliers. However, a *dedicated squaring design* is efficiently simplified due to the similarity of the bits in multiplicand and multiplier numbers [17]. An accurate squaring design as shown in Fig.4 can be transformed into an approximate design by replacing the exact FAs with approximate ones, either horizontally, vertically, or diagonally.

We approximate the conventional structure of the array-based squaring circuit by *vertically* replacing the conventional FAs with their approximate counterpart. For that, we use five well-known approximate FAs, i.e., approximate mirror adders, known as AMA1, AMA2, AMA3, AMA4, and AMA5 [18], that show superiority compared to others [16].

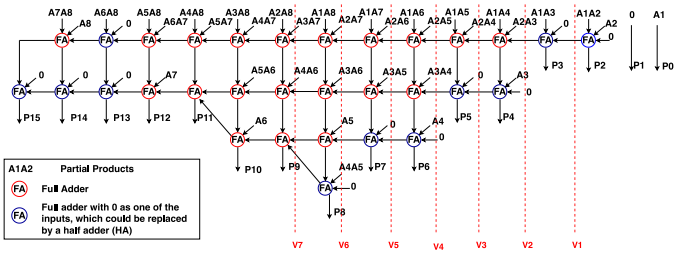

Fig. 5: Different configurations for "Columns of Approximate FAs" of *the AAS*

In this section, we illustrate various designs of an *8-bit approximate array squaring* (*AAS*) function. We propose to replace the FAs of each column with approximate ones gradually. For efficiency of reference, the vertical replacement of FAs, i.e., the level of approximation, is referred to as *V1* to *V7* as highlighted in Fig. 5. With seven levels of approximation and five well-known FAs, we generated 35 approximate squaring designs.

A. Accuracy of the Proposed Designs

We implemented all designs in MATLAB [19] to assess their accuracy. The accuracy of the designs are for the entire 8-bit input domain. The quality analysis of the approximate designs is performed using various error metrics proposed in the literature [20]. Namely, error rate (ER), mean error distance (MED), normalized mean error distance (NMED), mean relative error distance (MRED), and mean square error (MSE). The ER, NMED and MRED metrics range from 0 to 1 where the closer the value to zero, the lower the error. If the value of ER is closer to 1, i.e., 100%, this means the error is very frequent. On the other side, for the metrics MED, NMED, MRED and MSE, the larger the value the greater the magnitude of the error. Additionally, the values of MED and MSE for an 8-bit squaring unit logically range from 0 to 65,025 and 0 to 4,228,250,625, respectively. As shown in Fig. 6, the ER of AMA1-based designs ranges from 25% to 96% with an average of 72%. The average ER for AMA2-based designs is 85% while it is 92% for AMA3-based designs. The designs based on AMA4 and AMA5 have an ER of 72% and 61%, respectively. The average ER for the 35 AMA-based designs is 77%. The ER of AMA1 and AMA4-based designs are overlapped in Fig. 6.

Figure 7 shows the MED of various proposed designs. The MED for the 35 designs ranges between 1 and 721, averaging 91. The MED for AMA1-based designs ranges from 1 to 231, and from 2 to 405 for AMA2-based designs. AMA3-based designs have the highest MED which is 721. The designs based on AMA4 and AMA5 have a minimum MED of 1 and a maximum of 216 and 183, respectively. As shown in Fig. 8, the NMED for the 35 designs ranges from 1.5×10^{-5} to 1.1×10^{-2} with an average of 1.3×10^{-3}. Both AMA1- and AMA4-based designs have an NMED that ranges from 1.5×10^{-5} to 3.56×10^{-3} with an average of 9.8×10^{-4}. The NMED for AMA2-based designs ranges from 3.1×10^{-5} to 6.2×10^{-3} with an average of 1.6×10^{-3}. AMA3-based designs have the highest NMED which ranges from 4.6×10^{-5} to 1.1×10^{-2} with an average of 2.7×10^{-3}. The designs based on AMA5 have an NMED that ranges from 1.5×10^{-5} to 2.8×10^{-3} with an average of 7.4×10^{-4}.

Regarding MRED, the 35 designs have an average of 9.3×10^{-1} and ranges from 2.4×10^{-3} to 1.3×10^{1}, as shown in Fig. 9. The designs based on AMA1 have an average MRED of 5.8×10^{-2} that ranges from 4.8×10^{-3} to 1.7×10^{-1}. The MRED of AMA2-based designs ranges from 3.4×10^{-2} to 5.5 with an average of 1.4. The designs based on AMA3-AMA5 have a minimum value of 3.8×10^{-2}, 4.8×10^{-3}, and 2.4×10^{-3}, respectively. Similarly, these designs have maximum values of 1.3×10^{1}, 1.1×10^{-1}, and 6.2×10^{-2}, respec-

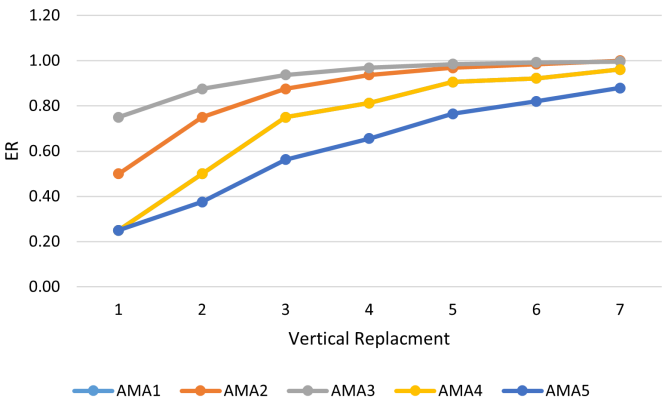

Fig. 6: Error Rate (ER) of the Various Proposed Designs

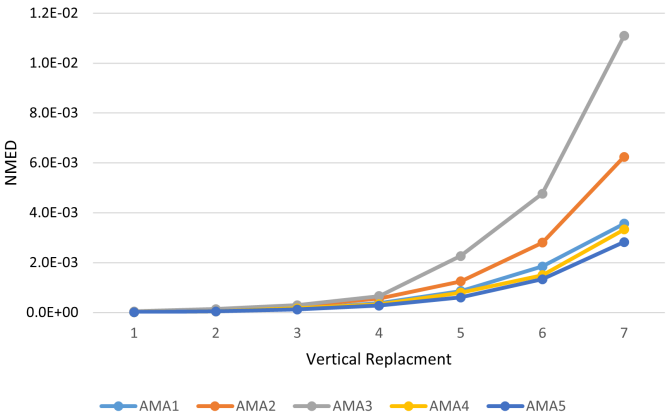

Fig. 8: Normalized Mean Error Distance (NMED) of the Various Proposed Designs

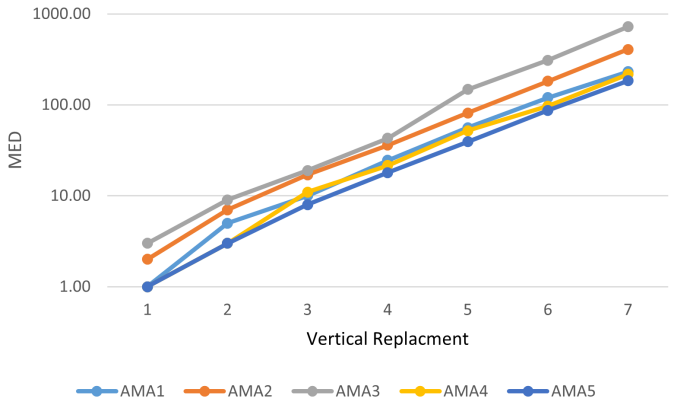

Fig. 7: Mean Error Distance (MED) of the Various Proposed Designs

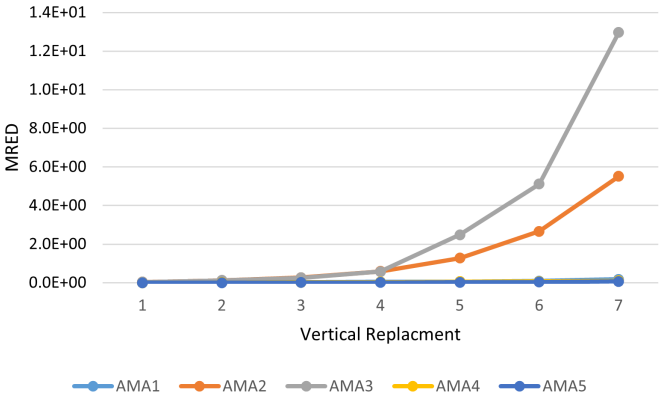

Fig. 9: Mean Error Relative Distance (MRED) of the Various Proposed Designs

TABLE II: Power, Area and Delay for Various Approximate Squaring Designs

Design	Area (μm^2)					Power (μW)					Delay (ns)				
	AMA1	AMA2	AMA3	AMA4	AMA5	AMA1	AMA2	AMA3	AMA4	AMA5	AMA1	AMA2	AMA3	AMA4	AMA5
V1	176.13	179.55	176.81	177.50	177.50	4.30	4.36	4.36	4.37	4.37	0.72	0.74	0.72	0.72	0.72
V2	171.00	177.84	171.68	172.37	173.74	4.39	4.29	4.38	4.25	4.40	0.69	0.73	0.69	0.69	0.70
V3	164.50	175.10	165.87	166.21	163.48	4.59	4.38	4.47	4.26	4.27	0.68	0.74	0.68	0.66	0.66
V4	158.00	172.37	159.37	154.58	153.22	4.65	4.57	4.59	4.08	4.06	0.66	0.72	0.68	0.60	0.61
V5	152.53	168.61	150.82	146.03	136.46	4.47	4.55	4.58	3.74	3.48	0.66	0.70	0.68	0.56	0.55
V6	147.06	164.84	141.59	123.80	119.70	4.12	4.71	4.07	3.16	3.10	0.63	0.71	0.66	0.50	0.51
V7	145.01	160.06	123.80	100.55	97.81	3.84	4.52	3.24	2.46	2.36	0.58	0.68	0.63	0.43	0.43

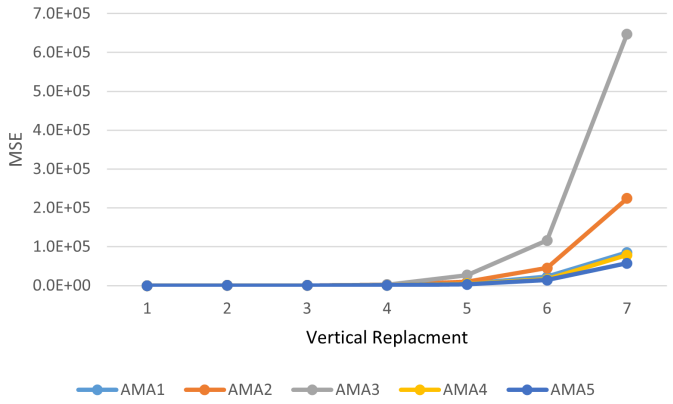

Fig. 10: The Mean Square Error (MSE) of the Various Proposed Designs

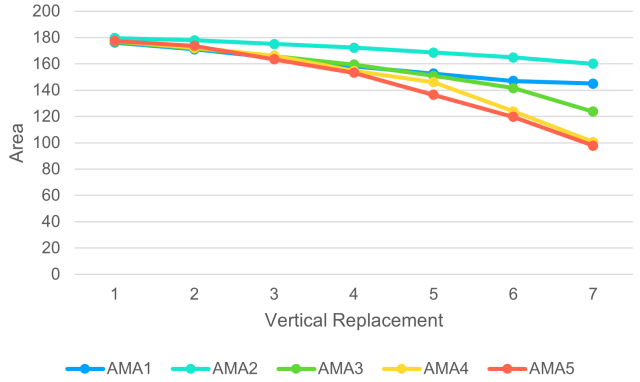

Fig. 11: Area of the Various Proposed Designs

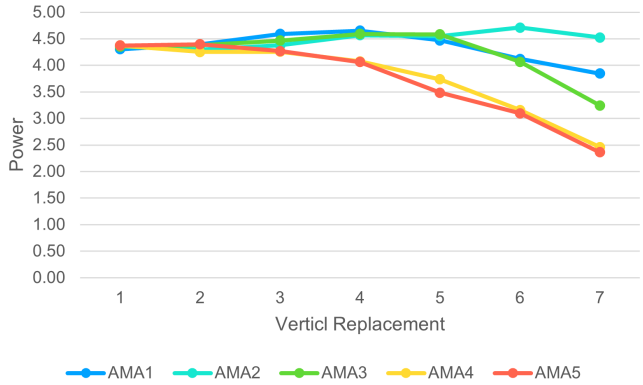

Fig. 12: Power of the Various Proposed Designs

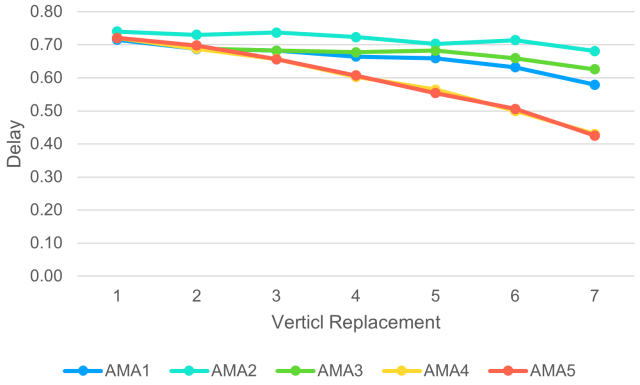

Fig. 13: Delay of the Various Proposed Designs

tively. The average MRED of these designs is 3.1, 4.1×10^{-2}, and 2.2×10^{-2}, respectively. As shown in Fig. 9 the MRED of AMA1, AMA4, and AMA5 designs have the same magnitude (10^{-2} or 10^{-3}) for the configuration from V1 to V7.

The MSE of the 35 designs ranges from 4 to 6.4×10^{5} with an average of 3.9×10^{4}. The minimum MSE of AMA1-AMA5 based designs are 4, 8, 12, 4, and 4, respectively. On the other hand, the maximum MSE of AMA1-AMA5 based designs are 8.5×10^{4}, 2.2×10^{5}, 6.4×10^{5}, 7.8×10^{4}, and 5.7×10^{4}, respectively. The average MSE is 1.6×10^{4}, 4.1×10^{4}, 1.1×10^{5}, 1.4×10^{4}, and 1.1×10^{4}, for AMA1-AMA5, respectively.

Since approximate computing is bounded by the concept of fail small or fail rare, we can notice that the proposed approximate squaring designs in general fail small. Additionally, for small levels of approximation, e.g., V1 and V2, the error was small and rare, i.e., failing small and rare.

B. Hardware Metrics

We implemented all of the approximate squaring designs in VHDL, where we used *Siemens QuestaSim* [21] for functional verification. On the other hand, the synthesis is performed using the *Cadence Innovus* tool [10] with 45nm GPDK.

Fig. 11 shows the areas of the proposed designs which range from $97.81\mu m^2$ to 179.55 μm^2 with an average of $156.16\mu m^2$. Thus, these designs have an average reduction of 14% and 57% compared to the exact squarer and multiplier, respectively. Fig. 12 shows the proposed designs' power, ranging from 2.36μW to 4.71μW with an average of 4.11μW. Thus, these designs have an average reduction of 6% and 57% compared to the exact squarer and multiplier, respectively. Fig. 13 shows the delay of the proposed designs which ranges from 0.43ps to 0.74ps with an average of 0.65ps. Thus, these designs have average reductions of 12% and 39% compared to the exact squarer and multiplier, respectively.

V. Application

To assess the performance of the proposed approximate squaring designs, their quality is analyzed using averaged root mean square of gray-scale images. An 8-bit gray-scale image (X) can be represented as a two-dimensional array of unsigned integers in image processing applications. Each pixel $x(i, j)$ has a value that ranges from 0 to 255, where i = 1,..., r and j = 1, ..., c are the coordinates of the pixel in an $r \times c$

	V1	V2	V3	V4	V5	V6	V7
AMA1	0.397	0.397	0.398	0.398	0.398	0.398	0.400
AMA2	0.397	0.398	0.398	0.398	0.399	0.401	0.406
AMA3	0.397	0.398	0.398	0.398	0.400	0.404	0.413
AMA4	0.397	0.397	0.397	0.397	0.397	0.396	0.394
AMA5	0.397	0.397	0.398	0.398	0.398	0.399	0.398

Fig. 14: Average Root Mean Square (ARMS) for 50 Images Evaluated using the Various Proposed Designs

image. The measure of information provided by an image is called *image energy*, which is quantified by calculating the root mean square (RMS) of image pixels as given in Eq. (1). Thus, the energy evaluation requires a large number of squaring operations. Image energy represents the rate of change in the color, brightness or magnitude of the pixels over limited ranges, which is used in various image processing algorithms [14], e.g., image segmentation, classification, fusion, and image distance measure. In Eq. (1), the total number of pixels is represented by $n = r \times c$. For a set of images, the average RMS is calculated by Eq. (2), for m images.

$$RMS = \frac{1}{n}\sqrt{\sum_{i,j} x^2(i,j)} \tag{1}$$

$$ARMS = \frac{1}{m}\sum_{k=1}^{m} RMS_k \tag{2}$$

We calculated the RMS for a set of fifty 8-bit gray-scale images, each of size 256 $\times$ 256 pixels. Fig. 14 shows the values of ARMS for the images evaluated using various squaring designs. The ARMS for an exact design is 0.40, and we consider a value of 0.40$\pm$threshold as acceptable. For a threshold of 0.05, we have all 35 designs with an acceptable ARMS, i.e., 0.45 $\geq$ ARMS $\geq$ 0.35.

VI. Conclusion

Approximate computing reduces execution time and energy consumption of error-resilient applications, e.g., digital signal processing, by slackening the quality requirements. In this paper, we have investigated and proposed various designs of an 8-bit array-based approximate squaring function that can be implemented with numerous levels of accuracy by changing the level of approximation, i.e., the number of approximate basic blocks. The proposed designs are examined by changing the number of columns with approximate full adders. The detailed analysis confirmed that the proposed designs have optimized results for both hardware and accuracy parameters, i.e., the proposed designs have reduced areas, powers, and delays compared to the exact array squarer, with an average of 14%, 6%, and 12%, respectively. Finally, the efficiency of the proposed designs is verified by employing them in a real-world application, i.e., averaged root mean square of gray-scale images. The obtained results verified its effectiveness and showed promising results. In approximate computing, the final application quality depends on the applied individual inputs, where for some values the output accuracy could be degraded significantly. Therefore, as a future direction, we will investigate the design of a run-time adaptive squaring function, based on the applied inputs. Moreover, we will explore designing circuits with different configurations, e.g., larger operands, targeting more applications.

References

[1] S. Mittal, "A survey of techniques for approximate computing," *ACM Computing Surveys*, vol. 48, no. 4, pp. 1–33, 2016.

[2] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*. Prentice Hall, 2007.

[3] J. W. Leis, *Communication Systems Principles using MATLAB*. John Wiley & Sons, 2018.

[4] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*. Prentice Hall Englewood Cliffs, 2002.

[5] V. Muralidharan and N. S. Kumar, "Design and implementation of low power and high speed multiplier using quaternary carry look-ahead adder," *Microprocessors and Microsystems*, vol. 75, p. 103054, 2020.

[6] B. Shao and P. Li, "Array-based approximate arithmetic computing: A general model and applications to multiplier and squarer design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 4, pp. 1081–1090, 2015.

[7] P. Balasubramanian, R. Nayar, and D. L. Maskell, "Approximate array multipliers," *Electronics*, vol. 10, no. 5, p. 630, 2021.

[8] T. Yamamoto, I. Taniguchi, H. Tomiyama, S. Yamashita, and Y. Hara-Azumi, "A systematic methodology for design and analysis of approximate array multipliers," in *Asia Pacific Conference on Circuits and Systems*. IEEE, 2016, pp. 352–354.

[9] M. Masadeh, Y. Elderhalli, O. Hasan, and S. Tahar, "A Quality-Assured Approximate Hardware Accelerators–Based on Machine Learning and Dynamic Partial Reconfiguration," *Journal of Emerging Technologies in Computing Systems*, vol. 17, no. 4, 2021.

[10] Cadence, "Innovus Implementation System," 2024. [Online]. Available: https://www.cadence.com/ko_KR/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html

[11] C.-L. Wey and M.-D. Shieh, "Design of a high-speed square generator," *IEEE Transactions on Computers*, vol. 47, no. 9, pp. 1021–1026, 1998.

[12] I.-C. Park and T.-H. Kim, "Multiplier-less and table-less linear approximation for square-related functions," *IEICE Transactions on Information and Systems*, vol. E93.D, no. 11, pp. 2979–2988, 2010.

[13] J. Langlois and D. Al-Khalili, "Carry-free approximate squaring functions with O(n) complexity and O(1) delay," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 5, pp. 374–378, 2006.

[14] A. Avramović, Z. Babić, D. Raič, D. Strle, and P. Bulić, "An approximate logarithmic squaring circuit with error compensation for DSP applications," *Microelectronics Journal*, vol. 45, no. 3, pp. 263–271, 2014.

[15] M. Masadeh, O. Hasan, and S. Tahar, "Input-Conscious Approximate Multiply-Accumulate (MAC) Unit for Energy-Efficiency," *IEEE Access*, vol. 7, pp. 147 129–147 142, 2019.

[16] M. Masadeh, O. Hasan, and S. Tahar, "Comparative study of approximate multipliers," in *Great Lakes Symposium on VLSI*. ACM, 2018, pp. 415–418.

[17] T. C. Chen, "A binary multiplication scheme based on squaring," *IEEE Transactions on Computers*, vol. C-20, no. 6, pp. 678–680, 1971.

[18] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, 2012.

[19] Mathworks, "MATLAB," 2024. [Online]. Available: https://www.mathworks.com/products/matlab.html

[20] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1760–1771, 2012.

[21] Siemens Digital Industries Software, "Questa Advanced Simulator," 2023. [Online]. Available: https://eda.sw.siemens.com/en-US/ic/questa/simulation/advanced-simulator/