Transactions Briefs

Design of Power and Area Efficient Approximate Multipliers

Suganthi Venkatachalam and Seok-Bum Ko, *Senior Member, IEEE*

***Abstract*— Approximate computing can decrease the design complexity with an increase in performance and power efficiency for error resilient applications. This brief deals with a new design approach for approximation of multipliers. The partial products of the multiplier are altered to introduce varying probability terms. Logic complexity of approximation is varied for the accumulation of altered partial products based on their probability. The proposed approximation is utilized in two variants of 16-bit multipliers. Synthesis results reveal that two proposed multipliers achieve power savings of 72% and 38%, respectively, compared to an exact multiplier. They have better precision when compared to existing approximate multipliers. Mean relative error figures are as low as 7.6% and 0.02% for the proposed approximate multipliers, which are better than the previous works. Performance of the proposed multipliers is evaluated with an image processing application, where one of the proposed models achieves the highest peak signal to noise ratio.**

***Index Terms*— Approximate computing, error analysis, low error, low power, multipliers.**

I. Introduction

In applications like multimedia signal processing and data mining which can tolerate error, exact computing units are not always necessary. They can be replaced with their approximate counterparts. Research on approximate computing for error tolerant applications is on the rise. Adders and multipliers form the key components in these applications. In [1], approximate full adders are proposed at transistor level and they are utilized in digital signal processing applications. Their proposed full adders are used in accumulation of partial products in multipliers.

To reduce hardware complexity of multipliers, truncation is widely employed in fixed-width multiplier designs. Then a constant or variable correction term is added to compensate for the quantization error introduced by the truncated part [2], [3]. Approximation techniques in multipliers focus on accumulation of partial products, which is crucial in terms of power consumption. Broken array multiplier is implemented in [4], where the least significant bits of inputs are truncated, while forming partial products to reduce hardware complexity. The proposed multiplier in [4] saves few adder circuits in partial product accumulation.

In [5], two designs of approximate 4-2 compressors are presented and used in partial product reduction tree of four variants of 8 $\times$ 8 Dadda multiplier. The major drawback of the proposed compressors in [5] is that they give nonzero output for zero valued inputs, which largely affects the mean relative error (MRE) as discussed later. The approximate design proposed in this brief overcomes the existing drawback. This leads to better precision. In static segment multiplier (SSM) proposed in [6], m-bit segments are derived from n-bit operands based on leading 1 bit of the operands. Then, $m \times m$ multiplication is performed instead of $n \times n$ multiplication, where $m<n$. Partial product perforation (PPP) multiplier in [7] omits k successive partial products starting from jth position, where $j \in [0, n\text{-}1]$ and $k \in [1, \min(n\text{-}j, n\text{-}1)]$ of a n-bit multiplier. In [8], 2 $\times$ 2 approximate multiplier based on modifying an entry in the Karnaugh map is proposed and used as a building block to construct 4 $\times$ 4 and 8 $\times$ 8 multipliers. In [9], inaccurate counter design has been proposed for use in power efficient Wallace tree multiplier. A new approximate adder is presented in [10] which is utilized for partial product accumulation of the multiplier. For 16-bit approximate multiplier in [10], 26% of reduction in power is accomplished compared to exact multiplier. Approximation of 8-bit Wallace tree multiplier due to voltage over-scaling (VOS) is discussed in [11]. Lowering supply voltage creates paths failing to meet delay constraints leading to error.

Previous works on logic complexity reduction focus on straightforward application of approximate adders and compressors to the partial products. In this brief, the partial products are altered to introduce terms with different probabilities. Probability statistics of the altered partial products are analyzed, which is followed by systematic approximation. Simplified arithmetic units (half-adder, full-adder, and 4-2 compressor) are proposed for approximation. The arithmetic units are not only reduced in complexity, but care is also taken that error value is maintained low. While systemic approximation helps in achieving better accuracy, reduced logic complexity of approximate arithmetic units consumes less power and area. The proposed multipliers outperforms the existing multiplier designs in terms of area, power, and error, and achieves better peak signal to noise ratio (PSNR) values in image processing application.

Error distance (ED) can be defined as the arithmetic distance between a correct output and approximate output for a given input. In [12], approximate adders are evaluated and normalized ED (NED) is proposed as nearly invariant metric independent of the size of the approximate circuit. Also, traditional error analysis, MRE is found for existing and proposed multiplier designs.

The rest of this brief is organized as follows. Section II details the proposed architecture. Section III provides extensive result analysis of design and error metrics of the proposed and existing approximate multipliers. The proposed multipliers are utilized in image processing application and results are provided in Section IV. Section V concludes this brief.

II. Proposed Architecture

Implementation of multiplier comprises three steps: generation of partial products, partial products reduction tree, and finally, a vector merge addition to produce final product from the sum and carry rows generated from the reduction tree. Second step consumes more power. In this brief, approximation is applied in reduction tree stage.

Manuscript received May 18, 2016; accepted December 19, 2016. Date of publication January 16, 2017; date of current version April 24, 2017. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Department of Electrical and Computer Engineering, University of Saskatchewan, Saskatoon, SK, Canada.

The authors are with the Department of Electrical and Computer Engineering, University of Saskatchewan, Saskatoon, SK S7N 5C5, Canada (e-mail: seokbum.ko@usask.ca; suganthi.venkat@usask.ca).

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.2016.2643639

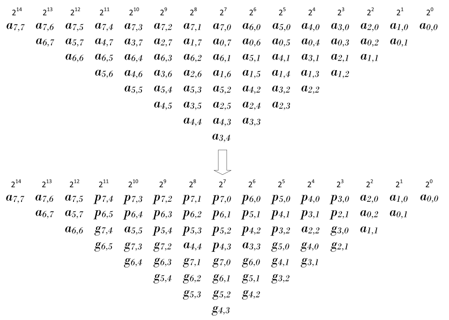
Fig. 1. Transformation of generated partial products into altered partial products.

TABLE I
PROBABILITY STATISTICS OF *Generate* SIGNALS

m	Probability of the *generate* elements being				P_{err}
	all zero	one 1	two 1's	three 1's and more	
2	0.8789	0.1172	0.0039	-	0.00390
3	0.8240	0.1648	0.0110	0.00024	0.01124
4	0.7725	0.2060	0.0206	0.00093	0.02153

A 8-bit unsigned[1] multiplier is used for illustration to describe the proposed method in approximation of multipliers. Consider two 8-bit unsigned input operands $\alpha = \sum_{m=0}^{7} \alpha_m 2^m$ and $\beta = \sum_{n=0}^{7} \beta_n 2^n$. The partial product $a_{m,n} = \alpha_m \cdot \beta_n$ in Fig. 1 is the result of AND operation between the bits of α_m and β_n.

From statistical point of view, the partial product $a_{m,n}$ has a probability of 1/4 of being 1. In the columns containing more than three partial products, the partial products $a_{m,n}$ and $a_{n,m}$ are combined to form *propogate* and *generate* signals as given in (1). The resulting *propogate* and *generate* signals form altered partial products $p_{m,n}$ and $g_{m,n}$. From column 3 with weight 2^3 to column 11 with weight 2^{11}, the partial products $a_{m,n}$ and $a_{n,m}$ are replaced by altered partial products $p_{m,n}$ and $g_{m,n}$. The original and transformed partial product matrices are shown in Fig. 1

$$\begin{aligned} p_{m,n} &= a_{m,n} + a_{n,m} \\ g_{m,n} &= a_{m,n} \cdot a_{n,m}. \end{aligned} \tag{1}$$

The probability of the altered partial product $g_{m,n}$ being one is 1/16, which is significantly lower than 1/4 of $a_{m,n}$. The probability of altered partial product $p_{m,n}$ being one is $1/16 + 3/16 + 3/16 = 7/16$, which is higher than $g_{m,n}$. These factors are considered, while applying approximation to the altered partial product matrix.

A. Approximation of Altered Partial Products $g_{m,n}$

The accumulation of *generate* signals is done columnwise. As each element has a probability of 1/16 of being one, two elements being 1 in the same column even decreases. For example, in a column with 4 *generate* signals, probability of all numbers being 0 is $(1 - pr)^4$, only one element being one is $4pr(1 - pr)^3$, the probability of two elements being one in the column is $6pr^2(1 - pr)^2$, three ones is $4pr^3(1-pr)$ and probability of all elements being 1 is pr^4, where pr is 1/16. The probability statistics for a number of *generate* elements m in each column are given in Table I.

Based on Table I, using OR gate in the accumulation of columnwise *generate* elements in the altered partial product matrix provides exact result in most of the cases. The probability of error (P_{err}) while using OR gate for reduction of *generate* signals in each column is also listed in Table I. As can be seen, the probability of misprediction is very low. As the number of *generate* signals increases, the error probability increases linearly. However, the value of error also rises. To prevent this, the maximum number of *generate* signals to be grouped by OR gate is kept at 4. For a column having m *generate* signals, $\lceil m/4 \rceil$ OR gates are used.

[1]The proposed approximate technique can be applied to signed multiplication including Booth multipliers as well, except it is not applied to sign extension bits.

TABLE II
TRUTH TABLE OF APPROXIMATE HALF ADDER

Inputs		Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	$Carry$	Sum	$Carry$	Sum	
0	0	0	0	0 ✔	0 ✔	0
0	1	0	1	0 ✔	1 ✔	0
1	0	0	1	0 ✔	1 ✔	0
1	1	1	0	1 ✔	1 ✗	1

TABLE III
TRUTH TABLE OF APPROXIMATE FULL ADDER

Inputs			Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	$x3$	$Carry$	Sum	$Carry$	Sum	
0	0	0	0	0	0 ✔	0 ✔	0
0	0	1	0	1	0 ✔	1 ✔	0
0	1	0	0	1	0 ✔	1 ✔	0
0	1	1	1	0	1 ✔	0 ✔	0
1	0	0	0	1	0 ✔	1 ✔	0
1	0	1	1	0	1 ✔	0 ✔	0
1	1	0	1	0	0 ✗	1 ✗	1
1	1	1	1	1	1 ✔	0 ✗	1

B. Approximation of Other Partial Products

The accumulation of other partial products with probability 1/4 for $a_{m,n}$ and 7/16 for $p_{m,n}$ uses approximate circuits. Approximate half-adder, full-adder, and 4-2 compressor are proposed for their accumulation. $Carry$ and Sum are two outputs of these approximate circuits. Since $Carry$ has higher weight of binary bit, error in $Carry$ bit will contribute more by producing error difference of two in the output. Approximation is handled in such a way that the absolute difference between actual output and approximate output is always maintained as one. Hence $Carry$ outputs are approximated only for the cases, where Sum is approximated.

In adders and compressors, XOR gates tend to contribute to high area and delay. For approximating half-adder, XOR gate of Sum is replaced with OR gate as given in (2). This results in one error in the Sum computation as seen in the truth table of approximate half-adder in Table II. A tick mark denotes that approximate output matches with correct output and cross mark denotes mismatch

$$\begin{aligned} Sum &= x1 + x2 \\ Carry &= x1 \cdot x2. \end{aligned} \tag{2}$$

In the approximation of full-adder, one of the two XOR gates is replaced with OR gate in Sum calculation. This results in error in last two cases out of eight cases. $Carry$ is modified as in (3) introducing one error. This provides more simplification, while maintaining the difference between original and approximate value as one. The truth table of approximate full-adder is given in Table III

$$\begin{aligned} W &= (x1 + x2) \\ Sum &= W \oplus x3 \\ Carry &= W \cdot x3. \end{aligned} \tag{3}$$

Two approximate 4-2 compressors in [5] produce nonzero output even for the cases where all inputs are zero. This results in high ED and high degree of precision loss especially in cases of zeros in all bits or in most significant parts of the reduction tree. The proposed 4-2 compressor overcomes this drawback.

In 4-2 compressor, three bits are required for the output only when all the four inputs are 1, which happens only once out of 16 cases.

TABLE IV
TRUTH TABLE OF APPROXIMATE 4-2 COMPRESSOR

Inputs				Approximate outputs		Absolute Difference
$x1$	$x2$	$x3$	$x4$	$Carry$	Sum	
0	0	0	0	0 ✔	0 ✔	0
0	0	0	1	0 ✔	1 ✔	0
0	0	1	0	0 ✔	1 ✔	0
0	0	1	1	1 ✔	0 ✔	0
0	1	0	0	0 ✔	1 ✔	0
0	1	0	1	0 ✗	1 ✗	1
0	1	1	0	0 ✗	1 ✗	1
0	1	1	1	1 ✔	1 ✔	0
1	0	0	0	0 ✔	1 ✔	0
1	0	0	1	0 ✗	1 ✗	1
1	0	1	0	0 ✗	1 ✗	1
1	0	1	1	1 ✔	1 ✔	0
1	1	0	0	1 ✔	0 ✔	0
1	1	0	1	1 ✔	1 ✔	0
1	1	1	0	1 ✔	1 ✔	0
1	1	1	1	1 ✗	1 ✗	1

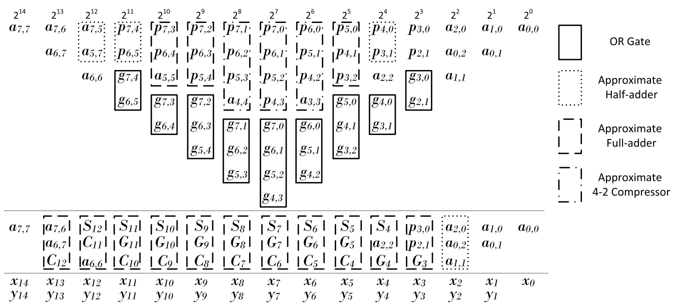

Fig. 2. Reduction of altered partial products.

This property is taken to eliminate one of the three output bits in 4-2 compressor. To maintain minimal error difference as one, the output "100" (the value of 4) for four inputs being one has to be replaced with outputs "11" (the value of 3). For Sum computation, one out of three XOR gates is replaced with OR gate. Also, to make the Sum corresponding to the case where all inputs are ones as one, an additional circuit $x1 \cdot x2 \cdot x3 \cdot x4$ is added to the Sum expression. This results in error in five out of 16 cases. $Carry$ is simplified as in (4). The corresponding truth table is given in Table IV

$$\begin{aligned} W1 &= x1 \cdot x2 \\ W2 &= x3 \cdot x4 \\ Sum &= (x1 \oplus x2) + (x3 \oplus x4) + W1 \cdot W2 \\ Carry &= W1 + W2. \end{aligned} \tag{4}$$

Fig. 2 shows the reduction of altered partial product matrix of 8 × 8 approximate multiplier. It requires two stages to produce sum and carry outputs for vector merge addition step. Four 2-input OR gates, four 3-input OR gates, and one 4-input OR gates are required for the reduction of *generate* signals from columns 3 to 11. The resultant signals of OR gates are labeled as G_i corresponding to the column i with weight 2^i. For reducing other partial products, 3 approximate half-adders, 3 approximate full-adders, and 3 approximate compressors are required in the first stage to produce Sum and $Carry$ signals, S_i and C_i corresponding to column i. The elements in the second stage are reduced using 1 approximate half-adder and 11 approximate full-adders producing final two operands x_i and y_i to be fed to ripple carry adder for the final computation of the result.

C. Two Variants of Multipliers

Two variants of multipliers are proposed. In the first case (Multiplier1), approximation is applied in all columns of partial products of n-bit multiplier, whereas in Multiplier2, approximate circuits are used in $n - 1$ least significant columns.

TABLE V
SYNTHESIS RESULTS OF EXACT, EXISTING, AND PROPOSED APPROXIMATE MULTIPLIERS

Multiplier Type	Area (μm^2)	Delay (ns)	Power (μW)	PDP (fJ)	APP ($\mu m^2 \cdot \mu W$)(10^5)
Exact	4859.28	0.68	1776.49	1208.01	86.32
Multiplier1	2158.56	0.47	503.15	236.48	10.86
Multiplier2	3319.20	0.66	1102.03	727.34	36.57
ACM1 [5]	2871.72	0.4	435.31	174.12	12.50
ACM2 [5]	3782.16	0.63	1250.70	787.94	47.30
SSM [6]	3953.88	0.69	1225.29	845.45	48.44
PPP [7]	4547.52	0.64	1570.79	1005.31	71.43
UDM [8]	3938.00	0.67	1318.51	883.40	51.92

TABLE VI
ERROR METRICS FOR 16-bit MULTIPLIER

Multiplier	Mean Relative Error	Normalized Error Distance
Multiplier1	7.63×10^{-2}	1.78×10^{-2}
Multiplier2	2.44×10^{-4}	7.10×10^{-6}
ACM1 [7]	16.6	4.96×10^{-2}
ACM2 [7]	2.30×10^{-3}	6.36×10^{-6}
SSM [8]	6.34×10^{-4}	1.07×10^{-4}
PPP [9]	8.98×10^{-4}	4.58×10^{-5}
UDM [10]	3.32×10^{-2}	1.39×10^{-2}

III. RESULTS AND DISCUSSION

All approximate multipliers are designed for $n = 16$. The multipliers are implemented in Verilog and synthesized using Synopsys Design Compiler and a TSMC 65 nm standard cell library at the typical process corner, with temperature 25 °C and supply voltage 1 V. From the Synopsys dc reports, we get area, delay, dynamic power and leakage power. Multiplier1 applies approximation in all columns, whereas in Multiplier2, approximation is applied in 15 least significant columns during partial product reduction. For the proposed multipliers, the altered partial products are generated and compressed using half-adder, full-adder, and 4-2 compressor structures to form final two rows of partial products. The efficiency of the proposed multipliers is compared with existing approximate multipliers [5]–[8]. Inexact compressor design 2 of [5] is used to design compressor based multipliers ACM1, where all columns are approximated and ACM2, where only 15 least significant columns are approximated. SSM [6] for $m = 12$ and $n = 16$ is designed for implementation. PPP design discussed in [7] for $j = 2$, $k = 2$ is designed and implemented under Dadda tree structure. In [8], the partial product matrix of 16-bit under designed multiplier (UDM) comprises approximate 2 × 2 partial products accumulated together with exact carry save adders. Exhaustive error analysis of the approximate multipliers is done using MATLAB.

Exact 16-bit multiplier is designed using Dadda tree structure. Table V compares all designs in terms of area, delay, power, power delay product (PDP), and area power product (APP). NED and MRE of the approximate multipliers are listed in Table VI. If high approximation can be tolerated for saving more power, Multiplier1 and ACM1 are the candidates to be considered. It can be seen that Multiplier1 has better APP, whereas ACM1 has better PDP. However, Multiplier1 has 64% lower NED and three orders of magnitude lower MRE, compared to ACM1. It should be noted that high values of MRE for ACMs are due to nonzero output for inputs with all zeros.

TABLE VII
RANKING OF APPROXIMATE MULTIPLIERS IN TERMS OF DESIGN AND ERROR METRICS

Approximate Multiplier Type	APP Gain	PDP Gain	NED	MRE
Multiplier1	1	2	6	6
Multiplier2	3	3	2	1
ACM1 [5]	2	1	7	7
ACM2 [5]	4	4	1	4
SSM [6]	5	5	4	2
PPP [7]	7	7	3	3
UDM [8]	6	6	5	5

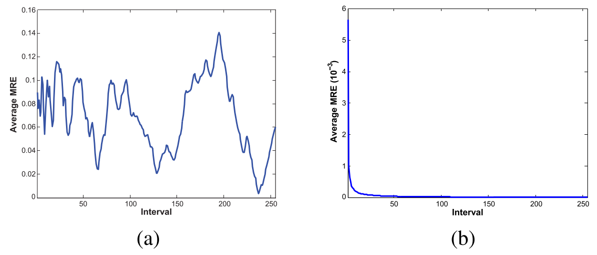

Fig. 3. MRE distribution of (a) Multiplier1 and (b) Multiplier2.

Multiplier2 offers 32% area savings and 38% power savings, over the exact multiplier. ACM2 provides 22% and 30% area and power savings, respectively. SSM has 19% area and 31% power savings over accurate multiplier. Perforated multiplier has 6% and 12% area and power savings, respectively. UDM provides 19% and 26% area and power savings. Multiplier2 has one order of lower MRE than ACM2, two orders of lower MRE than UDM, 73% lower MRE than PPP, and 62% lower MRE than SSM. NED of Multiplier2 outperforms all approximate multipliers except ACM2. ACM2 exhibits 10% lower NED than Multiplier2. Multiplier2 produces large ED relative to ACM2. However, lower MRE indicates that Multiplier2 has smaller relative error values.

Table VII gives a comprehensive comparison of approximate multipliers to get an idea of tradeoff between design metrics and error metrics. Multiplier1 delivers the lowest APP; Multiplier2 delivers the lowest MRE value. Overall, Multiplier2 has better PDP, APP, and MRE over ACM2, SSM, perforated multiplier, and UDM, with lower NED in most cases as well. For applications where high power savings are desired with more error tolerance, Multiplier1 can be used. For moderate power savings with better performance, Multiplier2 is suggested.

MRE distribution of 16-bit versions of Multiplier1 and Multiplier2 is shown in Fig. 3. All possible outputs ranging from 0 to 65535^2 are categorized into 255 intervals. MRE of Multiplier2 is significantly low at higher product values, as exact units are used in most significant part of the multiplier.

IV. APPLICATION—IMAGE PROCESSING

Geometric mean filter is widely used in image processing to reduce Gaussian noise [13]. The geometric mean filter is better at preserving edge features than the arithmetic mean filter. Two 16-bits per pixel gray scale images with Gaussian noise are considered. 3×3 mean filter is used, where each pixel of noisy image is replaced with geometric mean of 3×3 block of neighboring pixels centered around it. The algorithms are coded and implemented in MATLAB. Exact and approximate 16-bit multipliers are used to perform multiplication between 16-bit pixels. PSNR is used as figure

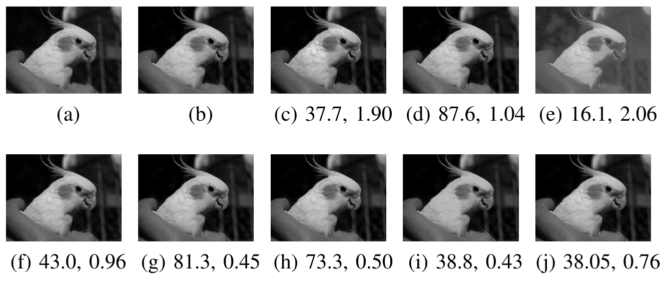

Fig. 4. (a) Input image-1 with Gaussian noise. Geometric mean filtered images and corresponding PSNR and energy savings in μJ using (b) exact multiplier, (c) Multiplier1, (d) Multiplier2, (e) ACM1, (f) ACM2, (g) SSM, (h) PPP, (i) UDM, and (j) VOS.

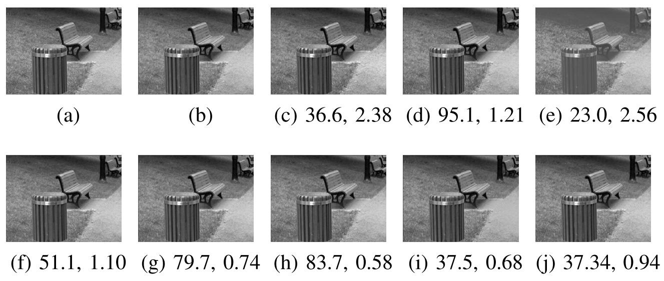

Fig. 5. (a) Input image-2 with Gaussian noise. Geometric mean filtered images and corresponding PSNR and energy savings in μJ using (b) exact multiplier, (c) Multiplier1, (d) Multiplier2, (e) ACM1, (f) ACM2, (g) SSM, (h) PPP, (i) UDM, and (j) VOS.

of merit to assess the quality of approximate multipliers. PSNR is based on mean-square error found between resulting image of exact multiplier and the images generated from approximate multipliers. Energy required by exact and approximate multiplication process while performing geometric mean filtering of the images is found using Synopsys Primetime. Further, exact multiplier is voltage scaled from 1 to 0.85 V (VOS), and its impact on energy consumption and image quality is computed.

The noisy input image and resultant image after denoising using exact and approximate multipliers, with their respective PSNRs and energy savings in μJ are shown in Figs. 4 and 5, respectively. Energy required for exact multiplication process for image-1 and image-2 is 3.24 and 2.62 μJ, respectively. Although ACM1 has better energy savings compared to Multiplier1, Multiplier1 has significantly higher PSNR than ACM1. Multiplier2 shows the best PSNR among all the approximate designs. Multiplier2 has better energy savings, compared to ACM2, PPP, SSM, UDM, and VOS. The intensity of image-1 being mostly on the lower end of the histogram causes poor performance of ACM multipliers. As the switching activity impacts most significant part of the design in VOS, PSNR values are affected.

V. CONCLUSION

In this brief, to propose efficient approximate multipliers, partial products of the multiplier are modified using *generate* and *propagate* signals. Approximation is applied using simple OR gate for altered *generate* partial products. Approximate half-adder, full-adder, and 4-2 compressor are proposed to reduce remaining partial products. Two variants of approximate multipliers are proposed, where approximation is applied in all n bits in Multiplier1 and only in $n-1$ least significant part in Multiplier2. Multiplier1 and Multiplier2 achieve significant reduction in area and power consumption compared with exact designs. With APP savings being 87% and 58% for Multiplier1 and Multiplier2 with respect to exact multipliers, they also outperform in APP in comparison with existing approximate designs. They are also found to have better precision when compared to existing approximate multiplier designs. The proposed multiplier designs can be used in applications with minimal loss in output quality while saving significant power and area.

References

[1] 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.

[2] E. J. King and E. E. Swartzlander, Jr., "Data-dependent truncation scheme for parallel multipliers," in *Proc. 31st Asilomar Conf. Signals, Circuits Syst.*, Nov. 1998, pp. 1178–1182.

[3] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low-error fixed-width modified booth multiplier," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 5, pp. 522–531, May 2004.

[4] 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.

[5] 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.

[6] 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.

[7] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Design-efficient approximate multiplication circuits through partial product perforation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 10, pp. 3105–3117, Oct. 2016.

[8] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power in a multiplier architecture," *J. Low Power Electron.*, vol. 7, no. 4, pp. 490–501, 2011.

[9] C.-H. Lin and C. Lin, "High accuracy approximate multiplier with error correction," in *Proc. IEEE 31st Int. Conf. Comput. Design*, Sep. 2013, pp. 33–38.

[10] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Proc. Conf. Exhibit. (DATE)*, 2014, pp. 1–4.

[11] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and analysis of circuits for approximate computing," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Oct. 2011, pp. 667–673.

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

[13] S. Suman *et al.*, "Image enhancement using geometric mean filter and gamma correction for WCE iamges," in *Proc. 21st Int. Conf., Neural Inf. Process. (ICONIP)*, 2014, pp. 276–283.