Microelectronics Journal 88 (2019) 1–8

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Implementation of energy-efficient approximate multiplier with guaranteed worst case relative error

Merin Loukrakpam [a,b,*], Madhuchhanda Choudhury [a]

[a] Department of Electronics and Communication Engineering, NIT Silchar, Assam, India
[b] Department of Electronics and Communication Engineering, Manipur Technical University, Imphal, Manipur, India

ARTICLE INFO

Keywords:
Approximate computing
Energy efficiency
Error resilience
Multiplier
Piecewise linear approximation

ABSTRACT

Existing design methods for approximate multipliers typically rely on exhaustive simulation to determine the approximation error. However, this approach is not tractable for complex designs. In this paper, a two-dimensional piecewise linear approximation method for multiplication that formally guarantees the maximum error is proposed. Based on this method, a design procedure to implement energy-efficient approximate multipliers is also proposed. Three 32-bit unsigned approximate multipliers with guaranteed maximum errors of 12.5%, 3.13% and 0.78% were realized using the design procedure. The efficiencies of the proposed designs were evaluated by comparing their parameters with that of the exact and state-of-the-art approximate multipliers. The proposed designs deliver up to 91.8% (62.1%) energy saving when compared with that of exact (approximate) multipliers. The effectiveness of the approximate multipliers was also assessed in an image smoothening application.

1. Introduction

Computing digital signal processing and machine learning algorithms on a battery operated device have been of great interest for applications like computer vision, speech recognition and medical imaging [1]. Energy minimization is one of the main design requirements for these power constrained devices [2]. Moreover, the energy/power-efficiency improvement is desired without compromising the performance (speed) which is a challenging task. One of the most used operations in these applications is multiplication [3]. Hence, improving the energy efficiency of multipliers is crucial for improving the overall energy efficiency of the system.

Interestingly, many of the signal processing and machine learning applications do not always require an exact computation. Either the exact result is not critical for the proper functioning of the system [4,5] or the inexact results often produce acceptable results for human perception [6]. Hence, such systems are resilient to a certain amount of error. This error-resilient property is leveraged to improve the energy efficiency of the multipliers in many cases by using approximate computing [7]. The use of approximate computing considers accuracy as a design parameter for trading with speed and energy.

Various approximation methods to improve the energy efficiency of multipliers have been reported in the literature. Approximate Array, Wallace-Tree, and Dadda-Tree multipliers have been explored. Some of the approaches are truncated designs of Array multipliers [4,8], a 4×4 approximate Wallace tree multiplier [9], and inexact Dadda multipliers based on approximate 4:2 compressors [10,11]. Although these inexact multipliers provided higher speeds and lower powers, they suffered from high approximation errors. Approximate multipliers based on partial-product accumulation have also been explored [12,13]. Recently, energy efficient signed approximate multipliers based on radix encoding were reported in Refs. [14,15].

Approximations based on the truncation of input operands to reduce the overall multiplication have also been studied. A dynamic segment method (DSM) was proposed in Ref. [16] in which the input numbers were segmented into two or three possible m-bits segments. In Ref. [17], a dynamic range unbiased multiplier (DRUM) was proposed in which m-bits segments were selected based on the leading one position of the numbers, and the least significant bits of the truncated values were set to one. A low energy truncation-based approximate multiplier (LETAM) was reported in Ref. [18] wherein the input operands were truncated to a fixed bit length smaller than that of the operands. The result was generated using smaller bit length addition and multiplication operations.

* Corresponding author. Department of Electronics and Communication Engineering, NIT Silchar, Assam, India.
E-mail address: merin1280@gmail.com (M. Loukrakpam).

https://doi.org/10.1016/j.mejo.2019.04.006
Received 5 September 2018; Received in revised form 7 April 2019; Accepted 9 April 2019
Available online 11 April 2019

These methods typically produced accurate results but still required multipliers of smaller bit length in their structures.

Approximate multipliers purely based on shift and add operations have also been investigated. Computing multiplication using approximate logarithmic and antilogarithmic converters were first proposed in Ref. [19] and later with improved accuracies in Refs. [20–22]. Recently, a rounding based approximate (RoBA) multiplier was proposed in Ref. [23] in which the multiplication was performed with the input operands rounded to the nearest powers of two. The approximation was realized using shift and add operations which resulted in lower delays. Low power error-tolerant approximate logarithmic multipliers were also proposed in Ref. [24].

All the aforementioned approximate multipliers evaluated the errors by means of exhaustive simulations. However, this approach is not tractable for complex designs. A simulation-based error analysis using a subset of input combinations does not guarantee an accurate result [25]. Hence, a formal approach is required to compute the error bound of the approximation method.

Recently, a search-based design to construct approximate multipliers that guarantee the worst case error was proposed in Ref. [25]. However, the method was applicable only to truncated designs of Array multipliers. Moreover, the search-based design relied on a rich library of small approximate multiplier units. Theoretical error analyses were reported in Refs. [17,23] for its respective designs. However, the maximum error in Ref. [17] was a function of the bit length of the operands and computed under the assumption of uniform distribution of the inputs. Although the maximum error reported in Ref. [23] was independent of the bit length of the operands, it was fixed at 11.1% for the approximation method.

The focus of this paper is to design energy-efficient approximate multipliers for error-resilient systems with a maximum error bound. A method to approximate multiplication within a desired maximum error bound using two-dimensional piecewise linear function is proposed. A formal analysis to compute the worst case arithmetic error of the approximation method is discussed. A design procedure is then developed to implement energy-efficient approximate multiplier without violating the error constraint. Three energy-efficient unsigned approximate multipliers were implemented using the design procedure for different error constraints. The efficiencies of the hardware were assessed by comparing the area, delay, power, energy, energy-delay product and area-delay-power product with those of exact and state-of-the-art approximate multipliers. The contributions of this paper can be summarized as follows:

1. A two-dimensional piecewise linear approximation method for multiplication that formally guarantees the maximum error;
2. A design procedure, based on the approximation method, to implement energy-efficient approximate multiplier that meets the error constraint;
3. Three energy-efficient unsigned approximate multipliers implemented using the design procedure with different error constraints.

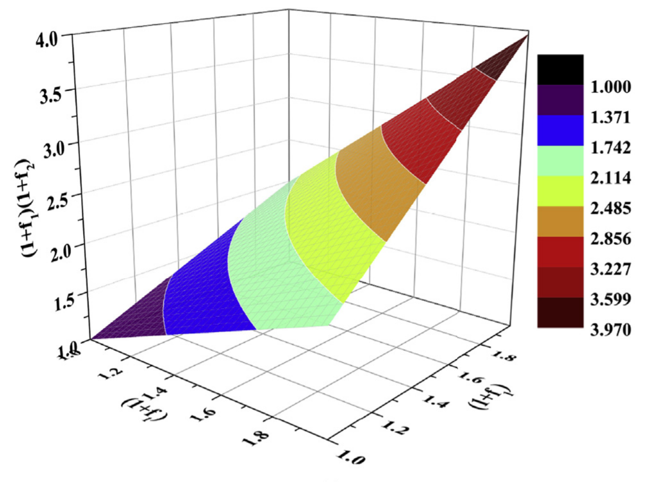
The rest of the paper is organized as follows. Section 2 discusses the proposed approximation method and the design procedure in detail. The proposed hardware implementations are presented in Section 3. Section 4 presents the characteristics of the proposed approximate multipliers. The effectiveness of the approximate multipliers in an image smoothening application is also studied in this Section. Finally, Section 5 concludes the paper.

2. Proposed approximation method and design procedure

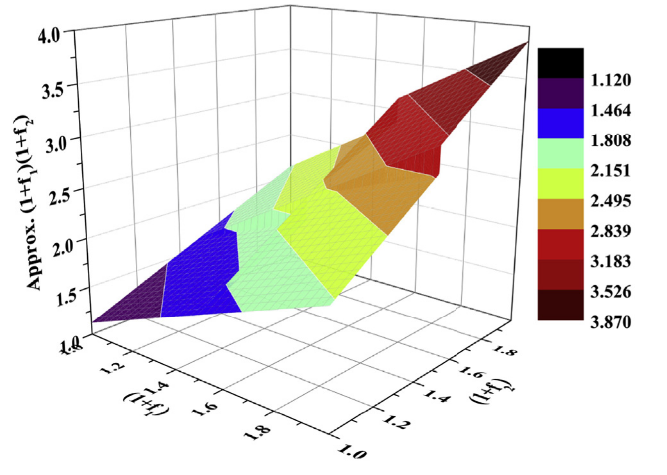
2.1. Proposed approximation method

A positive integer number N can be written as follows [19]:

$$N = 2^k(1+f), \quad (1)$$



(a)



(b)

Fig. 1. (a) Plot of $(1 + f_1)(1 + f_2)$ product. (b) Plot of two-dimensional piecewise linear approximation of $(1 + f_1)(1 + f_2)$ using 2 segments.

where k is the position of its leading one bit and f is a fraction, i.e., $0 \leq f < 1$. The multiplication of two unsigned numbers N_1 and N_2 can be computed as

$$N_1 \times N_2 = 2^{(k_1+k_2)}(1+f_1)(1+f_2). \quad (2)$$

Since k_1 and k_2 are integers, the term $2^{(k_1+k_2)}$ can be implemented using shifters. Fig. 1(a) shows the three dimensional plot of the function $y = (1 + f_1)(1 + f_2)$ formed by all possible values of f_1 and f_2 such that $0 \leq f_1, f_2 < 1$.

In this work, we propose to approximate the function $y = (1 + f_1)(1 + f_2)$ using two-dimensional piecewise linear (PWL) function. If the domain of f_1 and f_2 are divided into n segments, the plane formed by the ith segment of f_1 and the jth segment of f_2 is defined as

$$\hat{y} = a_j(1+f_1) + b_i(1+f_2) + c_{ij}. \quad (3)$$

Fig. 1(b) shows the plot of $\hat{y}$ using $n = 2$ segments of f_1 and f_2.

The corresponding values of f_1 and f_2 at the starting points of the ith and jth segments will be $f_1 = i/n$ and $f_2 = j/n$ respectively. In this work, a in (3) for a particular region is considered as the slope of $y = (1 + f_1)(1 + f_2)$ with respect to f_1 at the point $(f_1 = i/n, f_2 = j/n)$. Similarly, b is considered as the slope of $y = (1 + f_1)(1 + f_2)$ with respect to f_2 at the same point. Hence,

$$a_j = \partial y/\partial f_1 = 1 + f_2 = 1 + j/n; \tag{4}$$

$$b_i = \partial y/\partial f_2 = 1 + f_1 = 1 + i/n. \tag{5}$$

Now, the error of the approximation is computed as

$$\begin{aligned} E &= \hat{y} - y \\ &= a_j(1+f_1) + b_i(1+f_2) + c_{ij} - (1+f_1)(1+f_2). \end{aligned} \tag{6}$$

To find the maximum error, we solve $\partial E/\partial f_1 = 0$ and $\partial E/\partial f_2 = 0$. Hence, we get

$$\partial E/\partial f_1 = 0 \Rightarrow 1 + f_{2,\max} = a_j;$$

$$\partial E/\partial f_2 = 0 \Rightarrow 1 + f_{1,\max} = b_i. \tag{7}$$

Therefore $E_{\max}$ is computed as

$$\begin{aligned} E_{\max} &= a_j(1+f_{1,\max}) + b_i(1+f_{2,\max}) + c_{ij} - (1+f_{1,\max})(1+f_{2,\max}) \\ &= a_jb_i + b_ia_j + c_{ij} - b_ia_j \\ &= a_jb_i + c_{ij}. \end{aligned} \tag{8}$$

To make the error analysis simpler, the absolute error ($|E|$) is made symmetric by equating the error at the points $(i/n, j/n)$ and $((i+1)/n, (j+1)/n)$. Hence,

$$\begin{aligned} & a_j(1+i/n) + b_i(1+j/n) + c_i - (1+\frac{i}{n})(1+\frac{j}{n}) \\ & = (1+\frac{i+1}{n})(1+\frac{j+1}{n}) - a_i(1+\frac{i+1}{n}) - b_i(1+\frac{j+1}{n}) - c_i \\ & \Rightarrow 2c_i = 2a_jb_i + \frac{1}{n}(a_j+b_i) - 2a_j(b_i+\frac{1}{2n}) - 2b_i(a_j+\frac{1}{2n}) + \frac{1}{n^2} \\ & \Rightarrow c_{ij} = \frac{1}{2n^2} - a_jb_i. \end{aligned} \tag{9}$$

Substituting the values of a, b, and c from (4), (5) and (9) in (8), we have

$$E_{\max} = \frac{1}{2n^2}. \tag{10}$$

Eq. (10) reveals that the maximum error of $\hat{y}$ depends only on the number of segments of the two-dimensional PWL approximation.

2.2. Error analysis

Different metrics have been developed to analyze the approximation error [26], viz., the mean error distance (MED), the worst case error (WCE), the mean relative error distance (MRED), the normalized mean error distance (NMED) [27], and the worst case relative error (WCRE) [25]. The WCE and WCRE are crucial for time-critical as well as image and signal processing systems where low average error with high worst case error can lead to unacceptable results [25]. The WCE is defined as

$$\begin{aligned} WCE &= \max_{\forall N_1,N_2} |(N_1 \times N_2) - approx.(N_1 \times N_2)| \\ &= \max_{\forall N_1,N_2} 2^{(k_1+k_2)}|y - \hat{y}| \\ &= E_{\max} \max_{\forall N_1,N_2} 2^{(k_1+k_2)}. \end{aligned} \tag{11}$$

If K is the bit length of the input operands, the maximum value of k_1 and k_2 will be $K - 1$. Hence,

$$WCE = \frac{2^{2(K-1)}}{2n^2}. \tag{12}$$

Another important error metric is WCRE which is defined as

$$\begin{aligned} WCRE &= \max_{\forall N_1,N_2} \frac{|(N_1 \times N_2) - approx.(N_1 \times N_2)|}{(N_1 \times N_2)} \\ &= \max_{\forall N_1,N_2} \frac{|y - \hat{y}|}{y}. \end{aligned} \tag{13}$$

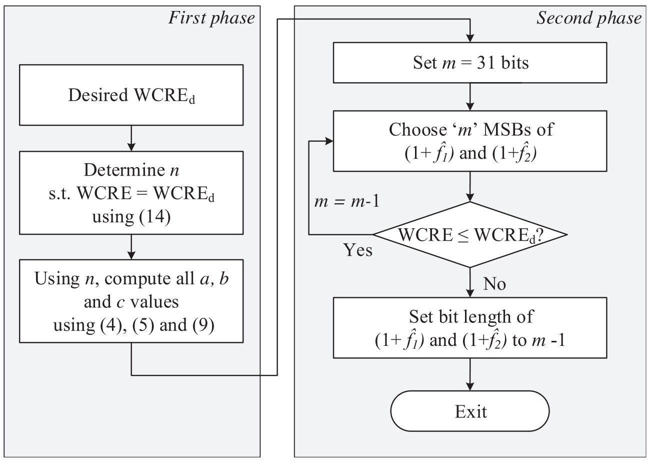

First phase
Second phase
Desired WCRE$_d$
Determine n s.t. WCRE = WCRE$_d$ using (14)
Using n, compute all a, b and c values using (4), (5) and (9)
Set m = 31 bits
Choose 'm' MSBs of $(1+\hat{f}_1)$ and $(1+\hat{f}_2)$
$m = m$-1
WCRE ≤ WCRE$_d$?
Yes
No
Set bit length of $(1+\hat{f}_1)$ and $(1+\hat{f}_2)$ to m -1
Exit

Fig. 2. Flowchart of the proposed design procedure.

The maximum value of (13) is computed by taking the maximum of the numerator, i.e., $E_{\max}$ and the minimum of the denominator, i.e., 1. Hence,

$$WCRE = \frac{1}{2n^2}. \tag{14}$$

It should be noted that the WCRE of the proposed approximate multiplier is independent of the bit length of the operands, and the value depends only on the number of segments of the two-dimensional PWL approximation.

2.3. Proposed design procedure

A design procedure based on (14) to implement energy-efficient approximate multiplier for the desired worst case relative error (WCRE$_d$) is proposed. Fig. 2 shows the flowchart of the design procedure. The procedure involves two phases. The first phase uses (14) to determine the minimum number of segments (n) required for approximating multiplication to meet the WCRE$_d$. Based on the minimum number of segments, the corresponding coefficients a, b, and c values of the two-dimensional PWL function are computed.

The second phase of the design procedure reduces the hardware resources by truncating the bit lengths of f_1 and f_2 without violating the error constraint. For this purpose, Eq. (3) is rewritten as

$$\begin{aligned} & (1+f_1)(1+f_2) \\ & \approx (1+f_1) + (a_j - 1)(1+\hat{f}_1) + (1+f_2) + (b_i - 1)(1+\hat{f}_2) - c_{ij}. \end{aligned} \tag{15}$$

The negative sign in front of c_{ij} is for mathematical convenience which ensures c_{ij} to be positive. The second phase starts with m-MSBs of $(1+\hat{f}_1)$ and $(1+\hat{f}_2)$ with $m = 31$ which means $\hat{f}_1 = f_1$ and $\hat{f}_2 = f_2$ initially. The bit lengths of $(1+\hat{f}_1)$ and $(1+\hat{f}_2)$ are then truncated by 1. A check is performed to verify if the worst case relative error is still within the WCRE$_d$. The process continues until the WCRE exceeds WCRE$_d$. The minimum bit lengths of $(1+\hat{f}_1)$ and $(1+\hat{f}_2)$ are then set to $m-1$.

Hence, the design procedure starts with the WCRE$_d$ and systematically computes the minimum number of segments, the corresponding coefficients a, b, and c, and the minimum bit lengths of $(1+\hat{f}_1)$ and $(1+\hat{f}_2)$ for approximating $(1+f_1)(1+f_2)$ within WCRE$_d$.

3. Proposed approximate multipliers

3.1. Piecewise linear approximate multipliers

Three approximate multiplier designs were implemented using the design procedure with different WCRE constraints. The desired WCRE

Table 1
The values of a, b, and c computed using the proposed design procedure with $n = 2$. The unique values of c are highlighted.

		f_1	
		$[0, 0.5)$ $b = 1.00$	$[0.5, 1)$ $b = 1.50$
f_2	$[0, 0.5)$ $a = 1.00$	$c = \mathbf{0.875}$	$c = \mathbf{1.375}$
	$[0.5, 1)$ $a = 1.50$	$c = 1.375$	$c = \mathbf{2.125}$

of 12.5% was used to implement the first approximate multiplier. The WCRE value chosen is comparable to that reported in Ref. [23]. The first phase of the design procedure revealed that $n = 2$ segments were required to meet the error constraint. The coefficients a, b, and c were then computed using $n = 2$. Table 1 displays their corresponding values. With f_1 and f_2 divided into 2 segments, the two-dimensional PWL approximation of $(1 + f_1)(1 + f_2)$ had four regions as shown in Table 1 with four values of c. However, only three values were unique as highlighted in the table. Since a and b depend on f_2 and f_1 regions respectively, they have 2 possible values. Two bits were used to represent a and b while five bits were required to represent c using the fixed point number representation. As per the second phase of the design procedure, a minimum of 3-bits of $(1+\widehat{f_1})$ and $(1+\widehat{f_2})$ were required to maintain the WCRE within the constraint.

The proposed approximation method was also used to implement an approximate multiplier with the desired WCRE of 5% which is comparable to that of [17] with 6 segments. Based on (14), the WCRE value could be achieved with $n = 3.16$. After rounding, $n = 4$ segments were used that guarantees the WCRE of 3.125% which is well within the error constraint. Table 2 captures the corresponding values of the coefficients. Three bits were needed to represent a and b while 7-bits were required for c. Moreover, only 10 values of c were unique out of the 16 generated values. The second phase of the design procedure revealed that the area cost could be further reduced by truncating the bit lengths of $(1+\widehat{f_1})$ and $(1+\widehat{f_2})$ to 5-bits.

An approximate multiplier with WCRE of 1% was also attempted in this work. Based on (14), the error constraint could be met using $n = 7.07$ segments. Eight segments were chosen to implement the approximate multiplier with the WCRE = 0.78% which meets the error constraint. The bit lengths of the corresponding coefficients were found to be 4-bits for a and b, and 9-bits for c. At least 7-bits of $(1+\widehat{f_1})$ and $(1+\widehat{f_2})$ were required to realize the approximate multiplier without violating the error constraint. The proposed eight segment approximate multiplier have 64 regions. Out of the 64 values of c only 36 uniques values were required to realize the approximation. In general, $n(n + 1)/2$ unique values are required for an n segment approximation.

The proposed approximate multipliers are named piecewise linear approximate multiplier (PLAM) suffixed with the number of segments used for the approximation, e.g., PLAM2, PLAM4, and PLAM8.

3.2. Hardware architecture

The proposed approximate multipliers were realized using shift and add operations. Fig. 3 shows the hardware architecture of the PLAM4 design. Two 32-bit inputs, N_1 and N_2, were considered. The leading one positions (k_1 and k_2) and the 31-bit fractional parts (f_1 and f_2) of the inputs were extracted using ExtractKF modules. Lookup tables were used to realize these modules. The extracted $(1 + f_1)$ and $(1 + f_2)$ values were used to compute the approximate $(1 + f_1)(1 + f_2)$ product. Fixed point Q2.31 number format, with 2-bits integer part and 31-bits fractional part, was used to implement the $(1 + f_1)(1 + f_2)$ approximation since its maximum value tends to 4 as shown in Fig. 1.

It can be noted from Table 2 that all the values of $(a-1)$ and $(b - 1)$ can be represented as sums of powers of 2. Hence, the terms $(a-1)(1+\widehat{f_1})$ and $(b-1)(1+\widehat{f_2})$ in (15) were implemented using shifters and 7-bit adders (A1 and A2 in Fig. 3). Since f_1 and f_2 were divided into 4 segments, two MSBs of the fractional parts were used to determine the respective segments of the inputs. The corresponding $(a-1)(1+\widehat{f_1})$ and $(b-1)(1+\widehat{f_2})$ were selected based on the determined segments of the inputs using multiplexers.

To realize $(1+f_1)+(a-1)(1+\widehat{f_1})$, a 7-bit adder (A3 in Fig. 3) was used to compute the 7 MSBs of the result while the 24 LSBs were obtained directly from the lower 24-bits of f_1. A similar structure was used to realize $(1+f_2)+(b-1)(1+\widehat{f_2})$. A 24-bit adder (A5 in Fig. 3) was used to add the lower 24 bits of $(1 + f_1)$ and $(1 + f_2)$. The carry bit from the result was added along with the previous summation results from A3 and A4 to generate 8 MSBs of $(1+f_1)+(a-1)(1+\widehat{f_1})+(1+f_2)+(b-1)(1+\widehat{f_2})$ (refer A6 in Fig. 3).

Based on the determined segments of the inputs, the corresponding 5-bits value of c was selected from the 10 unique values ($c1$, $c2$, …$c10$ in Fig. 3) using multiplexers, and then subtracted from the summation result of A6. The 24-bits of $(1 + f_1) + (1 + f_2)$ were then appended to the result to compute the final $(1+f_1)+(a-1)(1+\widehat{f_1})+(1+f_2)+(b-1)(1+\widehat{f_2})-c$.

The final approximate multiplication output was obtained by shifting the approximate $(1 + f_1)(1 + f_2)$ product by $k_1 + k_2$ bits using a shifter. Similar architectures were developed for the proposed PLAM2 and PLAM8 designs.

The architecture can be extended for signed operation by using additional hardware to find the absolute values of the operands before the unsigned operation and to apply proper sign to the unsigned result.

Table 2
The values of a, b, and c computed using the proposed design procedure with $n = 4$.

		f_1			
		$[0, 0.25)$ $b = 1.00$	$[0.25, 0.5)$ $b = 1.25$	$[0.5, 0.75)$ $b = 1.50$	$[0.75, 1)$ $b = 1.75$
f_2	$[0, 0.25)$ $a = 1.00$	$c = 0.96875$	$c = 1.21875$	$c = 1.46875$	$c = 1.71875$
	$[0.25, 0.5)$ $a = 1.25$	$c = 1.21875$	$c = 1.53125$	$c = 1.84375$	$c = 2.15625$
	$[0.5, 0.75)$ $a = 1.50$	$c = 1.46875$	$c = 1.84375$	$c = 2.21875$	$c = 2.59375$
	$[0.75, 1)$ $a = 1.75$	$c = 1.71875$	$c = 2.15625$	$c = 2.59375$	$c = 3.03125$

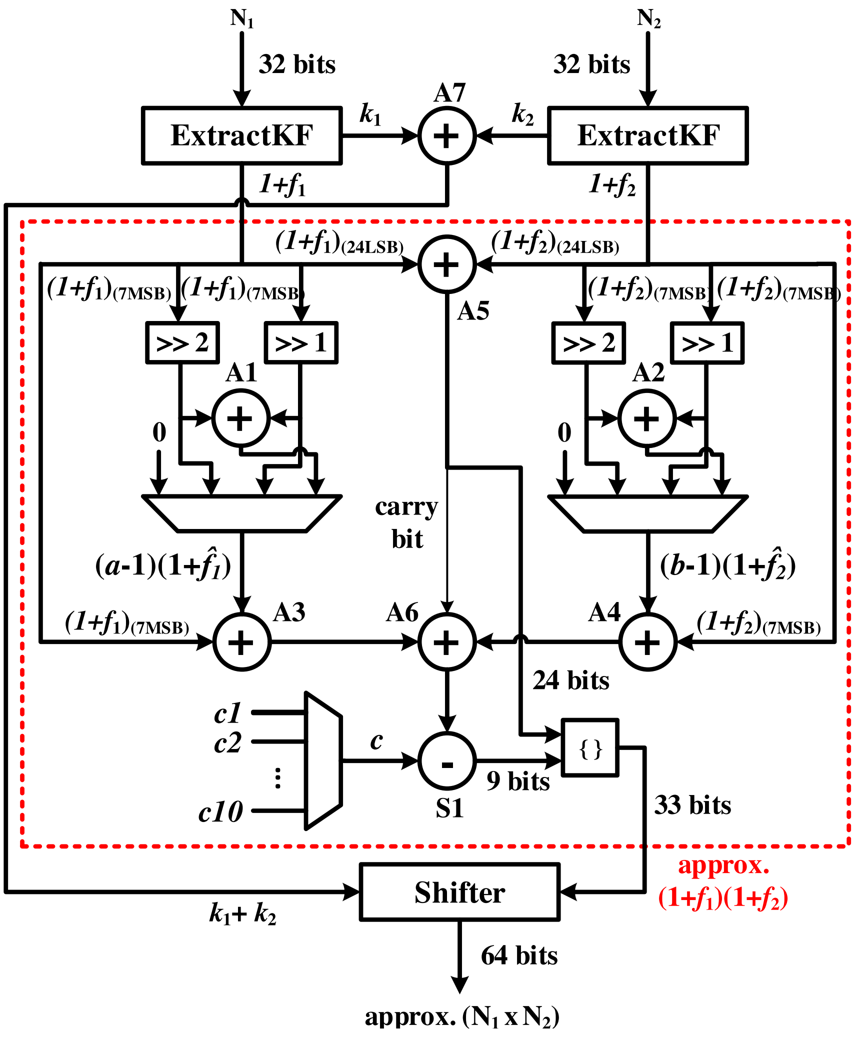


Fig. 3. Hardware architecture of the 32-bit PLAM4 design.

4. Results and discussion

4.1. VLSI implementation

The proposed 32-bit unsigned approximate multipliers were implemented using Verilog hardware description language. The designs were then synthesized in TSMC 65-nm CMOS technology using Synopsys Design Compiler. Gate-level simulations were then performed on the synthesized designs in Synopsys VCS (Verilog-Compiler-Simulator) using 100,000 random inputs of 32-bits, and the output errors were captured. The errors were then analyzed based on the mean error distance (MED), the normalized mean error distance (NMED), the mean relative error distance (MRED) and the worst case relative error (WCRE).

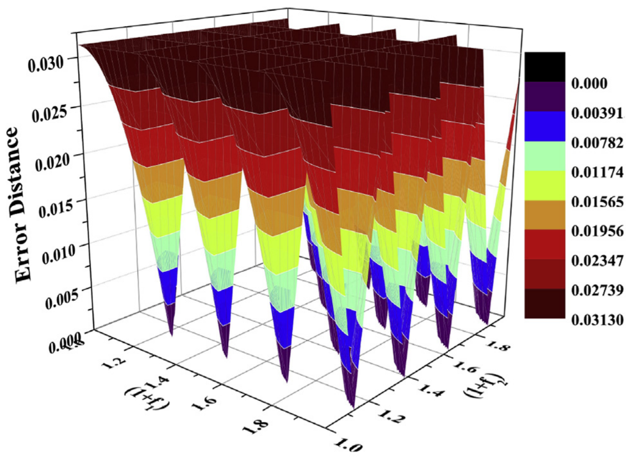
Fig. 4(a) shows the error profile of the proposed PLAM4 design for different inputs. The figure displays that the maximum error of $(1 + f_1)(1 + f_2)$ is the same for all the 16 regions. The relative error of the proposed hardware was also computed and captured in Fig. 4(b). It can be noted that the WCRE = 3.125% which aligns with (14) for n = 4. The MRED of the proposed hardware was computed as 0.927%.

4.2. Comparison with previous works

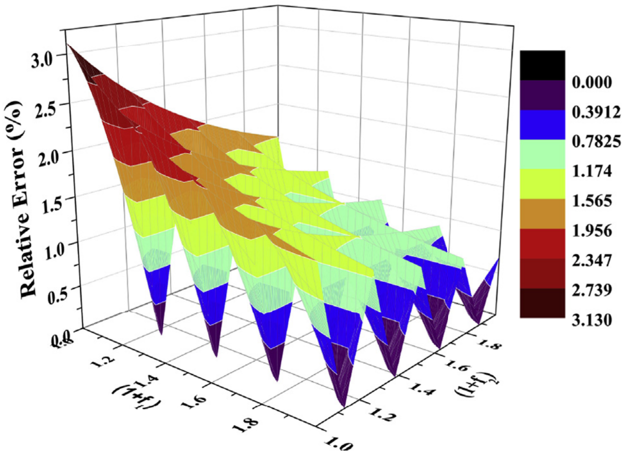
To evaluate the efficiencies, the proposed designs were compared with state-of-the-art approximate multipliers, viz.: DSM with 8 segments (DSM8) [16], DRUM with 6 segments (DRUM6) [17], unsigned RoBA (U-RoBA) multiplier [23], logarithmic multiplier (LM) [19] and approximate logarithmic multiplier with number of inexact bits, i.e., M = 20 (ALM-SOA20) [24]. The proposed designs were also compared with an exact 32-bit Wallace-tree multiplier. All the designs were implemented as 32-bit unsigned multipliers using Verilog hardware description language and synthesized in TSMC 65-nm CMOS technology using Synopsys Design Compiler. Errors were then analyzed by performing gate-level simulations on all the synthesized approximate multipliers using 100,000 random inputs of 32-bits. All the designs were synthesized and simulated at the typical process corner and at a nominal supply voltage of 1 V.

Table 3 highlights the synthesis results of PLAM2, PLAM4, and PLAM8 along with that of the selected state-of-the-art approximate multipliers and the exact Wallace-tree multiplier. The synthesis results were compared on the basis of area, power, delay, energy, energy-delay product (EDP) and area-delay-power product (ADP). The comparison results showed that DRUM6 had the least area while the proposed PLAM2 had the least power and delay. The proposed PLAM2 (PLAM4 and PLAM8) design exhibited an average reduction of 46.2% (44.6% and 28.4%) in delay when compared with that of the approximate multipliers.

The PLAM2 (PLAM4 and PLAM8) design showed 91.8% (90.2% and 83.0%) lower energy than that of the exact Wallace-tree multiplier. When compared with the DSM8 (DRUM6, U-RoBA, LM and ALM-



(a)



(b)

Fig. 4. (a) Error distance profile of the PLAM4 design. The error of the approximate $(1 + f_1)(1 + f_2)$ product is shown. (b) Relative error profile of the PLAM4 design.

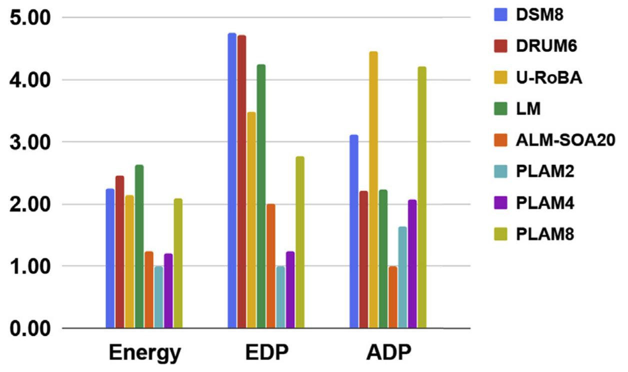


Fig. 5. Comparison of different approximate multipliers in terms of energy, energy-delay product (EDP) and area-delay-power product (ADP). The ratios were computed by setting the lowest value of each parameter to 1.

Table 4
Error analysis results of different 32-bit approximate multipliers.

	MED	NMED	MRED	WCRE
DSM8	2.20E+16	0.0012	0.53%	1.54%
DRUM6	6.30E+16	0.0034	1.47%	6.31%
U-RoBA	1.30E+17	0.0069	2.92%	11.11%
LM	1.70E+17	0.0093	3.86%	11.11%
ALM-SOA20	1.38E+17	0.0076	3.68%	12.12%
PLAM2	1.45E+17	0.0078	3.83%	12.50%
PLAM4	3.88E+16	0.0021	0.93%	3.13%
PLAM8	1.02E+16	0.0006	0.23%	0.78%

SOA20) design, energy and EDP of the proposed PLAM2 design were 55.6% (59.3%, 53.2%, 62.1% and 32.5%) and 78.9% (78.7%, 71.2%, 76.4% and 38.2%) lower respectively. The energy and EDP comparison also showed 46.5% (50.9%, 43.6%, 54.3% and 18.8%) and 73.8% (73.6%, 64.2%, 70.7% and 38.2%) respective reductions for the PLAM4 design. The proposed PLAM8 design showed reductions of 7.6% (15.2% and 2.5%) in energy, and 41.5% (41.0% and 20.1%) in EDP when compared with those of DSM8 (DRUM6 and U-RoBA).

In terms of ADP, the ALM-SOA20 design had the lowest value among the selected approximate multipliers while U-RoBA had the highest. The PLAM2 design showed a reduction of 63.4% in ADP when compared with that of U-RoBA. The PLAM4 design also showed lower ADP values when compared with that of DSM8, DRUM6, U-RoBA and LM designs exhibiting a maximum reduction of 53.5%. However, the PLAM8 design displayed greater ADP values when compared with that of DSM8, DRUM6 and LM. Nevertheless, it showed a reduction of 5.5% in ADP when compared with that of U-RoBA.

Fig. 5 summarizes the comparison of energy, EDP and ADP for the selected approximate multipliers in terms of ratios. The lowest value of each parameter was normalized to 1, and the ratios were computed accordingly. The comparison results highlight that the proposed design procedure can, therefore, be used to implement a range of low energy and hardware-efficient approximate multipliers with guaranteed worst case relative errors.

Error analysis on the selected approximate multipliers revealed that the PLAM8 design had the least WCRE among other designs under comparison as shown in Table 4. The design also had the least MED, NMED and MRED values.

4.3. Image processing application

The effectiveness of the proposed approximate multipliers and some of the state-of-the-art approximate multipliers was assessed by using

Table 3
Synthesis results of different 32-bit multipliers.

	Area (μm^2)	Power (mW)	Delay (ns)	Energy (pJ)	EDP (pJ $\times$ ns)	ADP (pJ $\times$ μm^2)
PLAM2	3958	1.03	1.63	1.67	2.72	6613
PLAM4	4174	1.20	1.68	2.01	3.38	8401
PLAM8	4907	1.60	2.17	3.48	7.55	17068
DSM8	3364	1.10	3.43	3.76	12.91	12656
DRUM6	2194	1.32	3.12	4.10	12.80	9000
U-RoBA	5064	1.35	2.65	3.57	9.45	18066
LM	2054	1.68	2.62	4.40	11.53	9041
ALM-SOA20	1637	1.12	2.21	2.48	5.47	4052
Wallace	15172	5.70	3.60	20.52	73.87	311329

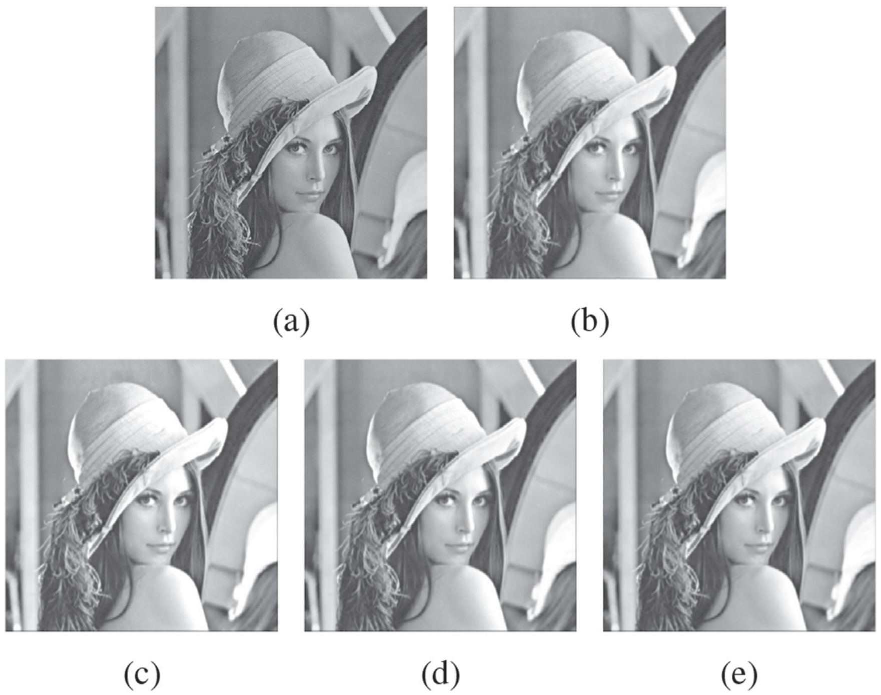


Fig. 6. Image smoothening using the proposed approximate multipliers. (a) Original image. Image smoothening using (b) exact multiplier, (c) the PLAM2 design, (d) the PLAM4 design, and (e) the PLAM8 design.

Table 5
PSNR (in dB) and MSSIM values of the outputs of image smoothening using different approximate multiplier designs.

	DSM8		DRUM6		U-RoBA		LM		ALM-SOA20		PLAM2		PLAM4		PLAM8	
	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM
Peppers	∞	1.00	34.82	0.97	43.27	1.00	20.00	0.83	10.80	0.57	38.46	0.97	43.40	1.00	49.08	1.00
House	∞	1.00	33.54	0.89	41.45	0.99	17.84	0.72	10.16	0.62	37.44	0.99	43.27	1.00	49.03	1.00
F-16	∞	1.00	33.25	0.85	40.60	0.95	16.82	0.72	7.56	0.61	37.19	0.91	43.07	1.00	48.88	1.00
Mandrill	∞	1.00	34.82	1.00	43.72	1.00	19.21	0.65	11.82	0.48	37.68	0.99	43.49	1.00	49.28	1.00
Lena	∞	1.00	34.63	0.96	44.47	1.00	19.73	0.74	11.21	0.64	37.58	0.99	43.37	1.00	49.21	1.00
Average	∞	1.00	34.21	0.93	42.70	0.99	18.72	0.73	10.31	0.58	37.67	0.97	43.32	1.00	49.10	1.00

them in an image smoothening application. The image smoothening output is determined using the following equation:

$$Y(i,j) = \frac{1}{60}\sum_{m=-2}^{m=2}\sum_{n=-2}^{n=2} X(i+m,j+n)M_{smooth}(m+3,n+3) \qquad (16)$$

where $X(i,j)$ and $Y(i,j)$ denote the pixel of the ith row and the jth column of input and output images respectively and M_{smooth} is given by

$$M_{smooth} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 4 & 4 & 4 & 4 \\ 1 & 4 & 12 & 4 & 7 \\ 1 & 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \qquad (17)$$

Five benchmark images from Ref. [28] were used for the image smoothening application. As an example, Fig. 6 presents the original image of Lena, and the output images produced by image smoothening using an exact multiplier (Fig. 6(b)) and the proposed approximate multipliers (Fig. 6(c)–(e)). The quality of the smoothening process may not be easily assessed by human eyes. To measure the quality of the approximate output images, two metrics, viz., the peak signal-to-noise ratio (PSNR) and mean structural similarity index metric (MSSIM) were used [29]. Table 5 displays the PSNR (in dB) and MSSIM results of different approximate multipliers utilized for image smoothening on the benchmark images. As the results show, the proposed PLAM8 design showed an average PSNR (MSSIM) of 49.10 dB (1.0) demonstrating a higher output image quality than the rest of the approximate multipliers except DSM8. In all the benchmarks, DSM8 showed the best performance as that of exact multiplication, hence having PSNR = ∞. The PLAM4 design also showed an average PSNR (MSSIM) of 43.32 dB (1.0) which is better than that of DRUM6, U-RoBA, LM and ALM-SOA20. The PLAM2 design showed an inferior average PSNR (MSSIM) of 37.67 dB (0.97) to that of DSM8, DRUM6 and U-RoBA. However, such PSNR (MSSIM) value is acceptable in many applications such as despeckling of synthetic aperture radar (SAR) images [30,31].

5. Conclusion

A two-dimensional piecewise linear approximation method for multiplication is proposed in this paper. A formal analysis to compute the worst case relative error of the approximation method is discussed. Based on this method, a design procedure to implement energy-efficient approximate multiplier with guaranteed worst case relative error is also proposed. Three 32-bit unsigned approximate multipliers, viz., PLAM2, PLAM4, and PLAM8 were implemented using the design procedure with guaranteed worst case relative errors of 12.5%, 3.13%, and 0.78% respectively. The proposed designs were compared with an exact and some state-of-the-art approximate multipliers. The proposed designs showed up to 91.8% lower energy than that of the exact multiplier. The PLAM2 design delivered up to 62.1% energy saving at the cost of higher approximation error when compared with that of the approximate multipliers. The PLAM8 design showed the least WCRE of 0.78% among the compared designs. The PLAM8 also exhibited the least MED, NMED and MRED among other designs. Furthermore, the effectiveness of the approximate multipliers was assessed in an image smoothening application. In our future work, the proposed approximate multipliers will be extended to support signed multiplication.

References

[1] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, Z. Zhang, Hardware for machine learning: challenges and opportunities, in: Custom Integrated Circuits Conference (CICC), IEEE, 2018, pp. 1–8.
[2] M. Alioto, Ultra-low power VLSI circuit design demystified and explained: a tutorial, IEEE Trans. Circ. Syst. I: Reg. Pap. 59 (1) (2012) 3–29.
[3] P. Kulkarni, P. Gupta, M. Ercegovac, Trading accuracy for power with an underdesigned multiplier architecture, in: VLSI Design (VLSI Design), 2011 24th International Conference on, IEEE, 2011, pp. 346–351.
[4] H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, C. Lucas, Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications, IEEE Trans. Circ. Syst. I: Reg. Pap. 57 (4) (2010) 850–862.
[5] R. Venkatesan, A. Agarwal, K. Roy, A. Raghunathan, MACACO: modeling and analysis of circuits for approximate computing, in: Proceedings of the International Conference on Computer-Aided Design, IEEE Press, 2011, pp. 667–673.
[6] K. Roy, A. Raghunathan, Approximate computing: an energy-efficient computing technique for error resilient applications, in: VLSI (ISVLSI), 2015 IEEE Computer Society Annual Symposium on, IEEE, 2015, pp. 473–475.
[7] R. Hegde, N.R. Shanbhag, Energy-efficient signal processing via algorithmic noise-tolerance, in: Low Power Electronics and Design, 1999. Proceedings. 1999 International Symposium on, IEEE, 1999, pp. 30–35.
[8] F. Farshchi, M.S. Abrishami, S.M. Fakhraie, New approximate multiplier for low power digital signal processing, in: Computer Architecture and Digital Systems (CADS), 2013 17th CSI International Symposium on, IEEE, 2013, pp. 25–30.
[9] C.-H. Lin, C. Lin, High accuracy approximate multiplier with error correction, in: Computer Design (ICCD), 2013 IEEE 31st International Conference on, IEEE, 2013, pp. 33–38.
[10] A. Momeni, J. Han, P. Montuschi, F. Lombardi, Design and analysis of approximate compressors for multiplication, IEEE Trans. Comput. 64 (4) (2015) 984–994.
[11] O. Akbari, M. Kamal, A. Afzali-Kusha, M. Pedram, Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers, IEEE Trans. Very Large Scale Integr. Syst. 25 (4) (2017) 1352–1361.
[12] C. Liu, J. Han, F. Lombardi, A low-power, high-performance approximate multiplier with configurable partial error recovery, in: Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014, IEEE, 2014, pp. 1–4.
[13] V. Camus, J. Schlachter, C. Enz, M. Gautschi, F. Gurkaynak, Approximate 32-bit floating-point unit design with 53% power-area product reduction, in: Proceedings of the 42nd European Solid-State Circuits Conference, IEEE, 2016, pp. 465–468.
[14] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, F. Lombardi, Design of approximate radix-4 booth multipliers for error-tolerant computing, IEEE Trans. Comput. 66 (8) (2017) 1435–1441.
[15] V. Leon, G. Zervakis, D. Soudris, K. Pekmestzi, Approximate hybrid high radix encoding for energy-efficient inexact multipliers, IEEE Trans. Very Large Scale Integr. Syst. 26 (3) (2018) 421–430.
[16] S. Narayanamoorthy, H.A. Moghaddam, Z. Liu, T. Park, N.S. Kim, Energy-efficient approximate multiplication for digital signal processing and classification applications, IEEE Trans. Very Large Scale Integr. Syst. 23 (6) (2015) 1180–1184.
[17] S. Hashemi, R. Bahar, S. Reda, DRUM: a dynamic range unbiased multiplier for approximate applications, in: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, IEEE Press, 2015, pp. 418–425.
[18] S. Vahdat, M. Kamal, A. Afzali-Kusha, M. Pedram, LETAM: a low energy truncation-based approximate multiplier, Comput. Electr. Eng. 63 (2017) 1–17.
[19] J.N. Mitchell, Computer multiplication and division using binary logarithms, IRE Trans. Electronic Comput. (4) (1962) 512–517.
[20] E.L. Hall, D.D. Lynch, S.J. Dwyer, Generation of products and quotients using approximate binary logarithms for digital filtering applications, IEEE Trans. Comput. 100 (2) (1970) 97–105.
[21] D.J. McLaren, Improved Mitchell-based logarithmic multiplier for low-power DSP applications, in: SOC Conference, 2003. Proceedings. IEEE International [Systems-On-Chip], IEEE, 2003, pp. 53–56.
[22] V. Mahalingam, N. Ranganathan, Improving accuracy in Mitchell's logarithmic multiplication using operand decomposition, IEEE Trans. Comput. 55 (12) (2006) 1523–1535.
[23] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, M. Pedram, RoBA multiplier: a rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing, IEEE Trans. Very Large Scale Integr. Syst. (2) (2017) 393–401.
[24] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, F. Lombardi, Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications, IEEE Trans. Circ. Syst. I: Reg. Pap. 65 (9) (2018) 2856–2868.
[25] V. Mrazek, Z. Vasicek, L. Sekanina, H. Jiang, J. Han, Scalable construction of approximate multipliers with formally guaranteed worst case error, IEEE Trans. Very Large Scale Integr. Syst. 99 (2018) 1–5.
[26] V. Mrazek, R. Hrbacek, Z. Vasicek, L. Sekanina, Evoapprox8b: library of approximate adders and multipliers for circuit design and benchmarking of approximation methods, in: Proceedings of the Conference on Design, Automation & Test in Europe, European Design and Automation Association, 2017, pp. 258–261.
[27] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, D. Grossman, EnerJ: approximate data types for safe and general low-power computation, in: ACM SIGPLAN Notices, vol. 46, ACM, 2011, pp. 164–174.
[28] Sipi.usc.edu, SIPI Image Database, 2016. [Online]. Available: http://sipi.usc.edu/database/.
[29] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[30] B. Xu, Y. Cui, Z. Li, B. Zuo, J. Yang, J. Song, Patch ordering-based SAR image despeckling via transform-domain filtering, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 8 (4) (2015) 1682–1695.
[31] P. Singh, R. Shree, A new SAR image despeckling using directional smoothing filter and method noise thresholding, Eng. Sci. Tech. Int. J. 21 (4) (2018) 589–610.