VMEO: Vector Modeling Errors and Operands for Approximate adders

Vishesh Mishra
Dept. of Computer Science and Engineering
IIT Kanpur, India
vishesh@cse.iitk.ac.in

Urbi Chatterjee
Dept. of Computer Science and Engineering
IIT Kanpur, India
urbic@cse.iitk.ac.in

***Abstract*—Approximate computing techniques are extensively used in computationally intensive applications. Addition architecture being the basic component of computational unit, has received a lot of interest from approximate computing community. Approximate adders are designed with the motivation to reduce area, power and delay of their accurate versions at the cost of bounded loss in accuracy. A major class of approximate adders are implemented using binary logic circuits that operate with a high degree of predictability and speculation. This paper is one of the early attempt to vector model error values that occur in approximate architectures and the inputs fed to them. In this paper, we propose two vectors namely Error Vectors (EVs) and the Input Conditioning Vectors (ICVs) that will form the mathematical foundation of several probabilistic error evaluation methodologies. In other words, the suggested vectors can be used to develop assessment methods to measure the performance of approximate circuits. Our proposed vectors when utilised to analyze approximate circuits, will provide a descriptive idea about (i) chances of error generation and propagation, (ii) the amount of error at specific bit locations and its impact on overall result. This is however not conceivable with existing state-of-the-art methodologies.**

***Index Terms*—Approximate Computing, Computational Unit, Approximate Adders, Vectors, Mathematical modeling**

I. Introduction

Approximate computing [1] has gathered a significant role in the design of power-aware architecture [2]. In a typical case of approximate computing deployed at hardware level, traditional accurate circuits are often replaced with energy-efficient approximate circuits [3]. These approximate circuits provide significantly lower computational time and additional power and energy benefits. Previous research works suggest that introduction of approximate computing at hardware level can play a major role in lowering the area, power and delay estimation of a VLSI design [4].

Modern day computer applications such as machine learning, artificial intelligence, scientific computing, and image processing deal with massive data. Furthermore, large number of convolution operations in these applications repeatedly invoke addition and multiplication architectures. This has been a major reason behind the popularity of approximate adder design among approximate computing community. Previous works have furthered the design of approximate adders ranging from low latency block-based designs to low power truncation-based architectures that come up with bounded imprecision in results [5], [6]. Although existing works claim the boundedness of error by evaluating their designs over error-resilient applications such as image processing and machine learning tools, the error may still get out-of-bound and affect the Quality of Results (QoR) in the end.

The evaluation of approximate addition architectures is majorly done using the error metrics [7], namely Mean Relative Error Distance ($MRED$), Mean Error Distance (MED) and Normalized Mean Error Distance ($NMED$). A large number of random inputs are fed to the targeted design and the error occurred at each iteration is calculated. Hereafter, the above-mentioned metrics are evaluated based on the statistical analysis of Error Distance (ED), where ED is the absolute gap between mean error and the error occurred at each iteration. Evaluation of these metrics require significant computational time since the results are to be generated from the scratch. Additionally, despite consuming sufficient time to analyze, they still fail to differentiate between the designs that have same error margins. The existing error metrics also struggle to pick the most appropriate approximate design for specific applications that usually deal with repetitive set of binary numbers. It is found that ED values do not vary for repetitive set of inputs, therefore the error metrics evaluation may not be helpful in driving conclusive inferences. Other than this, since such metrics only deal with the error analysis of design as a whole, they do not provide any probabilistic inference about the chances of error at specific bit locations and about correctness of sum bits. Furthermore, these metrics provide no conclusive inference about impact that occurred error can cause at specific bit locations and about the number of bits required to completely sink-in the occurred error.

A. Motivation and Contribution

As discussed earlier, the standard error metrics for reliability of approximate adders do not give specific inference about the behavior of approximations in contrast to varying input operands. Therefore, there is a need of error metrics *that do not restrict the error-analysis to mere numbers*. Rather, the error metrics should provide a descriptive idea about the (i) probable chances of error generation and/or propagation and (ii) impact of error occurred on bunch of bits out of the total sum bits. To address this, we propose an alternative mathematical foundation which is different from conventional

ED values computation. To the best of our knowledge, *this is the early attempt that presents vector modeling of errors and operands for approximate adders through VMEO*. These vectors can be utilised to design a probabilistic model for analyzing approximate circuits.

In summary, the novel contributions provided by this work are as follows:

- The foundation of any approximate adder is a 1-bit full adder circuit, thus we first investigate all the theoretical and real-world error-occurring cases corresponding to 1-bit full adder.
- The prospective error value for each case is then calculated. As a result, we obtain seven different error values that are eventually used to construct an error vector $(\overrightarrow{EV})$.
- Finally, we examine the input operands fed to an architecture to formulate the input conditioning vector $(\overrightarrow{ICV})$.

II. VMEO: Vector Modeling Errors and Operands

In this section, we initially discuss the mathematical basis behind the vector modeling of errors and operands. We later discuss the formulation of $\overrightarrow{EVs}$ and $\overrightarrow{ICVs}$.

The measure of accuracy in a cascaded system of full adders is typically determined by the relative performance of the individual full adders. Moreover, the accuracy of results provided by a full adder depends upon two factors namely: i)correctness of carry input fed to it and ii) the computational logic of full adder itself. Now, since in a cascaded system, 1-bit full adders gets the carry input from the carry output of the full adder preceding it, the individual performance of a full adder is also dependent upon the full adder preceding it.

- **Formulation of EVs:** 1 A full adder takes three inputs A, B, and carry input (C_{in}) to generate outputs sum-bit (S), carry output (C_{out}) respectively. This way, a full adder can encounter $2^3 = 8$ different input patterns. We analyze each input pattern separately and take into account all hypothetical error possibilities that can occur when compared with accurate output for the chosen input pattern. Thereafter, we compute the exact error (EE) value as follows:

$$EE = (S_{app} + 2C_{app}) - (S + 2C_{out}). \tag{1}$$

This results in six distinct values ranging from -3 to -3 (excluding 0). Here in Eqn.1, S_{app} are C_{app} are the error prone sum and carry bits and S, C_{out} are accurate sum and carry bits of full adder. Also a factor of 2 is multiplied to carry bits C_{app} and C_{out} respectively. This is done because carry bit is always weighted when compared to a sum bit. Table I shows all possible EE values if the error has definitely occurred. Also, if error-prone bits are equal to accurate bits, it means there is no error and EE value for this case comes out to be zero. Therefore, we now formulate a set X that contains all seven solutions of linear Eqn.1 given by:

$$X = \{-3, -2, -1, 0, 1, 2, 3\} \tag{2}$$

Now, keeping in mind the elements of set X, we define the error vector $(\overrightarrow{EVs})$ given by,

$$\overrightarrow{EV} = a_1.\vec{n_1} + a_2.\vec{n_2} + a_3.\vec{n_3} + a_4.\vec{n_4} + a_5.\vec{n_5} + a_6.\vec{n_6} + a_7.\vec{n_7} \tag{3}$$

where $a_i \epsilon \{0, 1\}$, $|\vec{n_i}| = 1$ and $\vec{n_i} \times \vec{n_j} = 0 \ \forall \ i, j \ \epsilon [1, 7]$. Here in Eqn.3, we have mapped different EE values to different unit vectors along hypothetically different directions. This is done because each EE value independently impacts the overall result in a way indifferent to other. We will use $(\overrightarrow{EV})$ to denote error of a cascaded system of full adders in upcoming section.

- **Formulation of ICVs:** The probable chances when a full adder may produce error prone sum bit and/or carry bit are already discussed earlier. However, the occurrence and further propagation of error greatly depends upon various input patterns. Clearly, if both inputs (A and B) fed to the accurate full adder are $(0, 0)$ or $(1, 1)$ then the carry-out is certainly accurate. Furthermore, if the input bits are $(0, 1)$ and $(1, 0)$ respectively, then error occurrence and propagation becomes probabilistic in nature. From this we infer that error generation/propagation and input bit patterns are greatly related to each other. Therefore, to clearly identify a specific input pattern, we associate operands pattern detector (OPD) corresponding to two distinct type of input patterns. We define OPD as follows:

$$OPD = A \oplus B \tag{4}$$

Clearly, OPD can have two values namely 0 or 1. Table II shows possible OPD values Therefore, we now define a set Y that contains solution of linear Eqn.4 given by:

$$Y = \{0, 1\} \tag{5}$$

Now, keeping in mind the elements of set Y, we define conditioning vector $(\overrightarrow{ICV})$ given by:

$$\overrightarrow{ICV} = b_1.\vec{m_1} + b_2.\vec{m_2} \tag{6}$$

where $b_i \epsilon \{0, 1\}$, $|\vec{m_i}| = 1$ and $\vec{m_i} \times \vec{m_j} = 0 \ \forall \ i, j \ \epsilon \{1, 2\}$. Here in Eqn.3, we have mapped different OPD values to different unit vectors along hypothetically different directions. This is done because each OPD value independently impacts the overall result in a all together different way. We will use $(\overrightarrow{ICV})$ to denote input pattern of a cascaded system of full adders in upcoming section.

III. Evaluation

In this section, we demonstrate the evaluation of $\overrightarrow{EV}$ and $\overrightarrow{ICV}$ with the help of an example. For this purpose, we consider two 8-bit inputs [01010110] and [10010111] respectively. Clearly, we require '8' 1-bit full adders to compute the addition of two considered numbers. Therefore, we assume that the addition is performed by a circuitry that will contain '8' 1-bit full adders. Now, we perform the addition via an accurate adder and a hypothetical approximate adder respectively. As a result, we get 8-bit sum and 8-bit carry out corresponding to

$[Cin, A, B]$	S	$Cout$	S_{app}	C_{app}	Exact Error (EE)
0 0 0	0	0	1	0	1
0 0 0	0	0	0	1	2
0 0 0	0	0	1	1	3
0 0 1	1	0	0	0	-1
0 0 1	1	0	1	1	2
0 0 1	1	0	0	1	1
0 1 0	1	0	0	0	-1
0 1 0	1	0	1	1	2
0 1 0	1	0	0	1	1
0 1 1	0	1	1	1	1
0 1 1	0	1	0	0	-2
0 1 1	0	1	1	0	-1
1 0 0	1	0	0	0	-1
1 0 0	1	0	1	1	2
1 0 0	1	0	0	1	1
1 0 1	0	1	1	1	1
1 0 1	0	1	0	0	-2
1 0 1	0	1	1	0	-1
1 1 0	0	1	1	1	1
1 1 0	0	1	0	0	-2
1 1 0	0	1	1	0	-1
1 1 1	1	1	0	1	-1
1 1 1	1	1	1	0	-2
1 1 1	1	1	0	0	-3

TABLE I
TRUTH TABLE FOR 1-BIT ACCURATE FULL ADDER$[(C_{in}, A, B) \rightarrow (S, C_{out})]$, POSSIBLE ERROR-PRONE SUM, CARRY BIT (S_{app}, C_{app}) AND CORRESPONDING EEs

A	B	Operands Pattern Detector (OPD)
0	0	0
0	1	1
1	0	1
1	1	2

TABLE II
POSSIBLE INPUT COMBINATIONS A, B AND CORRESPONDING $OPDs$

accurate and approximate adder. Thereafter, we perform the following steps:

STEP 1: Firstly, compute EEs and $OPDs$ with help of Eqn.1 and Eqn.4.

STEP 2: Secondly, note EE and OPD values corresponding to each full adder and then write $\overrightarrow{EV}$ and $\overrightarrow{ICV}$ for each full adder using Eqn.3 and Eqn.6 (starting from least significant bit side).

STEP 3: Finally, add the obtained individual $\overrightarrow{EVs}$ and $\overrightarrow{ICVs}$ to get the $\overrightarrow{EV}$ and $\overrightarrow{ICV}$ for 8-bit approximate adder design.

Tab.III contains $\overrightarrow{EVs}$ and $\overrightarrow{ICVs}$ corresponding to each full adder. Based on these values, we now formulate, $\overrightarrow{EV}$ and $\overrightarrow{ICV}$ for 8-bit adder as follows:

$$\overrightarrow{EV} = \vec{n_2} + 4\vec{n_4} + 2\vec{n_5} + \vec{n_6} \tag{7}$$

$$\overrightarrow{ICV} = 5\vec{m_1} + 3\vec{m_2} \tag{8}$$

Here in Eqn.8, $\vec{m_1}$ is multiplied by scalar 5 since exactly five values of $OPDs$ are zero and $\vec{m_2}$ is multiplied by scalar 3 because three values of $OPDs$ are 1. Eqn.7 can also be interpreted in a similar manner. Fig.1 depicts the ilusration example considered for evaluating $\overrightarrow{EV}$ and $\overrightarrow{ICV}$. This way, $\overrightarrow{EV}$ and $\overrightarrow{ICV}$ can be evaluated for any generic architecture. Later, these vectors can be utilised to design a mathematical model for accuracy analysis of approximate architectures.

OPD	EE	$\overrightarrow{ICV}$	$\overrightarrow{EV}$
1	1	$\vec{m_2}$	$\vec{n_5}$
0	0	$\vec{m_1}$	$\vec{n_4}$
0	-2	$\vec{m_1}$	$\vec{n_2}$
0	0	$\vec{m_1}$	$\vec{n_4}$
0	0	$\vec{m_1}$	$\vec{n_4}$
0	0	$\vec{m_1}$	$\vec{n_4}$
1	1	$\vec{m_2}$	$\vec{n_5}$
1	2	$\vec{m_2}$	$\vec{n_6}$

TABLE III
OPD, EE, AND CORRESPONDING $\overrightarrow{ICVs}$, $\overrightarrow{EVs}$

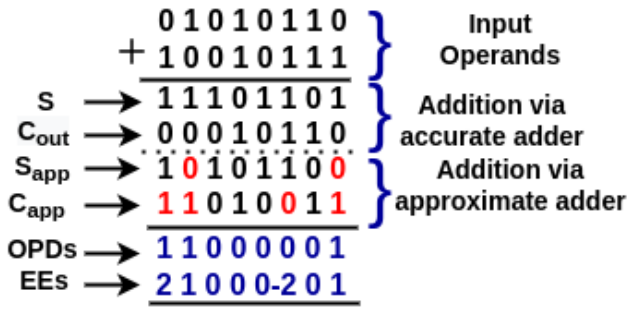

Fig. 1. Illustration example depicting evaluation of OPDs and EEs

IV. CONCLUSION

This article presents one of the earliest attempts to vector model error values that appear in approximate designs and the inputs provided to them. In this article, we present two vectors, called error vectors $(\overrightarrow{EVs})$ and input conditioning vectors $(\overrightarrow{ICVs})$, which will serve as the mathematical basis for a number of probabilistic error evaluation approaches. To put it another way, the suggested vectors can be utilised to create evaluation techniques to gauge how well approximation circuits operate. With current state-of-the-art approaches, this is not nevertheless practicable.

REFERENCES

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*, 2013, pp. 1–6.

[2] S. S. Dayapule, F. Yao, and G. Venkataramani, "Powerstar: Improving power efficiency in heterogenous processors for bursty workloads with approximate computing," in *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2019, pp. 175–182.

[3] M. Breuer, "Hardware that produces bounded rather than exact results," in *Design Automation Conference*, 2010, pp. 871–876.

[4] Z. Aizaz and K. Khare, "Area and power efficient truncated booth multipliers using approximate carry based error compensation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2021.

[5] R. Bhattacharjya, V. Mishra, S. Singh, K. Goswami, and D. S. Banerjee, "An approximate carry estimating simultaneous adder with rectification," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 2020, pp. 139–144.

[6] S. Singh, V. Mishra, S. Satapathy, D. Pandey, K. Goswami, D. S. Banerjee, and B. Jajodia, "Efcsa: An efficient carry speculative approximate adder with rectification," in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, 2022, pp. 1–7.

[7] 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, 2013.