Statistical Error Analysis for Low Power Approximate Adders

Muhammad Kamran Ayub[1], Osman Hasan[1], and Muhammad Shafique[2]

[1]School of Electrical Engineering & Computer Science, National University of Sciences and Technology, Islamabad, Pakistan

[2]Institute of Computer Engineering, Vienna University of Technology, Vienna, Austria

{14mseemayub,osman.hasan}@seecs.nust.edu.pk, muhammad.shafique@tuwien.ac.at

ABSTRACT

Low-power approximate adders provide basic building blocks for approximate computing hardware that have shown remarkable energy efficiency for error-resilient applications (like image/video processing, computer vision, etc.), especially for battery-driven portable systems. In this paper, we present a novel scalable, fast yet accurate analytical method to evaluate the output error probability of multi-bit low power adders for a predetermined probability of input bits. Our method recursively computes the error probability by considering the accurate cases only, which are considerably smaller than the erroneous ones. Our method can handle the error analysis of a wider-range of adders with negligible computational overhead. To ensure its rapid adoption in industry and academia, we have open-sourced our LabVIEW and MATLAB libraries.

Keywords: *Low Power, Approximate Computing, Probabilistic Analysis, Error, Performance, Scalability, Accuracy*

1. INTRODUCTION

New avenues of hardware optimizations are being revealed by alleviating the exactness in the traditional computing for a wider range of applications requiring heavy data computations. Recent research works conducted by both academia [8] and industry, e.g., Microsoft [2] [5], Intel [12], IBM [14] and others [3], have shown that a wider range of resource and power hungry applications, e.g., video or image processing, data mining, computer vision, big data analytics, mobile computing, deep learning networks, artificial intelligence etc., exhibit inherent error resilience. This can be leveraged to relax the error bounds by using approximate computing to benefit from optimizations in area, power and performance [4] [8]. The pliability in these applications lies because of various reasons, like the existence of redundant information or noise in a real-world bulk data, extent to which a particular application user tends to scrutinize an information, error attenuation characteristics of processed algorithm and lack of need to get a highly precise answer [16]. Thus, inexact computing with certain error bounds in place of precise computing still serves the purpose of producing acceptable quality output for significant amount of data.

Adders are one of the most frequently used arithmetic components in the above-mentioned applications. A lot of work has been done in decreasing the power, latency and area of a single-bit adder evolving a class of adders, known as Approximate Adders, which can be further classified into *low latency approximate adder (LLAA)* [15] [10] and *low power approximate adder (LPAA)* [7] [6]. In LPAA, optimization techniques involve logic complexity simplification of precise adders by reducing the number of transistors and internal node capacitances as a trade-off of adder accuracy. This reduction has a significant effect, especially in terms of size and power, when these single-bit adders are cascaded to form any multi-bit adder topology, e.g., traditional Ripple Carry Adder (RCA) and Carry Save Adder (CSA), which are used as building blocks of digital signal processors (DSP) and other applications [7] [6]. Besides the LPAAs, many high speed LLAAs leverage upon the fact that the carry propagation chain for the most input combinations is shorter than that for the worst case scenario [17]. Such adders use numerous small sub-adder blocks with no carry linkage. A recently proposed low latency Generic Accuracy Configurable Adder (GeAr) uses a similar approach to provide different approximation configurations enabling flexibility and trade-off in performance vs output quality [15].

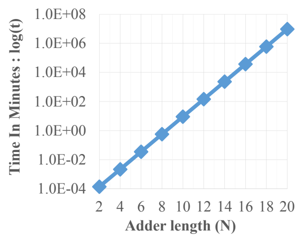
To maintain the output quality within bounds, error in every approximation technique needs to be pre-evaluated and compared with maximum acceptable error resilience of the given application before improvising it. Maximum error bounds typically depend on the type and resilience of the application under consideration and help in deciding the best suited approximation technique. Performance of any approximate adder is estimated in terms of error probability indicating expected erroneous outcomes. The analytical analysis of probability of error of an approximate adder gets more and more complex by increasing the number of stages in adder. In most of the LPAA related work, the error probability of a *single* stage has been analytically evaluated and compared with other existing versions [1]. While, exhaustive simulations have been carried out in order to estimate the error probability in multistage applications for performance metrics/case studies. A probability of error estimation in the sum and carry magnitude values has been done, which is then compared with their corresponding probability values obtained by exhaustive simulations to get error probability in [7]. Similarly, an analytical approach for error probability analysis of the low latency GeAr adder has been proposed in [11]. In this work, each term in the inclusion-exclusion equation has been solved to estimate error probabilities for any number of sub-adder blocks.

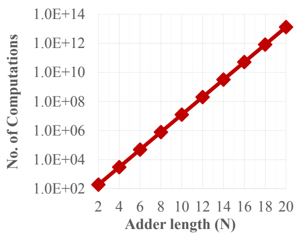
To the best of our knowledge, no analytical method for reporting the error probabilities of a multistage LPAA exists in the literature so far. Moreover, the above-mentioned analytical method, based on the principal of inclusion-exclusion, is not

DAC '17, June 18-22, 2017, Austin, TX, USA

© 2017 ACM. ISBN 978-1-4503-4927-7/17/06...$15.00

DOI: http://dx.doi.org/10.1145/3061639.3062319





(a) (b)

Figure 1: Increase in Exhaustive Simulation Time and Number of Computations (Addtions, Comparisons etc.) for LPAA vs. Adder length (N) (Intel Core i7 3rd Generation)

feasible for the analysis of low power multi-bit adders as the number of stages in these adders are much larger than the sub-adder stages in low latency adders, which in turn causes *an exponential expansion of equation terms.*
Though, exhaustive simulation can serve as a low effort solution for error analysis of applications with fewer stages of approximated bits in a multi-bit adder but for large number of approximation bits, it becomes inefficient with respect to resource and time consumption. Figure 1 illustrates this problem and we can observe an exponential increase in the simulation time and number of arithmetic computations involved as the width of the adder increases.

1.1 Our Novel Contributions

In order to overcome above mentioned challenges, we propose an *analytical error probability estimation approach for low power adders* that recursively computes the required probability without involving the principle of inclusion-exclusion. This results in *compact probability expressions* and keeps the analysis approach *scalable*. The approach is based on matrix arithmetic, which makes it quite *generic* in terms of handling any low power approximate full adder in multi-bit adder configuration. It also exhibits the *flexibility* to cater for probability estimation of multi-bit adders made up of different types of LPAAs which can be very useful in identifying the most optimal configuration of a multi-bit adder based on known probabilities of inputs. Finally, the proposed approach can also be used for the error *analysis of LLAAs*, like GeAr adders, and by avoiding the principal of inclusion-exclusion, can provide the corresponding error probabilities with *less computational overhead*. Also, the analysis complexity will further aggravate when these adders form an accelerator data path.

1.2 Open-Source Contribution

We have released LabVIEW and MATLAB libraries of our proposed methodology at https://sealpaa.sourceforge.io/ for state-of-the-art LPAAs. This release would not only be helpful for reproducible research comparisons in this direction but also for industry to analyze the performance of a particular LPAA before using it in their applications for design automation of complex approximate computing processors, and high-level synthesis.

2. PRELIMINARIES

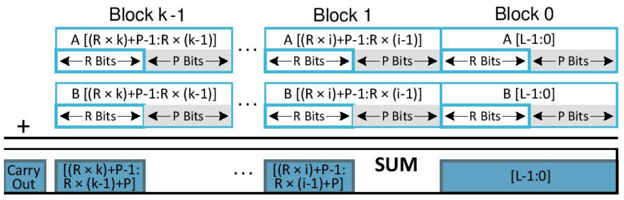
To facilitate the understanding of the proposed analysis method and its application in analyzing LPAAs and LLAAs, we describe the design methodologies for LPAA and LLAA (GeAr Adder) in this section.

2.1 Low Power Approximate Adder

Exploiting the advantage of relaxation of computational precision, LPAA reduces the logic complexity of precise adders at transistor level. This reduction in complexity decreases the power consumption by reducing the switch capacitances and leakages because of smaller size. Also, logic simplicities lead to shorter critical paths, which ensure voltage reduction without generating timing induced errors [6]. Similarly, architectures based on Algorithmic Noise Tolerance (ANT) [9], Significance Driven Computation (SDC) [13] and non uniform Voltage Over Scaling (VOS) have also been proposed in similar contexts.
Using the above methodologies, different versions of LPAAs have been proposed. Table 1 shows the truth table, representation of 5 low power adders (LPAA 1-5) topologies presented in [7], as well as 2 topologies (LPAA 6-7), presented in [1]. The cases with either sum or carry bit corrupted are marked as error cases and are represented in bold red. Approximate Adder 2 [7] and Approximate Adder 3 [1] have different architectures at the transistor level but present the same truth tables, due to which the latter is not included in the discussions. Table 2 compares the area, quality and power consumption for the single stage LPAA [7], which can be considered as another metric, along with accuracy, for LPAA selection in a multi-bit application.

Table 2: Characteristics of LPAA Proposed in [7]

LPAA Type	Error Cases	Power (nW)	Area (GE)
LPAA 1	2	771	4.23
LPAA 2	2	294	1.94
LPAA 3	3	198	1.59
LPAA 4	3	416	1.76
LPAA 5	4	0	0



Block k -1
Block 1
Block 0
A [(R × k)+P-1:R × (k-1)]
A [(R × i)+P-1:R × (i-1)]
A [L-1:0]
R Bits
P Bits
B [(R × k)+P-1:R × (k-1)]
B [(R × i)+P-1:R × (i-1)]
B [L-1:0]
+
Carry Out
[(R × k)+P-1: R × (k-1)+P]
[(R × i)+P-1: R × (i-1)+P]
SUM
[L-1:0]

Figure 2: Block Diagram of GeAr Adder Proposed in [15]

2.2 Low Latency Approximate Adder

LLAA takes advantage of using multiple independent sub-adders with or without overlapping and thus reduces carry propagation path. In LLAAs, we will discuss GeAr as it is a latest generic model and captures all of the prominent previously proposed LLAAs [15]. GeAr is a highly parameterizable adder design with accuracy configurability. While adding two N-bit operands, GeAr adder makes use of L-bit overlapped multiple sub-adder units working in parallel without waiting for carry from the previous sub-adder stage, where $L <= N$. Assuming R as the number of bits contributing to the final output while P are the number of overlapping bits, then, $L = R + P$ defines the length of the sub-adder. For a known L, R and N, the total number of k sub-adder stages is given as $k = ((N - L)/R) + 1$ [15]. Figure 2 shows a block diagram of the GeAr adder, where $i = 0, 1, .., k - 1$ represents the number of sub-adder blocks. The GeAr adder limits the carry propagation delay to L-bit sub-adders instead of N-bits.

3. CHALLENGES IN TRADITIONAL STATISTICAL ANALYSIS TECHNIQUES

The key challenge in the application of traditional methods for statistical analysis of multi-bit approximate adders is the dependency of each stage on all its previous stages. This is the case because the effect of an error in a particular stage, propagates through all the stages along the length of the adder and there is a chance that it might be statistically detected in more than one stages as illustrated in Figure 3, which shows

Table 1: Truth Tables of Different Single Bit LPAA as Proposed in [7] [1]

Inputs			AccuFA		LPAA 1		LPAA 2		LPAA 3		LPAA 4		LPAA 5		LPAA 6		LPAA 7	
A	B	Cin	Sum	C_{out}	Sum	C_{out}	Sum	C_{out}	Sum	C_{out}	Sum	C_{out}	Sum	C_{out}	Sum	C_{out}	Sum	C_{out}
0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1	0	1	0	1	0	0	0	1	1	1	0
0	1	0	1	0	0	1	1	0	0	1	0	0	1	0	1	0	1	0
0	1	1	0	1	0	1	0	1	0	1	1	0	1	0	0	1	1	1
1	0	0	1	0	0	0	1	0	1	0	0	1	0	1	1	0	1	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1	1
1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	0	0	0	1
1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1

Table 3: Number of Inclusion-Exclusion Equation Terms vs, Multiplications, Additions and Memory Elements required for Traditional Analytical Analysis of a Multi-Stage Adder

No. of stages	Terms	Multiplications	Additions	Memory Units
4	15	28	14	31
8	255	1016	254	511
12	4095	24564	4094	8191
16	65535	52427	65534	131071
20	1.04×10^9	10.5×10^6	1.04×10^9	2.10×10^6
24	16×10^9	201×10^6	16×10^9	33×10^6
28	268×10^9	3.70×10^9	268×10^9	536×10^6
32	40×10^{12}	68.7×10^9	40×10^{12}	8.5×10^9

the implementation of a multi-bit adder using single-bit low power adders as building blocks. There are two main challenges faced in calculating the probability of error based on input (operands and carry) at every stage:

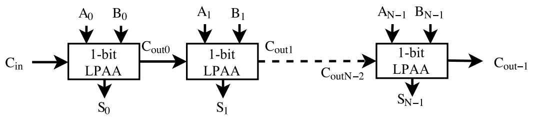
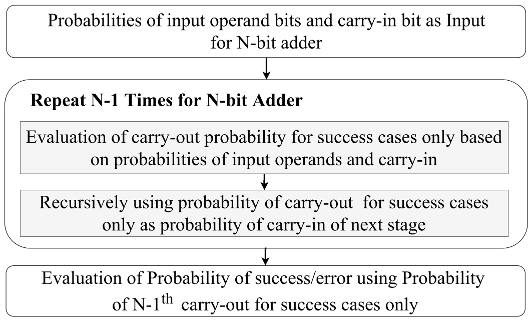


Figure 3: Block Diagram of Multi-Stage Adder

1. The probability of carry out needs to be calculated at every stage of the full adders as it does not remain the same as the one for carry-in for most approximate adders.
2. The overall probability of error of a multi-bit adder cannot be estimated by simply adding the error probabilities for all the stages as this would result in duplication of terms. Thus, if we are simply adding them then we have to extract the replicated error probabilities and the principle of inclusion-exclusion was used for this purpose in [11]. The problem in using the principle of inclusion-exclusion is the generation of a large number of terms, i.e., $\sum_{i=0}^{k-1} \binom{k}{1}$, with the increase in the number of stages k. Table 3 shows an elaboration of inclusion-exclusion terms vs number of stages. *This clearly shows that for the statistical analysis of a 32-bit adder, we have to deal with* 40×10^6 *terms, which is practically impossible.*

Based on the above-mentioned challenges, we cannot use a recursive algorithm for the error calculation, as we need the history of errors from all stages. Thus, we cannot reuse the memory resources as the memory requirements for error computation increase rapidly, i.e., by $\sum_{i=1}^{k} 2^i$, as the number of stages k of approximate adders increases. Table 3 also shows an estimate of multiplications, additions and memory elements needed for the statistical analysis in a traditional fashion. Moreover, an additional overhead for the computation of joint probabilities is also encountered while evaluating the error probability using the terms obtained from the principle of inclusion-exclusion.

It is important to note that the traditional analytical algorithm is an acceptable solution while dealing with overlapping sub-adder blocks, as was the case in [11], because the number of error generating stages that have to be resolved is far less than the ones obtained in multi-bit LPAAs.



Probabilities of input operand bits and carry-in bit as Input for N-bit adder
Repeat N-1 Times for N-bit Adder
Evaluation of carry-out probability for success cases only based on probabilities of input operands and carry-in
Recursively using probability of carry-out for success cases only as probability of carry-in of next stage
Evaluation of Probability of success/error using Probability of N-1th carry-out for success cases only

Figure 4: Flow Chart of Proposed Method

4. PROPOSED METHODOLOGY

To resolve the above mentioned challenges, our proposed analytical algorithm recursively estimates the probability of success for each stage in the multi-bit adder such that there is no dependency of the previous stage on the next. To accomplish this, it ensures to discard the erroneous or corrupted terms at every stage from propagating carry probability and thus also avoids the principle of inclusion-exclusion. This keeps the algorithm very simple and resource friendly with almost negligible computational time, which makes it quite scalable in terms of the number of bits (stages) in a multi-bit adder.

The first step in the proposed methodology, depicted in Figure 4, is to use the probabilities of the input operands and the carry-in bit at the 1^{st} stage to evaluate the probability of the carry-out bit for accurate cases only. These probabilities can in turn be used recursively as carry-in probabilities for the next stages along with the known input operand probabilities to recursively estimate the carry-out probabilities for accurate cases only for $N-1$ stages. It is worth mentioning that since we are only dealing with the accurate cases, so the carry-out probabilities keeps on decreasing because of the discarded error terms. The probability of error for the last stage can then be evaluated by subtracting the probability of success, obtained from the above-mentioned recursive process. Similarly, the probability of the output sum bits (if required) can also be evaluated involving every stage. Similar to other analysis techniques, we also consider that all the operand bits and the input carry bit to the first stage are statistically independent.

4.1 Mathematical Model

In this section, we provide the mathematical perspective of the error evaluation methodology described above. As depicted in Figure 3, consider a multi-stage adder with operand bits $A_0, A_1, ..., A_{N-1}$ and $B_0, B_1, ..., B_{N-1}$ and the carry bit C_{in}, with corresponding probabilities $P(A_i), P(B_i), P(C_{in})$ where $i = 0, 1, .., N-1$ for an N-bit adder. As the proposed method is based on a recursive approach, the probability of success for the carry-out bit evaluated in the current stage is utilized as the probability of success for the carry-in bit in the next stage. In our mathematical analysis, we deal with the carry bit to be "1" or "0" separately as the corresponding mathematical terms behave differently in LPAAs, as can be seen from Table 1. The rest of the mathematical treatment of the proposed methodology is going to be described using the following terminology:

- $P(C_{curr}) = P(C_{in})$ to be "1" at the current stage
- $P(\overline{C_{curr}}) = P(C_{in})$ to be "0" at the current stage
- $P(C_{next}) = P(C_{out})$ to be "1" at the current stage
- $P(\overline{C_{next}}) = P(C_{out})$ to be "0" at the current stage
- $P(Succ)$ = Probability of Success
- $P(\overline{Succ})$ = Probability of No Success

Using the values at any current stage, we need to recursively evaluate the probability of carry-out to be "1" or "0" for accurate cases only, i.e., $P(\overline{C_{next}} \cap Succ)$ and $P(C_{next} \cap Succ)$ for every stage. The equations for evaluation of these probabilities can be obtained for LPAAs using Table1. For instance, the following equations are for evaluating the carry-out probabilities using current stage parameters for LPAA 1.

$$\begin{aligned} P(\overline{C_{next}} \cap Succ) = P(\overline{A_i} \cap \overline{B_i} \cap (\overline{C_{curr}} \cap Succ)) + \\ P(\overline{A_i} \cap \overline{B_i} \cap (C_{curr} \cap Succ)) \end{aligned} \tag{1}$$

$$\begin{aligned} P(C_{next} \cap Succ) = P(\overline{A_i} \cap B_i \cap (C_{curr} \cap Succ)) + \\ P(A_i \cap \overline{B_i} \cap (C_{curr} \cap Succ)) + \\ P(A_i \cap B_i \cap (\overline{C_{curr}} \cap Succ)) + \\ P(A_i \cap B_i \cap (C_{curr} \cap Succ)) \end{aligned} \tag{2}$$

As, all input probabilities are assumed to be independent, Equations 1 and 2 can be re-written as

$$\begin{aligned} P(\overline{C_{next}} \cap Succ) = P(\overline{A_i}).P(\overline{B_i}).P(\overline{C_{curr}} \cap Succ) + \\ P(\overline{A_i}).P(\overline{B_i}).P(C_{curr} \cap Succ) \end{aligned} \tag{3}$$

$$\begin{aligned} P(C_{next} \cap Succ) = P(\overline{A_i}).P(B_i).P(C_{curr} \cap Succ) + \\ P(A_i).P(\overline{B_i}).P(C_{curr} \cap Succ) + \\ P(A_i).P(B_i).P(\overline{C_{curr}} \cap Succ) + \\ P(A_i).P(B_i).P(C_{curr} \cap Succ) \end{aligned} \tag{4}$$

For the first stage,

$$\begin{aligned} P(C_{curr} \cap Succ) = P(C_{in}) \\ P(\overline{C_{curr}} \cap Succ) = 1 - P(C_{in}) \end{aligned} \tag{5}$$

While for stages $2, 3, .., N-1$,

$$\begin{aligned} P(C_{curr} \cap Succ)_{curr\_stage} = P(C_{next} \cap Succ)_{prev\_stage} \\ P(\overline{C_{curr}} \cap Succ)_{curr\_stage} = P(\overline{C_{next}} \cap Succ)_{prev\_stage} \end{aligned} \tag{6}$$

Although, probability of success or error can be evaluated at any stage, however, we are mostly interested to analyze it for the complete multi-bit adder involving all the stages. Probability of success for an N-bit adder can be obtained after evaluating the probability of $N-1^{th}$ stage carry-output bearing accurate cases only. Using the probabilities $P(C_{next} \cap Succ)_{N-1}$ and $P(\overline{C_{next}} \cap Succ)_{N-1}$, evaluated using Equations 3 and 4 recursively, the probability of success for N-bit LPAA 1 can be evaluated as:

$$\begin{aligned} P(Succ) = P(\overline{A_N} \cap \overline{B_N} \cap (\overline{C_{next}} \cap Succ)_{N-1}) + \\ P(\overline{A_N} \cap \overline{B_N} \cap (C_{next} \cap Succ)_{N-1}) + \\ P(\overline{A_N} \cap B_N \cap (C_{next} \cap Succ)_{N-1}) + \\ P(A_N \cap \overline{B_N} \cap (C_{next} \cap Succ)_{N-1}) + \\ P(A_N \cap B_N \cap (\overline{C_{next}} \cap Succ)_{N-1}) + \\ P(A_N \cap B_N \cap (C_{next} \cap Succ)_{N-1}) \end{aligned} \tag{7}$$

While using the independence of inputs, Equation 7 can be re-written as:

$$\begin{aligned} P(Succ) = P(\overline{A_N}).P(\overline{B_N}).P(\overline{C_{next}} \cap Succ)_{N-1} + \\ P(\overline{A_N}).P(\overline{B_N}).P(C_{next} \cap Succ)_{N-1}) + \\ P(\overline{A_N}).P(B_N).P(C_{next} \cap Succ)_{N-1} + \\ P(A_N).P(\overline{B_N}).P(C_{next} \cap Succ)_{N-1} + \\ P(A_N).P(B_N).P(\overline{C_{next}} \cap Succ)_{N-1} + \\ P(A_N).P(B_N).P(C_{next} \cap Succ)_{N-1} \end{aligned} \tag{8}$$

Now the probability of error can be evaluated using the probability of success as:

$$P(Error) = 1 - P(Succ) \tag{9}$$

Error probability equations for any multi-bit LPAA can be derived by following the above-mentioned steps as well.
The proposed methodology is applied on a 4-bit multistage LPAA 1 for variable probabilities of input operand bits and carry-in bit as shown in Table 4. The evaluated carry-out probability of each stage, as shown in table, is used in the next stage. Probability of success is not required (NR) for all stages except the last one as the final/overall success of multi-bit adder is estimated in the last stage only. Moreover, carry-out probability of the last stage is also not required for probability of success calculation as described in the proposed methodology.

Table 4: An Example of Error Analysis for a 4-bit Multistage LPAA 1 using Proposed Methodology

Stage (i)	**0**	**1**	**2**	**3**
$P(A_i)$	0.9	0.5	0.4	0.8
$P(B_i)$	0.8	0.7	0.6	0.9
$P(\overline{C_{curr}} \cap Succ)$	0.5	0.02	0.1305	0.2064
$P(C_{curr} \cap Succ)$	0.5	0.85	0.7295	0.58574
$P(\overline{C_{next}} \cap Succ)$	0.02	0.1305	0.2064	NR
$P(C_{next} \cap Succ)$	0.85	0.7295	0.58574	NR
$P(Succ)$	NR	NR	NR	**0.738476**

4.2 Matrices Based Generalized Approach

It can be observed from Equations 3, 4 and 8, derived for LPAA 1 using its truth table, that certain terms, based on the truth table of particular LPAA, are added up where each term represents a particular case of the truth table. To facilitate the error analysis and make it generic for any LPAA, we redefine the above mentioned method using matrix theory. This step would allow us to do the error analysis of any LPAA by simply swapping three matrices obtained from the truth table of the specified LPAA. The following steps describe this approach, which is depicted using Pseudo code in Algorithm 1:

1. We start by forming a 1x8 matrix $M = [m_1, m_2, ..., m_8]$, such that $m_i = 1$ if C_{out} is "1" *AND* the case is a *Success* else $m_i = 0$ to discard the failure cases for all possible 8 cases of the truth table.
2. In a similar way, we declare another 1x8 matrix $K = [k_1, k_2, ..., k_8]$, such that $k_i = 1$ if C_{out} is "0" *AND* the case is a *Success* else $k_i = 0$.

Table 5: *M, K* and *L* Matrices Required for Analysis of LPAA 1-7 Proposed in [1] [7]

LPAA Type	**M Matrix**	**K Matrix**	**L Matrix**
LPAA 1	[0,0,0,1,0,1,1,1]	[1,1,0,0,0,0,0,0]	[1,1,0,1,0,1,1,1]
LPAA 2	[0,0,0,1,0,1,1,0]	[0,1,1,0,1,0,0,0]	[0,1,1,1,1,1,1,0]
LPAA 3	[0,0,0,1,0,1,1,0]	[0,1,0,0,1,0,0,0]	[0,1,0,1,1,1,1,0]
LPAA 4	[0,0,0,0,0,1,1,1]	[1,1,0,0,0,0,0,0]	[1,1,0,0,0,1,1,1]
LPAA 5	[0,0,0,0,0,1,0,1]	[1,0,1,0,0,0,0,0]	[1,0,1,0,0,1,0,1]
LPAA 6	[0,0,0,1,0,1,0,1]	[1,0,1,0,1,0,0,0]	[1,0,1,1,1,1,0,1]
LPAA 7	[0,0,0,0,0,0,1,1]	[1,1,1,0,1,0,0,0]	[1,1,1,0,1,0,1,1]

3. Similarly, we obtain a third matrix $L = [l_1, l_2, ..., l_8]$, such that $l_i = 1$ if the case is a *Success* else $l_i = 0$. These three matrices will remain constant irrespective of the number of bits of a particular LPAA. *M, K* and *L* matrices for different LPAA topologies are given in Table 5.
4. The next step is to evaluate probabilities of occurrence of all 8 possible outcomes of the truth table and use them to form the 1x8 *Input Probability Matrix (IPM)* as follow:

$$
\begin{aligned}
IPM = [&P(\overline{A_i}).P(\overline{B_i}).P(\overline{C_{curr}} \cap Succ),\\
&P(\overline{A_i}).P(\overline{B_i}).P(C_{curr} \cap Succ),\\
&P(\overline{A_i}).P(B_i).P(\overline{C_{curr}} \cap Succ),\\
&P(\overline{A_i}).P(B_i).P(C_{curr} \cap Succ),\\
&P(A_i).P(\overline{B_i}).P(\overline{C_{curr}} \cap Succ),\\
&P(A_i).P(\overline{B_i}).P(C_{curr} \cap Succ),\\
&P(A_i).P(B_i).P(\overline{C_{curr}} \cap Succ),\\
&P(A_i).P(B_i).P(C_{curr} \cap Succ)]
\end{aligned}
\tag{10}
$$

In the evaluation of the IPM matrix, we would use Equation 5 to obtain the probabilities for the first stage.

5. Now, we can simply use the dot product between M, K and IPM matrices to obtain the carry out probability:

$$
\begin{aligned}
P(C_{next} \cap Succ) &= [IPM].[M]\\
P(\overline{C_{next}} \cap Succ) &= [IPM].[K]
\end{aligned}
\tag{11}
$$

It can be noted here that, in case of LPAA 1, results of Equation 11 after expansion of dot product are equivalent to Equation 3 and 4 described in last section. However, now it is more generalized to use with multiple LPAAs.

6. Steps 4 and 5 are repeated $N-1$ times in case of an N-bit adder. For every iteration, we use the evaluated carry probabilities from current stage to evaluate the ones for the next stage as explained while describing Equation 6.
7. After $N-1$ iterations, the probability of success and error can be evaluated by dot product as follow:

$$
\begin{aligned}
P(Succ) &= [IPM].[L]\\
P(Error) &= 1 - P(Succ)
\end{aligned}
\tag{12}
$$

The probability of output sum bits can also be evaluated using a similar matrices based approach.

Algorithm 1 Pseudo-code for Matrices Based Generalized Method

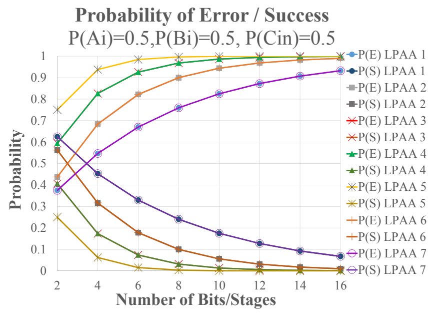
```
Input:
    $i$: $i^{th}$ bit of N-bit adder
    $P(A_i)$: Probability of operand A bits ($\vee i = 0, 1, .., N-1$)
    $P(B_i)$: Probability of operand B bits ($\vee i = 0, 1, .., N-1$)
    $P(C_{in})$: Probability of input carry bit
    $[M]$: Matrix as given in Table 5
    $[K]$: Matrix as given in Table 5
    $[L]$: Matrix as given in Table 5
Initialize:
    $i$: 0
    $P(C_{curr} \cap Succ) : P(C_{in})$
    $P(\overline{C_{curr}} \cap Succ) : 1 - P(C_{in})$
1:  while $i \leq N-1$ do
2:      $\overline{P(A_i)} = 1 - P(A_i)$
3:      $\overline{P(B_i)} = 1 - P(B_i)$
4:      Evaluate $IPM$ as given in Equation 10
5:      $P(C_{next} \cap Succ) = [IPM].[M]$
6:      $P(\overline{C_{next}} \cap Succ) = [IPM].[K]$
7:      $P(C_{curr} \cap Succ) = P(C_{next} \cap Succ)$
8:      $P(\overline{C_{curr}} \cap Succ) = P(\overline{C_{next}} \cap Succ)$
9:  end while
10: $P(Succ) = [IPM].[L]$
11: $(Error) = 1 - P(Succ)$
```

5. RESULTS AND DISCUSSION

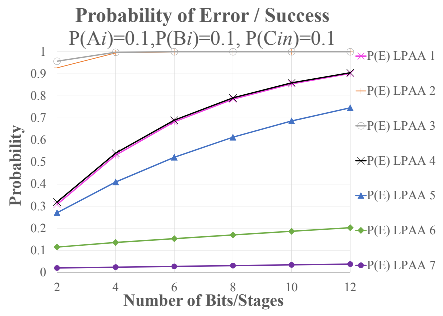
The proposed method was extensively validated using exhaustive simulation results for all proposed LPAAs using National Instruments® LabVIEW for equally and in-equally probable operand input bits scenarios as shown in Table 6. LabVIEW was chosen because of its rapid designing, prototyping and deployment concept which not only helps in testing the algorithms but also gives an interactive and user handy GUI which makes our online library easy to use with no requirement of code understanding. Figure 5(a) shows the probability of success and error values estimated using our proposed method considering all inputs as equally probable. As finite number of cases exist for equally probable inputs, so we were able to test every case via exhaustive simulations using $2^{2N}.2$ cases in total for N-bit un-symmetrical adders. We found ***a 100 percent match*** (up to any decimal precision) of simulated results with results of our method. Moreover, we also experimented with having different probabilities for each input operand bit as well as carry-in bit. For exhaustive simulations, input bits were generated assuming a uniform distribution. Due to the inability to exhaustively test all cases in this scenario, owing to computational limitations, we tested 1 Million cases to average out error probabilities for one input probability and the number of bits. Table 7 shows a comparison of the results obtained from the proposed method with exhaustive simulation for all input probabilities equal to 0.1. It can be clearly observed that the results *match till the* 3^{rd} *decimal place* and a further precision match can be obtained if the *number of test cases in simulation are increased beyond 1 Million.* We found these results quite encouraging because of their precise equivalence with the simulated results, which supports the soundness of the proposed method for evaluating the error of LPAAs. The main benefit of the analytically derived generic error equations from the proposed method is that they can be instantiated to obtain the error for any given value of the input probabilities and can be equally good for handling any stage adder without any significant computational requirement. Table 8 shows the total number of resources required for equally probable and varying input probability scenarios. Total number of iterations are equal to the number of bits in a multistage adder. When the operand bits bear identical probabilities, we can reduce the repeated multiplications as illustrated in Table 8. The execution time is approximately less than 1ms for any length of multistage adder being analyzed, which is insignificant as compared to simulation time as shown in Figure 1. As clearly indicated from Tables 3 and 8, these computations are *far less and linearly scalable* as compared to state-of-the-art traditional analysis technique using principle of inclusion-exclusion. Every adder exhibits different performance for a specific input probability value. In order to categorize each adder performance for low and high input probabilities, we analyzed them using our proposed methodology as shown in Figure 5(b,c). For example, even LPAA 7 and LPAA 1 exhibit the same performance when input operands are equally probable, Figure 5(b,c) depicts that the performance of LPAA 7 is better for low probabilities of input operand bits while LPAA 1 works better for high probability of input bits. As in MSBs, we tend to have more 0's, therefore LPAA 7 can be used in such cases. While for the bits expecting high probability of 1's, LPAA 1 seems to be a more preferable choice. Also, it is obvious from Figure 5(a,b,c) that LPAA 6 works optimally better for low, high and equally probable inputs, so it can be considered as a *"Four Season Adder"* as it can provide a good performance for any probability value. Similar results can be obtained for multiple input bit probability configurations and low power adders to optimally design a *hybrid multistage low power adder* using more than one type of LPAA. The designed hybrid adder performance can again be evaluated using our proposed method. However for equal probabilities of input operands, Figure 5(a)

Table 6: Accuracy Match of Proposed Method with Exhaustive Simulations

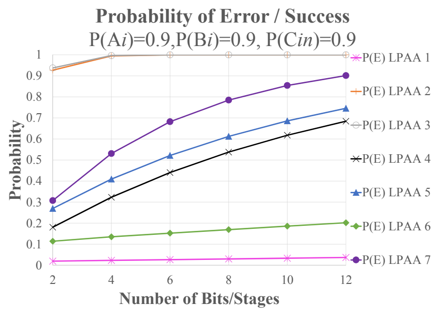
Accuracy Match	**Input Probabilities**	**Test Cases**	**No. of Simulation Cases**
Precisely up to any decimal place	Equally Probable	Finite	2^{2N+1}
Up to 3^{rd} decimal place	Not Equally Probable	Infinite	1 Million (can be increased for better precision match)



(a)



(b)



(c)

Figure 5: Probability of Success and Error for LPAAs (see their truth tables in Table 1). (a) Input Operands Equally Probable (b) Low Probability of Input Operands (c) High Probability of Input Operands

Table 7: Our Proposed Analytical Method vs Exhaustive Simulation Results ($A_i = B_i = C_{in} = 0.1$)

Number of	P(E) LPAA 1		P(E) LPAA 2		P(E) LPAA 3		P(E) LPAA 4		P(E) LPAA 5		P(E) LPAA 6		P(E) LPAA 7	
Bits/Stages	Analyt.	Sim.	Analyt.	Sim.	Analyt.	Sim.	Analyt.	Sim.	Analyt.	Sim.	Analyt.	Sim.	Analyt.	Sim.
2	0.30780	0.30746	0.9271	0.92740	0.95707	0.95701	0.31851	0.31900	0.27000	0.26999	0.1143	0.11483	0.01980	0.01960
4	0.53090	0.53151	0.99468	0.99460	0.99763	0.99762	0.54033	0.54036	0.40950	0.40808	0.13533	0.13557	0.02333	0.02329
6	0.68240	0.68172	0.99961	0.99961	0.99986	0.99987	0.68999	0.69022	0.52170	0.52258	0.15266	0.15279	0.02685	0.02710
8	0.78498	0.78501	0.99997	0.99998	0.99999	0.99999	0.79092	0.79070	0.61258	0.61194	0.16953	0.16939	0.03035	0.03029
10	0.85443	0.85405	0.99999	0.99999	0.99999	0.99999	0.85899	0.85906	0.68618	0.68635	0.18605	0.18617	0.03385	0.03386
12	0.90145	0.90159	0.99999	0.99999	0.99999	0.99999	0.90490	0.90531	0.74581	0.74552	0.20225	0.20221	0.03733	0.03763

Table 8: Resource Utilization of Proposed Method

Operand Bits Identical and Equally Probable		**Operand Bits with Different Probabilities**	
Multipliers	32	Multipliers	48
Adders	21	Adders	21
Memory Units	3	Memory Units	No. of bits +1

clearly shows that none of the LPAA is useful beyond 10-bits cascading because of very high probability of error in results.

6. CONCLUSION

Statistical error analysis allows us to quantize performance of an approximate adder before its utility for a particular application. However, it is not scalable to derive a closed form mathematical expression for large-bit LPAAs using existing analytical techniques, based on the principle of inclusion-exclusion due to the generation of a large number of terms with an increase in the number of adder stages. To overcome these limitations, we have presented a generic matrices based recursive approach for error analysis which can be used for the error evaluation of any multi-bit approximate adder. We have applied our proposed approach on multi-bit LPAAs and discussed the accuracy of results. Using our analysis approach, we also presented an optimal design of a multistage hybrid adder with different input probabilities based on more than one type of LPAAs. We have also open-sourced LabVIEW and MATLAB libraries of our methodology for its rapid adoption in further research and development.

7. REFERENCES

[1] H. A. Almurib, T. Kumar, and F. Lombardi. Inexact Designs for Approximate Low Power Addition by Cell Replacement. In *Design, Automation & Test in Europe*, pages 660–665. IEEE, 2016.

[2] J. Bornholt, T. Mytkowicz, and K. McKinley. Uncertain< t>: Abstractions For Uncertain Hardware and Software. *IEEE Micro*, pages 132–143, 2015.

[3] V. Chippa, S. Chakradhar, K. Roy, and A. Raghunathan. Analysis and Characterization of Inherent Application Resilience for Approximate Computing. In *Design Automation Conference*, pages 113:1–113:9. ACM, 2013.

[4] V. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. Chakradhar. Scalable Effort Hardware Design: Exploiting Algorithmic Resilience for Energy Efficiency. In *Design Automation Conference*, pages 555–560, 2010.

[5] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger. Architecture Support for Disciplined Approximate Programming. In *ACM SIGPLAN Notices*, pages 301–312, 2012.

[6] V. Gupta, D. Mohapatra, S. Park, A. Raghunathan, and K. Roy. Impact: IMprecise Adders for Low-Power Approximate Computing. In *Symposium on Low-power electronics and design*, pages 409–414, 2011.

[7] 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*, pages 124–137, 2013.

[8] J. Han and M. Orshansky. Approximate Computing: An Emerging Paradigm for Energy-Efficient Design. In *IEEE European Test Symposium (ETS)*, pages 1–6, 2013.

[9] R. Hegde and N. Shanbhag. Energy-Efficient Signal Processing via Algorithmic Noise-Tolerance. In *Symposium on Low power electronics and design*, pages 30–35, 1999.

[10] A. Kahng and S. Kang. Accuracy-Configurable Adder for Approximate Arithmetic Designs. In *Design Automation Conference*, pages 820–825. ACM, 2012.

[11] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, and J. Henkel. Probabilistic Error Modeling for Approximate Adders. *IEEE Transactions on Computers*, pages 515–530, 2016.

[12] A. Mishra, R. Barik, and S. Paul. iact: A Software-Hardware Framework for Understanding the Scope of Approximate Computing. In *Workshop on Approximate Computing Across the System Stack*, 2014.

[13] D. Mohapatra, G. Karakonstantis, and K. Roy. Significance Driven Computation: A Voltage-Scalable, Variation-Aware, Quality-Tuning Motion Estimator. In *Symposium on Low power electronics and design*, pages 195–200, 2009.

[14] R. Nair. Big Data Needs Approximate Computing: Technical Perspective. *Communications of the ACM*, pages 104–104, 2015.

[15] M. Shafique, R. Ahmad, W.and Hafiz, and J. Henkel. A Low Latency Generic Accuracy Configurable Adder. In *Design Automation Conference*, pages 86:1–86:6, 2015.

[16] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel. Invited-Cross-Layer Approximate Computing: From Logic to Architectures. In *Design Automation Conference*, pages 99:1–99:6, 2016.

[17] A. Verma, P. Brisk, and P. Ienne. Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design. In *Design, automation and test in Europe*, pages 1250–1255, 2008.