

Error Analysis of Approximate Array Multipliers

Mahmoud Masadeh¹, Osman Hasan^{1,2}, and Sofiène Tahar¹

¹Department of Electrical and Computer Engineering,
Concordia University, Montréal, Canada
{m_masa, o_hasan, tahar}@ece.concordia.ca

²Electrical Engineering and Computer Science,
National University of Science and Technology, Islamabad, Pakistan

TECHNICAL REPORT

August 2019

Abstract

Approximate computing is a nascent energy-efficient computing paradigm suitable for error-tolerant applications. However, the value of approximation error depends on the applied inputs where individual output error may reach intolerable level while the average output error is acceptable. Thus, it is critical to show the response of approximate design for various applied inputs where understanding the interdependence between the inputs and the approximation error could be utilized to control the output quality. In this report, we exhaustively analyze the accuracy of 20 different designs of 8-bit approximate array multipliers. We designed these multipliers based on various configurations including the *type* of approximable component and how much of the result to approximate, i.e., approximation *degree*. Accuracy analysis shows a strong correlation between the applied inputs and the magnitude of the observed error, i.e., error distance, normalized error distance and peak-to-signal-noise ration. We may utilize this input-dependency of approximation error, in controlling the quality of approximate computing by eliminating large magnitude errors and improving the quality of the final results.

Keywords— Approximate Computing, Approximate Multiplier, Accuracy Analysis, Input Dependency, Approximation Error

Contents

1	Introduction	4
2	Library of Approximate Multipliers	4
3	Error Analysis	5
3.1	Accuracy Metrics	6
3.2	Analysis of ED	7
3.3	Analysis of NED	9
3.4	Analysis of PSNR	11
4	Conclusion	13

1 Introduction

Approximate Computing (AxC) known as best-effort, inexact and tolerable computing is gaining a great attention due to the unprecedented benefits. This emerging computing paradigm is suitable for processing the gigantic data produced in the era of “IoT”, where the world produces 2.5 quintillion bytes of data per day [1]. Many modern applications including web searching, image processing, machine learning, and computer vision show a tendency to accept imperfect results. Approximate computing is a viable method of exploiting error tolerance in such applications to trade accuracy for energy savings and performance improvements.

Multiple approximation techniques have been proposed at different abstraction layers of the computing stack [2]. At the software layer, techniques like loop unrolling and code approximation are commonly employed while, at the hardware layer, techniques like approximation of the functional units, e.g., adders [3], dividers [4] and multipliers [5], are commonly used. A detailed description of approximation techniques and their computing layers can be found in [6].

In [7], we designed a set of 8-bit and 16-bit approximate multipliers with acceptable average output quality and reduced area, delay and power consumption. These multipliers are designed based on three design settings: (1) the type of approximate full adder (FA) used to construct the multiplier; (2) the architecture of the multiplier, i.e., array or tree; and (3) the placement of sub-modules of approximate and exact multipliers in the target multiplier module. We were able to design approximate multipliers, which are suitable to applications with intrinsic error resiliency. We used these designs in an image processing application and obtained interesting results, thus they are applicable to other domains. However, utilizing a specific approximate design for the whole range of the applied inputs could introduce a large magnitude error for specific inputs where such error is undesirable. In this report, we investigate the relationship between the range of design input values and their associated approximation errors in order to identify the individual inputs which cause a large approximation error. For that, we rely on a library of approximate multipliers which encompasses 20 different designs of 8-bit approximate array multipliers. We were able to clearly show the dependency of various approximation metrics on the input data.

The rest of this report is organized as follows: Section 2 describes our library of approximate multipliers with 20 designs. The analysis of various error metrics of the approximate designs is explained in Section 3. Finally, some conclusions are drawn in Section 4.

2 Library of Approximate Multipliers

Aiming to design 8-bit and 16-bit approximate multipliers, in [7] we investigated five approximate mirror adders (AMA) [8], named AMA1–AMA5, three approximate XOR/XNOR based full adders (AXA) [9], named AXA1–AXA3 and three inexact adders (InXA) [10], named InXA1–InXA3. These FAs constitute the basic building blocks to design array multipliers. Moreover, we utilized these 11 approximate FAs to

Table 1: Library of approximate multipliers with 20 configurations based on *Degree* and *Type* knobs

Approximate Design		Degree			
		D1	D2	D3	D4
Type	AMA1	<i>Design1</i>	<i>Design2</i>	<i>Design3</i>	<i>Design4</i>
	AMA2	<i>Design5</i>	<i>Design6</i>	<i>Design7</i>	<i>Design8</i>
	AMA3	<i>Design9</i>	<i>Design10</i>	<i>Design11</i>	<i>Design12</i>
	AMA4	<i>Design13</i>	<i>Design14</i>	<i>Design15</i>	<i>Design16</i>
	AMA5	<i>Design17</i>	<i>Design18</i>	<i>Design19</i>	<i>Design20</i>

construct approximate compressors, i.e., 4-to-3, 5-to-3, 6-to-3, 7-to-3 and 8-to-4, which are the basic blocks for tree multipliers. The approximable building blocks, i.e., FAs and compressors, were used to construct approximate multipliers, i.e., array or tree architecture, where the final result has a variable number of approximable bits, i.e., 7-bits, 8-bits, 9-bits and 16-bits.

In this report, in order to investigate the relationship between the applied inputs to an approximate design and the associated error, we select 8-bit approximate array multiplier designs based on the five approximate mirror adders (AMA1, AMA2, AMA3, AMA4 and AMA5), where the final result has 7-bits, 8-bits, 9-bits and 16-bits of the results being approximated. We call such designs our “Library Approximate Multipliers”.

Generally, an approximate design with N knobs, i.e., $K1, K2, \dots, KN$, will have $|K1| \times |K2| \times \dots \times |KN|$ different design settings. The 8-bit approximate multipliers of our library have two knobs; $K1$ which represents the *type* of the approximate block used to build the approximate design, and $K2$ which is the approximation *degree* of the design. Thus, $K1 = \{AMA1, AMA2, AMA3, AMA4, AMA5\}$, i.e., chosen from the low power approximate full adders [8] and $K2 = \{D1, D2, D3, D4\}$, where $D1$ has 7 bits approximated out of 16-bit result, while $D2, D3$, and $D4$ have 8, 9, and 16 approximate bits, respectively. We have $|K1| \times |K2| = 5 \times 4 = 20$ different settings (as shown in Table 1) with enhanced design characteristics, i.e., reduced power consumption and execution time, compared to the exact design [7].

3 Error Analysis

In this section, we are going to analyze the accuracy of the approximate library, i.e., 20 designs of 8-bit approximate multipliers, based on various error metrics. Then, the accuracy of the approximate designs is analyzed to verify its input-dependency which is an essential consideration in approximate computing. Based on that, we identify the accuracy metrics that are the most suitable to be used in different situations/applications.

3.1 Accuracy Metrics

Approximation introduced accuracy as a new design metric for digital designs. There are several application dependent *error metrics* used in AxC to quantify approximation errors and evaluate design accuracy [10]. For example, considering an approximate design with two inputs, i.e., X and Y , of n -bit each, where the exact result is (P) and the approximate result is (P'), these error metrics include:

- Error Rate (ER): Also called error probability, is the percentage of erroneous outputs among all outputs.
- Error Distance (ED): The arithmetic difference between the exact output and the approximate output for a given input. ED can be given by:

$$ED = |P - P'| \quad (1)$$

- Mean Error Distance (MED): The average of ED values for a set of outputs obtained by applying a set of inputs. MED is an effective metric for measuring the implementation accuracy of a multiple-bit circuit design, which is obtained as:

$$MED = \frac{1}{2^{2n}} \sum_{i=1}^{2^{2n}} |ED_i| \quad (2)$$

- Normalized Error Distance (NED): The normalization of MED by the maximum result that an exact design can have (P_{max}). NED is an invariant metric independent of the size of the circuit, therefore, it is used for comparing circuits of different sizes, and it is expressed as:

$$NED = \frac{MED}{P_{max}} \quad (3)$$

- Relative Error Distance (RED): The ratio of ED to the accurate output, given by:

$$RED = \frac{ED}{P} = \frac{|P - P'|}{P} \quad (4)$$

- Mean Relative Error Distance (MRED): The average value of all possible relative error distances (RED):

$$MRED = \frac{1}{2^{2n}} \sum_{i=1}^{2^{2n}} |RED_i| \quad (5)$$

- Mean Square Error (MSE): It is defined as the average of the squared ED values:

$$MSE = \frac{1}{2^{2n}} \sum_{i=1}^{2^{2n}} |P_i - P'_i|^2 = \frac{1}{2^{2n}} \sum_{i=1}^{2^{2n}} |ED_i|^2 \quad (6)$$

- Peak Signal-to-Noise Ratio (PSNR): The peak signal-to-noise ratio is a fidelity metric used to measure the quality of the output images, and given by:

$$PSNR = 10 * \log_{10}\left(\frac{255^2}{MSE}\right) \quad (7)$$

Table 2: Accuracy metrics for the library of approximate multipliers

Approx Degree	FA Type	ER	MED	NED	MRED	MSE	PSNR
D1	AMA1	0.931	102	0.0165	0.0380	1.69E+04	39.35
	AMA2	0.978	101	0.0213	1.0447	1.44E+04	39.97
	AMA3	0.996	200	0.0416	2.8055	4.76E+04	34.78
	AMA4	0.932	51	0.0083	0.0189	4.11E+03	45.58
	AMA5	0.870	44	0.0069	0.0148	3.14E+03	46.80
D2	AMA1	0.962	246	0.0366	0.0823	9.61E+04	32.10
	AMA2	0.990	227	0.0515	2.1470	7.23E+04	33.26
	AMA3	0.999	474	0.1077	7.0425	2.68E+05	27.65
	AMA4	0.963	107	0.0164	0.0350	1.79E+04	39.15
	AMA5	0.922	93	0.0138	0.0280	1.38E+04	40.32
D3	AMA1	0.977	538	0.0758	0.1679	4.53E+05	25.80
	AMA2	0.996	464	0.0976	4.2929	2.97E+05	27.41
	AMA3	1.000	1010	0.2280	13.4428	1.17E+06	21.75
	AMA4	0.977	185	0.0286	0.0582	5.27E+04	34.30
	AMA5	0.952	185	0.0261	0.0488	5.34E+04	34.56
D4	AMA1	0.9882	13162	1.1862	2.1	3.27E+08	5.54
	AMA2	0.9998	16902	5.0033	272.6	4.00E+08	8.18
	AMA3	0.9999	17211	3.9303	180.7	4.13E+08	7.36
	AMA4	0.9882	6334	0.4705	0.6039	7.91E+07	10.38
	AMA5	0.9825	8096	0.5309	0.6642	1.17E+08	8.85

Next, we present an error analysis for our library of approximate designs in order to observe the dependency of approximation error on the applied inputs. Such observations could be leveraged to control approximation quality by designing an input-aware adaptive approximate computing. Table 2 summarizes various error metrics, i.e., ER, MED, NED, MRED and PSNR, that we obtained based on an exhaustive simulation for 20 approximate designs. The shown values are averaged over the full range of possible inputs, and provide useful insights about the accuracy of designs. In the sequel, we describe the ED, NED and PSNR error metrics one by one in detail.

3.2 Analysis of ED

Figure 1 shows the histogram distribution of the error distance (ED) for the approximate multipliers given in Table 1. The ED for the designs based on *AMA1* with D1,

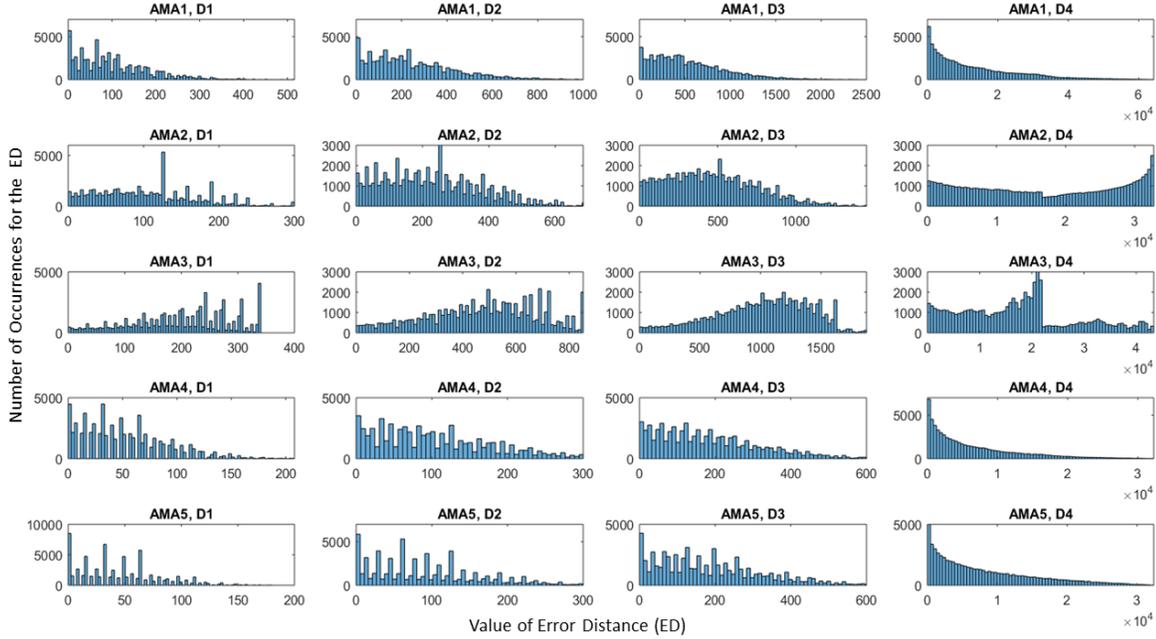


Figure 1: Histogram distribution of error distance (ED) for the library of approximate multipliers

D2, D3 and D4 have the average of 102, 246, 538 and 13162, respectively. While the ED for the designs based on *AMA2* with D1, D2, D3 and D4 have the average of 101, 227, 464 and 16902, respectively. Designs based on *AMA3* with D1, D2, D3 and D4 have the average ED of 200, 474, 1010 and 17211, respectively. However, the designs based on *AMA4* with D1, D2, D3 and D4 have the average ED of 51, 107, 185 and 6334, respectively, which is quite low. *AMA5 based designs with D1, D2, D3 and D4 have the lowest average of ED which is 44, 93, 185 and 8096, respectively.*

We notice that the error distance varies for different inputs, for example, *Design1* (top-left corner of Figure 1) shows that the ED varies from 0 to 518 with an average of 102. If the value of ED was input-independent, then it would be constant for all input values. Such input-dependency is also present in the other designs, i.e., *Design2–Design20*. Moreover, we notice that the variation in ED based on approximation degree for a specific type (represented horizontally in Figure 1) is more evident than the variation between different types for a specific degree (represented vertically in Figure 1), i.e., ED is correlated with the degree more than with the type. Clearly, the ED increases (almost doubles) while increasing the approximation degree for any design type. Ideally, for every input, there exist a specific design (among the 20 designs) with a minimal error distance (ED) which is the most suitable to be used in error-tolerant applications. However, changing the most suitable design for every input is impractical due to the associated overhead.

3.3 Analysis of NED

As explained previously in Equation 3, NED is an invariant metric, which is independent of the size of the approximate design. Figures 2–6 show the NED for different approximate multipliers, utilizing *AMA1*–*AMA5* types, respectively. Each figure depicts a graphical representation for four approximation degrees, i.e., *D1*, *D2*, *D3* and *D4*. Since NED should be computed over a range of inputs rather than a single input, we clustered every 16 consecutive inputs as a group to be able to compute the average of NED over it.

Each bar -from the 256 bars- in these figures represents the average error (NED) of the approximate design based on 16 consecutive values of the first input, i.e., *Input1*, and 16 consecutive values of the second input, i.e., *Input2*. A high error value, e.g., $NED \geq 20\%$, indicates an unacceptable error in most applications. These designs are thus not selected. Now, for clarification purposes, we consider NED with a value of $\leq 100\%$ as our threshold.

Figure 2 shows the NED for the designs based on *AMA1*, where the design based on *D1* has a NED average and maximum NED of 1.7% and 26.2%, respectively. Similarly, the *D2* based design has an average and maximum NED values of 3.7% and 51%, respectively, while the *D3* based design has an average of 7.6%. However, one out of the 256 clusters has a value greater than our specified threshold, i.e., $NED \leq 100\%$. Other clusters have values less than 71.6%. Whereas, for the *D4* based design, 79 clusters out of 256 have a $NED > 100\%$, with an average of 118.6%.

Figure 3 shows the NED for designs based on the *AMA2* type. The design based on *D1* has an average of 2.1% with a maximum value of 68%, while the *D2* based design has an average of 5.1% with one cluster having a value of 164%, and the other 255 clusters have a maximum value of 87.8%. Similarly, the *D3* based design has an average of 9.8% with 4 clusters having a value $\geq 100\%$, and the remaining 252 clusters have a maximum value of 70.3%.

The NED for designs based on *AMA3* type are shown in Figure 4. The *D1* based design has an average of 4.1% with one cluster having a value of 128%, while the other 255 clusters have a maximum value of 49.3%. Similarly, the *D2* based design has an average of 10.8% with 3 clusters having a value $> 100\%$, and the other 253 clusters have a maximum value of 99%. The *D3* based design has a large average of NED, which is around 22.8% and many clusters have $> 100\%$ NED values. Moreover, the *D4* based design has a large error for most of the inputs.

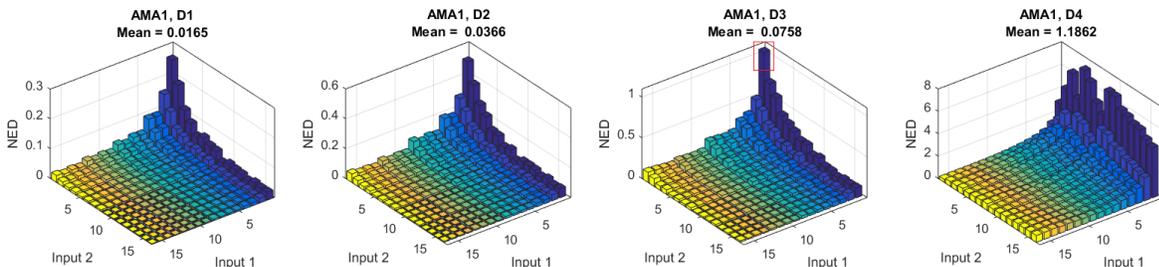


Figure 2: NED for approximate multiplier based on *AMA1* type and *D1*, *D2*, *D3* and *D4* approximation degrees

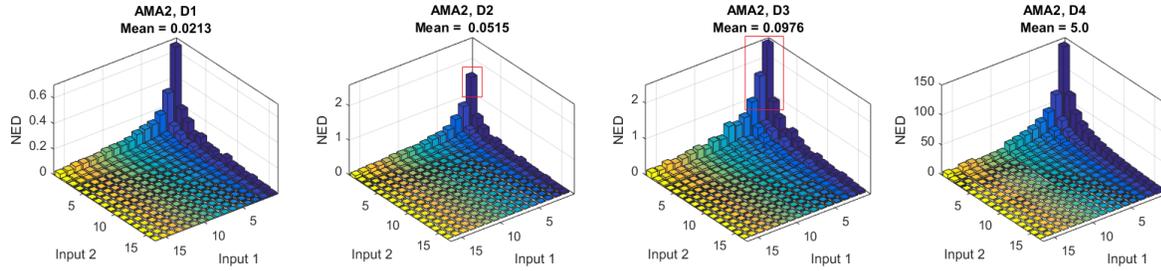


Figure 3: NED for approximate Multiplier based on *AMA2* type and D1, D2, D3 and D4 approximation degrees

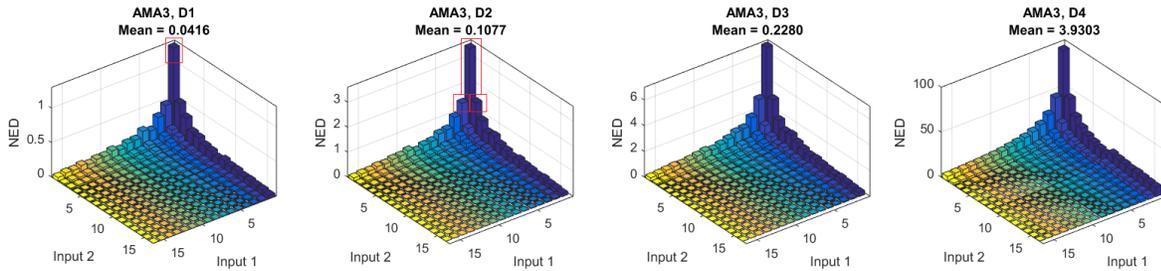


Figure 4: NED for approximate Multiplier based on *AMA3* type and D1, D2, D3 and D4 approximation degrees

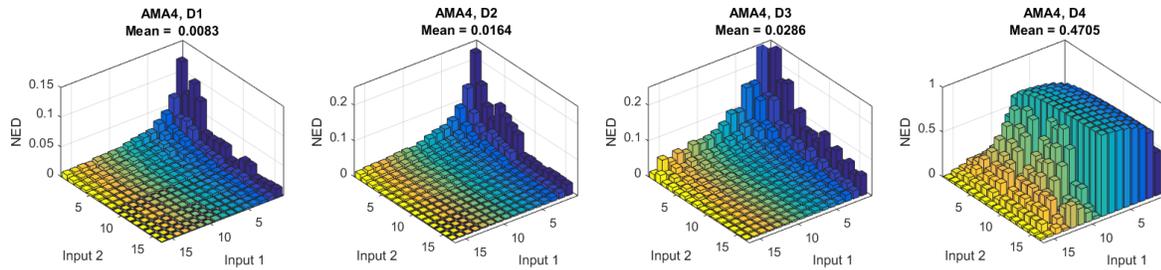


Figure 5: NED for approximate Multiplier based on *AMA4* type and D1, D2, D3 and D4 approximation degrees

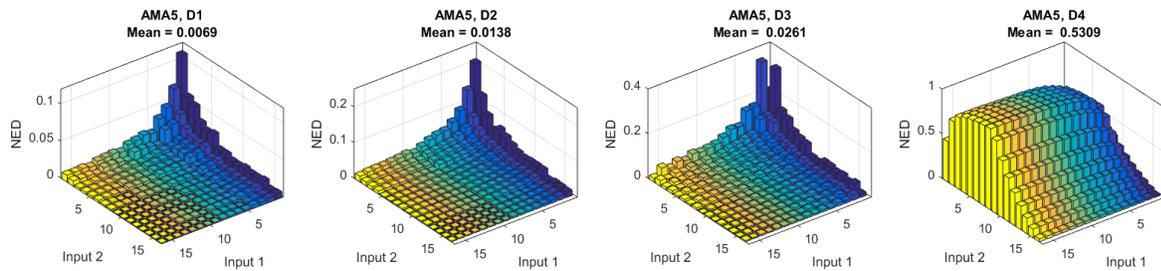


Figure 6: NED for approximate Multiplier based on *AMA5* type and D1, D2, D3 and D4 approximation degrees

Figures 5 and 6 show the NED for designs based on *AMA4* and *AMA5*, respectively. As depicted graphically, all designs have a low value of NED. Figure 5 shows that the maximum values of NED for *AMA4* based designs are 12.5%, 23%, 37% and 91% for *D1–D4* based designs, respectively. Similarly, as shown in Figure 6, the maximum values of NED for *AMA5* based designs are 11%, 21%, 35% and 91% for *D1–D4* based designs, respectively. As we noticed, for a given error threshold, e.g., $NED \leq 100\%$, the average is acceptable for most of the input clusters. However, there are some cases where it has a large error value and such cases should be identified.

3.4 Analysis of PSNR

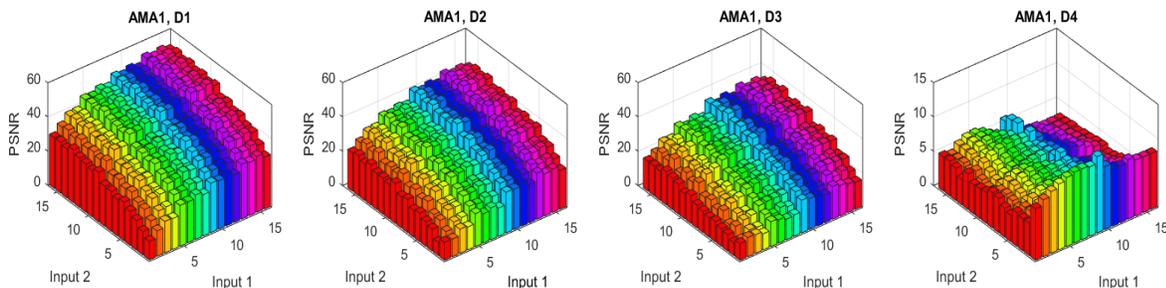


Figure 7: PSNR for approximate multiplier based on *AMA1* type and D1, D2, D3 and D4 approximation degrees

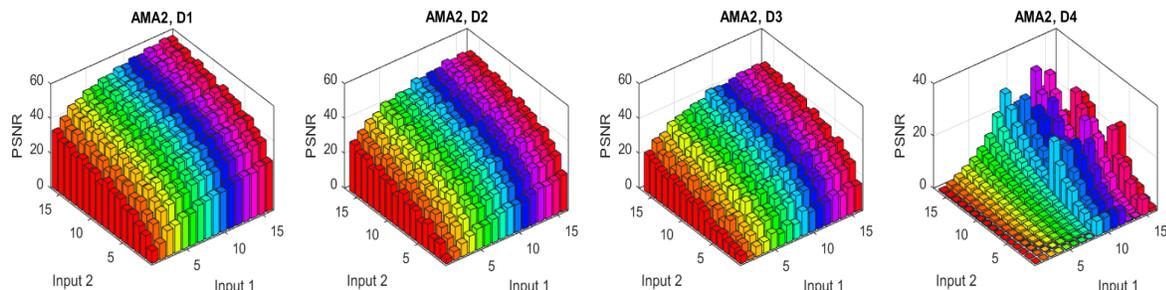


Figure 8: PSNR for approximate multiplier based on *AMA2* type and D1, D2, D3 and D4 approximation degrees

The Peak Signal-to-Noise Ratio (PSNR), as given in Equation 7, depends on the Mean Square Error (MSE). Figures 7–11 show the PSNR for different approximate multipliers utilizing *AMA1–AMA5* types, respectively. For image processing applications, PSNR is an indication for image quality. Thus, it could be used as TOQQ metric while controlling the quality of approximate computing as we proposed in [11]. A low value of PSNR indicates a low image quality associated with a large MSE. Similar to [12], we consider $PSNR \geq 25$ as our threshold for an “acceptable” quality.

Figure 7 shows the PSNR for designs based on the *AMA1* type. The design based on *D1* has an average value of 39.4dB with a minimum value of 10.7dB. It has 19 out of 256 input combinations with $PSNR < 25$ dB. Similarly, the design based on *D2* has

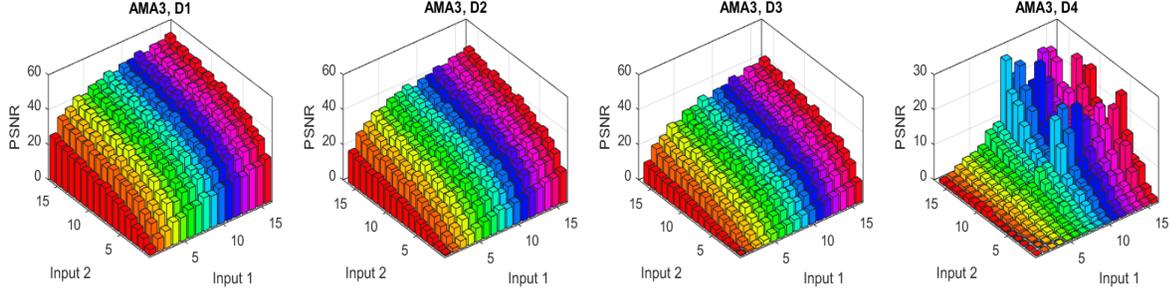


Figure 9: PSNR for approximate multiplier based on $AMA3$ type and D1, D2, D3 and D4 approximation degrees

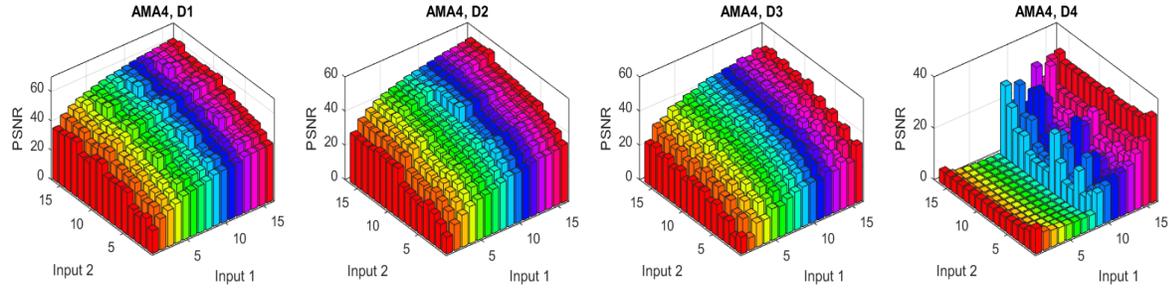


Figure 10: PSNR for approximate multiplier based on $AMA4$ type and D1, D2, D3 and D4 approximation degrees

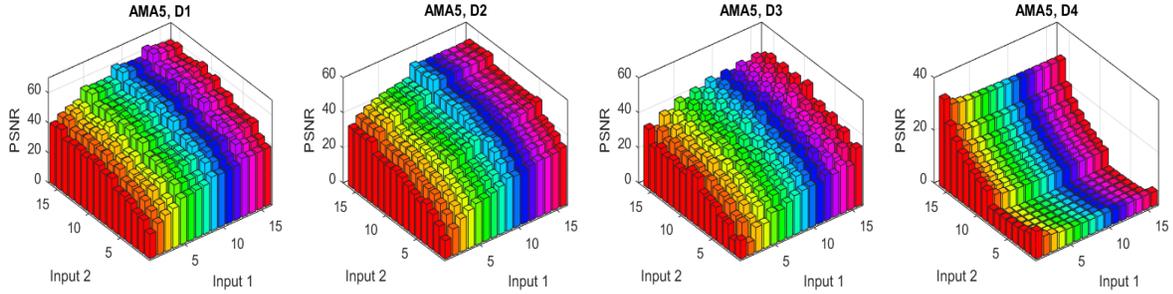


Figure 11: PSNR for approximate multiplier based on $AMA5$ type and D1, D2, D3 and D4 approximation degrees

an average of 32.1dB with a minimum value of 7.7dB, and 54 input combinations with PSNR < 25dB. The $D3$ based design has an average of 25.8dB for the PSNR, with 115 clusters out of the 256 clusters having PSNR < 25dB. Regarding the $D4$ based design, all clusters have a low output quality, with PSNR < 25dB and a maximum value of 11dB.

The PSNR for designs based on $AMA2$ are depicted in Figure 8, where the $D1$ based design has an average of 40dB with a minimum value of 7dB. It has 22 input combinations with PSNR < 25dB. Similarly, the design based on $D2$ has an average of 33.2dB with 55 input combinations having PSNR < 25dB. The $D3$ based design has an average of 27.4dB with 99 input combinations having a PSNR < 25dB. Regarding the $D4$ based design, only 7 clusters have a PSNR > 25dB.

Figure 9 depicts a graphical representation for the PSNR of designs based on *AMA3*. The *D1* based design has an average of 34.8dB with 46 input combinations having PSNR < 25dB, while the *D2* based design has an average of 27.7dB with 92 input combinations having PSNR < 25dB. The *D3* based design has 151 input combinations, which violate the PSNR, i.e., PSNR < 25, while the *D4* based design has only 6 clusters with a PSNR > 25dB.

The PSNR for designs based on *AMA4* are shown in Figure 10. The design based on *D1* has an average of 45.6dB with a minimum value of 15.5dB. It has only 6 input combinations with PSNR < 25dB. Similarly, the *D2* based design has an average of 39.1dB with a minimum value of 9.7dB, and 18 input combinations with PSNR < 25dB. The *D3* based design has an average of 34.3dB for the PSNR, and 43 input combinations with PSNR < 25dB. Regarding the *D4* based design, 230 clusters have a low output quality, with PSNR < 25dB and an average value of 10.4dB.

Figure 11 shows the PSNR for *AMA5* based designs. The *D1* based design has an average of 46.8dB with a minimum value of 16.3dB, where it has 5 input combinations with PSNR < 25dB. Similarly, the design based on *D2* has an average of 40.3dB with a minimum value of 10.4dB and 14 input combinations have PSNR < 25dB. The design with *D3* has an average of 34.6dB for the PSNR, and 33 input combinations have PSNR < 25dB. Regarding the *D4* based design, 239 clusters have a low output quality with an average of 8.8dB.

Based on the above analysis for the PSNR of 20 different approximate designs, we notice that every design has some input combinations that violate the specified quality constraint, e.g., the PSNR should be at least 25dB. Thus, we should avoid such input-dependent cases with low output quality when requiring a high quality result.

4 Conclusion

The emergence of approximate computing has led to the design of interesting designs with low power dissipation, delays and high performance. However, this paradigm requires paying attention to the quality of individual outputs although the average output quality is being acceptable. With this concern, we investigated the relationship between the full range of applicable inputs and their associated approximation errors, where we noticed a strong correlation between them.

References

- [1] C. Dobre and F. Xhafa, “Intelligent services for big data science,” *Future Generation Computer Systems*, vol. 37, pp. 267 – 281.
- [2] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, and J. Henkel, “Architectural-space exploration of approximate multipliers,” in *International Conference on Computer-Aided Design*. ACM, 2016, pp. 1–8.
- [3] N. Zhu, W. Goh, W. Zhang, K. Yeo, and Z. H. Kong, “Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 8, pp. 1225–1229, 2010.
- [4] M. Imani, R. Garcia, A. Huang, and T. Rosing, “Cade: Configurable approximate divider for energy efficiency,” in *2019 Design, Automation Test in Europe Conference Exhibition*, 2019, pp. 586–589.
- [5] S. Venkatachalam and S. B. Ko, “Design of power and area efficient approximate multipliers,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 5, pp. 1782–1786, 2017.
- [6] S. Mittal, “A survey of techniques for approximate computing,” vol. 48, no. 4, pp. 1–33, 2016.
- [7] M. Masadeh, O. Hasan, and S. Tahar, “Comparative study of approximate multipliers,” in *Great Lakes Symposium on VLSI*. ACM, 2018, pp. 415–418.
- [8] 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, 2013.
- [9] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, “Approximate xor/xnor-based adders for inexact computing,” in *IEEE International Conference on Nanotechnology (IEEE-NANO)*, 2013, pp. 690–693.
- [10] H. A. F. Almurib, T. N. Kumar, and F. Lombardi, “Inexact designs for approximate low power addition by cell replacement,” in *Design, Automation and Test in Europe*, 2016, pp. 660–665.
- [11] M. Masadeh, O. Hasan, and S. Tahar, “Using machine learning for quality configurable approximate computing,” in *Design, Automation & Test in Europe*. IEEE/ACM, 2019, pp. 1554–1557.
- [12] Y. Xu, J. D. Deng, and M. Nowostawski, “Quality of service for video streaming over multi-hop wireless networks: Admission control approach based on analytical capacity estimation,” in *International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2013, pp. 345–350.