RoBA Multiplier: A Rounding-Based Approximate Multiplier for High-Speed yet Energy-Efficient Digital Signal Processing

Reza Zendegani, Mehdi Kamal, Milad Bahadori, Ali Afzali-Kusha, and Massoud Pedram

***Abstract*— In this paper, we propose an approximate multiplier that is high speed yet energy efficient. The approach is to round the operands to the nearest exponent of two. This way the computational intensive part of the multiplication is omitted improving speed and energy consumption at the price of a small error. The proposed approach is applicable to both signed and unsigned multiplications. We propose three hardware implementations of the approximate multiplier that includes one for the unsigned and two for the signed operations. The efficiency of the proposed multiplier is evaluated by comparing its performance with those of some approximate and accurate multipliers using different design parameters. In addition, the efficacy of the proposed approximate multiplier is studied in two image processing applications, i.e., image sharpening and smoothing.**

***Index Terms*— Accuracy, approximate computing, energy efficient, error analysis, high speed, multiplier.**

I. Introduction

ENERGY minimization is one of the main design requirements in almost any electronic systems, especially the portable ones such as smart phones, tablets, and different gadgets [1]. It is highly desired to achieve this minimization with minimal performance (speed) penalty [1]. Digital signal processing (DSP) blocks are key components of these portable devices for realizing various multimedia applications. The computational core of these blocks is the arithmetic logic unit where multiplications have the greatest share among all arithmetic operations performed in these DSP systems [2]. Therefore, improving the speed and power/energy-efficiency characteristics of multipliers plays a key role in improving the efficiency of processors.

Many of the DSP cores implement image and video processing algorithms where final outputs are either images or videos prepared for human consumptions. This fact enables us to use approximations for improving the speed/energy efficiency. This originates from the limited perceptual abilities of human beings in observing an image or a video. In addition to the image and video processing applications, there are other areas where the exactness of the arithmetic operations is not critical to the functionality of the system (see [3], [4]). Being able to use the approximate computing provides the designer with the ability of making tradeoffs between the accuracy and the speed as well as power/energy consumption [2], [5].

Applying the approximation to the arithmetic units can be performed at different design abstraction levels including circuit, logic, and architecture levels, as well as algorithm and software layers [2]. The approximation may be performed using different techniques such as allowing some timing violations (e.g., voltage overscaling or overclocking) and function approximation methods (e.g., modifying the Boolean function of a circuit) or a combination of them [4], [5]. In the category of function approximation methods, a number of approximating arithmetic building blocks, such as adders and multipliers, at different design levels have been suggested (see [6]–[8]).

In this paper, we focus on proposing a high-speed low-power/energy yet approximate multiplier appropriate for error resilient DSP applications. The proposed approximate multiplier, which is also area efficient, is constructed by modifying the conventional multiplication approach at the algorithm level assuming rounded input values. We call this rounding-based approximate (RoBA) multiplier. The proposed multiplication approach is applicable to both signed and unsigned multiplications for which three optimized architectures are presented. The efficiencies of these structures are assessed by comparing the delays, power and energy consumptions, energy-delay products (EDPs), and areas with those of some approximate and accurate (exact) multipliers. The contributions of this paper can be summarized as follows:

1) presenting a new scheme for RoBA multiplication by modifying the conventional multiplication approach;
2) describing three hardware architectures of the proposed approximate multiplication scheme for sign and unsigned operations.

The rest of this paper is organized as follows. Section II discusses the related works about approximate multipliers. The proposed scheme of the approximate multiplication, its hardware implementations, and its accuracy results are presented in Section III. In Section IV, the characteristics of the proposed approximate multiplier compared with the accurate and

Manuscript received January 6, 2016; revised April 26, 2016 and June 26, 2016; accepted June 27, 2016. The work of R. Zendegani, M. Kamal, M. Bahadori, and A. Afzali-Kusha was supported by the Iran National Science Foundation.

R. Zendegani, M. Bahadori, and A. Afzali-Kusha are with the School of Electrical and Computer Engineering, University of Tehran, Tehran 19967-15433, Iran (e-mail: zendegani_reza@ut.ac.ir; milad.bahadori@ut.ac.ir; afzali@ut.ac.ir).

M. Kamal is with the School of Electrical and Computer Engineering, University of Tehran, Tehran 19967-15433, Iran and also with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran 19538–33511, Iran (e-mail: mehdikamal@ut.ac.ir).

M. Pedram is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089 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.2016.2587696

approximate multipliers, and also its effectiveness in image processing applications are studied. Finally, the conclusion is drawn in Section V.

II. Prior Works

In this section, some of the previous works in the field of approximate multipliers are briefly reviewed. In [3], an approximate multiplier and an approximate adder based on a technique named broken-array multiplier (BAM) were proposed. By applying the BAM approximation method of [3] to the conventional modified Booth multiplier, an approximate signed Booth multiplier was presented in [5]. The approximate multiplier provided power consumption savings form 28% to 58.6% and area reductions from 19.7% to 41.8% for different word lengths in comparison with a regular Booth multiplier. Kulkarni *et al*. [6] suggested an approximate multiplier consisting of a number of 2×2 inaccurate building blocks that saved the power by 31.8%–45.4% over an accurate multiplier.

An approximate signed 32-bit multiplier for speculation purposes in pipelined processors was designed in [7]. It was 20% faster than a full-adder-based tree multiplier while having a probability of error of around 14%. In [8], an error-tolerant multiplier, which computed the approximate result by dividing the multiplication into one accurate and one approximate part, was introduced, in which the accuracies for different bit widths were reported. In the case of a 12-bit multiplier, a power saving of more than 50% was reported. In [9], two approximate 4:2 compressors for utilizing in a regular Dadda multiplier were designed and analyzed.

The use of approximate multipliers in image processing applications, which leads to reductions in power consumption, delay, and transistor count compared with those of an exact multiplier design, has been discussed in the literature. In [10], an accuracy-configurable multiplier architecture (ACMA) was suggested for error-resilient systems. To increase its throughput, the ACMA made use of a technique called carry-in prediction that worked based on a precomputation logic. When compared with the exact one, the proposed approximate multiplication resulted in nearly 50% reduction in the latency by reducing the critical path. Also, Bhardwaj *et al*. [11] presented an approximate Wallace tree multiplier (AWTM). Again, it invoked the carry-in prediction to reduce the critical path. In this work, AWTM was used in a real-time benchmark image application showing about 40% and 30% reductions in the power and area, respectively, without any image quality loss compared with the case of using an accurate Wallace tree multiplier (WTM) structure.

In [12], approximate unsigned multiplication and division based on an approximate logarithm of the operands have been proposed. In the proposed multiplication, the summation of the approximate logarithms determines the result of the operation. Hence, the multiplication is simplified to some shift and add operations. In [13], a method for increasing the accuracy of the multiplication approach of [12] was proposed. It was based on the decomposition of the input operands. This method considerably improved the average error at the price of increasing the hardware of the approximate multiplier by about two times.

In [16], a dynamic segment method (DSM) is presented, which performs the multiplication operation on an m-bit segment starting from the leading one bit of the input operands. A dynamic range unbiased multiplier (DRUM) multiplier, which selects an m-bit segment starting from the leading one bit of the input operands and sets the least significant bit of the truncated values to one, has been proposed in [17]. In this structure, the truncated values are multiplied and shifted to left to generate the final output. In [18], an approximate 4×4 WTM has been proposed that uses an inaccurate 4:2 counter. In addition, an error correction unit for correcting the outputs has been suggested. To construct larger multipliers, this 4×4 inaccurate Wallace multiplier can be used in an array structure.

Most of the previously proposed approximate multipliers are based on either modifying the structure or complexity reduction of a specific accurate multiplier. In this paper, similar to [12], we propose performing the approximate multiplication through simplifying the operation. The difference between our work and [12] is that, although the principles in both works are almost similar for unsigned numbers, the mean error of our proposed approach is smaller. In addition, we suggest some approximation techniques when the multiplication is performed for signed numbers.

III. Proposed Approximate Multiplier

A. Multiplication Algorithm of RoBA Multiplier

The main idea behind the proposed approximate multiplier is to make use of the ease of operation when the numbers are two to the power n (2^n). To elaborate on the operation of the approximate multiplier, first, let us denote the rounded numbers of the input of A and B by A_r and B_r, respectively. The multiplication of A by B may be rewritten as

$$A \times B = (A_r - A) \times (B_r - B) + A_r \times B + B_r \times A - A_r \times B_r. \quad (1)$$

The key observation is that the multiplications of $A_r \times B_r$, $A_r \times B$, and $B_r \times A$ may be implemented just by the shift operation. The hardware implementation of $(A_r - A) \times (B_r - B)$, however, is rather complex. The weight of this term in the final result, which depends on differences of the exact numbers from their rounded ones, is typically small. Hence, we propose to omit this part from (1), helping simplify the multiplication operation. Hence, to perform the multiplication process, the following expression is used:

$$A \times B \cong A_r \times B + B_r \times A - A_r \times B_r. \quad (2)$$

Thus, one can perform the multiplication operation using three shift and two addition/subtraction operations. In this approach, the nearest values for A and B in the form of 2^n should be determined. When the value of A (or B) is equal to the $3 \times 2^{p-2}$ (where p is an arbitrary positive integer larger than one), it has two nearest values in the form of 2^n with equal absolute differences that are 2^p and 2^{p-1}. While both values lead to the same effect on the accuracy of the proposed multiplier, selecting the larger one (except for the case of $p = 2$) leads to a smaller hardware implementation for determining the nearest rounded value, and hence, it is

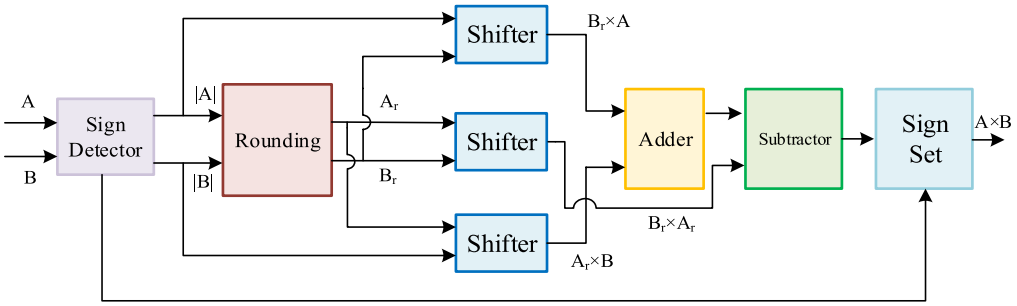

Fig. 1. Block diagram for the hardware implementation of the proposed multiplier.

considered in this paper. It originates from the fact that the numbers in the form of $3 \times 2^{p-2}$ are considered as do not care in both rounding up and down simplifying the process, and smaller logic expressions may be achieved if they are used in the rounding up.

The only exception is for three, which in this case, two is considered as its nearest value in the proposed approximate multiplier.

It should be noted that contrary to the previous work where the approximate result is smaller than the exact result, the final result calculated by the RoBA multiplier may be either larger or smaller than the exact result depending on the magnitudes of A_r and B_r compared with those of A and B, respectively. Note that if one of the operands (say A) is smaller than its corresponding rounded value while the other operand (say B) is larger than its corresponding rounded value, then the approximate result will be larger than the exact result. This is due to the fact that, in this case, the multiplication result of $(A_r - A) \times (B_r - B)$ will be negative. Since the difference between (1) and (2) is precisely this product, the approximate result becomes larger than the exact one. Similarly, if both A and B are larger or both are smaller than A_r and B_r, then the approximate result will be smaller than the exact result.

Finally, it should be noted the advantage of the proposed RoBA multiplier exists only for positive inputs because in the two's complement representation, the rounded values of negative inputs are not in the form of 2^n. Hence, we suggest that, before the multiplication operation starts, the absolute values of both inputs and the output sign of the multiplication result based on the inputs signs be determined and then the operation be performed for unsigned numbers and, at the last stage, the proper sign be applied to the unsigned result. The hardware implementation of the proposed approximate multiplier is explained next.

B. Hardware Implementation of RoBA Multiplier

Based on (2), we provide the block diagram for the hardware implementation of the proposed multiplier in Fig. 1 where the inputs are represented in two's complement format. First, the signs of the inputs are determined, and for each negative value, the absolute value is generated. Next, the rounding block extracts the nearest value for each absolute value in the form of 2^n. It should be noted that the bit width of the output of this block is n (the most significant bit of the absolute value of an n-bit number in the two's complement format is zero). To find the nearest value of input A, we use the following equation to determine each output bit of the rounding block:

$$
\begin{aligned}
A_r[n-1] &= \overline{A[n-1]} \cdot A[n-2] \cdot A[n-3] \\
&\quad + A[n-1] \cdot \overline{A[n-2]} \\
A_r[n-2] &= (\overline{A[n-2]} \cdot A[n-3] \cdot A[n-4] \\
&\quad + A[n-2] \cdot \overline{A[n-3]}) \cdot \overline{A[n-1]} \\
&\vdots \\
A_r[i] &= (\overline{A[i]} \cdot A[i-1] \cdot A[i-2] + A[i] \cdot \overline{A[i-1]}) \cdot \prod_{i=i+1}^{n-1} \overline{A[i]} \\
&\vdots \\
A_r[3] &= (\overline{A[3]} \cdot A[2] \cdot A[1] + A[3] \cdot \overline{A[2]}) \cdot \prod_{i=4}^{n-1} \overline{A[i]} \\
A_r[2] &= A[2] \cdot \overline{A[1]} \cdot \prod_{i=3}^{n-1} \overline{A[i]} \\
A_r[1] &= A[1] \cdot \prod_{i=2}^{n-1} \overline{A[i]} \\
A_r[0] &= A[0] \cdot \prod_{i=1}^{n-1} \overline{A[i]}.
\end{aligned} \tag{3}
$$

In the proposed equation, $A_r[i]$ is one in two cases. In the first case, $A[i]$ is one and all the bits on its left side are zero while $A[i-1]$ is zero. In the second case, when $A[i]$ and all its left-side bits are zero, $A[i-1]$ and $A[i-2]$ are both one. Having determined the rounding values, using three barrel shifter blocks, the products $A_r \times B_r$, $A_r \times B$, and $B_r \times A$ are calculated. Hence, the amount of shifting is determined based on $\log_2^{A_r} - 1$ (or $\log_2^{B_r} - 1$) in the case of A (or B) operand. Here, the input bit width of the shifter blocks is n, while their outputs are $2n$.

A single $2n$-bit Kogge-Stone adder is used to calculate the summation of $A_r \times B$ and $B_r \times A$. The output of this adder and the result of $A_r \times B_r$ are the inputs of the *subtractor* block whose output is the absolute value of the output of the proposed multiplier. Because A_r and B_r are in the form of 2^n, the inputs of the *subtractor* may take one of the three input patterns shown in Table I. The corresponding output patterns are also shown in Table I.

The forms of the inputs and output inspired us to conceive a simple circuit based on the following expression:

$$
\begin{aligned}
\text{out} = (P \text{ XOR } Z) \text{ AND } (\{(P \ll 1) \text{ XOR } (P \text{ XOR } Z)\} \text{ or} \\
\{(P \text{ AND } Z) \ll 1\})
\end{aligned} \tag{4}
$$

TABLE I
ALL POSSIBLE CASES FOR $A_r \times B_r$ AND $A_r \times B + B_r \times A$ VALUES

Input 1 (A_r×B+B_r×A)	Input 2 (A_r×B_r)	Output
000…11…xxx	000…10…000	000…01…xxx
000…11…xxx	000…01…000	000…10…xxx
000…10…xxx	000…01…000	000…01…xxx

where P is $A_r \times B + B_r \times A$ and Z is $A_r \times B_r$. The corresponding circuit for implementing this expression is smaller and faster than the conventional subtraction circuit.

Finally, if the sign of the final multiplication result should be negative, the output of the subtractor will be negated in the *sign set* block. To negate values, which have the two's complement representation, the corresponding circuit based on $\bar{X}+1$ should be used. To increase the speed of negation operation, one may skip the incrementation process in the negating phase by accepting its associated error. As will be seen later, the significance of the error decreases as the input widths increases. In this paper, if the negation is performed exactly (approximately), the implementation is called signed RoBA (S-RoBA) multiplier [approximate S-RoBA (AS-RoBA) multiplier].

In the case where the inputs are always positive, to increase the speed and reduce the power consumption, the *sign detector* and *sign set* blocks are omitted from the architecture, providing us with the architecture called unsigned RoBA (U-RoBA) multiplier. In this case, the output width of the *rounding* block is $n+1$ where this bit is determined based on $A_r[n] = A[n-1] \cdot A[n-2]$. This is because in the case of unsigned $11x\ldots x$ (where x denotes do not care) with the bit width of n, its rounding value is $10\ldots0$ with the bit width of $n+1$. Therefore, the input bit width of the shifters is $n+1$. However, because the maximum amount of shifting is $n-1$, $2n$ is considered for the output bit width of the shifters.

C. Accuracy of RoBA Multiplier

In this section, inaccuracies of the three architectures discussed above are considered. The inaccuracies of the U-RoBA multiplier and S-RoBA multiplier, which originate from omitting the term $(A_r - A) \times (B_r - B)$ from the accurate multiplication of $A \times B$, are the same. Hence, the error is

$$\text{error}(A, B) = \frac{(A_r - A)(B_r - B)}{AB}. \tag{5}$$

Assuming A_r and B_r are equal to 2^n and 2^m, respectively, the maximum error occurs when A and B are equal to 3×2^n and 3×2^m, respectively. In this case, both A_r and B_r have the maximum arithmetic difference from their corresponding inputs. Thus

$$\max\{\text{error}(A,B)\} = \frac{(2^n - 3 \times 2^{n-2})(2^m - 3 \times 2^{m-2})}{(3 \times 2^{n-2}) \times (3 \times 2^{m-2})} = \frac{1}{9}. \tag{6}$$

Therefore, the maximum error for these two architectures is $\%11.\bar{1}$, which is the same as that of [12].

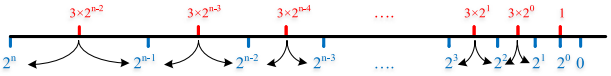

Fig. 2. Numbers (top numbers) and their corresponding possible round values.

In the case of the AS-RoBA multiplier, the error includes an additional term due to the approximate negation (approximate negation). Therefore, in the worst case (where both inputs are negative), one may obtain the maximum error from

$$\text{error}(A,B) = \frac{(\bar{A}_r - \bar{A})(\bar{B}_r - \bar{B})}{AB} + \frac{\bar{A} + \bar{B} + 1}{AB}. \tag{7}$$

Compared with (5), the second term comes from the negation approximation obtained from the following relation:

$$A \times B = (\bar{A}+1)(\bar{B}+1) = \bar{A} + \bar{B} + 1 + \bar{A} \times \bar{B} \approx \bar{A} \times \bar{B} \tag{8}$$

which shows the error as $\bar{A} + \bar{B} + 1$. Hence, in the case where at least one of the inputs is negative, the AS-RoBA multiplier error is larger than that of the two other RoBA multiplier types. Also, when both of the inputs are negative, although the final result will be positive, one still needs to negate the negative inputs. Based on this formulation, when one of the inputs is -1, the maximum error, which is 100%, occurs. To reduce the maximum error of this case, one may use a detector to identify the case when one of the inputs is -1, and bypass the multiplication process and generate the output by negating the other input. It is clear that this solution has some delay and power consumption overhead.

In addition to the maximum error, the occurrence rate of the maximum error condition (which we shall simply call the maximum error rate) is obtained as the ratio of the number of maximum error occurrences to the total number of outputs. This error rate is another accuracy measurement parameter. Here, all the input combinations are assumed to occur. In the case of n-bit U-RoBA multiplier, there are $n-1$ cases for each input where the rounded value has the maximum difference to the actual number (see Fig. 2). The maximum error occurs when these numbers are the input operands. This corresponds to $(n-1)^2$ cases. In the case of S-RoBA multiplier, for each operand, there are $2(n-2)$ cases where the rounded operand has the maximum error. Hence, similar to the U-RoBA multiplier, the maximum error occurs when both of the rounded operands have the maximum error that makes the number of maximum error occurrence equal to $(2(n-2))^2$. Finally, in the case of the AS-RoBA multiplier, as mentioned before, the maximum error happens when one of the inputs is -1. Hence, the number of maximum error occurrences is equal to $2 \times 2^{n-1} - 1$ $(2^n - 1)$.

Table II shows the maximum error rates for the three RoBA multiplier architectures for the input bit width of 8-, 16-, 24-, and 32-bit multipliers. As the results show, the rate of the maximum error decreases as the bit length increases. Also, among the architectures, the AS-RoBA multiplier has the maximum error rate.

On the other hand, in the cases of the U-RoBA and S-RoBA multipliers when the absolute value of the input operand of

TABLE II
MAXIMUM ERROR RATES (%) FOR THE RoBA MULTIPLIER ARCHITECTURES

Input bit width →	8	16	24	32
U-RoBA multiplier	7.40e-04	5.2e-06	1.8e-12	5.2e-15
S-RoBA multiplier	0.21	1.8e-05	6.8e-12	1.9e-14
AS-RoBA multiplier	0.39	1.5e-03	5.9e-06	2.3e-08

TABLE III
PASS RATES (%) FOR THE RoBA MULTIPLIER ARCHITECTURES

Input bit width →	8	16	24	32
U-RoBA multiplier	6.9	0.050	2.98e-6	1.5e-6
S-RoBA multiplier	12.1	0.097	5.72e-4	2.98e-6
AS-RoBA multiplier	3.02	> 0.024	> 1.4e-4	> 7.45e-7

TABLE IV
MRE, MED, NMED, MSE, ACC_{inf}, VARIANCE, AND ERROR RATE OF DIFFERENT 32-bit APPROXIMATE MULTIPLIER DESIGNS

	MRE	MED	NMED	MSE	ACC_{inf}	Variance	Error rate
U-RoBA	2.92%	1.3E+17	0.0069	4.8E+34	60.0%	6.0E-04	~1
S-RoBA	2.91%	3.2E+16	0.0017	3.0E+33	58.0%	6.0E-04	~1
AS-RoBA	2.93%	3.2E+16	0.0017	3.0E+33	58.0%	6.0E-04	~1
Mitchell [12]	3.86%	1.7E+17	0.0093	7.7E+34	58.8%	1.2E-06	~1
DSM8 [16]	0.53%	2.2E+16	0.0012	1.0E+33	60.8%	5.6E-06	~1
DRUM6 [17]	1.47%	6.3E+16	0.0034	1.0E+34	60.0%	1.1E-04	~1
HAAM [18]	2.04%	3.8E+17	0.0188	4.5E+35	85.8%	1.0E-03	0.29

the multiplier is in the form of the 2^m, the output result of the RoBA multiplier is exact [see (5)]. Hence, the numbers of correct outputs in the cases of the U-RoBA multiplier and S-RoBA multipliers are $2(n+1)2^n-(n+1)^2$ and $n2^{n+2}-4n^2$, respectively. In the case of the AS-RoBA multiplier, when both inputs are positive, the multiplier behaves similar to the other two RoBA multiplier architectures, and hence, when one of the inputs is in the form of 2^m, the output is exact. In addition, there are some other combinations that lead to the correct output. One example of such cases is $(A-A_R)(\bar{B}-\bar{B}_R)+A=1$. Analytically finding all the combinations with correct (exact) output is extremely difficult, and hence, for the AS-RoBA multiplier, we use the lower bound of the correct output number that is equal to $n2^n-n^2$.

Next, the *passing rates*, defined as the ratio of the number of correct output occurrences to the total number of distinct outputs [19], for the proposed multiplier architectures are given in Table III. As the results show, by increasing the bit width, the rate of correct results is reduced. Compared with the maximum error, however, the rate at which the correct results are produced (i.e., the passing rate) is higher. As could be expected, the AS-RoBA multiplier has the lowest pass rate, while the pass rate of the S-RoBA multiplier is larger than the others. It should be noted that the pass rate of the method proposed in [12] is the same as that of the U-RoBA multiplier.

Table IV shows mean relative error (MRE), mean error distance (MED), normalized MED (NMED) [21], mean square error (MSE), ACC_{inf} (which measures the error significance as the Hamming distance) [19], variance, and error rate of different approximate multiplier designs. For extracting these metrics, 100K input combinations of inputs were selected from a uniform distribution. Here, we compare the accuracy of the proposed multipliers with DSM8 (DSM with a segment size of 8) [16], DRUM6 (DRUM with a segment size of 6) [17], the method proposed in [12] (denoted by Mitchell), and the approximate multiplier proposed in [18] (denoted by HAAM). Note that, DSM8, DRUM6, Mitchell, and HAAM all are unsigned multipliers.

As Table IV shows, except for the error rate and ACC_{inf}, the DSM8 provides the highest accuracy in terms of all the error metrics. The minimum error rate belongs to the HAAM architecture, while the minimum value for ACC_{inf} is for (A)S-RoBA. Also, the values for U-RoBA, DSM8, and DRUM6 are almost equal. It should be noted that the accuracy of the U-RoBA multiplier is slightly smaller than that of the (A)S-RoBA multiplier. This is due to the smaller range of the signed numbers compared with that of the unsigned numbers for the same bit width. In addition, although the accuracy of the U-RoBA is smaller than those of the DSM8 and DRUM6, its delay and energy values are lower.

Finally, the percentages of the outputs with the relative error (RE) smaller than a specific value for the 32-bit approximate multiplier designs are shown in Table V. They indicate that the best (the next best) accuracy belongs to DSM8 (DTUM6) whose all of its outputs have REs smaller than 2% (6%). In the cases of the proposed multipliers in this paper, almost all of the approximate outputs have RE values smaller than 10%.

TABLE V
PERCENTAGES OF THE OUTPUTS WITH RE SMALLER THAN A SPECIFIC VALUE FOR DIFFERENT 32-bit APPROXIMATE MULTIPLIER DESIGNS

	RE<0.5%	RE<1%	RE<2%	RE<4%	RE<6%	RE<8%	RE<10%	RE<12%
U-RoBA	18.1%	29.9%	47.5%	71.0%	85.8%	94.8%	99.3%	100.0%
S-RoBA	18.1%	30.0%	47.7%	71.0%	86.0%	94.9%	99.4%	100.0%
AS-RoBA	17.9%	30.0%	47.7%	71.3%	86.1%	95.0%	99.4%	100.0%
DSM8 [16]	47.2%	97.6%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
DRUM6 [17]	21.1%	40.3%	70.8%	98.3%	100.0%	100.0%	100.0%	100.0%
HAAM [18]	70.4%	70.4%	70.5%	72.0%	72.0%	100.0%	100.0%	100.0%
Mitchell [12]	12.1%	20.9%	35.3%	57.4%	74.6%	88.1%	97.5%	100.0%

IV. RESULTS AND DISCUSSION

A. Hardware Implementation

To evaluate the efficacy of the proposed multiplier, the three RoBA multiplier implementations were compared with some approximate and exact multipliers. Baugh Wooley based on Wallace tree architecture (as an exact signed) and Wallace (as an exact unsigned) multipliers were selected as the exact multipliers. Also, in the case of approximate multipliers, DSM8 [16], DRUM6 [17], and HAAM [18] were chosen. Since [12] has not provided any hardware implementation,

TABLE VI
POSTLAYOUT DESIGN PARAMETERS OF DIFFERENT 32-bit MULTIPLIER DESIGNS

	Delay		Power		Area		Energy		EDP		PDA	
	(ns)	Ratio	(mW)	Ratio	(μm2)	Ratio	(pJ)	Ratio	(ns×pJ)	Ratio	(pJ×μm2)	Ratio
U-RoBA	1.12	**1.00**	5.37	1.23	13224	2.32	6.0	**1.00**	6.7	**1.00**	79534	2.03
S-RoBA	1.78	1.59	9.21	2.10	22800	4.00	16.4	2.73	29.2	4.33	373779	9.53
AS-RoBA	1.37	1.22	7.6	1.74	14161	2.48	10.4	1.73	14.3	2.12	147444	3.76
DSM8	1.44	1.29	4.38	**1.00**	8742	1.53	6.3	1.05	9.1	1.35	55138	1.41
DRUM6	1.31	1.17	5.25	1.20	5700	**1.00**	6.9	1.14	9.0	1.34	39202	**1.00**
HAAM	2.95	2.63	17.58	4.01	23194	4.07	51.9	8.62	153.0	22.71	1202864	30.68
Wallace	1.69	1.51	17.3	3.95	33124	5.81	29.2	4.86	49.4	7.34	968446	24.70
Baugh Wooly	1.72	1.54	17.42	3.98	33306	5.84	30.0	4.98	51.5	7.65	997928	25.46

we excluded it from this part of the study. The multipliers were implemented using Verilog hardware description language and then synthesized using Synopsys design compiler with the option of synthesizing with the minimum delay objective under a 45-nm technology [14]. Next, the postlayout design parameters of the considered multipliers were extracted by exploiting Cadence system-on-chip encounter. The extracted design parameters of these multipliers are reported in Table VI. It should be mentioned that in this paper, the supply voltage was 1.1 V (based on the NanGate 45-nm technology [14]), while the frequency was selected using the reported delay for each multiplier (see Table VI).

The results reveal that the minimum delay, energy, and EDP belong to the U-RoBA while DSM8 has the best power consumption and DRUM8 has the minimum area and PDA. The delay, energy, and EDP of the U-RoBA are about 22% (15%), 5% (13%), and 26% (25%) lower than those of DSM8 (DRUM6). In contrast, the power (area and PDA) of DSM8 (DRUM6) is (are) about 18% (57% and 51%) lower. Also, the negation operation leads to larger design parameters for S-RoBA and AS-RoBA compared with those of U-RoBA, DSM8, and DRUM6. Also, HAAM has the worst design parameters due to its array structure.

The results also indicate that the exact multipliers have considerably larger design parameters compared with those of the proposed U-RoBA and AS-RoBA. In the case of the S-RoBA multiplier, the delay is, on average, 3.4% larger than that of the Baugh Wooley multiplier due to the use of the exact negation operation. Except for the delay parameter, other design parameters of the S-RoBA multiplier are better than those of the Bough Wooley multiplier. On the other hand, the power, area, energy, EDP, and PDA of the S-RoBA multiplier, are about 47%, 32%, 45%, 43%, and 63%, respectively, lower than those of the Bough Wooley multiplier.

Finally, Table VII shows the breakdown of the power, delay, and area of different units of both the AS-RoBA and S-RoBA multipliers. As the results reveal, the shifter unit has the highest delay, power, and area among the units of the multipliers.

TABLE VII
BREAKDOWN OF THE POWER, DELAY, AND AREA OF AS-RoBA AND S-RoBA

		Sign Detector	Rounding	Shifter	Adder	Subtractor	Sign Set
AS-RoBA	Power	7.3%	11.4%	40.1%	29.9%	7.1%	4.1%
	Delay	3.8%	16.7%	34.8%	34.1%	6.8%	3.8%
	Area	5.2%	9.0%	66.6%	13.7%	3.9%	1.6%
S-RoBA	Power	9.5%	5.5%	32.1%	16.6%	8.0%	28.3%
	Delay	15.5%	16.1%	23.8%	23.3%	5.7%	15.5%
	Area	8.3%	6.6%	61.4%	11.7%	3.9%	8.0%

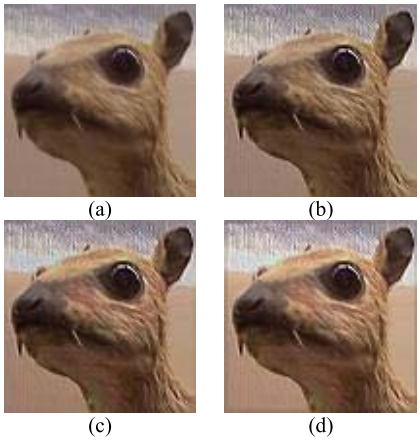

Fig. 3. Image sharpening using the proposed approximate architecture. (a) Original image. Image sharpening utilizing (b) exact multiplier, (c) S-RoBA multiplier, and (d) AS-RoBA multiplier.

B. Image Processing Applications

To evaluate the feasibility of the proposed multiplier in real applications, we compared the performances of the RoBA multiplier architectures in two image processing applications of smoothing and sharpening with those of the corresponding exact ones. For sharpening, two different methods were invoked. In the first one, each pixel of the sharp image was extracted from [15]

$$Y(i,j) = 2\cdot X(i+m, j+n) - \frac{1}{273}\sum_{m=-2}^{2}\sum_{n=-2}^{2} X(i+m, j+n) \cdot \text{Mask}_{\text{Sharpening},1}(m+3, n+3) \quad (9)$$

where the $X(i,j)$ $[Y(i,j)]$ indicates the pixel of the ith row and jth column of input (output) image and $\text{Mask}_{\text{sharpening},1}$

TABLE VIII

PSNR AND MSSIM VALUES FOR THE SHARPENING ALGORITHMS WHEN DIFFERENT APPROXIMATE MULTIPLIER STRUCTURES ARE USED

	$\text{Mask}_{\text{Sharpening,1}}$								$\text{Mask}_{\text{Sharpening,2}}$							
	RoBA multiplier		DRUM6 multiplier		DSM8 multiplier		Mitchell multiplier		S-RoBA multiplier		AS-RoBA multiplier		DRUM6 multiplier		DSM8 multiplier	
Image	PSNR(dB)	MSSIM	PSNR(dB)	MSSIM	PSNR(dB)	MSSIM	PSNR(dB)	MSSIM	PSNR(dB)	MSSIM	PSNR(dB)	MSSIM	PSNR(dB)	MSSIM	PSNR(dB)	MSSIM
Airplane (F-16)	39.3	1.0	35.2	1.0	∞	1.0	30.1	0.9	26.7	0.8	25.4	0.8	35.9	0.8	∞	1.0
Girl	49.5	1.0	42.3	1.0	∞	1.0	36.7	0.9	37.6	0.9	30.1	0.9	43.0	1.0	∞	1.0
House	40.6	1.0	36.3	1.0	∞	1.0	29.3	0.9	27.9	0.8	23.4	0.8	36.5	0.9	∞	1.0
Mandrill	44.3	1.0	37.8	1.0	∞	1.0	26.3	0.9	32.2	1.0	24.9	1.0	37.5	1.0	∞	1.0
Peppers	42.8	1.0	38.2	1.0	∞	1.0	32.6	0.9	30.2	1.0	25.0	1.0	38.2	0.9	∞	1.0
Vd-Orig	45.1	1.0	38.7	1.0	∞	1.0	31.5	0.9	32.9	1.0	25.1	1.0	38.6	0.9	∞	1.0
Lena	43.4	1.0	37.9	1.0	∞	1.0	31.8	0.9	31.1	0.9	24.7	0.9	37.8	0.9	∞	1.0
Average	43.6	1.0	38.1	1.0	∞	1.0	31.2	0.9	31.2	0.9	25.5	0.9	38.2	0.9	∞	1.0

TABLE IX

PSNR AND MSSIM VALUES FOR THE SMOOTHING ALGORITHM WHEN DIFFERENT APPROXIMATE MULTIPLIER STRUCTURES ARE USED

	RoBA multiplier		DRUM6 multiplier		DSM8 multiplier		Mitchell multiplier	
Image	PSNR(dB)	MSSIM	PSNR(dB)	MSSIM	PSNR(dB)	MSSIM	PSNR(dB)	MSSIM
Airplane (F-16)	40.60	0.95	33.25	0.85	∞	1.00	16.82	0.72
Girl	48.96	1.00	38.78	0.98	∞	1.00	24.86	0.84
House	41.45	0.99	33.54	0.89	∞	1.00	17.84	0.72
Mandrill	43.72	1.00	34.82	1.00	∞	1.00	19.21	0.65
Peppers	43.27	1.00	34.82	0.97	∞	1.00	20.00	0.83
Vd-Orig	45.53	1.00	35.05	0.98	∞	1.00	20.89	0.68
Lena	44.47	1.00	34.63	0.96	∞	1.00	19.73	0.74
Average	44.00	0.99	34.98	0.95	∞	1.00	19.91	0.74

is an $n \times n$ coefficient sharpening matrix given by

$$\text{Mask}_{\text{sharpening,1}} = \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}. \quad (10)$$

In the second method, each output pixel is determined from

$$Y(i,j) = \frac{1}{256} \sum_{m=-2}^{2} \sum_{n=-2}^{2} X(i+m, j+n) \cdot \text{Mask}_{\text{sharpening,2}}(m+3, n+3) \quad (11)$$

where the sharpening matrix is [15]

$$\text{Mask}_{\text{sharpening,2}} = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -4 & -16 & -24 & -16 & -4 \\ -6 & -24 & 476 & -24 & -6 \\ -4 & -16 & -24 & -16 & -4 \\ -1 & -4 & -6 & -4 & -1 \end{bmatrix}. \quad (12)$$

In the case of $\text{Mask}_{\text{Sharpening,1}}$, all the values of the matrix are positive, and hence, all the three RoBA multiplier architectures lead to the same results, while in the case of the $\text{Mask}_{\text{sharpening,2}}$, both S-RoBA and AS-RoBA multipliers may be utilized leading to different image qualities.

First, as an example, consider the sharpening described above for the original image of Vd-Orig shown in Fig. 3. The sharpened images for the second approach when the exact multiplier, S-RoBA multiplier, and AS-RoBA multiplier were used are also given in Fig. 3(b)–(d). As Fig. 3(b)–(d) reveals, the betterness of the sharpening process may not be easily recognized by human eyes. Next, we report the peak signal-to-noise ratio (PSNR) and mean structural similarity index metric (MSSIM [20]) of the sharpened pictures for the two sharpening matrices for seven images in Table VIII. It should be noted that the reported PSNRs are determined based on comparing the sharpened image obtained using the exact multipliers to the sharpened image obtained using the considered approximate multiplier structures. Also, the MSSIM values closer to one indicate higher qualities for the approximate output image.

As the results show, in the case of the positive numbers, the average of PSNR (MSSIM) of the proposed multiplier is more than 43 dB (0.99). Although, in the case of negative numbers, the quality of the images is lower, the PSNRs (MSSIM) in all the cases are more than 20 dB (0.91), which is acceptable in many applications [15]. In all the benchmarks, the DSM8 provides the highest output quality providing the same performance as that of the exact multiplication that has the PSNR values of ∞. The Mitchell multiplier supports only the unsigned operation, and hence, its results have been reported only for the first sharpening algorithm. The results reveal the lowest quality for this multiplier. Also, our proposed approximate multiplier yields higher (lower) output PSNR values compared with those of the DRUM6 in the case of the first (second) sharpening algorithm.

For the second application of the smoothing, we have utilized the following equation to determine the smoothed output image [15]:

$$Y(i,j) = \frac{1}{60} \sum\nolimits_{m=-2}^{2} \sum_{n=-2}^{2} X(i+m, j+n) \cdot \text{Mask}(m+3, n+3). \quad (13)$$

Here, again $X(i, j)$ $[Y(i, j)]$ is the pixel of the ith row and jth column of input (output) image and $Mask_{\text{Smoothing}}$ is an $n \times n$ coefficient smoothing matrix given by

$$\text{Mask}_{\text{smoothing}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 4 & 4 & 4 & 4 \\ 1 & 4 & 12 & 4 & 7 \\ 1 & 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \quad (14)$$

Because every coefficient is positive, all the three RoBA multiplier architectures lead to the same output image quality. Table IX shows the PSNR and MSSIM of the smoothing process using the considered approximate multiplier structures for the seven images compared with the case of using the exact multiplier. As the results reveal, all the PSNRs (MSSIMs) are higher than 40 (0.99) demonstrating small errors for the proposed multiplier. The output quality of the RoBA in all the benchmark images is better than those of the DRUM6 and Mitchell multipliers. However, similar to the sharpening application, the DSM8 multiplier provides the highest output quality.

V. Conclusion

In this paper, we proposed a high-speed yet energy efficient approximate multiplier called RoBA multiplier. The proposed multiplier, which had high accuracy, was based on rounding of the inputs in the form of 2^n. In this way, the computational intensive part of the multiplication was omitted improving speed and energy consumption at the price of a small error. The proposed approach was applicable to both signed and unsigned multiplications. Three hardware implementations of the approximate multiplier including one for the unsigned and two for the signed operations were discussed. The efficiencies of the proposed multipliers were evaluated by comparing them with those of some accurate and approximate multipliers using different design parameters. The results revealed that, in most (all) cases, the RoBA multiplier architectures outperformed the corresponding approximate (exact) multipliers. Also, the efficacy of the proposed approximate multiplication approach was studied in two image processing applications of sharpening and smoothing. The comparison revealed the same image qualities as those of exact multiplication algorithms.

References

[1] M. Alioto, "Ultra-low power VLSI circuit design demystified and explained: A tutorial," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 1, pp. 3–29, Jan. 2012.

[2] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 124–137, Jan. 2013.

[3] 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. I, Reg. Papers*, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[4] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and analysis of circuits for approximate computing," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2011, pp. 667–673.

[5] F. Farshchi, M. S. Abrishami, and S. M. Fakhraie, "New approximate multiplier for low power digital signal processing," in *Proc. 17th Int. Symp. Comput. Archit. Digit. Syst. (CADS)*, Oct. 2013, pp. 25–30.

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

[7] D. R. Kelly, B. J. Phillips, and S. Al-Sarawi, "Approximate signed binary integer multipliers for arithmetic data value speculation," in *Proc. Conf. Design Archit. Signal Image Process.*, 2009, pp. 97–104.

[8] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in *Proc. IEEE Int. Conf. Electron Devices Solid-State Circuits (EDSSC)*, Dec. 2010, pp. 1–4.

[9] 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.

[10] K. Bhardwaj and P. S. Mane, "ACMA: Accuracy-configurable multiplier architecture for error-resilient system-on-chip," in *Proc. 8th Int. Workshop Reconfigurable Commun.-Centric Syst.-Chip*, 2013, pp. 1–6.

[11] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in *Proc. 15th Int. Symp. Quality Electron. Design (ISQED)*, 2014, pp. 263–269.

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

[13] V. Mahalingam and N. Ranganathan, "Improving accuracy in Mitchell's logarithmic multiplication using operand decomposition," *IEEE Trans. Comput.*, vol. 55, no. 12, pp. 1523–1535, Dec. 2006.

[14] *Nangate 45nm Open Cell Library*, accessed on 2010. [Online]. Available: http://www.nangate.com/

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

[16] 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.

[17] 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, 2015, pp. 418–425.

[18] C.-H. Lin and I.-C. Lin, "High accuracy approximate multiplier with error correction," in *Proc. 31st Int. Conf. Comput. Design (ICCD)*, 2013, pp. 33–38.

[19] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. 49th Design Autom. Conf. (DAC)*, Jun. 2012, pp. 820–825.

[20] 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.

[21] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Trans. Comput.*, vol. 62, no. 9, pp. 1760–1771, Sep. 2013.

Reza Zendegani was born in Mashhad, Iran. He received the B.Sc. degree in electrical engineering from the Ferdowsi University of Mashhad, Mashhad, in 2013 and the M.Sc. degree in electrical engineering from the University of Tehran, Tehran, Iran, in 2015, where he is currently pursuing the Ph.D. degree in digital systems.

His current research interests include VLSI implementation of arithmetic circuits, low-power VLSI architectures, variation-tolerant signal processing circuits, and fault-tolerant systems design.

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

He is currently an Assistant Professor with the School of Electrical and Computer Engineering, University of Tehran. His current research interests include reliability in nanoscale design, application specific instruction set processor design, hardware/software co-design, and low-power design.

Milad Bahadori received the M.Sc. degree in electrical and electronics engineering from the Sharif University of Technology, Tehran, Iran, in 2011, and the Ph.D. degree in electrical and electronics engineering from the University of Tehran, Tehran, in 2015.

He was a Research Assistant with the Sharif Integrated Circuits Design Center, Sharif University of Technology, from 2009 to 2012, where he was involved in digital systems design. He joined the Low-Power High-Performance Nanosystems Laboratory, University of Tehran, in 2012, as a Research Assistant. His current research interests include low-power high-performance VLSI design, reliability in nanoscale design, near-threshold computing, and high-performance low power arithmetic circuits design.

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

He was a Post-Doctoral Fellow with the University of Michigan from 1994 to 1995. He has been with the University of Tehran, Tehran, since 1995, where he is currently a Professor with the School of Electrical and Computer Engineering and the Director of the Low-Power High-Performance Nanosystems Laboratory. He was a Research Fellow with the University of Toronto, Toronto, ON, Canada, in 1998, and the University of Waterloo, Waterloo, ON, Canada, in 1999. His current research interests include low-power high-performance design methodologies from the physical design level to the system level for nanoelectronics era.

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 with the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA. He holds ten U.S. patents and has authored four books, 12 book chapters, and over 140 archival and 350 conference papers. 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 the 1996 Presidential Early Career Award for Scientists and Engineers and an ACM Distinguished Scientist. He and his students have received six conference and two IEEE Transactions Best Paper Awards for the research. He has served on the Technical Program Committee of a number of premiere conferences. 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 of the *ACM Transactions on Design Automation of Electronic Systems*.