TOSAM: An Energy-Efficient Truncation- and Rounding-Based Scalable Approximate Multiplier

Shaghayegh Vahdat, Mehdi Kamal, Ali Afzali-Kusha, and Massoud Pedram

***Abstract*— A scalable approximate multiplier, called truncation- and rounding-based scalable approximate multiplier (TOSAM) is presented, which reduces the number of partial products by truncating each of the input operands based on their leading one-bit position. In the proposed design, multiplication is performed by shift, add, and small fixed-width multiplication operations resulting in large improvements in the energy consumption and area occupation compared to those of the exact multiplier. To improve the total accuracy, input operands of the multiplication part are rounded to the nearest odd number. Because input operands are truncated based on their leading one-bit positions, the accuracy becomes weakly dependent on the width of the input operands and the multiplier becomes scalable. Higher improvements in design parameters (e.g., area and energy consumption) can be achieved as the input operand widths increase. To evaluate the efficiency of the proposed approximate multiplier, its design parameters are compared with those of an exact multiplier and some other recently proposed approximate multipliers. Results reveal that the proposed approximate multiplier with a mean absolute relative error in the range of 11%–0.3% improves delay, area, and energy consumption up to 41%, 90%, and 98%, respectively, compared to those of the exact multiplier. It also outperforms other approximate multipliers in terms of speed, area, and energy consumption. The proposed approximate multiplier has an almost Gaussian error distribution with a near-zero mean value. We exploit it in the structure of a JPEG encoder, sharpening, and classification applications. The results indicate that the quality degradation of the output is negligible. In addition, we suggest an accuracy configurable TOSAM where the energy consumption of the multiplication operation can be adjusted based on the minimum required accuracy.**

***Index Terms*— Accuracy configurable, approximate multiplier, area efficient, low energy, scalable, truncating.**

I. Introduction

POWER consumption is one of the critical design constraints in designing digital systems. Approximate computing (AC) is one of the approaches which may be used to reduce energy consumption and/or increase the speed. Since the computing result may not be correct, AC can be exploited in error-resilient applications. Examples of these applications include audio and image processing [1], machine learning [2], and data mining [3]. More specifically, in many signal processing applications, a large portion of the energy consumption is caused by arithmetic operations (e.g., up to almost 75% of the total energy consumption of a fast Fourier transform architecture [4]). Among these operations, multiplication, which is used repeatedly, is a high latency and energy consuming operation [5]. This makes approximate multipliers good candidates for being employed in error-tolerant signal processing units.

Generally, a multiplication operation consists of three steps. In the first step, the partial products are generated based on the input operands. In the second step, the partial products are accumulated until only two rows remain. In the final step, the remained two rows are summed by employing a (fast) adder. One may apply the approximation to each of these steps. Approximation can be invoked in the first step to decrease the number of partial products [1], [6], [7] or to decrease the complexity of their generation [8]. Approximation may be applied in the second step of the multiplication process to decrease the latency or power consumption of the reduction levels. One of these approaches is to utilize approximate compressors [9]–[12]. The latency and power consumption of the multiplication operation are highly affected by the architecture of the adder used in the final step of the multiplication process. Hence, one may also employ an approximate adder in the final step to improve the power consumption of the multiplier [13].

In this paper, we present an approximation technique for decreasing the number of partial products. In the proposed approximate algorithm, input operands are truncated to h and t bits according to the position of their leading one bit, where these truncated values are employed for the multiplication and addition operations. In addition, to reduce the error resulting from the truncation operation, we find the approximate amount of the truncated values by rounding them. These simplifications result in higher accuracy and performance compared to those of the state-of-the-art approximate multipliers. Moreover, the proposed approximate multiplier has a nearly normal error distribution with near zero mean value. The calculation core of the proposed multiplier performs multiplication and addition operations on truncated and rounded numbers and the result

Manuscript received June 11, 2018; revised September 26, 2018 and November 24, 2018; accepted December 26, 2018. *(Corresponding author: Ali Afzali-Kusha.)*

S. Vahdat, M. Kamal, and A. Afzali-Kusha are with the School of Electrical and Computer Engineering, University of Tehran, Tehran 14399-57131, Iran (e-mail: vahdat_s@ut.ac.ir; m.kamal@ut.ac.ir; afzali@ut.ac.ir).

M. Pedram is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90007 USA (e-mail: pedram@usc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2018.2890712

is shifted to the left to generate the final output. Because the arithmetic operations are performed on the truncated values, the calculation core of the proposed multiplier is small and consumes less energy compared to that of the exact multiplier. Also, the accuracy of the proposed method is mainly dependent on t and h parameter values and is not significantly affected by the width of the input operands. This provides a scalability feature for the proposed multiplier. Key contributions of this paper may be summarized as follows.

1) A new scheme for the scalable approximate multiplier, which finds the position of the leading one bit and exploits both truncation and rounding operations to improve the accuracy of the multiplication operation.
2) Exploration of t (truncation) and h (rounding) parameters to find a tradeoff between accuracy, delay, and energy consumption.
3) Presenting hardware implementation of truncation- and rounding-based scalable approximate multiplier (TOSAM) for both signed and unsigned operations.
4) Investigating design parameters of the proposed multiplier for image processing and classification applications.

The rest of this paper is organized as follows. Some of the prior works on approximate multipliers are reviewed in Section II. In Section III, the details of the algorithm of the proposed approximate multiplier are described while its hardware implementation and error analysis are presented in Sections IV and V, respectively. The efficacy of the suggested multiplier in several image processing and classification applications as well as its delay, area, and power are analyzed in Section VI. Finally, this paper is concluded in Section VII.

II. Related Work

In this section, we review some of the research efforts on designing approximate multipliers. In the dynamic segment method (DSM) structure [1], the input operands were truncated to m bits based on the position of their leading one bit where a fixed-width multiplication was performed on the truncated values. This way of truncation made the generated output always less than the exact one, making the mean relative error (MRE) negative. This is an undesired feature due to the fact that for the approximate arithmetic units with the Gaussian error distribution, it is better to have the mean error close to zero for having a higher signal-to-noise ratio (SNR) when dealing with digital signal processing applications [15]. In the dynamic range unbiased multiplier (DRUM) structure [6], to mitigate the error resulted from the truncation operation for pushing the MRE toward zero, the least significant bit of the truncated input was set to "1." In LETAM structure [5], the input operands were truncated and in the multiplication step, half of the partial products were omitted. Hence, the delay and power consumption were improved compared to those of the DSM and DRUM structures due to omitting the partial products. In RoBA multiplier [7], the input operands were rounded to the nearest power of two where the output was produced by some shift, add, and subtraction operations. In this structure, the number of elements that should be summed to generate the final result was reduced compared to the exact multiplier leading to better energy and speed. As another approach, to improve the speed and area of the multiplier, the least significant bits of the partial products were eliminated [14].

A straightforward way to generate the partial products is to multiply each bit of the multiplier by the multiplicand, which can be performed simply by performing logical AND operation. Another approach is to encode the multiplier in higher radixes and multiply the encoded multiplier by the multiplicand. As the radix increases, encoding the multiplier becomes more complex. Hence, to decrease this complexity, one may use approximate encoders to generate partial products [16]. In [17], the partial products of an approximate radix-4 Booth multiplier were generated and accumulated approximately. Also, in [18], an approximate radix-8 Booth multiplier was proposed which used approximate adders to produce the least significant bits of the triple multiplicand. In [8], the most significant bits of the multiplier were encoded using exact radix-4 encoding and the least significant bits were encoded using an approximate higher radix encoding which rounded the least significant bits to the nearest power of two.

In [9], four approximate 4:2 compressors were proposed and exploited in the reduction levels of the multiplier. In [10], an approximate 4:2 compressor was proposed and employed in the accumulation step and an error recovery module was added to improve the accuracy of multiplication. In [11], several approximate 5:3 compressors were used in an approximate 15:4 compressor utilized in the main approximate multiplier structure. It should be mentioned that to increase the accuracy, accurate compressors were exploited to produce the most significant bits of the result. In [12], several approximate compressors have been proposed. Also, an algorithm was suggested to design efficient approximate multipliers composed of these compressors. In [19], several approximate adders were considered as the building blocks of the approximate multiplier and the design space was explored to find the optimum design.

To improve the speed of multiplication, one approach is to change the numbering system to the logarithmic one to perform addition instead of multiplication. In this method, the logarithm of the input operands is generated, their sum is calculated, and an antilogarithm operation is performed on their sum to generate the final result. The complexity of this method originates from generating the logarithm and antilogarithm steps. The accuracy of the multiplier depends on the accuracy of these steps. Several studies have been conducted on how to find the logarithm and antilogarithm of a number [20], [21]. Mitchell [22] proposed a simple approximate method to calculate the logarithm and antilogarithm of a number and used it to generate the multiplication results (Mitchell multiplier). Since then, some studies have been conducted on improving the Mitchell-based logarithmic multipliers [23], [24].

In this paper, we propose an approximate multiplier that finds the position of the leading one bits of the input operands, truncates and rounds them with different widths, and performs some shift, add, and small fixed-width multiplication operations to generate the multiplication result.

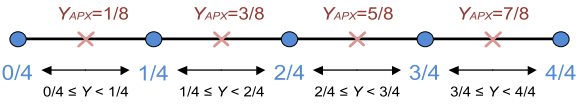

Fig. 1. Amount of Y_{APX} according to the amount of Y for the case of $S = 4$.

III. Proposed Approximate Multiplier

A. TOSAM

Each positive integer number (N) can be represented as

$$N = \sum_{i=0}^{k} 2^i x_i \tag{1}$$

where k denotes the position of its leading one bit and x_i is the ith bit of N. By factoring 2^k from (1), it is rewritten as

$$N = 2^k \left(\sum_{i=0}^{k} 2^{i-k} x_i \right) = 2^k \times X \tag{2}$$

where X is a fractional number between 1.0 and 2.0. Based on (2), the result of multiplying A by B may be calculated as

$$A \times B = 2^{k_A + k_B} \times X_A \times X_B. \tag{3}$$

Widths of X_A and X_B are the same as A and B making the calculation of the exact value of $X_A \times X_B$ time and power consuming. We propose calculating the approximate amount of this term based on the fractional parts of X_A and X_B. In the remainder of this paper, we represent the fractional part of X as Y obtained from

$$Y = X - 1. \tag{4}$$

For example, assume that $X = (1.1101)_2$. In this case, $Y = (0.1101)_2$. To generate the approximate value of Y, we divide this range (0.0–1.0) into S equal segments where S is a power of two represented by

$$S = 2^h \tag{5}$$

where h denotes an arbitrary positive integer which is one of our design parameters. It is obvious that the length of each segment is equal to $1/S$. We propose to generate the approximate value of Y as

$$Y_{\mathrm{APX}} = \frac{2m-1}{2S} \quad \text{if} \quad \frac{m-1}{S} \le Y < \frac{m}{S}, \quad m = 1, 2, \ldots, S. \tag{6}$$

For a better illustration, the approximate amounts of Y for the case where S is equal to 4, is depicted in Fig. 1.

To find Y_{APX}, it is required to consider only h most significant bits of Y. For example, when $S = 4 (h = 2)$, if two most significant bits of Y are zero, it means that $0 \le Y < 1/4$. Hence, we choose $1/8 = (0.001)_2$ as Y_{APX}. When two most significant bits of Y are "10," which implies that $2/4 \le Y < 3/4$, hence, Y_{APX} is approximated as $5/8 = (0.101)_2$. In other words, the value of Y_{APX} is obtained simply by truncating Y to h bits and inserting a "1" bit to the right side of the truncated Y. As a result, the width of Y_{APX} will be equal to $h + 1$ bits. Making use of (4), (3) is rewritten as

$$A \times B = 2^{k_A + k_B} \times (1 + Y_A + Y_B + Y_A \times Y_B). \tag{7}$$

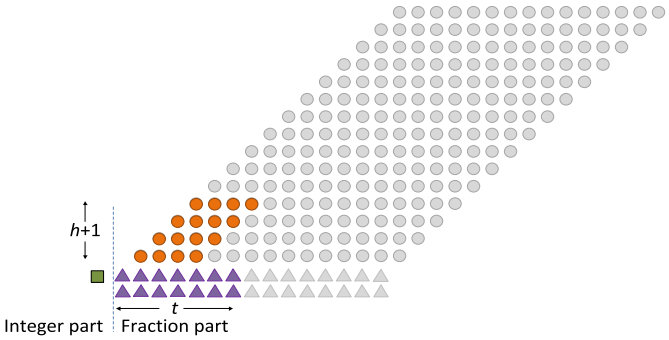

Fig. 2. Dot diagram of term $1 + (Y_A)_t + (Y_B)_t + (Y_A)_{\mathrm{APX}} \times (Y_B)_{\mathrm{APX}}$ where $t = 7$ and $h = 3$.

Now, the approximate of (7) may be expressed as

$$A \times B \approx 2^{k_A + k_B} \times (1 + Y_A + Y_B + (Y_A)_{\mathrm{APX}} \times (Y_B)_{\mathrm{APX}}). \tag{8}$$

To improve the speed of calculation, we truncate Y_A and Y_B to t bits, where in the rest of this paper, we denote by $(Y_A)_t$ and $(Y_B)_t$. Hence, we modify (8) as

$$A \times B \approx (A \times B)_{\mathrm{APX}} = 2^{k_A + k_B} \times \begin{pmatrix} 1 + (Y_A)_t + (Y_B)_t + \\ (Y_A)_{\mathrm{APX}} \times (Y_B)_{\mathrm{APX}} \end{pmatrix} \tag{9}$$

where the width of $(Y_A)_{\mathrm{APX}}$ ($(Y_B)_{\mathrm{APX}}$) is $h + 1$ bits. To have a better understanding, the dot diagram of the proposed algorithm for the case where $t = 7$ and $h = 3$ compared to that of an exact 16-bit multiplier is depicted in Fig. 2. The green square shows the "1" bit in the term $1 + (Y_A)_t + (Y_B)_t + (Y_A)_{\mathrm{APX}} \times (Y_B)_{\mathrm{APX}}$. Orange circles denote partial products of $(Y_A)_{\mathrm{APX}} \times (Y_B)_{\mathrm{APX}}$, whereas purple triangles show the bits of $(Y_A)_t$ and $(Y_B)_t$. Gray circles and triangles are omitted and are not considered in the calculations. As shown in Fig. 2, in the exact 16-bit multiplier, the number of partial products is equal to 256, which must be summed to generate the final result while in the proposed method, only 31 of the partial products are kept (which amounts to ~88% partial products reduction). This reduction rate will rise as the bit length of the multiplier input operands increases. As an example, the steps of multiplying A by B for the case of $t = 7$ and $h = 3$ are depicted in Fig. 3. In the rest of this paper, we denote our proposed structures by TOSAM (X, Y) where X and Y correspond to h and t.

The accuracy of the proposed approach depends on the values of the parameters t and h. Therefore, in the error analysis section (Section V), we will find a relationship between t and h parameters to achieve an almost high accuracy while having an acceptable speed and energy consumption.

Finally, the proposed multiplication approach is feasible for the case of unsigned operands. To use it for signed multipliers, one may find the absolute value of the input operands, multiply them by the proposed algorithm, and fix the sign of the final result according to the sign of the input operands. Finding the exact absolute value of the input operands may degrade the speed of calculation and, hence, we produce it according to the method presented in [7].

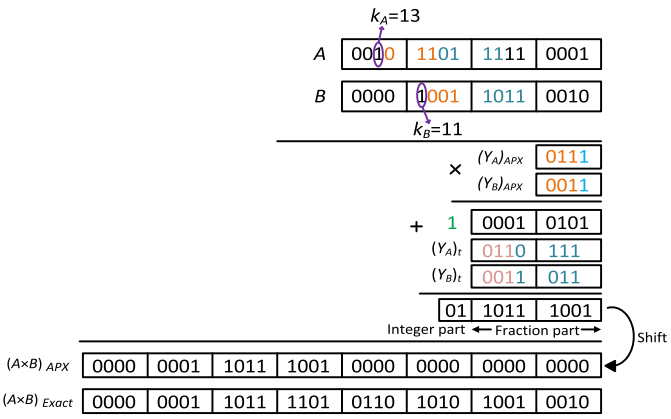

Fig. 3. Numeric example of 16-bit TOSAM(3, 7) with $A = 11761$ and $B = 2482$. The approximate result $[(A \times B)_{\text{APX}}]$ is equal to 28 901 376 while the exact result $[(A \times B)_{\text{Exact}}]$ is equal to 29 190 802. In this case, the absolute error is 289 426 which is about 0.99% of the exact output (the error is less than 1% in this case).

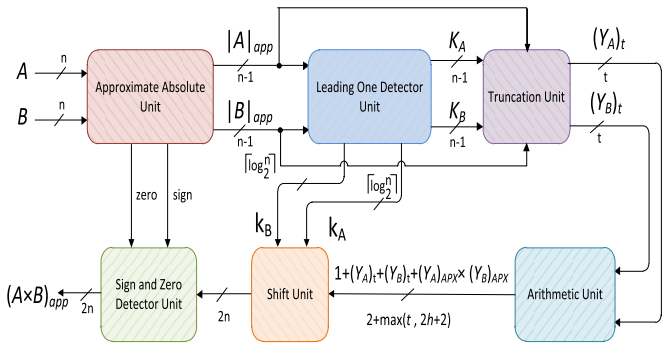

Fig. 4. Block diagram of the proposed approximate signed multiplier.

IV. Hardware Implementation

The block diagram of the proposed signed approximate multiplier is depicted in Fig. 4. First, the approximate absolute value of the input operands $(|A|_{\text{app}}, |B|_{\text{app}})$ is determined using the Approximate Absolute Unit, similar to the one exploited in [7]. In this unit, the bits of the input are inverted if the input is negative and they are not changed if the input is positive. $|A|_{\text{app}}$ and $|B|_{\text{app}}$ are injected to the Leading-One Detector Unit [25] and the positions of their leading one bits are found using

$$K[i] = \left(\bigwedge_{j=i+1}^{n-2} \overline{I[j]} \right) \wedge I[i] \text{ for } 0 \le i \le n-2 \qquad (10)$$

where I can be either $|A|_{\text{app}}$ or $|B|_{\text{app}}$. Only one bit of the signal K is "1" revealing the position of the input leading one bit. By using the K_A and K_B signals in a lookup table, k_A and k_B signals needed for (7) can be generated. The schematic of the Leading-One Detector Unit for 8-bit input operands is depicted in Fig. 5. For example, assume that $|A|_{\text{app}} = (011001)_2$, in this case $K_A = (010000)_2$ and $k_A = (100)_2 = 4$. Signals $|A|_{\text{app}}$, $|B|_{\text{app}}$, K_A, and K_B are then applied to the Truncation Unit [25] to produce $(Y_A)_t$ and $(Y_B)_t$. Assume that the input and output of this unit are I and $(Y)_t$. In this case, the ith bit

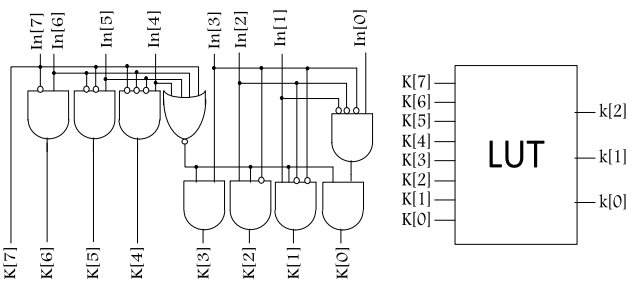

Fig. 5. Schematic of the Leading-One Detector Unit for 8-bit input operands.

of the output can be generated using

$$(Y)_t[i] = \bigvee_{j=0}^{n-2} (K[j] \wedge I[j+i-t]) \text{ for } i < t. \qquad (11)$$

Signals $(Y_A)_t$ and $(Y_B)_t$ are then exerted to the Arithmetic Unit to calculate the term $1 + (Y_A)_t + (Y_B)_t + (Y_A)_{\text{APX}} \times (Y_B)_{\text{APX}}$. It should be noted that the h most significant bits of $(Y_A)_{\text{APX}}$ and $(Y_B)_{\text{APX}}$ are the same as the h most significant bits of $(Y_A)_t$ and $(Y_B)_t$ whose rightmost bits are always "1." Hence, there is no need to add extra hardware to generate $(Y_A)_{\text{APX}}$ and $(Y_B)_{\text{APX}}$ signals which are produced by simple wiring.

In the Shift Unit, the output of the Arithmetic Unit is shifted to left by $k_A + k_B$ to produce the term $2^{k_A+k_B} \times (1 + (Y_A)_t + (Y_B)_t + (Y_A)_{\text{APX}} \times (Y_B)_{\text{APX}})$ [see (9)]. In the Sign and Zero Detector Unit, the sign of the output is set according to the sign of the multiplier input operands and also the output is set to zero if at least one of the inputs is zero. In the case of the unsigned multipliers, the Approximate Absolute Unit should be omitted and the Sign and Zero Detector Unit should be replaced by a Zero Detector Unit.

TOSAM can be implemented in an accuracy configurable structure. In order to implement an accuracy configurable structure of TOSAM, all of its units should be designed for the largest desired h and t values such that the design can work in all operation modes. We suggest a configurable TOSAM structure with three different operating modes of T2, T6, and T9 corresponding to TOSAM(0, 2), TOSAM(2, 6), and TOSAM(5, 9), respectively. The Truncation and the Shift Units of the configurable TOSAM should be designed for the largest t and h values ($h = 5$ and $t = 9$ in this case). In the Arithmetic Unit, some of the adders and logical AND gates should be power gated based on the operating mode to make the design more power efficient.

The reduction levels of the partial products based on the operating modes are depicted in Fig. 6. In the last level, a fast 9-bit adder is employed. To decrease its switching activity, some of its inputs are set to "0" with a transmission gate (TG), based on the operating mode.

In the T2 mode, only the purple partial products are accumulated, only the purple adders are active (not power gated), and all of the inputs of the 9-bit adder are set to "0." The 10 least significant bits of the result are set to "0" using the

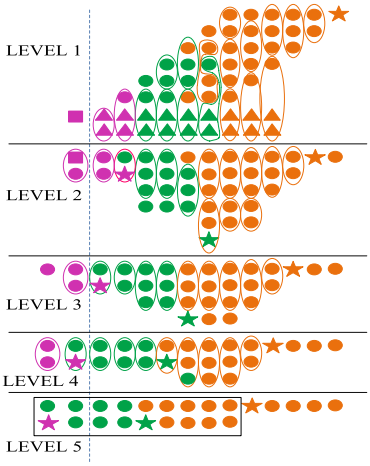

Fig. 6. Reduction levels of accuracy configurable TOSAM with three different operating modes.

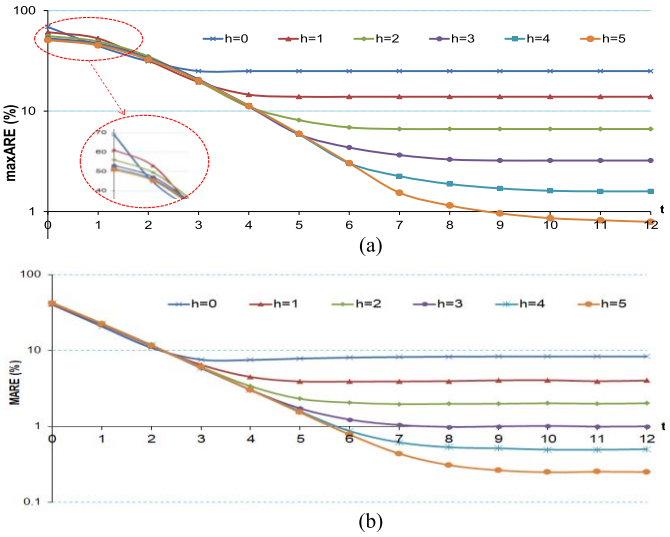

Fig. 7. maxARE and MARE of the proposed approximate multipliers versus t with h as the running parameter.

TGs and purple stars are passed through the TGs to generate four most significant bits of the output.

In the T6 mode, only the green and purple partial products are generated and summed to compose the final output and the orange adders are power gated. In addition, the orange inputs of the 9-bit adder are set to "0." Also, in the eighth column of LEVEL1, there are two orange circles that should be set to "0" by TGs when operating in the T6 mode. In this mode, the six least significant bits of the result are set to "0," green stars are passed through TGs to generate four intermediate bits of the output, and the four most significant bits of the result are produced by the 9-bit adder.

In the T9 mode, all parts are active. The orange stars are passed through the TGs to generate the five least significant bits of the result and the other bits are produced by the 9-bit adder.

Note that the least significant bits of $(Y_A)_{\text{APX}}$ and $(Y_B)_{\text{APX}}$, which depend on the operating mode, should be rounded (set to "1"). It is simply done by performing a logical OR operation on the corresponding bit and the operating mode. For example, when T6 signal is "1," the logical OR operation sets the corresponding bit of $(Y_A)_{\text{APX}}$ and $(Y_B)_{\text{APX}}$ to "1."

V. Accuracy Analysis

In this section, first, error analysis results of the proposed algorithm, using MATLAB, are discussed. Next, the scalability of the proposed method is studied by changing the bit length of the multiplier and measuring its relative error. Finally, the accuracy of the proposed method is compared with other approximate structures. To determine the accuracy of the studied n-bit multipliers, one million uniform random numbers in the range of 0 to $2^n - 1$ were injected. In the case of 8-bit multipliers, the accuracies were extracted by applying 65 536 numbers.

To measure the accuracy of the proposed algorithm, one may use the RE parameter defined by

$$\text{RE} = \frac{P_{\text{app}} - P}{P} \tag{12}$$

where P and P_{app} represent the exact and approximate values of the result, respectively. We denote the absolute of RE values as absolute relative error (ARE). Using (7), (9), and (12), the RE of the proposed approximate algorithm may be expressed by (13), as shown at the bottom of this page.

We have measured the maximum (maxARE) and the mean (MARE) of the ARE values for unsigned 32-bit multipliers with different t and h parameters whose results are depicted in Fig. 7. As shown in Fig. 7, for a constant h value ($h > 0$), MARE decreases as t increases until it reaches $h+4$ where after this point, the rate of maxARE and MARE changes decreases. This hints us to choose $t = h+4$ as the best choice for a better tradeoff between the accuracy, speed, and power consumption of the multiplier. In the rest of this paper, we have considered $t = h+4$ for $h > 0$. In the case of $h = 0$, the least MARE is achieved for $t = 3$.

To show the scalability of the proposed algorithm, we have reported MARE and the variance of ARE (VARE) versus the width of the unsigned multipliers in Table I. As the results reveal, the accuracy of the multiplier is strongly (weakly) dependent on the h value (bit length of the multiplier). It can be observed that by increasing h by one, the MARE value is almost halved. The accuracy of the multiplier also depends on t (see Fig. 7). It should be mentioned that the arithmetic

$$\begin{aligned}\text{RE} &= \frac{2^{k_A+k_B} \times (1 + (Y_A)_t + (Y_B)_t + (Y_A)_{\text{APX}} \times (Y_B)_{\text{APX}})}{2^{k_A+k_B} \times (1 + Y_A + Y_B + Y_A \times Y_B)} - 1 \\ &= \frac{((Y_A)_t - Y_A) + ((Y_B)_t - Y_B) + ((Y_A)_{\text{APX}} \times (Y_B)_{\text{APX}} - Y_A \times Y_B)}{(1 + Y_A + Y_B + Y_A \times Y_B)}\end{aligned} \tag{13}$$

TABLE I

MARE AND VARE OF THE PROPOSED APPROXIMATE MULTIPLIER WITH DIFFERENT WIDTHS AND DIFFERENT h AND t VALUES

TOSAM *Architecture* (*h*,*t*)	8-bit		16-bit		32-bit	
	MARE (%)	*VARE (%)*	*MARE (%)*	*VARE (%)*	*MARE (%)*	*VARE (%)*
(0,2)	10.12	43.73	10.91	46.55	10.90	46.63
(0,3)	7.66	28.23	7.60	28.78	7.61	28.81
(1,5)	4.06	8.38	3.95	7.61	3.95	7.60
(2,6)	2.11	2.29	2.06	2.00	2.06	2.00
(3,7)	1.12	0.65	1.05	0.51	1.05	0.52
(4,8)	0.62	0.20	0.53	0.13	0.53	0.13
(5,9)	0.37	0.06	0.26	0.03	0.27	0.03

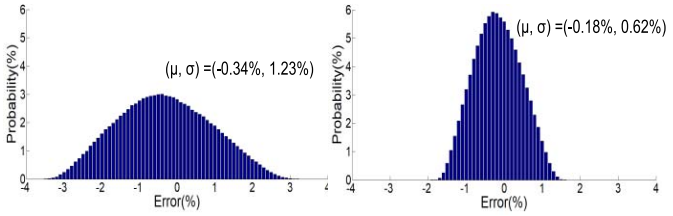

Fig. 8. Error distribution of (a) TOSAM(3,7) and (b) TOSAM(4,8) structures.

TABLE II

MAXARE, MRE, MARE, VARE, MAX_NED, AND NED OF DIFFERENT APPROXIMATE 32-BIT MULTIPLIERS

Architecture	*maxARE (%)*	*MRE (%)*	*MARE (%)*	*VARE (%)*	*max_NED*	*NED*
DSM(3)	36.00	-16.1	16.10	40.43	0.2344	0.0399
DRUM(3)	56.25	2.1	11.90	79.96	0.2344	0.0281
TOSAM(0,2)	31.25	-9.1	10.90	46.63	0.3125	0.0309
DSM(4)	21.00	-8.4	8.35	12.14	0.1211	0.0204
TOSAM(0,3)	25.00	-3.3	7.61	28.81	0.2500	0.0213
DRUM(4)	26.56	0.5	5.90	18.26	0.1211	0.0141
DSM(5)	11.42	-4.3	4.25	3.32	0.0615	0.0103
LETAM(3)	9.72	-4.0	4.00	2.54	0.0859	0.0104
TOSAM(1,5)	13.89	-0.7	3.95	7.60	0.1250	0.0104
U-RoBA	11.10	0	2.89	6.37	0.0625	0.0069
DRUM(5)	12.86	0.1	2.94	4.45	0.0615	0.0070
LETAM(4)	6.25	-2.4	2.37	0.88	0.0508	0.0061
DSM(6)	5.97	-2.2	2.15	0.87	0.0310	0.0052
TOSAM(2,6)	6.87	-0.6	2.06	2.00	0.0664	0.0053
DRUM(6)	6.35	0	1.47	1.11	0.0310	0.0035
DSM(7)	3.05	-1.1	1.08	0.22	0.0156	0.0026
TOSAM(3,7)	3.65	-0.3	1.05	0.52	0.0342	0..0027
DRUM(7)	3.15	0	0.73	0.28	0.0156	0.0018
DSM(8)	1.54	-0.5	0.54	0.06	0.0078	0.0013
TOSAM(4,8)	1.88	-0.2	0.53	0.13	0.0173	0.0013
DRUM(8)	1.57	0	0.37	0.07	0.0078	0.0009
TOSAM(5,9)	0.96	-0.1	0.27	0.03	0.0087	0.0007
LETAM(8)	0.81	-0.2	0.24	0.01	0.0051	0.0006

units of 8-, 16-, and 32-bit TOSAM structures with constant h and t parameters are the same with almost equal MARE and VARE parameters (see Table I). This provides us with higher speed and energy consumption improvements for multipliers with higher widths. In addition, this provides us with more opportunity to increase the accuracy for the TOSAM structures compared to the exact one for a given improvement. More specifically, it is due to the fact that when increasing the multiplier width, the delay of the exact multiplier increases at higher rates compared to those of the TOSAM structures.

Error distributions of the proposed approximate 32-bit multiplier in the case of $h = 3$ and $h = 4$ are depicted in Fig. 8. As expected, the error distribution of TOSAM(3,7) is wider than TOSAM(4,8) while having a smaller peak value. In other words, TOSAM(3,7) structure produces outputs with higher RE values which results in a wider error distribution. Both designs have almost normal error distribution with an MRE value near zero.

To evaluate the accuracy of TOSAM compared to those of the other approximate multipliers, we have measured their mean of RE values (MRE), maxARE, MARE, VARE, maximum normalized error distance (max_NED), and NED [26] using random vectors. The results are presented in Table II (sorted based on their MARE values). We denote DSM (DRUM) structures by DSM(m) (DRUM(m)), where m shows the bit length of the truncated (rounded) input operands. In the rest of this paper, we use the MARE parameter to compare the accuracy of the approximate multipliers.

VI. RESULTS AND DISCUSSION

In this section, the design parameters of the proposed structure are compared with some state-of-the-art approximate multipliers. In this paper, we have considered the prior works that have almost similar MARE to that of our proposed structure. All the considered multipliers have been described by Verilog HDL and synthesized using Synopsys Design Compiler in a 45-nm Nangate technology [27] at typical corner. The delay, power, and area of the proposed structure have been extracted and compared with those of the exact Wallace multiplier, LETAM [5], DRUM [6], DSM [1], RoBA [7], and DQ4:2C4 [9] architectures. We have implemented both signed and unsigned 8-, 16-, and 32-bit multipliers to show how the delay, power, and area are improved by increasing the width of the multiplier operands. The exact signed multiplier and signed DQ4:2C4 were implemented based on Baugh-Wooley algorithm. It should be noted that in [1], no hardware implementation for signed DSM was reported. Hence, we compared the design parameters of our proposed signed multiplier with those of signed DQ4:2C4 [9], AS-RoBA [7], signed DRUM [6], and signed LETAM [5] algorithms.

A. Design Parameters Extraction

1) 8-Bit Multipliers: We have reported design parameters of the considered exact and approximate unsigned 8-bit multipliers in Table III. As the results reveal, by employing the proposed method, we can improve the delay, area, energy, energy-delay product (EDP), and power-delay-area (PDA) product of the multiplication by up to 32%, 56%, 77%, 84%, and 90%, respectively, compared to those of the exact 8-bit multiplier. Also, our proposed multipliers operate at higher frequencies compared to the other approximate multipliers with almost the same accuracy level. For example, TOSAM(0,3) improves delay and energy about 32% and 43% compared to those of DRUM(4). TOSAM(0,3) improves the energy consumption and area occupation about 8% and 30%

TABLE III

DELAY, POWER, AREA, ENERGY, EDP, PDA, AND MARE OF DIFFERENT UNSIGNED 8-BIT MULTIPLIERS

Architecture	Delay (ns)	Power (μW)	Area (μm^2)	Energy (fJ)	EDP (fJ×ns)	PDA ($pJ\times\mu m^2$)	*MARE* (%)
Wallace	0.85	360	417	306	260	127.6	0.0
DSM(3)	0.80	128	182	102	82	18.6	14.4
DRUM(3)	0.70	104	143	73	51	10.4	12.6
TOSAM(0,2)	0.58	120	186	70	40	12.9	10.1
DQ4:2C4	0.57	186	281	106	60	29.8	8.1
TOSAM(0,3)	0.68	144	198	98	67	19.4	7.7
DSM(4)	1.08	205	233	221	239	51.6	6.8
DRUM(4)	1.00	172	208	172	172	35.8	6.4
TOSAM(1,5)	0.88	231	291	203	179	59.2	4.1
DSM(5)	1.22	328	355	400	488	142.1	3.0
LETAM(3)	1.00	270	310	270	270	83.7	2.9

TABLE IV

DELAY, POWER, AREA, ENERGY, EDP, PDA, AND MARE OF DIFFERENT SIGNED 8-BIT MULTIPLIERS

Architecture	Delay (ns)	Power (μW)	Area (μm^2)	Energy (fJ)	EDP (fJ×ns)	PDA ($pJ\times\mu m^2$)	*MARE* (%)
Wallace	0.89	384	416	342	304	142	0.0
DRUM(3)	1.15	236	290	271	312	79	13.4
TOSAM(0,2)	0.71	221	252	157	111	40	12.1
DQ4:2C4	0.58	195	293	113	66	33	1046.2
TOSAM(0,3)	0.82	234	247	192	157	47	9.1
DRUM(4)	1.37	382	389	523	717	204	7.1
TOSAM(1,5)	1.02	306	269	312	318	84	6.0
LETAM(3,4)	1.28	438	399	561	718	224	2.7

TABLE V

DELAY, POWER, AREA, ENERGY, EDP, PDA, AND MARE OF DIFFERENT UNSIGNED 16-BIT MULTIPLIERS

Architecture	Delay (ns)	Power (mW)	Area (μm^2)	Energy (pJ)	EDP (pJ×ns)	PDA ($pJ\times\mu m^2$)	*MARE* (%)
Wallace	1.22	2.08	1785	2.54	3.10	4530	0.0
DSM(3)	0.97	0.20	344	0.19	0.19	67	16.1
DRUM(3)	0.88	0.13	257	0.11	0.10	29	11.9
TOSAM(0,2)	0.74	0.16	342	0.12	0.09	40	10.9
DQ4:2C4	0.82	0.80	1078	0.65	0.53	703	9.1
DSM(4)	1.16	0.29	442	0.34	0.40	151	8.3
TOSAM(0,3)	0.84	0.21	423	0.18	0.15	76	7.6
DRUM(4)	1.12	0.27	381	0.30	0.34	115	5.9
DSM(5)	1.42	0.42	525	0.59	0.84	310	4.2
LETAM(3)	1.16	0.39	608	0.46	0.53	278	4.0
TOSAM(1,5)	1.00	0.35	532	0.35	0.35	185	4.0
DRUM(5)	1.36	0.43	532	0.59	0.80	313	2.9
U-RoBA	1.05	0.55	1438	0.57	0.60	826	2.9
LETAM(4)	1.25	0.44	620	0.54	0.68	337	2.4
DSM(6)	1.50	0.63	716	0.94	1.41	672	2.1
TOSAM(2,6)	1.21	0.38	564	0.46	0.56	261	2.1

compared to those of DQ4:2C4. Also, TOSAM designs with less MARE values can operate at higher frequencies and consume less energy compared to DSM designs. TOSAM(1,5) can operate at 14% higher frequency and consume 25% less energy compared to LETAM(3) with almost the same accuracy level.

The results of implementing exact and approximate signed 8-bit multipliers are given in Table IV which indicates that the proposed method improves the speed, area, and energy consumption of signed multiplication up to 20%, 39%, and 54%, respectively, compared to those of the exact multiplier. Also, TOSAM(1,5) can operate at 34% higher

TABLE VI

DELAY, POWER, AREA, ENERGY, EDP, PDA, AND MARE OF DIFFERENT SIGNED 16-BIT MULTIPLIERS

Architecture	Delay (ns)	Power (mW)	Area (μm^2)	Energy (pJ)	EDP (pJ×ns)	PDA ($pJ\times\mu m^2$)	*MARE* (%)
Wallace	1.22	2.16	1794	2.64	3.21	4728	0
DRUM(3)	1.48	0.38	647	0.56	0.83	361	11.9
TOSAM(0,2)	0.95	0.32	473	0.30	0.29	144	10.9
DQ4:2C4	0.83	0.77	1067	0.64	0.53	683	3745.3
TOSAM(0,3)	1.05	0.37	458	0.39	0.41	178	7.6
DRUM(4)	1.79	0.55	761	0.98	1.76	749	5.9
LETAM(3,4)	1.38	0.67	724	0.93	1.28	672	4.0
TOSAM(1,5)	1.23	0.55	604	0.68	0.84	412	4.0
DRUM(5)	1.99	0.77	918	1.52	3.03	1398	2.9
AS-RoBA	1.20	0.98	1469	1.17	1.41	1720	2.9
LETAM(4,5)	1.45	0.8	869	1.16	1.68	1008	2.4
TOSAM(2,6)	1.35	0.69	700	0.93	1.25	650	2.1

frequency and consume 40% less energy compared to DRUM(4) with almost the same MARE values. The MARE value of signed DQ4:2C4 is extremely high owing to the fact that the compressors employed in this multiplier generate the carry signals approximately. This causes that, in some cases, the output becomes positive while it should be negative and vice versa giving rise to high degradation in the accuracy of multiplication.

2) 16-Bit Multipliers: We have implemented the considered exact and approximate unsigned 16-bit multipliers and reported their design parameters in Table V. The proposed method improves the speed, area, and energy up to 39%, 81%, and 95%, respectively, compared to those of the exact multiplier. TOSAM(0,2) operates at 11% higher frequency, consumes 80% less energy, and occupies 68% less area compared to those of DQ4:2C4. TOSAM(1,5) structure with MARE value equal to 4% improves the speed of the multiplication by about 18% while consuming 86% less energy as well as occupying 70% less area compared to the exact Wallace multiplier. Also, the delay of the high accuracy TOSAM(2,6) structure, with MARE value equal to 2.1%, is smaller than that of the exact Wallace multiplier while consuming about 82% less energy. TOSAM structure, on average, improves the delay, power, area, energy, EDP, and PDA parameters by 22%, 87%, 74%, 89%, 91%, and 97%, respectively, compared to the exact Wallace multiplier. In addition, the delays of the proposed structures are lower than those of the prior approximate multipliers with almost the same accuracy level. For example, TOSAM(2,6) structure can operate at about 24% higher frequency compared to that of DSM(6) while consuming 51% less energy and occupying 21% less area. In addition, although the delays of the TOSAM(1,5) and U-RoBA multiplier are almost the same, the area and power consumption of the TOSAM(1,5) are 63% and 36% lower, respectively, compared to those of U-RoBA multiplier. Moreover, the proposed multipliers outperform LETAM structures in terms of the speed, area, and power consumption.

We have reported the design parameters of the exact and approximate implementations of the signed 16-bit multipliers in Table VI. Again, the proposed method can improve the

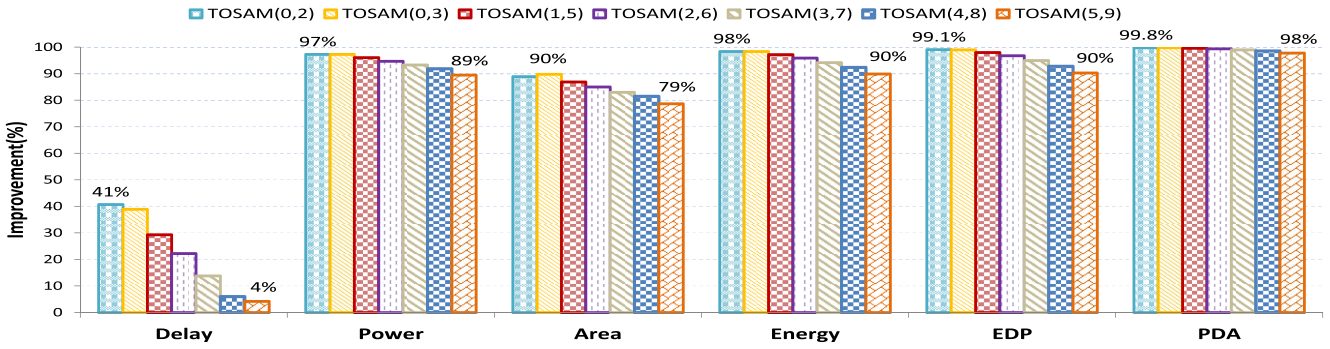

Fig. 9. Delay, power, area, energy, EDP, and PDA improvements of the proposed multipliers compared to the exact unsigned 32-bit multiplier.

TABLE VII
DELAY, POWER, AREA, ENERGY, EDP, PDA, AND MARE OF DIFFERENT UNSIGNED 32-BIT MULTIPLIERS

Architecture	Delay (ns)	Power (mW)	Area (μm^2)	Energy (pJ)	EDP (pJ×ns)	PDA (pJ×μm^2)	*MARE* (%)
Wallace	1.67	10.26	7618	17.13	28.61	130528	0.00
DSM(3)	1.20	0.30	640	0.35	0.42	227	16.10
DRUM(3)	1.08	0.20	520	0.22	0.23	112	11.90
TOSAM(0,2)	0.99	0.27	844	0.27	0.27	229	10.90
DQ4:2C4	1.02	3.03	4070	3.09	3.15	12579	9.12
DSM(4)	1.40	0.46	874	0.64	0.90	562	8.35
TOSAM(0,3)	1.02	0.28	780	0.28	0.29	219	7.61
DRUM(4)	1.33	0.36	738	0.48	0.64	352	5.90
DSM(5)	1.60	0.58	1004	0.92	1.47	924	4.25
LETAM(3)	1.28	0.50	1080	0.64	0.82	688	4.00
TOSAM(1,5)	1.18	0.40	999	0.47	0.56	474	3.95
DRUM(5)	1.55	0.56	944	0.87	1.35	819	2.89
U-RoBA	1.25	1.23	4739	1.54	1.92	7286	2.94
LETAM(4)	1.42	0.61	1208	0.87	1.23	1048	2.37
DSM(6)	1.72	0.78	1131	1.34	2.30	1513	2.15
TOSAM(2,6)	1.30	0.54	1146	0.71	0.92	810	2.06
DRUM(6)	1.69	0.75	1059	1.26	2.13	1335	1.47
DSM(7)	1.88	0.94	1287	1.76	3.31	2265	1.08
TOSAM(3,7)	1.44	0.69	1294	1.00	1.43	1288	1.05
DRUM(7)	1.85	0.96	1235	1.77	3.27	2182	0.73
DSM(8)	1.92	1.18	1494	2.27	4.35	3385	0.54
TOSAM(4,8)	1.57	0.83	1411	1.31	2.05	1845	0.53
DRUM(8)	1.93	1.17	1402	2.26	4.36	3166	0.37
TOSAM(5,9)	1.60	1.08	1625	1.73	2.76	2808	0.27
LETAM(8)	1.65	1.10	1651	1.82	2.99	2997	0.24

speed, power, area, energy, EDP, and PDA of signed 16-bit multiplication up to 22%, 85%, 74%, 88%, 91%, and 97%, respectively, compared to those of the exact one. Also, signed TOSAM(1,5) with an average ARE equal to ∼4% can operate at almost the same speed as the exact multiplier while consuming 74% less energy and occupying 66% less area. TOSAM(1,5) improves the energy and area about 42% and 59%, respectively, compared to the case of AS_RoBA while operating at an almost the same frequency.

3) 32-Bit Multipliers: We have reported the design parameters of implementing the considered unsigned 32-bit multipliers in Table VII. As the results suggest, employing TOSAM(0,2) improves the delay, power, area, energy, EDP, and PDA of multiplication by 41%, 97%, 89%, 98%, 99%, and 99.8%, respectively, compared to those of the exact Wallace multiplier. Also, employing TOSAM structure [TOSAM(0,2) to TOSAM(5,9)] leads, on average, to about 22%, 94%, 85%, 95%, 96%, and 99.2% improvements of delay, power, area, energy, EDP, and PDA, respectively, compared to the those of exact multiplier.

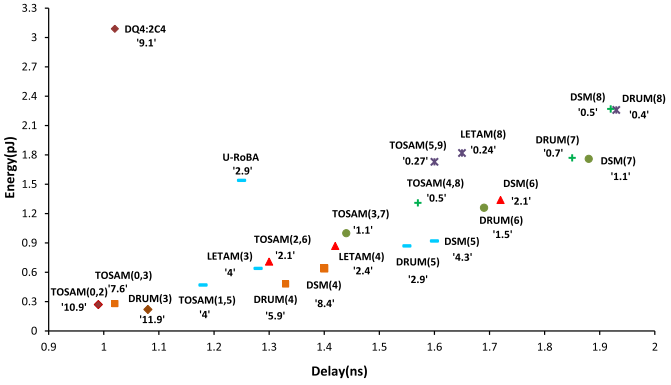

Fig. 10. Energy and delay comparison of the proposed and other approximate multipliers.

To demonstrate the efficacy of the proposed multiplier, its delay, power, area, energy, EDP, and PDA improvements (based on Table VII), compared to the exact Wallace multiplier are depicted in Fig. 9. As shown in Fig. 9, for all of the structures, the reduction in energy consumption and area occupation is more than ∼90% and ∼79%, respectively, while the speed of multiplication is improved in the range of 4%–41%.

Also, the results reveal that the proposed multipliers outperform other approximate structures in terms of speed and energy consumption while having almost the same MARE values. To demonstrate this better, we have depicted a diagram in Fig. 10 showing the delay and energy consumption of the proposed structure compared to the other considered approximate multipliers. In Fig. 10, the structures that have almost equal MARE values are shown with the same sign and color. The MARE value of each design is written beside it, inside a single quotation mark. For example, TOSAM(2,6), DSM(6), and LETAM(4) structures shown with red triangle sign have almost similar MARE values. Among these multipliers, TOSAM(2,6) outperforms the others in terms of delay and energy consumption. For the desired accuracy, a TOSAM structure is available which has better speed compared to the other approximate multipliers.

By comparing the results in Tables III, V, and VII, it is observed that the delay, area, and energy savings are improved

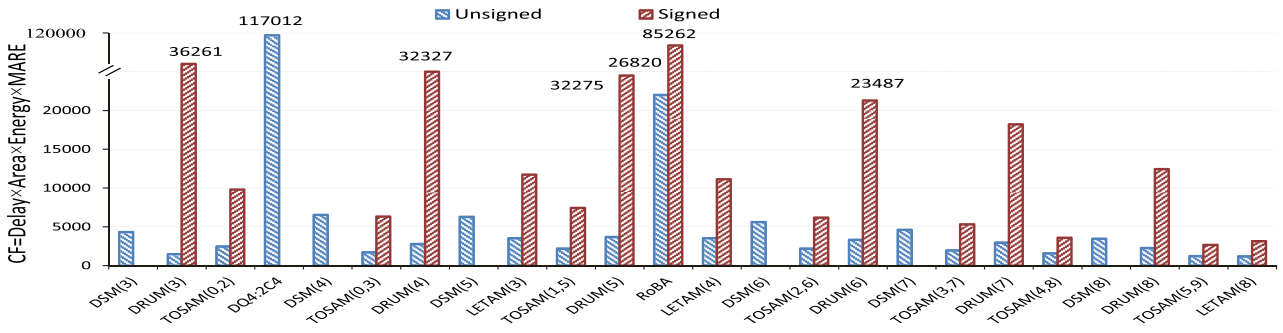

Fig. 11. CF parameters for the considered unsigned and signed approximate 32-bit multipliers.

TABLE VIII

DELAY, POWER, AREA, ENERGY, EDP, PDA, AND MARE OF DIFFERENT SIGNED 32-BIT MULTIPLIERS

Architecture	Delay (ns)	Power (mW)	Area (μm^2)	Energy (pJ)	EDP (pJ×ns)	PDA (pJ×μm^2)	*MARE* (%)
Wallace	1.65	10.6	7619	17.49	28.9	133256	0.00
DRUM(3)	1.89	0.62	1378	1.17	2.2	1615	11.90
TOSAM(0,2)	1.16	0.58	1156	0.67	0.8	776	10.92
DQ4:2C4	1.03	3.03	4114	3.12	3.2	12839	11275.10
TOSAM(0,3)	1.15	0.62	1002	0.72	0.8	719	7.61
DRUM(4)	2.15	0.78	1526	1.67	3.6	2553	5.90
LETAM(3,4)	1.54	0.91	1347	1.41	2.2	1894	4.01
TOSAM(1,5)	1.40	0.83	1159	1.16	1.6	1347	3.95
DRUM(5)	2.35	1.09	1831	2.56	6.0	4690	2.93
AS-RoBA	1.46	2.66	5208	3.88	5.7	20226	2.89
LETAM(4,5)	1.70	1.10	1478	1.87	3.2	2764	2.37
TOSAM(2,6)	1.52	0.96	1343	1.47	2.2	1968	2.06
DRUM(6)	2.55	1.28	1922	3.26	8.3	6273	1.47
TOSAM(3,7)	1.70	1.18	1482	2.01	3.4	2973	1.05
DRUM(7)	2.65	1.65	2154	4.37	11.6	9418	0.73
TOSAM(4,8)	1.73	1.36	1665	2.35	4.1	3917	0.53
DRUM(8)	2.78	1.89	2305	5.25	14.6	12111	0.37
TOSAM(5,9)	1.80	1.64	1854	2.95	5.3	5473	0.27
LETAM(8,7)	1.95	1.70	2019	3.32	6.5	6693	0.24

as the width of the multiplier increases. This is due to the fact that by increasing the multiplier width, the number of partial products increases quadratically, resulting in a high increase in the delay and energy consumption of the exact multiplier. In the proposed multiplier, the calculation core remains unchanged as the width of the multiplier increases (for the desired accuracy level). Hence, higher gains in speed, area occupation, and energy consumption are achieved as the multiplier width increases.

The design parameters of the signed 32-bit multipliers are reported in Table VIII. As the results reveal, all of the proposed approximate signed multipliers, on average, consume 91% less energy and occupy 82% less area compared to the exact signed multiplier. Also, TOSAM structures have less energy and area compared to the DRUM, DQ4:2C4, and AS-RoBA multipliers with almost the same accuracy level. For example, the delay, area, and energy of TOSAM(1,5) are almost 4%, 78%, and 70% lower, respectively, compared to those of the AS-RoBA multiplier.

By comparing the results of Tables VII and VIII, it is observed that, on average, the delay, power, and area of the signed TOSAM structures are increased almost 15%, 87%, and 21%, respectively, compared to those of the unsigned ones.

TABLE IX

DELAY, POWER, AND ENERGY COMPARISON OF DIFFERENT OPERATING MODES OF 32-BIT ACCURACY CONFIGURABLE TOSAM

Operating mode	Delay (ns)	Power (µW)	Energy (fJ)
T2	1.60	146	234
T6	2.05	165	338
T9	2.30	234	538

To compare the efficacy of different approximate multipliers as a function of accuracy and design parameters, we define the cost function (CF) as

$$\text{CF} = \text{Delay} \times \text{Energy} \times \text{Area} \times \text{MARE}. \tag{14}$$

The results for the approximate unsigned and signed 32-bit multipliers have been presented in Fig. 11. It is worthy to mention that the MARE of signed DQ4:2C4 is significantly higher than those of other considered approximate multipliers, and hence, we have not depicted its CF parameter in Fig. 11. To show the differences between the CF parameters better, we have split the vertical axis into two parts, below 20 000 and between 20 000 and 120 000. The value of CF parameters higher than 20 000 is written beside each bar. As the results reveal, the CF values of the proposed multiplier are smaller than those of the other approximate multipliers with almost the same MARE values, except for the unsigned LETAM(8) whose CF value is almost the same as that of the unsigned TOSAM(5,9) and DRUM(3) which has smaller area occupation compared to TOSAM(0,2). This shows the superiority of the proposed method compared to the other considered approximate multipliers.

4) Accuracy Configurable TOSAM: In this section, we have implemented an accuracy configurable 32-bit TOSAM in HSPICE in the 45-nm technology and measured its delay and power consumption for all of its operating modes. The considered configurable TOSAM has three different operating modes, called T2, T6, and T9 corresponding to TOSAM(0,2), TOSAM(2,6), and TOSAM(5,9), respectively. The results are reported in Table IX. As was expected, operating in the lowest accuracy mode (T2) results in the highest speed and the lowest power consumption compared to the other modes.

Fig. 12. Sharpened Female benchmark employing exact and approximate multipliers.

(a) (b) (c) (d)

Fig. 13. Compressed (a) and (c) exact and (b) and (d) approximate images of Baboon and Lena benchmarks generated using exact and TOSAM(0,2) multipliers.

B. Applications

1) Image Processing Application: As mentioned before, AC is a solution for improving the efficiency of computation in image processing applications. To assess the efficacy of the proposed multipliers, we have exploited them in different image processing applications such as sharpening and the JPEG image compression. In this section, several benchmarks such as Baboon and Lena [28] were employed. In the sharpening application, peak-signal-to-noise ratio (PSNR) and Structural SIMilarity (SSIM) [29] of the approximate outputs compared to the exact output images were extracted using MATLAB simulations.

In the sharpening application, each output pixel can be extracted from [30]

$$Y(i,j) = 2X(i,j) - \frac{1}{273}\sum_{m=-2}^{2}\sum_{n=-2}^{2} \times\left(\begin{matrix} X(i+m, j+n)\times \\ \text{Mask}_{\text{sharpening}}(m+3, n+3)\end{matrix}\right) \quad (15)$$

where $X(i,j)$ and $Y(i,j)$ denote the pixel of the ith row and the jth column of the input and output images, respectively, and $\text{Mask}_{\text{sharpening}}$ is a 5×5 sharpening coefficient matrix given by

$$\text{Mask}_{\text{sharpening}} = \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}. \quad (16)$$

The results of sharpening images using different approximate unsigned 8-bit multipliers have been presented in Table X. Also, we have implemented a sharpening unit by employing the Multiplier and Accumulator (MAC) module and measured the energy consumption of the MAC unit for different unsigned 8-bit multipliers. These results are also reported in Table X. For all the benchmarks, the SSIM is greater than 0.9 suggesting that the proposed approximate multipliers can do well in such applications without concerning about degrading the quality of the output image. TOSAM(0,2), with high-quality output images, improves the energy consumption significantly compared to the other multipliers. As an example, the output images of sharpening Female benchmark are depicted in Fig. 12. The differences between the exact and approximate sharpened images are slight and the quality of the output images is not degraded significantly.

As another application, we have compressed several images by employing JPEG algorithm and utilized the signed 16-bit approximate multipliers in the Discrete Cosine Transform (DCT) Unit. The PSNR and SSIM of the compressed images compared to those of the primary input images were extracted and reported in Table XI. As expected, by increasing the h value, PSNR and SSIM of the compressed images are improved. The results reveal that PSNR and SSIM of the approximate images compared to those of the exact ones are degraded slightly. Among the proposed multipliers, the maximum PSNR and SSIM reductions are 1.39 dB and 0.012, respectively, and occur for Lena and Baboon benchmarks compressed using TOSAM(0,2). These compressed images are shown in Fig. 13 to show that the difference between the exact and approximate compressed images is negligible and cannot be sensed by the human eye. As the results reveal, TOSAM(0,2) improves the energy consumption of the DCT

TABLE X

COMPARISONS OF PSNR (DECIBEL) AND SSIM AS WELL AS ENERGY (OF THE MAC UNIT) OF THE SHARPENING APPLICATION WHEN DIFFERENT UNSIGNED 8-BIT MULTIPLIERS ARE USED

Benchmark	*Aerial*		*Clock*		*Moon Surface*		*Tree*		*Female*		*Baboon*		*House*		*Splash*		Energy (fJ)
	PSNR	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	
Wallace	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	615
TOSAM(0,2)	22.9	0.96	22.0	0.92	23.5	0.91	23.2	0.94	22.6	0.97	23.7	0.98	20.8	0.94	24.1	0.97	212
TOSAM(1,5)	28.5	0.96	26.0	0.91	29.6	0.95	28.8	0.95	33.7	0.97	27.8	0.98	28.1	0.94	28.9	0.95	533
DQ4:2C4	20.0	0.94	20.9	0.93	21.0	0.91	22.1	0.94	22.3	0.90	21.2	0.96	20.1	0.80	23.2	0.97	246
DSM(3)	18.3	0.95	18.9	0.93	18.9	0.94	19.0	0.95	18.3	0.92	19.0	0.97	17.4	0.92	21.1	0.96	363
DSM(4)	26.8	0.99	24.2	0.97	26.8	0.97	26.1	0.97	25.3	0.89	27.4	0.99	25.1	0.97	27.4	0.99	580
DSM(5)	32.8	□ 1	30.2	0.98	32.8	0.99	32.8	0.99	32.6	0.97	33.4	□ 1	31.6	0.99	33.9	□ 1	859
DRUM(4)	24.5	0.95	19.1	0.90	24.0	0.90	24.6	0.92	22.4	0.90	24.6	0.97	23.5	0.90	23.0	0.94	470
LETAM(3)	31.1	0.99	30.0	0.99	31.7	0.99	31.5	0.99	29.9	0.98	31.7	□ 1	30.0	0.99	32.8	□ 1	657

TABLE XI

COMPARISONS OF PSNR (DECIBEL) AND SSIM AS WELL AS ENERGY (OF THE DCT UNIT) OF THE JPEG ENCODER WHEN DIFFERENT SIGNED 16-BIT MULTIPLIERS ARE USED

Benchmark	*Baboon*		*Boat*		*House*		*Lena*		*Peppers*		*Splash*		*Energy (pJ)*
	PSNR	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	
Exact	26.21	0.913	28.65	0.942	30.39	0.955	32.27	0.973	30.29	0.979	33.23	0.984	836
TOSAM(0,2)	25.56	0.901	27.68	0.931	29.23	0.944	30.88	0.967	29.47	0.974	32.20	0.977	187
TOSAM(1,5)	25.97	0.910	28.25	0.938	29.97	0.951	31.74	0.971	29.99	0.977	32.83	0.981	330
TOSAM(2,6)	26.13	0.912	28.55	0.941	30.27	0.954	32.14	0.973	30.20	0.978	33.14	0.983	388
DRUM(4)	25.48	0.904	27.64	0.931	29.22	0.944	30.84	0.966	29.44	0.974	32.25	0.976	339
DRUM(5)	26.07	0.912	28.47	0.94	30.16	0.954	32.01	0.972	30.13	0.978	33.06	0.983	440
AS-RoBA	26.00	0.910	28.38	0.939	30.07	0.952	31.86	0.972	30.02	0.977	32.98	0.982	319
LETAM(3,4)	26.19	0.912	28.61	0.941	30.32	0.954	32.21	0.973	30.25	0.978	33.18	0.984	388
LETAM(4,5)	26.21	0.912	28.64	0.942	30.35	0.955	32.26	0.973	30.27	0.979	33.21	0.984	395

TABLE XII

ACCURACY AND ENERGY CONSUMPTION COMPARISON OF MNIST CLASSIFICATION APPLICATION FOR DIFFERENT SIGNED 16-BIT MULTIPLIERS

Architecture	Wallace	TOSAM(0,2)	TOSAM(1,5)	DRUM(4)	LETAM(3,4)
Accuracy (%)	94.9	93.8	94.6	94.7	94.5
Energy(nJ)	64.3	14.9	21.5	29.7	25.0

part up to 77%, while the quality of its output images is almost the same as the exact compressed.

2) Classification Application: To estimate the efficacy of the proposed approximate multiplier in classification applications, we have trained a neural network offline with 50 hidden neurons for the MNIST data set [31]. Then, we have employed signed 16-bit approximate multipliers in the structure of the network and measured the classification accuracy of the network for the test samples. Also, the energy consumption of the neurons was measured by employing Synopsys Design Compiler. The results are reported in Table XII. As the results reveal, the classification accuracy of the network is not degraded significantly by employing the TOSAM structures while the energy consumption has been improved up to 77%. Also, the TOSAM structures have lower energy consumption compared to DRUM(4) and LETAM(3, 4) while their classification accuracies are almost the same.

VII. CONCLUSION

In this paper, we suggested a low-energy and area-efficient approximate multiplier in which the input operands were truncated with two different lengths, t and h, and then rounded to the nearest odd numbers to reduce the error resulted by the truncation operation. The proposed multiplier was scalable and outperformed other approximate multipliers in terms of speed, area, and energy. The proposed 32-bit multiplier, on average, improved the energy consumption 95% compared to the exact Wallace multiplier while occupied 85% less area. The latency and power consumption of the multiplication were improved in the range of 4%–41% and 89%–97%, respectively, compared to those of the exact multiplication. Comparing with the exact multiplier, the speed, area, and energy improvements of the proposed multiplier became better as the multiplier width increased. That was due to the simple and scalable calculation core of the proposed multiplier. Also, the high accuracy of the proposed multiplier made is a good choice to be exploited in image processing and classification applications.

REFERENCES

[1] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 6, pp. 1180–1184, Jun. 2015.

[2] S. Xu and B. C. Schafer, "Exposing approximate computing optimizations at different levels: From behavioral to gate-level," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 11, pp. 3077–3088, Nov. 2017.

[3] A. Raghunathan and K. Roy, "Approximate computing: Energy-efficient computing with good-enough results," in *Proc. IEEE 19th Int. On-Line Test. Symp. (IOLTS)*, Chania, Greece, Jul. 2013, p. 258.

[4] D. Jeon, "Energy-efficient digital signal processing hardware design," Ph.D. dissertation, Dept. Elect. Eng., Michigan Univ., Ann Arbor, MI, USA, 2014.

[5] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "LETAM: A low energy truncation-based approximate multiplier," *Comput. Elect. Eng.*, vol. 63, pp. 1–17, Oct. 2017.

[6] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, Nov. 2015, pp. 418–425.

[7] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram, "RoBa multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 393–401, Feb. 2017.

[8] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, "Approximate hybrid high radix encoding for energy-efficient inexact multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 3, pp. 421–430, Mar. 2018.

[9] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 4, pp. 1352–1361, Apr. 2017.

[10] M. Ha and S. Lee, "Multipliers with approximate 4–2 compressors and error recovery modules," *IEEE Embedded Syst. Lett.*, vol. 10, no. 1, pp. 6–9, Mar. 2018.

[11] 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.

[12] D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro, and N. Petra, "Approximate multipliers based on new approximate compressors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 12, pp. 4169–4182, Dec. 2018.

[13] D. Esposito, D. De Caro, E. Napoli, N. Petra, and A. G. M. Strollo, "On the use of approximate adders in carry-save multiplier-accumulators," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Baltimore, MD, USA, May 2017, pp. 1–4.

[14] H. J. Ko and S. F. Hsiao, "Design and application of faithfully rounded and truncated multipliers with combined deletion, reduction, truncation, and rounding," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 5, pp. 304–308, May 2011.

[15] A. Lingamneni, A. Basu, C. Enz, K. V. Palem, and C. Piguet, "Improving energy gains of inexact DSP hardware through reciprocative error compensation," in *Proc. 50th ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, Austin, TX, USA, May/Jun. 2013, pp. 1–8.

[16] 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. Comput.*, vol. 66, no. 8, pp. 1435–1441, Aug. 2017.

[17] S. Venkatachalam, H. J. Lee, and S.-B. Ko, "Power efficient approximate booth multiplier," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Florence, Italy, May 2018, pp. 1–4.

[18] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate radix-8 booth multipliers for low-power and high-performance operation," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2638–2644, Aug. 2016.

[19] I. Alouani, H. Ahangari, O. Ozturk, and S. Niar, "A novel heterogeneous approximate multiplier for low power and high performance," *IEEE Embedded Syst. Lett.*, vol. 10, no. 2, pp. 45–48, Jun. 2018.

[20] K. H. Abed and R. E. Siferd, "CMOS VLSI implementation of a low-power logarithmic converter," *IEEE Trans. Comput.*, vol. 52, no. 11, pp. 1421–1433, Nov. 2003.

[21] M. Combet, H. Van Zonneveld, and L. Verbeek, "Computation of the base two logarithm of binary numbers," *IEEE Trans. Electron. Comput.*, vol. EC-14, no. 6, pp. 863–867, Dec. 1965.

[22] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Trans. Electron. Comput.*, vol. EC-11, no. 4, pp. 512–517, Aug. 1962.

[23] Z. Babic, A. Avramovic, and P. Bulic, "An iterative Mitchell's algorithm based multiplier," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Sarajevo, Serbia, Dec. 2008, pp. 303–308.

[24] M. S. Kim, A. A. Del Barrio, R. Hermida, and N. Bagherzadeh, "Low-power implementation of Mitchell's approximate logarithmic multiplication for convolutional neural networks," in *Proc. 23rd Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jeju, South Korea, Jan. 2018, pp. 617–622.

[25] S. Vahdat, M. Kamal, A. Afzali-Kusha, M. Pedram, and Z. Navabi, "TruncApp: A truncation-based approximate divider for energy efficient DSP applications," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Lausanne, Switzerland, Mar. 2017, pp. 1635–1638.

[26] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 984–994, Apr. 2015.

[27] Nangate. *45 nm Open Cell Library*. Accessed: 2012. [Online]. Available: http://www.nangate.com

[28] *The USC-SIPI Image Database*. Accessed: Jun. 2017. [Online]. Available: http://sipi.usc.edu/database

[29] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[30] H. R. Myler and A. R. Weeks, *The Pocket Handbook of Image Processing Algorithms in C*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2009.

[31] Y. LeCun, C. Cortes, and C. J. C. Burges. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available:http://yann.lecun.com/exdb/mnist

Shaghayegh Vahdat received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Tehran, Tehran, Iran, in 2014 and 2016, respectively, where she is currently working toward the Ph.D. degree in electrical engineering at the School of Electrical and Computer Engineering.

Her current research interests include approximate computing, neuromorphic computing, hardware implementation of neural systems, robustness of neural networks at the presence of process variation, and low-power high-performance design of digital arithmetic units.

Mehdi Kamal received the B.Sc. degree in computer engineering from the Iran University of Science and Technology, Tehran, Iran, in 2005, the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, in 2007, and the Ph.D. degree in computer engineering from the University of Tehran, Tehran, in 2013.

He is currently an Assistant Professor at the School of Electrical and Computer Engineering, University of Tehran. His current research interests include reliability in nanoscale design, approximate computing, neuromorphic computing, design for manufacturability, embedded systems design, and low-power design.

Ali Afzali-Kusha received the B.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 1988, the M.Sc. degree in electrical engineering from the University of Pittsburgh, Pittsburgh, PA, USA, in 1991, and the Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 1994.

From 1994 to 1995, he was a Post-Doctoral Fellow at the University of Michigan. Since 1995, he has been at the University of Tehran, where he is currently a Professor of the School of Electrical and Computer Engineering and the Director of the Low-Power High-Performance Nanosystems Laboratory. He was a Research Fellow at the University of Toronto, Toronto, ON, Canada, and the University of Waterloo, Waterloo, ON, Canada, in 1998 and 1999, respectively. His current research interests include low-power high-performance design methodologies from the physical design level to the system level, new memory as well as digital design and implementation paradigms.

Massoud Pedram received the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 1991.

He is currently the Stephen and Etta Varra Professor at the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA. He has authored 4 books, 12 book chapters, and over 140 archival and 350 conference papers. He holds 10 U.S. patents. His current research interests include low-power electronics, energy-efficient processing, and cloud computing to photovoltaic cell power generation, energy storage, and power conversion, and RT level optimization of VLSI circuits to synthesis and physical design of quantum circuits.

Dr. Pedram was a recipient of six conference awards and two IEEE Transactions Best Paper Awards for the research with his students, and the 1996 Presidential Early Career Award for Scientists and Engineers and an ACM Distinguished Scientist. He was the Founding Technical Program Co-Chair of the 1996 International Symposium on Low-Power Electronics and Design and the Technical Program Chair of the 2002 International Symposium on Physical Design. He currently serves as the Editor-in-Chief for the *ACM Transactions on Design Automation of Electronic Systems*. He has served on the Technical Program Committee of a number of premier conferences in his field.