POLITECNICO DI TORINO
Repository ISTITUZIONALE

Design and Evaluation of Approximate Logarithmic Multipliers for Low Power Error-Tolerant Applications

(Article begins on next page)

28 April 2024

Design and Evaluation of Approximate Logarithmic Multipliers for Low Power Error-Tolerant Applications

Weiqiang Liu, *Senior Member, IEEE,* Jiahua Xu, Danye Wang, Chenghua Wang, Paolo Montuschi, *Fellow*, *IEEE* and Fabrizio Lombardi, *Fellow, IEEE*

Abstract—In this work, the designs of both non-iterative and iterative approximate logarithmic multipliers (LMs) are studied to further reduce power consumption and improve performance. Non-iterative approximate LMs (ALMs) that use three inexact mantissa adders, are presented. The proposed iterative approximate logarithmic multipliers (IALMs) use a set-one adder in both mantissa adders during an iteration; they also use lower-part-or adders and approximate mirror adders for the final addition. Error analysis and simulation results are also provided; it is found that the proposed approximate LMs with an appropriate number of inexact bits achieve a higher accuracy and lower power consumption than conventional LMs using exact units. Compared with conventional LMs with exact units, the normalized mean error distance (NMED) of 16-bit approximate LMs is decreased by up to 18% and the power-delay product (PDP) has a reduction of up to 37%. The proposed approximate LMs are also compared with previous approximate multipliers; it is found that the proposed approximate LMs are best suitable for applications allowing larger errors, but requiring lower energy consumption and low power. Approximate Booth multipliers fit applications with less stringent power requirements, but also requiring smaller errors. Case studies for error-tolerant computing applications are provided.

Index Terms—Approximate computing, logarithmic multiplier, inexact adder, low Power, error-tolerant applications.

I. INTRODUCTION

COMPUTING systems are encountering substantial difficulties to further improve performance at current power consumption levels; on-chip power consumption has become prohibitively high for efficient computation in today's integrated circuits. Approximate (or inexact) computing [1] is a promising approach to possibly resolve power issues; exactness is not so strict for increasing popular cognitive applications related to human perception, such as multimedia signal processing, machine learning and pattern recognition [2]. Therefore, power consumption and IC performance can be further improved by relaxing the requirement of a strict accuracy; approximate computing generates good enough results that do not require full accuracy and correctness. Approximate techniques have been applied at several levels including circuits, architectures and algorithm/software [3-4]. At circuit level, the design of approximate arithmetic units has received a significant research interest due to its importance in many computing applications; typical applications, such as DSP and machine learning algorithms, have an extensive number of arithmetic processing involving addition (or accumulation) and multiplication.

As basic operations of an arithmetic processor, addition and multiplication are very important for achieving high performance; therefore, they have been extensively studied for approximate computing and reducing power consumption. Error metrics including the error rate (ER), error distance (ED), mean error distance (MED), normalized mean error distance (NMED) [5], and the worst case error (WCE) [6] have been proposed for evaluating the designs of approximate arithmetic circuits.

Approximate adders have been extensively studied in the technical literature to attain reduction in power consumption and delay [7]. Approximate adder designs mainly include speculative adders [8-11] and non-speculative transistor-level full adders [12-13]; an approximate floating-point adder has also been studied [14].

The operation of multiplication is more complex than addition. Approximate design techniques can be applied to different parts of a conventional multiplier, such as operands [15-20], partial product (PP) generation [21-24], PP tree [25-30] and compressors [31-34]. Currently, artificial intelligence related methods have been proposed for approximate multiplication such as those using genetic programming [6]

W. Liu, J. Xu, D. Wang, and C. Wang are with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China (e-mail: {liuweiqiang, xujiahua, wangdanye, chwang}@nuaa.edu.cn).

P. Montuschi is with the Department of Control and Computer Engineering, Politecnico di Torino, Turin, 10129, Italy. (e-mail: paolo.montuschi@polito.it).

F. Lombardi is with the Department of Electrical and Computer Engineering Northeastern University, Boston, MA 02115, USA. (e-mail: lombardi@ece.neu.edu).

[35]. Multipliers using a logarithmic transformation (that converts multiplication into addition) show inherent approximate characteristics. The logarithmic multiplier (LM) has been first proposed by Mitchell [15]; it performs multiplication using shifts and addition only and by converting the operands to approximate logarithmic numbers. As the logarithmic transformation is approximate, an inherent error is unavoidable in the results. However, the area, delay and power are usually improved significantly at the cost of precision. State-of-the-art designs of LMs improve accuracy using either a fine piecewise linear approximation [16-18], or an iterative technique [19, 36].

However, no investigation has been conducted on the design of LMs using inexact parts and in particular, two interesting questions remain unanswered:

- How is the performance of a logarithmic multiplier affected if inexact units are used in the design?
- Are approximate logarithmic multipliers (ALMs) or approximate normal binary multipliers better in terms of different performance metrics?

In this work, inexact adders are used to compute the least significant bits of the mantissa adder in the logarithmic multipliers so that performance and power can be further improved. The iterative technique is also applied in the ALMs to improve the accuracy. Error analysis and evaluation are presented to validate the proposed ALM designs. Case studies for image processing and pattern recognition (*i.e.*, *k*-means clustering) are also investigated.

This paper has been extended significantly from the previous conference version [37]; the main differences are summarized as follows:

- The designs of the proposed ALMs are detailed and further explained;
- The error probability density distributions (PDDs) of ALMs are provided to explain the error analysis results;
- The relationships between M_1, M_2 and M_3 for IALM are studied and discussed;
- Comparison is provided for both 8-bit and 16-bit designs with previous approximate multipliers;
- Two case studies including image processing and *k*-means clustering are presented to show the validity of the proposed designs.

The rest of the paper is organized as follows. Section II reviews conventional non-iterative and iterative logarithmic multipliers. The design of an approximate logarithmic multiplier is presented in Section III. The design of iterative approximate logarithmic multipliers is also presented in this section. Error analysis and simulation results of the approximate logarithmic multipliers are given in Section IV. The proposed approximate LMs are also compared with previous approximate multipliers in this section. The applications of approximate LMs to digital image processing and *k*-means clustering are presented in Section V. Section VI concludes this paper.

II. REVIEW

Both the conventional Mitchell logarithmic multiplier and the iterative logarithmic multiplier are briefly reviewed in this section.

A. Non-Iterative Logarithmic Multiplier

Mitchell's algorithm [15] is referred to as a non-iterative logarithmic multiplier in this work. Assume A and B are two fixed-point operands of the logarithmic multiplier; they can be expressed as follows [15]:

$$A = 2^{k_1}(1 + x_1), 0 \le x_1 < 1 \tag{1}$$

$$B = 2^{k_2}(1 + x_2), 0 \le x_2 < 1 \tag{2}$$

where, the values of k_1 and k_2 are referred to as the characteristics of A and B, respectively, and they are the position of the leading one in its unsigned binary representation, *i.e.*, they represent the most significant operand bits [36]. x_1 and x_2 are the fractional parts of the resultant logarithmic approximation and in the range of $[0, 1)$. The values of the characteristic and the fractional part uniquely determine a binary logarithm of the two operands as follows:

$$log_2(A) = k_1 + log_2(1 + x_1) \tag{3}$$

$$log_2(B) = k_2 + log_2(1 + x_2) \tag{4}$$

The logarithm of a product, *i.e.*, P, is equal to the sum of the logarithms of the multiplier and the multiplicand, so

$$P = A \times B = 2^{k_1 + k_2}(1 + x_1) \times (1 + x_2) \tag{5}$$

$$log_2(P) = k_1 + k_2 + log_2(1 + x_1) + log_2(1 + x_2) \tag{6}$$

As $log_2(1 + x) \approx x$ when $0 \le x < 1$, the approximate product can be calculated as follows:

$$log_2(P) \approx k_1 + k_2 + x_1 + x_2 \tag{7}$$

which can be calculated by adders.

An 8-bit example of Mitchell's algorithm is shown in Fig. 1, where $102 \times 160 \approx 15,140$ (=16,320). Note that Mitchell logarithmic multiplier always underestimates the correct value [15].
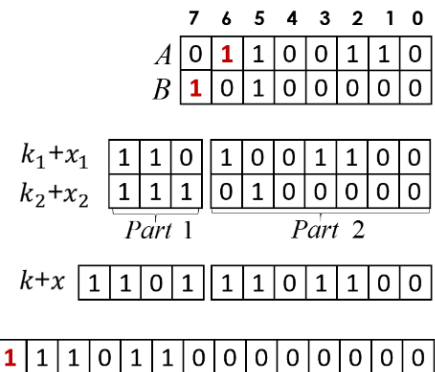

Fig. 1. An 8-bit example of Mitchell's algorithm.

Mitchell logarithmic multiplier mainly includes four steps: leading one detection, binary to logarithm conversion, mantissa addition and logarithm to binary conversion. The structure of a 16-bit Mitchell multiplier is shown in Fig. 2.

The leading one detector (LOD) finds the left-most one bit; then the binary-logarithm converter (BLC) produces the logarithms of the original operand numbers. The multiplication is then performed by the mantissa addition in the logarithmic

domain. The results are then decoded by the logarithm-binary converter (LBC) into binary numbers as final product.
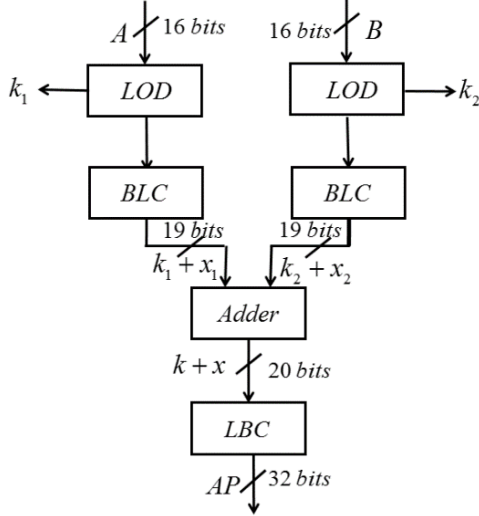

Fig. 2. A 16-bit non-iterative logarithmic multiplier [37].

B. Iterative Logarithmic Multiplier

Iterative approaches have been proposed to improve the accuracy at the cost of duplicated hardware [36]. As mentioned previously, the product of a LM is approximate. Errors are introduced by ignoring the correction parts. The exact product can be rewritten as follows [15]:

$$P = 2^{k_1+k_2}(1 + x_1 + x_2) + 2^{k_1+k_2}(x_1 \times x_2), x_1 + x_2 < 1 \quad (8)$$

$$P = 2^{1+k_1+k_2}(x_1 + x_2) + 2^{k_1+k_2}(1 - x_1) \times (1 - x_2), x_1 + x_2 \geq 1 \quad (9)$$

where, $2^{k_1+k_2}(x_1 \times x_2)$ and $2^{k_1+k_2}(1 - x_1) \times (1 - x_2)$ are the correction parts. Let U and V be defined as follows:

$$U = 2^{k_1}x_1, V = 2^{k_2}x_2, x_1 + x_2 < 1 \quad (10)$$

$$U = 2^{k_1}(1 - x_1), V = 2^{k_2}(1 - x_2), \ x_1 + x_2 \geq 1 \quad (11)$$

The correction parts can be calculated as follows:

$$P_L = U \times V \quad (12)$$

which can be calculated using the same LM design as shown in Fig. 2. P_L is then added to the original LM product (*i.e.*, $P_H = A \times B$) to increase the accuracy of the final product.

III. APPROXIMATE LOGARITHMIC MULTIPLIERS

The performance and power consumption of logarithmic multipliers are further improved by using inexact units during the mantissa addition; three inexact adders, namely, lower-part-or adder (LOA) [12], approximate mirror adder-A3 (MAA3) [13] and set-one adder (SOA) are used. The SOA is proposed with truncated BLC in this work for approximate LMs. Both non-iterative and iterative approximate LMs are then proposed based on these inexact adders.

LOA and MAA3 are selected due to their performance when considering both accuracy and hardware metrics as reported in [7]. SOA is proposed in this work to achieve a symmetric error distribution. These adders have very small complexity, which are best suitable for implementation of low power approximate logarithmic multipliers.

A. Non-Iterative Approximate Logarithmic Multipliers

1) ALM Using LOA (ALM-LOA)

ALM-LOA uses the lower-part-or adder (LOA) [12] in the mantissa adder. An n-bit LOA consists of two parts, *i.e.*, an *m*-bit inexact adder and an (*n-m*)-bit exact adder (EA) (Fig. 3). The (*n-m*)-bit adder is used for the (*n-m*) most significant bits of the sum, while the *m*-bit adder consists of OR gates to compute the addition of the least significant *m* bits (*i.e.*, the lower *m*-bit adder is an array of *m* 2-input OR gates). The logic expressions for the lower *m*-bit adder are as follows:

$$Sum[m - 1{:}0] = a[m - 1{:}0] + b[m - 1{:}0] \quad (13)$$

$$Cin = a[m - 1]b[m - 1] \quad (14)$$

As the inexact adder only uses an OR-gate, its complexity and critical path are reduced significantly, leading to significantly lower power consumption. This is achieved at the cost of a lower accuracy. The ER, MED and WCE are as follows:

$$ER = 1 - \left(\frac{3}{4}\right)^m \quad (15)$$

$$MED = 3 \times 2^{m-4} - \frac{1}{8} \quad (16)$$

$$WCE = 2^{m-1} \quad (17)$$

The LOD and BLC are very important computational steps to guarantee the correctness of the converted logarithmic operands. Therefore, accurate LOD and BLC designs are used in ALM-LOA to avoid large errors. The error characteristics of the approximate LMs are further studied in the next section.
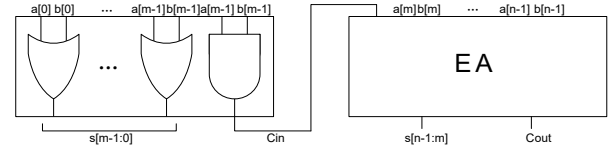

Fig. 3. LOA with *m* inexact bits (modified from [37]).

2) ALM Using MAA3 (ALM-MAA3)

ALM-MAA3 uses the approximate mirror adder (MAA3) [13] in the mantissa adder. An *n*-bit MAA3 also consists of two parts, *i.e.*, an *m*-bit inexact adder and an (*n-m*)-bit exact adder (Fig. 4). The (*n-m*)-bit adder is used for the (*n-m*) most significant bits of the sum; while the sum of the *m*-bit adder is actually one of its input. The logic expressions for the lower *m*-bit adder are as follows:
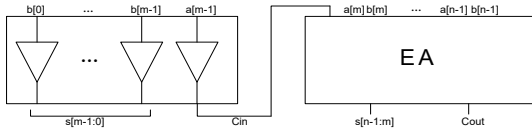
$$Sum[m - 1{:}0] = b[m - 1{:}0] \quad (18)$$

$$Cin = a[m - 1] \quad (19)$$

For the same reason as mentioned above, accurate LOD and BLC designs are used in ALM-MAA3. Its complexity and critical path are reduced significantly and smaller than ALM-LOA. This also leads to lower power consumption. The ER, MED and WCE of LM-MAA3 are as follows:

$$ER = 1 - \left(\frac{1}{2}\right)^m \quad (20)$$

$$MED = 2^{m-2} \quad (21)$$

$$WCE = 2^{m-1} \quad (22)$$

Fig. 4. MAA3 with m inexact bits (modified from [37]).

3) ALM Using SOA (ALM-SOA)

ALM-SOA uses both the set-one adder as mantissa adder and a truncated BLC (TBLC). An n-bit SOA consists of two parts, $i.e.$, an m-bit inexact adder and an $(n-m)$-bit exact adder (Fig. 5). The sum of the m-bit adder is set to logic one. The logic expressions for the lower m-bit adder are as follows:

$$Sum[m-1:0] = 1 \tag{23}$$

$$Cin = a[m-1]b[m-1] \tag{24}$$

The SOA is first proposed for the approximate LM in this paper. As the product from LM is always smaller than its exact counterpart, SOA is employed to compensate the negative errors. The ER, MED and WCE of LM-SOA are as follows:

$$ER = 1 - \left(\frac{1}{2}\right)^m \tag{25}$$

$$MED = \frac{2^m}{3} - \frac{1}{3 \times 2^m} \tag{26}$$

$$WCE = 2^{m-1} \tag{27}$$

As the lower significant sum bits of SOA are set to one already, it is not necessary to perform an exact binary-logarithm conversion; therefore, TBLC is used in the ALM-SOA to further reduce its complexity. The number of truncated bits is the same as the number of inexact bits in SOA, $i.e.$, m. However, for LOA and MAA3 based ALMs, as the lower significant bits are still processed with approximate adders, a truncated BLC cannot be used.
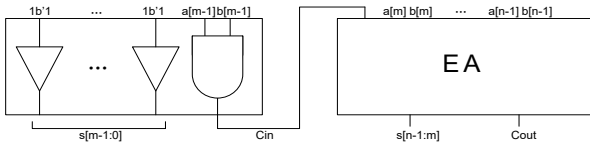
Fig. 5. SOA with m inexact bits (modified from [37]).

An example of a 16-bit ALM using SOA is given in Fig. 6, in which the operation is given by $12,237 \times 1,597 \approx 17,825,280$ ($=19,542,489$). The number of inexact bits of this multiplier are 11 out of the 32 bits of a conventional multiplier, $i.e.$, $M=11$. As shown in Fig. 6, the location of the leading one is found and then it is considered as the integer part of the logarithmic number, so the remaining part of the original binary number is regarded as the fractional part. In ALM-SOA, the fractional part is truncated and the sum is set to one directly so ultimately reducing the power consumption.

Fig. 6. A 16-bit example of ALM-SOA ($M=11$) with $A=12,237$ and $B=1,597$. AP: approximate product. EP: exact product.

B. Iterative Approximate LMs (IALMs)

As shown in the last section, the accuracy of ALM is affected by applying different number of inexact bits in the mantissa adder to achieve an even more efficient design compared with conventional LMs. However, the error of ALMs in general, is large. In this section, three iterative ALMs (IALMs), namely, IALM-S (IALM using SOA and Exact Adder3), IALM-SL (IALM using SOA and LOA) and IALM-SM (IALM using SOA and MAA3), are proposed to improve the accuracy of ALM. The proposed design of an iterative logarithmic multiplier is shown in Fig. 7, where Adder1, Adder2 and Adder3 can be designed approximately. For all IALMs, the numbers of inexact bits of Adder1, Adder2 and Adder3 are denoted as M_1, M_2 and M_3, respectively. As Adder2 is used for the correction parts, M_2 can be generally larger than M_1 as the corresponding bits of the original product are already approximate. The relationship between these three Ms is further studied in the next section. In Fig. 7, the Estimator is an approximate unit that is used to record the carry-in bit from Adder1 and to choose the correct correction part. If $c_{in}=0$, x_1 and x_2 can be used. If $c_{in}=1$, the one's complement of x_1 and x_2 are used instead. The inexact parts include BLC, Adder1, Adder2, Adder3 and Estimator, which are highlighted as shaded blocks (units) in Fig. 7.
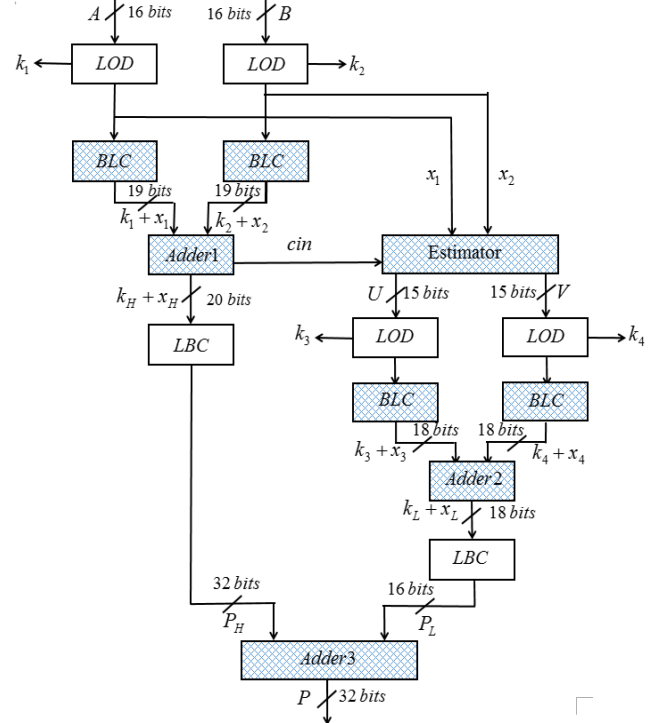
Fig. 7. The proposed 16-bit IALM [37].

1) IALM Using SOA and Exact Adder3 (IALM-S)

IALM-SL is shown in Fig. 7. As SOA has the least complexity compared with the other two inexact adders, both exact Adder1 and Adder2 are replaced with SOA with M_1 and M_2, respectively. TBLCs are used for both binary to logarithm conversions before Adder1 and Adder2. The numbers of the truncated bits of TBLCs are also M_1 and M_2, respectively. As the error is sensitive to the operation of Adder3, an exact Adder3 is used in IALM-S ($M_3=0$ in this case).

2) IALM Using SOA and LOA (IALM-SL)

IALM-SL is similar to IALM-S, SOAs and TBLCs are used for Adder1 and Adder2. The numbers of truncated bits of TBLCs are the same as the number of inexact bits of SOA. The difference is in Adder3; in this case, it uses LOA with different M_3; so as Adder3 is sensitive to errors, SOA is not utilized.

3) IALM Using SOA and MAA3 (IALM-SM)

In IALM-SM, Adder1 and Adder2 are the same as IALM-SL, where they are replaced with SOAs and TBLCs. TBLCs are used for the binary to logarithm conversion. However, Adder3 uses a MAA3 instead of LOA. SOA is also not utilized in Adder3 due to its low accuracy. The accuracy and power consumption of both IALM-SL and IALM-SM are mostly dependent on M_1, M_2 and M_3. The tradeoff between error and the power-delay product is further studied in the next section.

IV. EVALUATION AND ANALYSIS

A. Error Analysis

For approximate designs, several metrics have been proposed to measure the error of approximate adders and multipliers including the mean error distance (MED), the relative error distance (RED), the normalization of MED (NMED) [5] and the worst case error (WCE) [6]. NMED and RED are defined as follows.

- The NMED is defined as the normalized MED by the maximum output of the accurate design.
- RED is defined as the ED over the absolute accurate result.

TABLE I
ERROR CHARACTERISTICS OF 8-BIT NON-ITERATIVE ALMS WITH DIFFERENT MS USING BOTH EXHAUSTIVE AND MONTE CARLO SIMULATION

Design	M	Exhaustive Simulation					Monte Carlo Simulation				
		NMED (10^{-2})	MRED (10^{-2})	P_{RED} (%)	ER (%)	WCE	NMED (10^{-2})	MRED (10^{-2})	P_{RED} (%)	ER (%)	WCE
LM	0	0.93	3.76	36.59	93.09	4096	0.93	3.77	36.47	92.23	4096
	1	0.91	3.73	37.00	93.09	4096	0.91	3.74	36.25	93.35	4020
	2	0.90	3.68	37.94	93.09	4225	0.90	3.67	37.27	92.55	4029
ALM-LOA	3	0.92	3.68	38.16	93.09	4605	0.90	3.63	38.28	93.03	4410
	4	1.01	3.91	32.23	93.09	5369	1.01	3.93	32.35	92.95	5241
	5	1.32	4.94	24.40	93.09	6897	1.34	5.00	23.78	93.38	6897
	6	2.16	7.88	16.26	93.09	9953	2.18	7.94	16.35	93.03	9762
	7	4.11	14.42	13.47	93.09	16320	4.09	14.47	13.27	92.81	16198
	1	0.88	3.65	38.21	93.87	4096	0.87	3.62	38.52	93.80	3968
	2	0.87	3.58	39.64	94.65	4288	0.88	3.60	39.57	95.10	4232
ALM-MAA3	3	0.90	3.61	39.24	95.24	4672	0.91	3.62	39.17	95.43	4584
	4	1.01	3.96	31.96	95.63	5440	1.02	3.99	32.17	95.87	5223
	5	1.33	5.21	23.33	95.88	6976	1.32	5.21	23.20	95.93	6706
	6	2.22	8.77	13.78	96.02	10048	2.22	8.79	13.88	96.11	10015
	7	4.15	16.79	8.17	96.11	16320	4.09	16.50	8.42	96.32	15939
	1	0.87	3.50	40.54	96.61	4032	0.85	3.50	40.77	96.27	4026
	2	0.81	3.23	45.54	97.43	4223	0.81	3.20	46.33	97.10	4223
ALM-SOA	3	0.78	3.06	44.42	98.06	4605	0.79	3.09	44.28	98.27	4573
	4	0.85	3.44	35.62	98.42	5369	0.86	3.42	35.87	98.48	5369
	5	1.31	5.53	22.43	98.80	7680	1.33	5.44	22.68	98.78	7680
	6	2.61	11.16	11.89	98.96	15104	2.61	11.31	11.68	98.82	14880
	7	5.46	23.28	6.35	99.01	28416	5.49	23.44	6.27	99.22	28032

Mean RED (MRED) and P_{RED} are usually used to evaluate the error distribution of approximate multipliers. P_{RED} is the probability of obtaining a RED smaller than a specific percentage value, which is 2% in this work.

The error results of 8-bit ALMs are obtained by exhaustive simulation. However, an exhaustive simulation for 16-bit multipliers is infeasible and a Monte Carlo simulation with an uniform input distribution is usually utilized. As for the effectiveness of the Monte Carlo method, 8-bit ALMs using 6,000 input vectors (out of 65,536 all possible inputs) are considered for comparison with exhaustive simulation. As shown in Table I, the relative errors between exhaustive simulation and Monte Carlo simulation are less than 5%. The NMEDs have been obtained by simulation using both of these two methods and are shown in Figs. 8-10. Monte Carlo simulation generates results that are very close to exhaustive simulation; therefore, the error analysis of 16-bit multipliers is pursued by using Monte Carlo simulation (400,000,000 simulation runs out of 4,294,967,296).
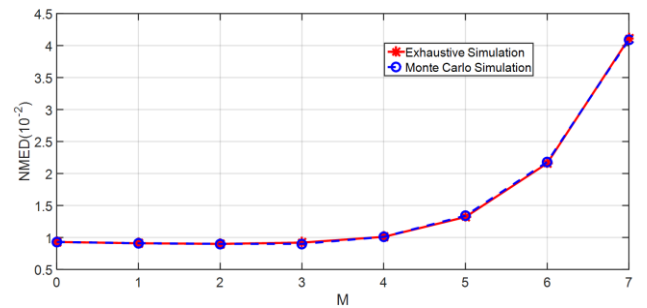
Fig. 8. NMED of 8-bit ALM-LOAs from both exhaustive and Monte Carlo simulations.
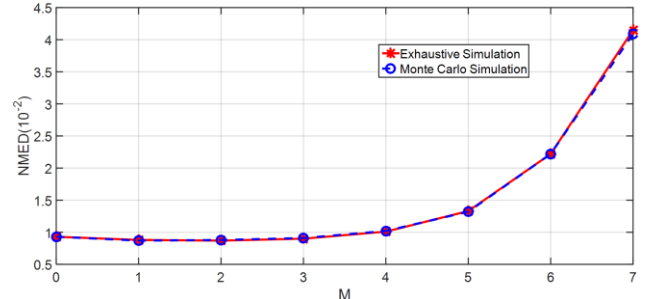
Fig. 9. NMED of 8-bit ALM-MAA3s from both exhaustive and Monte Carlo simulations.
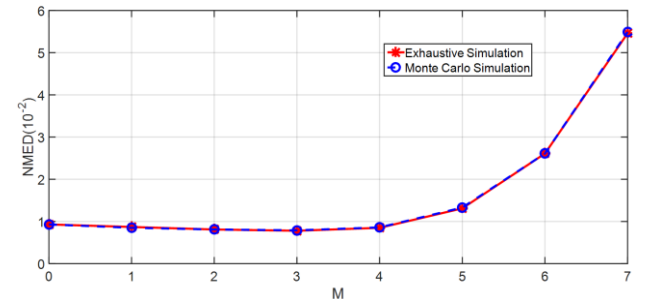
Fig. 10. NMED of 8-bit ALM-SOAs from both exhaustive and Monte Carlo simulations.

1) Error Analysis of ALMs

The error metrics of the proposed non-iterative approximate logarithmic multipliers are provided in Table II for 8-bit and 16-bit designs. Table II shows that with an increase of the number of inexact bits (*i.e.*, M) in 8-bit designs, the accuracy of all three ALMs generally decreases (*i.e.*, NMED, MRED, ER and WCE increase, while P_{RED} decreases). However, when M<4

for 8-bit designs, the errors of all three designs are smaller than the LM using exact units, as also found in Fig. 11 for the NMED of 8-bit ALMs with M from 1 to 5.

TABLE II
ERROR CHARACTERISTICS OF 8-BIT AND 16-BIT NON-ITERATIVE APPROXIMATE LOGARITHMIC MULTIPLIERS WITH DIFFERENT NUMBER OF INEXACT BITS (M)

LMs	M	8-bit LMs					16-bit LMs				
		NMED (10^{-2})	MRED (10^{-2})	P_{RED} (%)	ER (%)	WCE	NMED (10^{-3})	MRED (10^{-2})	P_{RED} (%)	ER (%)	WCE (10^6)
LM	0	0.93	3.76	36.59	93.09	4096	9.256	3.85	35.17	99.77	268.44
ALM-LOA	2	0.90	3.68	37.94	93.09	4225	9.256	3.85	35.17	99.77	268.47
	4	1.01	3.91	32.23	93.09	5369	9.253	3.84	35.19	99.77	268.71
	6	2.16	7.88	16.26	93.09	9953	**9.250**	3.84	35.22	99.77	269.86
	7	4.11	14.42	13.47	93.09	16320	**9.250**	3.84	35.24	99.77	271.38
	8	-	-	-	-	-	9.255	3.84	35.25	99.77	274.50
	9	-	-	-	-	-	9.277	3.85	35.21	99.77	280.89
	10	-	-	-	-	-	9.357	3.88	35.02	99.77	293.47
	11	-	-	-	-	-	9.640	3.98	33.65	99.77	318.63
	13	-	-	-	-	-	13.634	5.43	18.84	99.77	469.45
	15	-	-	-	-	-	41.636	15.10	8.15	99.77	1073.73
ALM-MAA3	2	0.87	3.58	39.64	94.65	4288	9.253	3.85	35.18	99.80	268.45
	4	1.01	3.96	31.96	95.63	5440	9.251	3.84	35.20	99.83	268.74
	6	2.22	8.77	13.78	96.02	10048	9.248	3.84	35.24	99.85	269.93
	7	4.15	16.79	8.17	96.11	16320	**9.247**	3.84	35.26	99.86	271.45
	8	-	-	-	-	-	9.253	3.84	35.26	99.86	274.60
	9	-	-	-	-	-	9.276	3.85	35.21	99.86	280.89
	10	-	-	-	-	-	9.360	3.88	34.89	99.87	293.45
	11	-	-	-	-	-	9.650	3.99	33.20	99.87	318.64
	13	-	-	-	-	-	13.704	5.56	20.89	99.87	469.63
	15	-	-	-	-	-	41.652	16.98	6.04	99.87	1073.77
ALM-SOA	2	0.81	3.23	**45.54**	97.43	4223	9.248	3.84	35.23	99.95	268.47
	3	**0.78**	**3.06**	44.42	98.06	4605	9.240	3.84	35.28	99.96	268.56
	4	0.85	3.43	35.62	98.42	5369	9.226	3.83	35.36	99.97	268.74
	6	2.61	11.16	11.89	98.96	15104	9.160	3.80	35.77	99.97	269.86
	7	5.46	23.28	6.34	99.01	28416	9.083	3.77	36.24	99.97	271.47
	8	-	-	-	-	-	8.942	3.71	37.13	99.97	274.58
	9	-	-	-	-	-	8.701	3.61	38.76	99.97	280.76
	10	-	-	-	-	-	8.347	3.46	41.54	99.97	293.44
	11	-	-	-	-	-	**8.055**	**3.33**	**41.92**	99.97	318.65
	13	-	-	-	-	-	13.186	5.36	22.06	99.97	520.03
	15	-	-	-	-	-	54.596	22.43	5.57	99.97	1878.98

For 16-bit designs, when M<9, all three ALMs have smaller errors compared with an exact LM. A 16-bit ALM-SOA has a smaller error compared with an exact LM when M<12; and it has the smallest NMED and MRED when M=11. ALM-SOA with M=11 reduces the NMED and MRED by up to 12% and 13%, respectively, compared with a LM made of exact units. As shown in Fig. 12 for 16-bit designs with M in a range from 1 to 12, both ALM-LOA and ALM-MAA3 have similar error characteristics. They have a larger error when M>9 compared with a conventional LM. ALM-SOA has significantly smaller errors compared with all other LMs, especially when 6<M<12. This result clearly shows that ALMs using approximate parts can be even more accurate than a LM using exact parts.

The error probability density distributions (PDD) of LM, ALM-LOA, ALM-MAA3 and ALM-SOA are shown in Figs. 13-16. ALM-LOA and ALM-MAA3 are studied with M=2 and

ALM-SOA is studied with M=3, as these designs have small errors (Table II). A conventional LM has a unidirectional error PDD, *i.e.,* all errors are negative. The errors produced by ALMs can be either negative or positive; however, their ALM-LOA and ALM-MAA3 have significantly more negative errors than positive errors. By using SOA, more positive errors are introduced in ALM-SOA; the largest positive error distance (ED) of ALM-LOA and ALM-MAA3 with M=2 is 510, while ALM-SOA with M=3 has a largest positive ED of 1776. This is the reason by which the ALM-LOA and ALM-MAA3 introduce a significantly smaller (positive) ED than ALM-SOA.
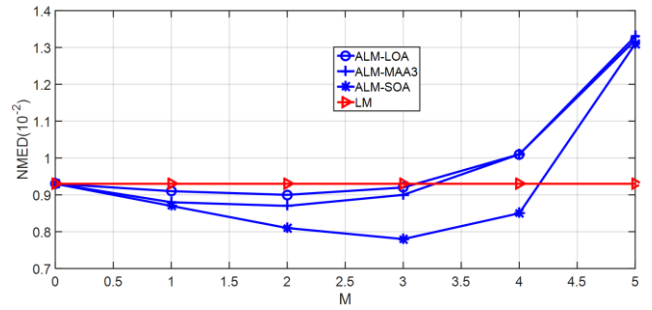

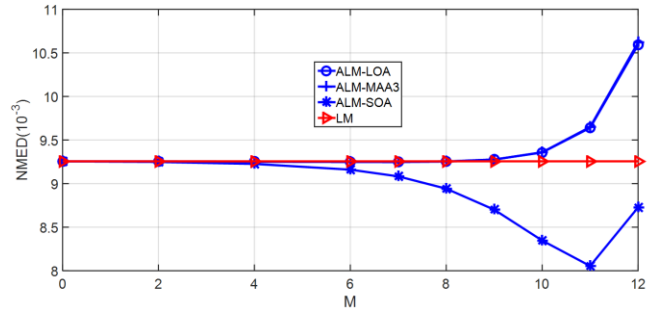Fig. 11. NMED of the proposed 8-bit approximate LMs.

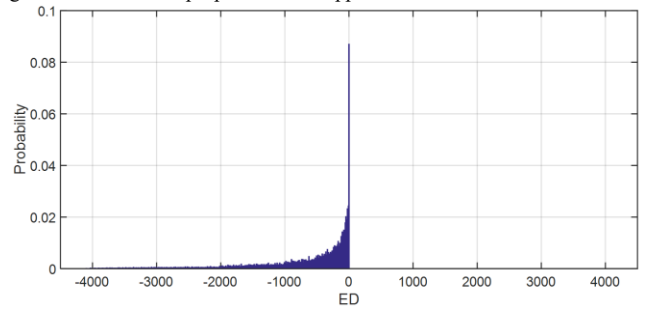Fig. 12. NMED of the proposed 16-bit approximate LMs.

Fig. 13. Error PDD of 8-bit LM.

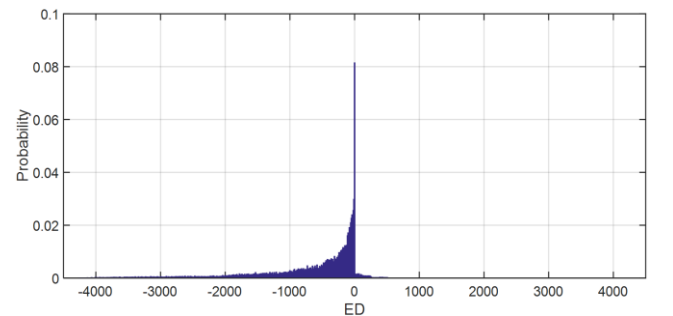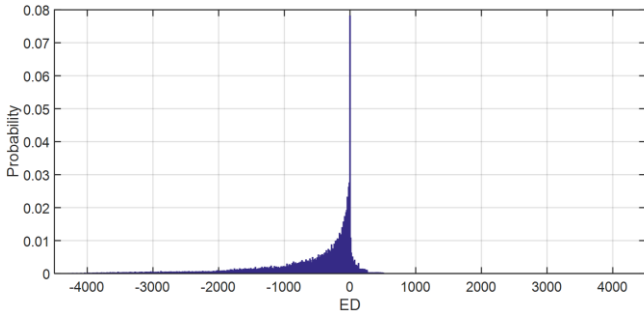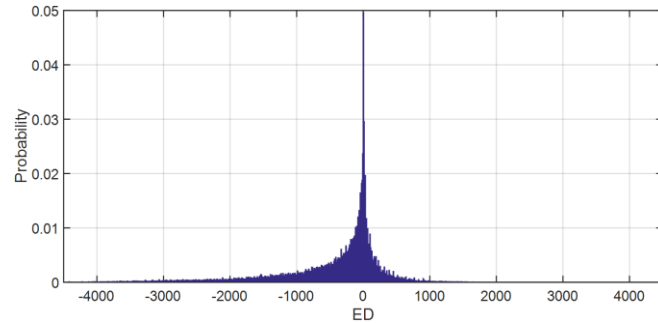Fig. 14. Error PDD of 8-bit ALM-LOA (M=2).

Fig. 15. Error PDD of 8-bit ALM-MAA3 (M=2).

Fig. 16. Error PDD of 8-bit ALM-SOA (M=3).

The bidirectional error PDD of ALM-SOA is nearly symmetric for both negative and positive errors; the total error can be now reduced because negative and positive errors cancel each other during computation. Therefore, ALM-SOA shows lower values in error when M is between 6 and 12 compared with all other LMs. The highest probability occurs when ED=0, so correct results still have a very high probability for both a conventional exact LM and approximate LMs.

2) Error Analysis of IALMs

The error metrics of the proposed iterative approximate logarithmic multipliers are reported in Table III for 16-bit designs. To evaluate the errors of IALM-S, an exact Adder2 ($M_2=0$) is considered first. As for IALM-S ($M_2=0$) in Table III, the accuracy is improved when $M_1<8$; moreover, when $M_1=5$ and 6 it has the smallest error. Therefore, for IALM-S ($M_1=5$) and IALM-S ($M_1=6$), M_2 is changed with these two fixed M_1 values. When $M_1=5$ and $M_2=11$, IALM-S has the smallest NMED; the error of IALM with an exact Adder2 is significantly larger than IALM with an inexact Adder2 using SOA. This is consistent with the error analysis as discussed previously that IALMs using approximate parts have more accurate results than ILMs using all exact units.

The error metrics of IALM-SL and IALM-SM are assessed at $M_1=5$ and $M_2=11$ by varying M_3 (Table IV). Using the iterative technique, the accuracy has been significantly increased compared with non-iterative LMs; their P_{RED} are all larger than 98% when $M_3<24$, while the P_{RED} of ALMs are all less than 50%. When $M_3<18$, the NMEDs of both IALM-SL and IALM-SM are similar to IALM-S with $M_1=5$ and $M_2=11$. The smallest NMED is achieved around $M_3=16$. The IALM-SL with $M_1=5$, $M_2=11$ and $M_3=16$ reduces the NMED by up to 18% compared with an ILM made of exact units. Therefore, IALM-SL and IALM-SM with $M_3=15$, 16 and 17 are studied by additional simulation in Section IV-B.

TABLE III
ERROR CHARACTERISTICS OF 16-BIT IALM-S

	M_1	NMED (10^{-4})	MRED (10^{-4})	P_{RED} (%)	ER (%)	WCE (10^6)
IALM-S **($M_2=0$)**	0	3.50	14.93	99.98	99.158	16.84
	3	3.36	14.27	99.98	99.980	16.99
	5	**3.21**	13.50	99.98	99.992	17.72
	6	**3.21**	**13.38**	99.98	99.993	18.78
	7	3.50	14.47	99.98	99.996	20.82
	8	4.71	19.28	99.99	99.997	24.93
	9	8.22	33.38	99.99	99.998	33.49
	12	77.85	289.13	44.96	99.999	4261.28
	M_2	NMED (10^{-4})	MRED (10^{-4})	P_{RED} (%)	ER (%)	WCE (10^6)
IALM-S **($M_1=5$)**	0	3.21	13.50	99.98	99.992	17.72
	3	3.20	13.46	99.98	99.996	17.69
	6	3.17	13.31	99.98	99.996	17.69
	8	3.11	13.01	99.98	99.996	17.61
	9	3.04	12.70	99.98	99.996	18.12
	10	2.94	12.29	99.98	99.997	17.69
	11	**2.88**	**12.08**	99.98	99.997	17.97
	12	3.21	13.51	99.98	99.997	24.43
	M_2	NMED (10^{-4})	MRED (10^{-4})	P_{RED} (%)	ER (%)	WCE (10^6)
IALM-S **($M_1=6$)**	0	3.21	13.38	99.98	99.993	18.78
	3	3.20	13.35	99.98	99.997	18.77
	6	3.17	13.23	99.98	99.997	18.60
	8	3.12	13.00	99.98	99.998	18.53
	9	3.06	12.74	99.98	99.998	18.60
	10	2.98	12.43	99.98	99.998	18.53
	11	**2.96**	**12.39**	99.98	99.998	18.63
	12	3.35	14.09	99.98	99.998	24.63

TABLE IV
ERROR CHARACTERISTICS OF 16-BIT IALM-SL AND IALM-SM WITH $M_1=5$ AND $M_2=11$

	IALM-SL						IALM-SM				
M_3	NMED (10^{-4})	MRED (10^{-4})	P_{RED} (%)	ER (%)	WCE (10^6)	M_3	NMED (10^{-4})	MRED (10^{-4})	P_{RED} (%)	ER (%)	WCE (10^6)
0	2.881	12.08	99.98	99.997	20.83	0	2.881	12.08	99.98	99.997	20.83
4	2.882	12.08	99.98	99.996	20.98	4	2.882	12.07	99.98	99.994	20.83
8	2.882	12.08	99.98	99.996	20.96	8	2.881	12.08	99.98	99.992	20.83
12	2.880	12.11	99.98	99.996	20.98	12	2.881	12.12	99.98	99.992	20.83
15	2.871	12.59	99.92	99.997	20.80	15	2.881	12.59	99.92	99.992	20.83
16	**2.869**	13.16	99.85	99.997	20.95	16	2.881	13.12	99.86	99.992	20.98
17	2.873	14.24	99.72	99.997	20.80	17	2.884	14.14	99.74	99.993	20.9
18	2.885	16.17	99.48	99.997	20.90	18	2.895	16.00	99.51	99.993	20.98
19	2.927	19.54	99.05	99.997	20.89	19	2.934	19.31	99.10	99.994	20.95
20	3.062	25.27	98.29	99.997	20.89	20	3.058	25.08	98.35	99.995	20.95
24	12.154	102.96	85.75	99.998	28.96	24	10.165	110.80	84.66	99.997	29.37

The relationships between M_1, M_2 and M_3 are shown in Figs. 17-18; M_1, M_2 and M_3 have different error effects on the IALMs. As M_2 increases from 0 to 11, the NMED decreases and the most precise results are obtained when $M_1=5$, $M_2=11$ and $M_3=16$.

M_1 plays the most important role, while M_2 is second to M_1 and M_3 has the smallest impact on the error results. M_2 in Adder2 is used to compensate the errors, so it can be larger than M_1. In Adder3, the lower bits are already approximate; so it is not necessary to perform an accurate calculation. As Adder3 is a 32-bit adder, the number of inexact bits can be significantly larger than M_1 and M_2. Generally, the IALMs can be designed with $M_1 < M_2 < M_3$ to achieve more accurate results.
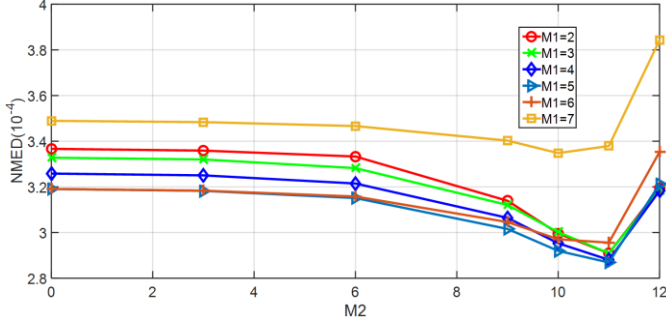

Fig. 17. NMED of 16-bit IALM-SL. M_3 is selected for the most accurate result for each pair of M_1 and M_2.
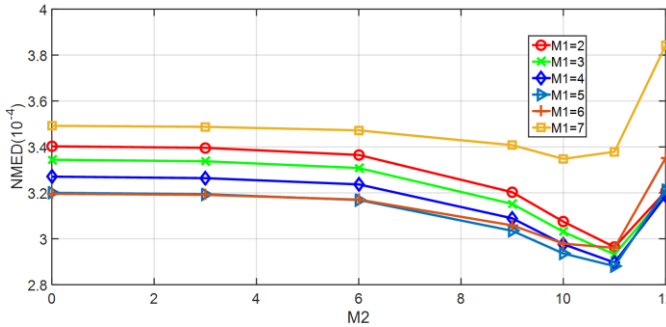

Fig. 18. NMED of 16-bit IALM-SM. M_3 is selected for the most accurate result for each pair of M_1 and M_2.

B. Hardware Evaluation

The proposed ALMs including IALMs are described at gate-level in Verilog HDL and verified by Synopsys VCS. Both designs are then synthesized by the Synopsys Design Compiler using the NanGate 45 nm Open Cell Library. The average power consumption is found using the Synopsys Power Compiler with a back annotated switching activity file generated from the random input vectors.

1) Results for ALMs

The critical path delay, area, power consumption and PDP are reported in Table V for 8-bit and 16-bit ALMs with different values of M. Hardware evaluation of both the exact multiplier (EM) from DesignWare and the exact Booth multiplier (EBM) is also provided; note that EBM performs signed multiplication, while an LM (approximate or exact) computes unsigned multiplication. This signed multiplication scheme is included for completeness in comparison. With an increase of M, the power consumption and PDP of all three designs for both 8-bit and 16-bit decrease, i.e., the PDPs of approximate designs are significantly lower than a conventional LM and the exact multipliers. The critical path delay is also reduced, so leading to an overall better performance. The PDPs of ALM-SOA are significantly better than for ALM-LOA and ALM-MAA3. As the NMED of ALM-SOA is the smallest when $M=11$, this

TABLE V
HARDWARE RESULTS OF 8-BIT AND 16-BIT NON-ITERATIVE ALMS WITH DIFFERENT NUMBER OF INEXACT BITS (Ms)

Designs	M	8-bit multipliers				16-bit multipliers			
		Power (μW)	Delay (ns)	Area (μm2)	PDP (fJ)	Power (μW)	Delay (ns)	Area (μm2)	PDP (fJ)
EM	0	115.40	0.80	350.85	92.30	641.28	1.51	1376.82	968.33
EBM [38]	0	188.00	0.70	788.4	131.60	634.00	1.02	2645.00	646.68
LM	0	70.05	0.68	331.44	59.06	168.12	1.00	781.24	168.12
ALM-LOA	2	68.55	0.68	316.81	46.61	183.32	0.95	848.00	174.15
	4	62.75	0.65	315.48	40.79	162.60	0.98	772.20	159.35
	6	56.67	0.60	286.75	34.00	153.18	1.01	720.59	154.71
	7	49.63	0.62	269.72	30.77	148.08	0.98	721.66	145.12
	8	-	-	-	-	147.55	0.96	701.71	141.65
	9	-	-	-	-	148.23	0.91	751.45	134.89
	10	-	-	-	-	149.32	0.88	757.03	131.40
	11	-	-	-	-	139.74	0.90	733.63	125.77
	13	-	-	-	-	139.13	0.85	738.15	118.26
	15	-	-	-	-	138.51	0.79	733.89	109.42
ALM-MAA3	2	67.79	0.67	321.59	45.42	164.08	1.02	759.43	167.36
	4	63.24	0.66	319.20	41.74	175.12	0.99	840.83	173.37
	6	50.09	0.59	258.55	29.55	161.89	0.95	786.56	153.80
	7	42.00	0.55	218.12	23.10	158.75	0.93	779.38	147.64
	8	-	-	-	-	157.51	0.92	780.98	144.91
	9	-	-	-	-	148.68	0.96	746.40	142.73
	10	-	-	-	-	149.63	0.92	747.99	137.66
	11	-	-	-	-	133.00	0.90	680.16	119.70
	13	-	-	-	-	137.36	0.88	720.59	120.88
	15	-	-	-	-	107.36	0.76	557.54	81.59
ALM-SOA	2	58.65	0.71	287.28	41.64	172.51	0.96	826.20	165.61
	4	46.39	0.60	239.13	27.83	145.86	0.97	703.57	141.48
	6	31.75	0.51	185.40	16.19	134.97	0.95	660.21	128.22
	7	16.09	0.39	108.79	6.28	122.76	0.91	636.80	111.71
	8	-	-	-	-	114.54	0.90	601.96	103.09
	9	-	-	-	-	103.64	0.90	558.33	93.28
	10	-	-	-	-	97.15	0.90	535.46	87.44
	11	-	-	-	-	89.03	0.81	515.24	72.11
	13	-	-	-	-	69.44	0.75	427.46	52.08
	15	-	-	-	-	32.14	0.52	231.42	16.71

TABLE VI
HARDWARE RESULTS OF 16-BIT IALMS.

Designs	M_1	M_2	M_3	Power (μW)	Delay (ns)	Area (μm2)	PDP (fJ)
ILM	0	0	0	469.31	2.04	2089.70	957.39
IALM-SL	5	11	15	333.01	1.82	1538.28	606.08
	5	11	16	328.61	1.83	1529.23	**601.36**
	5	11	17	332.99	1.85	1543.86	616.03
IALM-SM	5	11	15	345.79	1.83	1602.65	632.80
	5	11	16	334.65	1.84	1534.82	615.76
	5	11	17	329.78	1.86	1525.24	613.39

specific design shows the best tradeoff between error and power consumption for non-iterative ALMs.

Note that the PDPs of 16-bit ALM-SOA (M>10) are less than half of the PDP of the conventional LM. Therefore, this also

confirms that the mantissa adder plays a very important role in the overall complexity of the logarithmic multiplier.

2) Results for IALMs

According to the error analysis of Section IV-A, the IALMs with M_1=5, M_2=11, and 14<M_3<18 show good error characteristics. These IALM designs are evaluated by simulation (Table VI).

The power, delay and area of both IALM-SL and IALM-SM are smaller than conventional iterative LMs made of exact units. When considering the error characteristics, both the NMEDs and PDPs of approximate iterative LMs with a truncated BLC, inexact Adder1, Adder2 and Adder3 are better than conventional iterative LMs with exact units. So, approximate design techniques can improve both performance and accuracy of iterative logarithmic multipliers. The design of IALM-SL with M_3=16 is the most efficient design in terms of PDP; it reduces the PDP up to 37% compared with an ILM made of exact units. These approximate logarithmic multipliers are further compared with approximate Booth multipliers in the next subsection.

C. Comparison

Both 8-bit and 16-bit approximate multipliers are compared in this section.

For the 8-bit designs, ALM-SOA with M=3 achieves a good tradeoff between power consumption and accuracy; so it is selected for comparison with previous approximate designs including R4ABM04 [25], R4ABM11 [28], R4ABM12 [29], R4ABM1 and R4ABM2 (both with an approximate factor of 8) [24], R8ABM1, R8ABM2-C5 [23], DRUM (K=6) [20], 8-bit approximate multipliers designed using artificial intelligence methods (mul8x8_444, mul8x8_188 and mul8x8_198) [6].

For 16-bit designs, the 16-bit ALM-SOA with M=10 and 11 show a good tradeoff between power consumption and accuracy, so they are selected along with 16-bit IALM-SLs with M_1=5, M_2=11, and 14<M_3<17 for comparison with previous approximate multiplier designs. R4ABM1 and R4ABM2 (both with an approximate factor of 14) [24], R8ABM2-C15 [23], DRUM (k=6) [20], mul16x16_12, mul16x16_24 and mul16x16_43 [35] designed from artificial intelligence methods are selected and compared.

R4ABM04, R4ABM11, and R4ABM12 are fixed-width truncated Booth multipliers with simple error compensation. R4ABM1 and R4ABM2 are two latest radix-4 approximate Booth multipliers with an accuracy that can be adjusted by approximate factor, i.e., p. R8ABM1 and R8ABM2 are radix-8 approximate Booth multipliers. DRUM is a multiplier based on an approximate operand, namely, the dynamic range unbiased multiplier. DRUM (K=6) is selected, in which 6 bits following the first 1 (to include the first one) are utilized in the multiplication; mul8x8_444, mul8x8_188 and mul8x8_198 for 8-bit comparison, and mul16x16_12, mul16x16_24 and mul16x16_43 for 16-bit comparison are chosen for their similar NMED as other approximate multipliers for comparison purposes. Both the exact multiplier (EM) from DesignWare and the exact Booth multiplier (EBM) are included for comparison.

In these cases, all designs are described in Verilog and synthesized by the Synopsys Design Compiler using the NanGate 45 nm Open Cell Library. The power consumption, delay, area, PDP, NMED, P_{RED} and WCE of 8-bit and 16-bit multipliers are reported in Table VII and Table VIII respectively. Fig. 19 and Fig. 20 show the comparison with both NMED and PDP. Both unsigned and signed multipliers have the same dynamic range and the signed design can be converted rather easily to its unsigned version [39]. Therefore, the signed multiplication schemes are also included for completeness in comparison.

TABLE VII
COMPARATIVE PERFORMANCES OF 8-BIT APPROXIMATE MULTIPLIERS.

Unsigned Designs	Power (μW)	Delay (ns)	Area (μm^2)	PDP (fJ)	NMED (10^{-3})	P_{RED} (%)	WCE
Exact Multiplier	115.4	0.80	350.85	92.30	0	100	0
LM (M=0)	70.1	0.68	331.44	59.06	9.25	36.59	4096
ALM-SOA (M=3)	45.0	0.66	224.78	29.70	7.77	44.42	4605
DRUM(K=6) [20]	108.6	0.98	571.10	106.43	3.20	84.56	2000
mul8x8_444 [6]	62.0	1.24	319.20	76.88	1.30	79.35	541
mul8x8_188 [6]	44.8	1.25	245.25	56.00	3.52	57.24	1304
mul8x8_198 [6]	**40.0**	**0.53**	**217.32**	**21.2**	8.78	30.68	2882
Signed Designs	Power (μW)	Delay (ns)	Area (μm^2)	PDP (fJ)	NMED (10^{-3})	P_{RED} (%)	WCE
Exact Booth Multiplier [38]	188.0	0.70	788.40	131.60	0	100	0
R4ABM04 [25]	125.4	0.66	612.07	82.76	1.54	59.71	192
R4ABM11 [28]	118.6	0.64	603.02	75.90	1.76	57.23	256
R4ABM12 [29]	124.1	0.65	608.34	80.67	1.98	53.68	320
R8ABM1 [23]	106.0	0.63	462.84	66.78	**0.30**	**92.74**	144
R8ABM2-C5 [23]	97.3	0.59	408.84	57.41	1.04	67.82	**82**
R4ABM1 (p=8) [24]	138.0	0.58	581.70	80.04	4.27	40.30	1003
R4ABM2 (p=8) [24]	127.4	0.58	538.60	73.89	4.09	37.16	1003
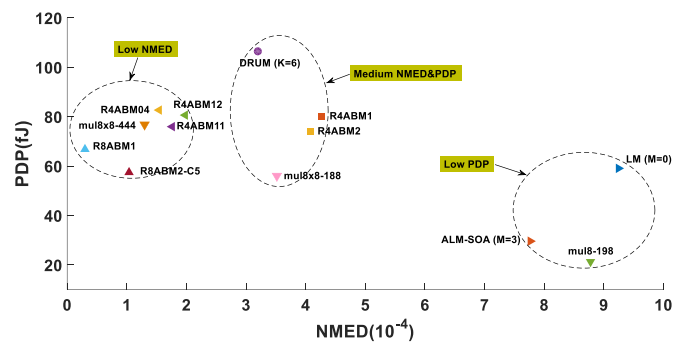
Fig. 19. The NMEDs and PDPs of 8-bit approximate multipliers.

For 8-bit multipliers, all approximate designs improve PDP significantly compared with exact multipliers. As shown in Table VII and Fig. 19, the ALM-SOA with M=3 has the smallest PDP among multipliers except mul8x8_198 [6]. However, mul8x8_198 is significantly more inaccurate. The truncated Booth multipliers, *i.e.*, R4ABM04, R4ABM11, and R4ABM12, show a good tradeoff between PDP and NMED. The best design is R8ABM2-C5; in general, approximate Booth

multipliers are more accurate than ALMs. DRUM is not attractive for 8-bit designs due to the cost of converting the operands.

As shown in Table VIII and Fig. 20, for 16-bit designs, the non-iterative ALMs in general have significantly lower power consumption than all other approximate multipliers. ALM-SOA with M=11 has the smallest PDP and is also the fastest approximate multiplier. DRUM is more accurate than ALM-SOA; however, its PDP is larger. mul16x16_43 is close to ALM-SOA while IALMs significantly improve the accuracy compared with ALMs, DRUM and mul16x16_43. Although the PDPs of IALMs are similar to the approximate Booth multipliers, the NMEDs of IALMs are much larger. However, IALMs have the best P_{RED}, which are all larger than 99.85%. Generally, IALMs have moderate NMED and PDP.

Approximate Booth multipliers have in general the smallest NMED and good P_{RED}. 16-bit R4ABM2 has the smallest NMED of 6×10^{-6} among all compared 16-bit approximate multipliers. Generally, the NMED of non-iterative ALMs is one order larger than iterative ALMs and two orders larger than approximate Booth multipliers.

TABLE VIII
COMPARATIVE PERFORMANCES OF 16-BIT APPROXIMATE MULTIPLIERS.

Unsigned Designs	Power (µW)	Delay (ns)	Area (µm²)	PDP (fJ)	NMED (10^{-4})	P_{RED} (%)	WCE (10^6)
EM	641.3	1.51	1377	968	0	100	0
LM (M=0)	168.1	1.00	781	168	92.56	35.17	268.44
ALM-SOA (M=10)	97.2	0.90	535	88	83.47	41.54	293.44
ALM-SOA (M=11)	**89.0**	**0.81**	**515**	**72**	80.55	41.92	318.65
ILM (0,0,0)	469.3	2.04	2090	957	3.50	**99.98**	16.84
IALM-SL (5,11,15)	333.0	1.82	1538	606	2.87	99.92	20.80
IALM-SL (5,11,16)	328.6	1.83	1529	601	2.87	99.85	20.95
IALM-SM (5,11,15)	345.8	1.83	1603	633	2.88	99.92	20.83
IALM-SM (5,11,16)	334.7	1.84	1535	616	2.88	99.86	20.98
DRUM(K=6) [20]	124.7	1.19	858	148	35.29	70.89	132.71
mul16x16_12	928.6	0.91	1641	845	0.26	99.08	0.41
mul16x16_24	476.7	0.82	864	391	2.54	91.59	4.20
mul16x16_43 [35]	122.3	0.62	326	76	71.25	36.30	84.31
Signed Designs	**Power (µW)**	**Delay (ns)**	**Area (µm²)**	**PDP (fJ)**	**NMED (10^{-4})**	**P_{RED} (%)**	**WCE (10^6)**
EBM [38]	634.0	1.02	2645	647	0	100	0
R4ABM04 [25]	427.3	0.95	1939	406	0.53	97.72	0.23
R4ABM11 [28]	404.4	0.94	1859	380	0.22	99.17	0.13
R4ABM12 [29]	394.6	0.95	1808	374	0.23	99.10	0.14
R8ABM1 [23]	376.7	1.23	1516	463	0.19	99.77	0.32
R8ABM2-C15 [23]	217.3	1.18	912	256	0.57	98.79	0.73
R4ABM1 (p=14) [24]	516.6	0.95	2169	490	0.09	95.89	**0.11**
R4ABM2 (p=14) [24]	479.7	0.92	2004	441	**0.06**	93.44	**0.11**

By comparing 8-bit and 16-bit designs, for applications allowing a large error and a small power consumption, non-iterative approximate LMs are the best choice; for applications requiring more accuracy and a larger power consumption, approximate multipliers (based on accurate binary multipliers) should be used. Although IALMs show significant

improvement in terms of accuracy over non-iterative LMs, their large PDPs make them less attractive compared with approximate Booth multipliers.
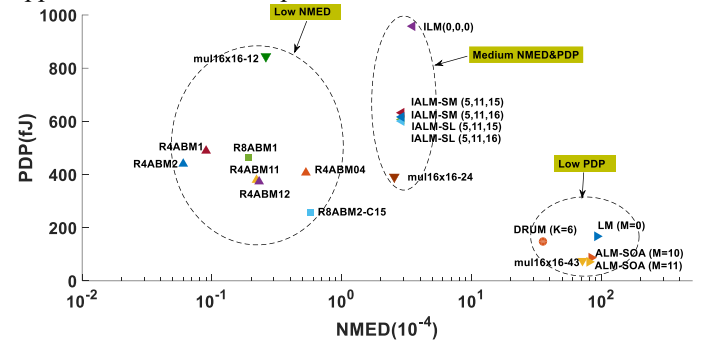

Fig. 20. The NMEDs and PDPs of 16-bit approximate multipliers.

V. CASE STUDIES

The proposed approximate logarithmic multipliers can be applied to error-tolerant applications such as multimedia/signal processing, data mining and analysis, machine learning and patter recognition. In this section, two case studies into image processing and K-means clustering are provided to validate the proposed designs.

A. Image Processing

The 8-bit ALMs are applied to image processing to assess their validity; two identical images are multiplied on a pixel-by-pixel basis to blend them into one single output image. The quality of the output image is described by the peak signal-to-noise ratio (PSNR). The images processed by using the 8-bit ALMs are shown in Figs. 21-22; the resulting PSNRs are reported in Table IX.
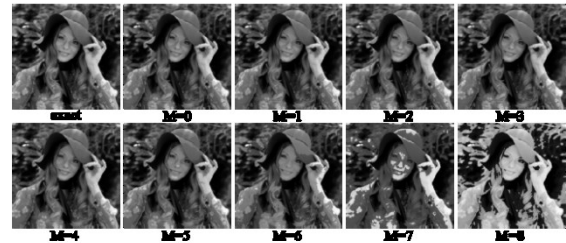
Fig. 21. The multiplied images using the 8-bit ALM-LOA and ALM-MAA3 with different Ms.

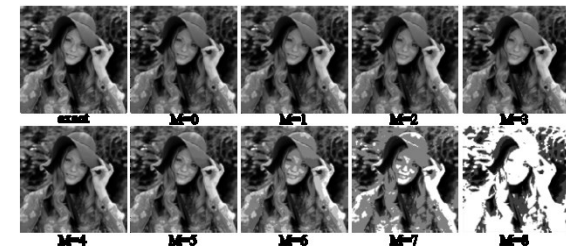
Fig. 22. The multiplied images using the 8-bit ALM-SOA with different Ms.

The most accurate result is produced by ALM-LOA and 8-bit ALM-MAA3 with M=2 and by ALM-SOA with M=4. This is consistent with the error analysis results provided in Table I. For smaller values of M, ALM-SOA produces results that are more accurate; however, for larger values of M, ALM-LOA and

ALM-MAA3 provide better results.

The quality of the processed image does not decrease significantly when M<5. From Figs. 21-22, detection of the differences when M<5 is hard, so showing the validity of the proposed designs for image processing.

TABLE IX
PSNR OF THE OUTPUT IMAGES OF 8-BIT ALMs WITH DIFFERENT NUMBER OF INEXACT BITS (M)

Design	M	PSNR(dB)	Designs	M	PSNR(dB)
8-bit ALM-LOA and ALM-MAA3	0	32.6555	8-bit ALM-SOA	0	32.6555
	1	33.0601		1	33.5270
	2	**33.0699**		2	34.1488
	3	32.9066		3	35.1394
	4	32.3358		4	**35.5746**
	5	30.6913		5	31.0756
	6	27.4286		6	23.6532
	7	21.3979		7	16.1922
	8	15.3423		8	7.2628

Note that the PSNR for the exact multiplier is infinite, as the PSNR is computed by dividing the mean-squared error (MSE). Generally, when the PSNR has a value larger than 30dB then it is considered to be acceptable for 8-bit image applications.

B. K-means Clustering

K-means clustering is a method for cluster analysis in data mining; it partitions n observations into k clusters with the nearest mean [40]. In the k-means clustering, the proposed 16-bit ALMs are applied to calculate the squared deviation between points belonging to different clusters. The F-measure value [41] is used as metric to evaluate the clustering results. It considers both the precision and the recall of the test; so, the F-measure score can be interpreted as a weighted average of the precision and recall. The best value of the F-measure score is 1 and its worst value is 0. Each F-measure value is the average of 50 experiments for each data set. In this work, several University of California Irvine (UCI) benchmark datasets [42] are selected to test the k-means clustering using ALMs. The F-measure results are listed in Table X; the ALMs are chosen with small NMED as found in Table II (ALM-LOA/MAA3 with M=6, 7, 8 and ALM-SOA with M=9, 10, 11).

For the first dataset Statlog (Heart), all LMs produce better results compared to exact multipliers; the proposed ALM-SOA provides the best results. For the second dataset Wholesales customers, ALM-LOA and ALM-MAA3 provide similar results as LM, so significantly better than results generated by exact multipliers. However, ALM-SOA does not provide as good results as other LMs. For the third dataset Balance Scale, ALM-SOA provides the best results than the exact multiplier and other LMs. For this dataset, the conventional LMs and ALM-LOA/ALM-MAA3 produce the worst results. For the fourth dataset Iris, the proposed ALMs generate more accurate results than conventional LM, although the results are worse than the exact multiplier. These results show that the proposed ALMs can provide better results than exact multipliers. However, this is also dependent on specific applications;

different approximate LMs may be suitable for different applications.

TABLE X
F-MEASURE OF K-MEANS CLUSTERING USING 16-BIT ALMs WITH DIFFERENT NUMBER OF INEXACT BITS (M)

16-bit Designs	M	Dataset			
		Stat log (Heart)	Wholesale customers	Balance Scale	Iris
Exact Multiplier	-	0.5373	0.5813	0.5418	0.8276
LM	0	0.6407	0.7170	0.4367	0.4820
ALM-LOA and ALM-MAA3	6	0.6407	0.7170	0.4711	0.4820
	7	0.6407	0.7170	0.4803	0.4820
	8	0.6407	0.7171	0.4353	0.4820
ALM-SOA	9	0.6577	0.6495	0.5687	0.5375
	10	0.6479	0.5640	0.5687	0.5384
	11	0.6727	0.4955	0.5687	0.5384

VI. CONCLUSION

New designs of both non-iterative and iterative approximate logarithmic multipliers have been proposed and analyzed in this paper. Three types of inexact adder are used in the mantissa addition in the non-iterative approximate LMs, denoted as ALM-LOA, ALM-MAA3 and ALM-SOA. In the design of ALM-SOA, the truncated binary-logarithm converter (TBLC) has been applied with no loss of accuracy. To further improve the accuracy of approximate LMs, an iterative technique has been used; so both Adder1 and Adder2 use TBLC and SOA, while Adder3 uses either LOA or MAA3. All approximate LMs have been analyzed using different error metrics. It has been first found that approximate LMs, *i.e.,* ALMs and IALMs using an appropriate number of inexact bits, are more accurate than conventional LMs and ILMs made of exact units. The proposed IALMs with an approximate number of inexact bits can improve both the NMED (up to 12%) and the PDP (up to 37%) over conventional designs.

The proposed designs have also been compared with approximate multipliers (that are based on accurate binary Booth multipliers). The comparison results addressed the interesting concern of selecting the type of approximate designs by a designer. This paper has shown that based on the comparison results for applications allowing a large error but requiring a small power consumption, non-iterative approximate LMs are the best choice; for applications requiring more accuracy while allowing a larger power consumption, approximate multipliers based on accurate binary multipliers should be used.

As the errors for the LMs and ALMs are large, compensation is required for many error-sensitive applications. The iterative approach is a first step to achieve the goal; however, the extensive hardware resources by the IALM make it less attractive. Therefore, we believe the results for the IALM can be useful for finding more efficient error compensation methods as future works.

REFERENCES

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

[2] V. Chippa, S. Chakradhar, K. Roy and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proc. 50th Annual Design Automation Conference (DAC)*, 2013, Article 113, 9 pages.

[3] S. Venkataramani, S. Chakradhar, K. Roy and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in *Proc. 52nd Annual Design Automation Conference (DAC)*, 2015, Article 120, 6 pages.

[4] Q. Xu, N.-S. Kim and T. Mytkowicz, "Approximate Computing: A Survey," *IEEE Design & Test*, vol. 33, no.1, pp. 8-22, 2016.

[5] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Trans. Computers*, vol. 63, no. 9, pp. 1760-1771, 2013.

[6] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "EvoApprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in Proc. Design, Automation & Test in Europe (DATE), 2017, pp. 258-261.

[7] H. Jiang, C. Liu, L. Liu, F. Lombardi and J. Han, "A review, classification, and comparative evaluation of approximate arithmetic circuits", *ACM J. Emerging Technologies in Computing Systems*, vol. 13, no. 4, Article 60, Aug. 2017.

[8] S.-L. Lu, "Speeding up processing with approximation circuits," *Computer*, vol. 37, no. 3, pp. 67-73, 2004.

[9] A. Verma, P. Brisk and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. Design, Automation, and Test in Europe (DATE)*, 2008, pp. 1250-1255.

[10] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo and Z. H. Kong, "Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing," *IEEE Trans. VLSI Syst.*, vol. 18, no. 8, pp.1225-1229, 2010.

[11] K. Du, P. Varian and K. Mohanram, "High-performance reliable variable latency carry select addition," in *Proc. Design, Automation, and Test in Europe (DATE)*, 2012, pp. 1257-1262.

[12] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie and C. Lucas, "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," *IEEE Trans. Circuits Syst.: Part I Regular Papers*, vol. 57, no. 4, pp. 850-862, 2010.

[13] V. Gupta, D. Mohapatra, A. Raghunathan and K. Roy, "Low power digital signal processing using approximate adders," *IEEE Trans. CAD*, vol. 32, no. 1, pp. 124-137, 2013.

[14] W. Liu, L. Chen, C. Wang, M. O'Neill and F. Lombardi, "Design and analysis of floating-point adders," *IEEE Trans. Computers*, vol. 65, pp. 308- 314, Jan. 2016.

[15] J. Mitchell, "Computer multiplication and division using binary logarithms", *IRE Transactions on Electronic Computers*, vol. EC-11, no. 4, pp. 512-517, 1962.

[16] B. Nam, H. Kim, and H. Yoo, "Power and area-efficient unified computation of vector and elementary functions for handheld 3D graphics systems", *IEEE Trans. Comput.*, vol. 57, no. 4, pp. 490-504, 2008.

[17] R. Gutierrez, and J. Valls, "Low cost hardware implementation of logarithm approximation", *IEEE Trans. VLSI Systems*, vol. 19, no. 12, pp. 2326-2330, 2011.

[18] J. Low, and C. Jong, "Unified Mitchell-based approximation for efficient logarithmic conversion circuit". *IEEE Trans. Computers*, vol. 64, no. 6, pp. 1783-1797, 2015.

[19] M. Sullivan, and E. E. Swartzlander, Jr., "Truncated error correction for flexible approximate multiplication", in *Proc. Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers*, 2012, pp. 355–359.

[20] S. Hashemi, R. Bahar and S. Reda, "DRUM: A Dynamic Range Unbiased Multiplier for Approximate Applications," in *Proc. IEEE/ACM International Conference on Computer Design (ICCD)*, 2015, pp. 418 - 425.

[21] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *Proc. 24th Int. Conf. VLSI Design*, 2011, pp. 346–351.

[22] K. Bhardwaj, P. Mane and Jörg Henkel, "Power- and Area-Efficient Approximate Wallace Tree Multiplier for Error-Resilient Systems," in *Proc. 15th IEEE Int. Symp. Quality Electronic Design (ISQED)*, 2014, pp. 263-269.

[23] H. Jiang, J. Han, F. Qiao, F. Lombardi, "Approximate radix-8 Booth multipliers for low-power and high performance operation," *IEEE Trans. Computers*, vol. 65, pp. 2638-2644, Aug. 2016.

[24] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han and F. Lombardi, "Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing," *IEEE Trans. Computers*, vol. 66, pp.1435-1441, Aug. 2017.

[25] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low error fixed-width modified Booth multiplier," *IEEE Trans. VLSI Systems*, vol. 12, no. 5, pp. 522–531, 2004.

[26] M. J. Schulte and E. E. Swartzlander Jr., "Truncated multiplication with correction constant," in *Proc. Workshop VLSI Signal Process. VI*, 1993, pp. 388–396.

[27] E. J. King and E. E. Swartzlander Jr., "Data dependent truncated scheme for parallel multiplication," in *Proc. 31st Asilomar Conf. Signals, Circuits Syst.*, 1998, pp. 1178–1182.

[28] J.-P. Wang, S.-R. Kuang, and S.-C. Liang, "High-accuracy fixed-width modified Booth multipliers for lossy applications," *IEEE Trans. VLSI Systems*, vol. 19, no. 1, pp. 52–60, 2011.

[29] Y.-H. Chen and T.-Y. Chang, "A high-accuracy adaptive conditional probability estimator for fixed-width Booth multipliers," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 59, no. 3, pp. 594-603, 2012.

[30] G. Zervakis, K. Tsoumanis, S. Xydis, N. Axelos and K. Pekmestzi, "Approximate multiplier architectures through partial product perforation: power-area tradeoffs analysis," in *Proc. ACM Great Lake Symp. VLSI*, 2015, pp. 229-232.

[31] C.-H. Lin and I.-C. Lin, "High accuracy approximate multiplier with error correction," in *Proc. IEEE/ACM International Conference on Computer Design (ICCD)*, 2013, pp. 33–38.

[32] A. Momeni, J. Han, P. Montuschi and F. Lombardi, "Design and Analysis of Approximate Compressors for Multiplication," *IEEE Trans. Computers*, vol. 64, no. 4, pp. 984–994, 2015.

[33] C. Liu, J. Han and F. Lombardi, "A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery," *in Proc. Design, Automation, and Test in Europe (DATE)*, 2014, pp. 1-4.

[34] R. Marimuthu, Y. E. Rezinold and P. Mallick, "Design and Analysis of Multiplier Using Approximate 15-4 Compressor," *IEEE Access*, vol.5, pp. 1027-1036, 2017.

[35] M. Ceska, J. Matyas, V. Mrazek, L. Sekanina, Z. Vasicek, and T Vojnar, "Approximating Complex Arithmetic Circuits with Formal Error Guarantees: 32-bit Multipliers Accomplished," in *Proc. Int. Conf. Computer Aided Design (ICCAD)*, 2017, pp. 416-423.

[36] S. Ahmed, S. Kadam, and M. Srinivas, "An iterative logarithmic multiplier with improved precision" in *Proc. IEEE 23nd Symposium on Computer Arithmetic (ARITH)*, 2016, pp. 104-111.

[37] W. Liu, J. Xu, D. Wang and F. Lombardi, "Design of Approximate Logarithmic Multipliers", in *Proc. ACM/IEEE Great Lakes Symp. VLSI (GLSVLSI)*, 2017, pp. 47-52.

[38] W. Yeh and C. Jen, "High-speed Booth encoded parallel multiplier design," *IEEE Trans. Comput.*, vol. 49, no. 7, pp. 692-701, Jul. 2000.

[39] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Trans. Computers*, vol. C-22, pp. 1045–1047, Dec. 1973.

[40] E. Forgy, "Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications," *Biometrics*, vol. 21, pp. 768-769, 1965.

[41] L. Rushi, D. Snehlata and M. Latesh, "Class imbalance problem in data mining: review," *Int. J. Computer Science and Network*, vol. 2, pp. 83-87, 2013.

[42] K. Bache and M. Lichman, UCI machine learning repository, University of California, Irvine. http://archive.ics.uci.edu/ml, 2013.

Weiqiang Liu (M'12-SM'15) received the B.Sc. degree in Information Engineering from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China and the Ph.D. degree in Electronic Engineering from the Queen's University Belfast (QUB), Belfast, UK, in 2006 and 2012, respectively. In Dec. 2013, he joined the College of Electronic and Information Engineering, NUAA, where he is currently an Associate Professor. He was a Research Fellow in the Institute of Electronics,

Communications and Information Technology (ECIT) at QUB from Aug. 2012 to Nov. 2013. He has published one research book by Artech House and over 50 leading journal and conference papers. His paper was finalist in the Best Paper Contest of IEEE ISCAS 2011 and he is the co-author of a Best Paper Candidate of ACM GLSVLSI 2015. He serves as an Associate Editor of *IEEE Transactions on Computers* (TC), the leader of The Multimedia Team at TC Editorial Board, and the Guest Editors of two special issues of *IEEE Transactions on Emerging Topics in Computing*. He has been a technical program committee member for several international conferences including ARITH, ASAP, ISCAS, and ICONIP. He is a member of IEEE Circuits & Systems for Communications (CASCOM) Technical Committee. His research interests include emerging technologies in computing systems and VLSI design for digital signal processing and cryptography.

Jiahua Xu was born in Jiangsu Province, China in 1996. She is currently pursuing her B.Sc. degree in Information Engineering at College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China. She was an visiting student in Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, from Sep. 2016 to Dec. 2016. Her current research interests include approximate computing circuits and low-power VLSI design for signal processing and pattern recognition.

Danye Wang was born in Anhui Province, China, in 1996. She is pursuing her B.Sc. degree in Information Engineering at College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China. Her current research interests include approximate computer arithmetic circuits, low power integrated circuits design, radio frequency integrated circuit design and signal integrity.

Chenghua Wang received the B.Sc. and M.Sc. degrees from Southeast University, Nanjing, China, in 1984 and 1987, respectively. In 1987, he joined the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, where he became a full Professor in 2001. He has published 6 books and over 100 technical papers in journals and conference proceedings. He is the recipient of more than ten teaching and research awards at the national and provincial level. His current research interests include testing of integrated circuits, and circuits and systems for communications and signal processing.

Paolo Montuschi (M'90-SM'07-F'14) is a Full Professor in the Department of Control and Computer Engineering and a Member of the Board of Governors (BoG) at Politecnico di Torino. Previously, he served as Chair of the Department from 2003 to 2011. His research interests include computer arithmetic and architectures, computer graphics, electronic publications, semantics and education. He is an IEEE Fellow, an IEEE Computer Society (CS) Golden Core member, a recipient of the "Distinguished Service" and the "Spirit of the Computer Society" awards. Currently, he is serving as Editor-in-Chief of *IEEE Transactions on Computers* (2015-18), as the Chair of the CS Awards Committee (2017-18), and as a member of the IEEE Publications Services and Products Board (2018-20). Previously, he served in a number of Boards and Committees, including the IEEE Products and Services Committee and the BoG of the CS. He is a life member of the International Academy of Sciences of Turin and of Eta Kappa Nu (the Honor Society of IEEE). In March 2017, he co-founded the first HKN Student Chapter in Italy.

Fabrizio Lombardi graduated in 1977 from the University of Essex (UK) with a B.Sc. (Hons.) in Electronic Engineering. In 1977 he joined the Microwave Research Unit at University College London, where he received the Master in Microwaves and Modern Optics (1978), the Diploma in Microwave Engineering (1978) and the Ph.D. from the University of London (1982). He is currently the holder of the International Test Conference (ITC) Endowed Chair Professorship at Northeastern University, Boston. During 2007-2010 Dr. Lombardi was the Editor-In-Chief of the *IEEE Transactions on Computers*. Currently, he is the Editor-in-Chief of the *IEEE Transactions on Nanotechnology* and the inaugural Editor-in-Chief of the *IEEE Transactions on Emerging Topics in Computing*. He currently serves as an elected Member of the Board of Governors of the IEEE Computer Society. His research interests are bio-inspired and nano manufacturing/computing, VLSI design, testing, and fault/defect tolerance of digital systems. He has extensively published in these areas and coauthored/edited seven books. He is a Fellow of IEEE.