



Performance Evaluation of Wordlength Reduction Based Area and Power Efficient Approximate Multiplier for Mobile Multimedia Applications

R. Ramya¹ · S. Moorthi¹

Received: 9 August 2018 / Revised: 7 May 2019 / Accepted: 8 May 2019 / Published online: 15 May 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

Hardware multiplier circuits decide the speed and power consumption in the execution of digital signal processing algorithms. The desirable feature of reduced area and power consumption for battery-driven multimedia gadgets can be realized by replacing the power hungry multiplier circuits with approximate multiplier circuits. The approximation techniques reduce the complexity of the design and improve the energy efficiency of the circuit. This paper proposes an area and power efficient approximate unsigned integer multiplier architecture based on wordlength reduction. It is designed to meet a pre-specified error performance with improved area and power reduction compared with similar designs. It is extended further for the signed multiplier architecture. The circuit characteristics are analyzed to establish the suitability of the proposed design for low-power applications. Synthesis results show that the proposed unsigned multiplier consumes 65% less power than the exact Wallace multiplier. The area requirement of the proposed multiplier reduces by 50% compared to an exact multiplier. The multiplier is tested for image filtering to establish the efficacy of the design in multimedia applications.

Keywords Approximate multiplier · Multimedia · Wordlength · PSNR · SSIM

1 Introduction

Approximate computing represents a paradigm shift in low-power digital system design. The renewed interest is because of the fast development of VLSI systems for

✉ S. Moorthi
srimoorthi@nitt.edu

R. Ramya
407114003@nitt.edu

¹ VLSI Systems Research Laboratory, Department of Electrical and Electronics Engineering, National Institute of Technology, Tiruchirappalli, Tamil Nadu 620 015, India

communication and digital signal processing. The growing performance gap among application complexity, VLSI technology and battery technology is the constraint in delivering satisfactory performance at low power and at high speed. The all-pervasive connectivity to the Internet through wireless communication technologies saw a revolution in the use of mobile multimedia gadgets for communication and entertainment. It necessitated the extended time of operation of these devices from battery power that changed the design parameters of VLSI circuits from maximum speed to low power. Low-power systems become an area of active research in all stocks of computing and approximate computing emerged as a promising approach for the design of low-power high-speed multimedia digital communication systems. The present research in approximate computing covers almost all areas of computing like operating systems, compilers, networks, architectures, microelectronic circuits and components [4, 14]. Approximate computing becomes a necessity because of a large and growing class of error-resilient applications in multimedia signal processing, graphics, data mining and wireless communication [19]. A common feature of error-resilient applications is that an exact result often not necessary, but an approximate result is sufficient. One of the studies conducted in this direction reported that about 83% of the runtime of error-tolerant applications is spent on computations that can be approximated [2].

Hardware-based approximate computing permits the use of hardware architectures and circuits that do not meet the design specifications exactly. The resulting approximation errors may be due to the errors induced either due to timing errors or due to the functional approximations made in the implementation of hardware building blocks. Accordingly, the hardware approximation techniques for power reduction can be classified into two groups. The first group comprises supply bias voltage scaling techniques, and the second group represents the circuit complexity reduction techniques. Voltage scaling is considered the best way to reduce energy consumption, and it is still an active area of research [10, 17]. Voltage over-scaling methods try to reduce the bias voltage in CMOS circuits in such a way that it does not cause timing errors. CMOS circuits fabricated using fabrication technologies of <60 nm show reliability issues and unstable functionality under low-voltage operation. Since the speeding up and the energy reduction in VLSI circuits fast approaching its physical limits [7], the voltage over scaling may find limited applications in the future. Thus, there is renewed interest in research in the second class of approximate computing methods that deals with the hardware complexity reduction. Here, approximate architectures and circuits replace the exact at the implementation stage compromising the functional equivalence between specification and implementation.

In multimedia processing, the important signals are audio and video signals. Considering the logarithmic response characteristics of human auditory and visual sensory organs, it is possible to apply approximate techniques to process large dynamic range signals like images and music. These approximations may result in small or no detectable degradation in the quality of image or music as far as human processing is considered. The important arithmetic operations in multimedia signal processing are addition, subtraction, multiplication, and multiply and accumulate. The multiplication operation is the hardest among them and is the core operation in many of the signal processing algorithms, including convolution, squaring, filtering and fast Fourier transform (FFT). By sacrificing some accuracy, the error-tolerant multiplier

circuits can attain large reduction in both the power consumption and area in multimedia systems, and they may generate satisfactory results rather than accurate results. Thus, to achieve improved performance in terms of area and power consumption in multimedia signal processing, it is essential to adopt approximation techniques in the design of hardware multipliers.

In this paper, an area and power efficient approximate multiplier architecture suitable for mobile multimedia applications is proposed. The area and power reduction are achieved by reducing the wordlength of the input operands. The N -bit input operand is reduced to an $N/2$ -bit operand by right-shifting a block of continuous $N/2$ -bits beginning with highest priority '1' bit present in the higher-order $N/2$ bits to the lower-order $N/2$ bits. This enables the use of an $N/2$ -bit multiplier in place of an N -bit multiplier with a little overhead of additional hardware required for wordlength reduction of input operands. The reduced area and power consumption thus achieved in the design of the approximate multiplier making it very attractive for application-specific integrated circuit (ASIC) implementation.

The paper is organized as follows: The related research in approximate multiplier design is reviewed in Sect. 2. The description of the proposed multiplier architecture and its error characteristics are presented in Sect. 3. Section 4 deals with the hardware implementation of the proposed approximate multiplier, and circuit characteristics are analyzed. Section 5 deals with image filtering applications implemented using the proposed multiplier unit, and performance is analyzed using quantitative assessment of image quality. Finally, the conclusions are presented in Sect. 6.

2 Related Works

Several approximate multiplier designs were proposed in the studies [1, 9, 12]. A comparative performance analysis of approximate multipliers was elaborated in [8]. Due to the renewed interest in this area of research, an increased number of publications are reported recently. Some of the major contributions in approximate multiplier design are discussed here. Parag Kulkarni et al. [11] proposed a 2×2 underdesigned approximate multiplier block and used this block to construct large inaccurate multipliers. The inaccurate multipliers have shown to achieve an appreciable power savings over an accurate multiplier. Reza Zendegani et al. [21] presented one unsigned and two signed rounding-based approximate multiplier (RoBA) architectures based on rounding of the inputs in the form of 2^n . The RoBA architecture is a multiplier-less approach, and the multiplication operation can be performed using a rounding block, 3 shifters and 2 addition/subtraction operations. Venkatachalam and Ko [18] proposed two variants of approximate multipliers. The partial products of the multiplier are altered using generate and propagate signals, and the accumulation of generate signals is done column-wise. Approximate 4:2 compressors and adders are used to accumulate the remaining partial products. In the first variant, approximation is applied to all the columns of the partial product, and in variant 2, approximation is not applied to most significant column of partial products. Even though multiplier design2 has lower relative error compared to multiplier design1, it offers less area and power savings as compared to multiplier design1. Momeni et al. [15] presented two approximate 4:2

compressors and used these compressors in developing four approximate multiplier designs. The error and circuit-level performance are analyzed for a regular Dadda multiplier. Narayanamoorthy et al. [16] proposed dynamic segment method (DSM), static segment method (SSM), enhanced static segment method (ESSM)-based approximation multipliers for various DSP and classification applications. A scalable dynamic range unbiased multiplier (DRUM) for approximate applications such as image filtering, JPEG compression and perceptron classifier is proposed in [5]. Following a similar design approach as in DSM, the DRUM multiplier design makes use of two N -bit leading one detectors (LODs) to locate the presence of most significant '1' in both the inputs. The location of the most significant '1' is used to select a fixed-length block of consecutive bits beginning with the leading one, and the approximate values of the remaining lower bits are set to its expectation value. This makes the error distribution unbiased, and hence, it gives better error performance compared to DSM. The hardware of DRUM multiplier includes expensive N -bit LODs, N -bit encoders, N -bit MUX, barrel shifter and an exact multiplier. The presence of N -bit LODs and N -bit encoder increases the complexity of the circuit and the power performance of the circuit.

It is shown that the DSM has sufficient accuracy for image processing applications [16]; we propose an approximate multiplier circuit that has the same accuracy as that of DSM and better circuit characteristic than both DSM and DRUM. The proposed approximate multiplier architecture is modeled using Verilog HDL. The circuit characterization performed using evaluation of area, power and delay performance of the circuit. The error characterization is performed using standard error characterization techniques. The error and circuit characteristic of the proposed design is compared with other recent state-of-the-art designs to demonstrate its performance.

3 Proposed Approximate Multiplier

The approximate multiplier architecture can be represented by three sub-blocks, where the first sub-block is the wordlength reduction logic which reduces the input operand wordlength. The second sub-block is an arithmetic unit which is an exact multiplier block of $N/2$ -bit wordlength, and the last sub-block is a correction logic block to compensate for the reduction of input operand wordlength.

In the proposed approximate multiplier, the wordlength of the N -bit input operands is reduced to $N/2$ bits and multiplication is done with a single $N/2$ -bit multiplier instead of N -bit multiplier. The block diagram of unsigned approximate multiplier is shown in Fig. 1.

The wordlength reduction logic is composed of shift derive (SD) logic and an N -bit right-barrel shifter. The SD logic is comprised of the following blocks: $N/2$ -bit priority encoder, a binary-to-excess one converter, and a 2:1 MUX. The $N/2$ -bit priority encoder receives the higher-order $N/2$ bits as the inputs. The index of the active bit of highest priority present in the higher-order $N/2$ -bit inputs is encoded by the $N/2$ -bit priority encoder. Figure 2a depicts the truth table and Boolean expression of a 4:2 priority encoder for an 8-bit multiplier. Similarly, the truth table and Boolean expression of an 8:3 priority encoder for a 16-bit multiplier are correspondingly given in Fig. 2b. In order to obtain the amount of shift needed to obtain the truncated input

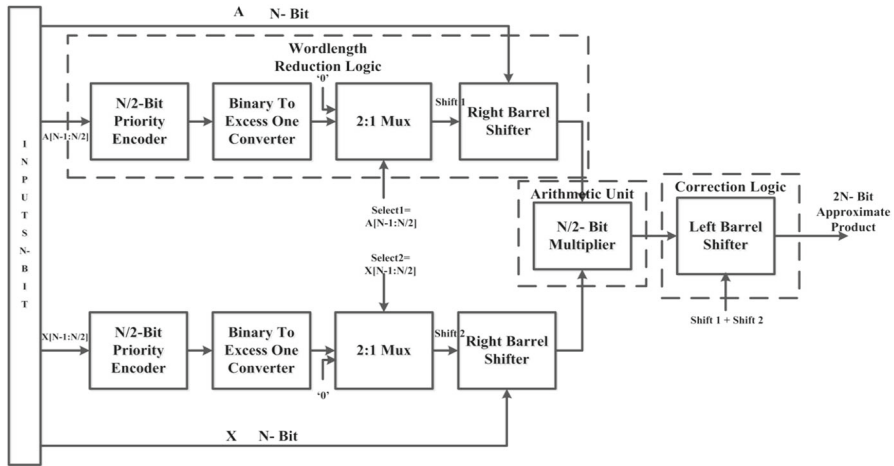


Fig. 1 Block diagram of unsigned approximate multiplier

operands, a binary-to-excess one converter and a 2:1 multiplexer (MUX) are used. The select lines needed for the 2:1 MUX are obtained by performing the bit-wise OR operation of the higher-order $N/2$ bits of the input operand. In case of input operands with wordlength less than $N/2$ bits, the select value will be set as '0' and the 2:1 MUX selects the input as '0' and the right-barrel shifter will not provide any shift and lower-order $N/2$ bits of the input operands are read into the input registers of the multiplier. For operand size greater than $N/2$ bits, then select value will be set as '1' and the 2:1 MUX will select the required shift amount from the binary-to-excess one converter. The $N/2$ successive bits starting with the highest priority '1' bit are right-shifted to $N/2$ -bit LSB registers and are read into the multiplier input registers. The multiplier unit performs $N/2$ -bit multiplication, and the N -bit product is fed to the correction logic. The correction logic is made up of $2N$ -bit left-barrel shifter that expands the approximate N -bit product to a $2N$ -bit product by left shifting. The left-barrel shifter left-shifts the approximate product by an amount equal to the number of bit positions that is the sum of right shifts applied to both the input operands, and the final $2N$ -bit approximate product is obtained at the output.

The above design can be extended for signed numbers by inserting a two's complement block at the input of each branch. At the output, the product value may be negated if necessary. For further simplifying the circuit complexity, we propose modified shift derive (MSD) logic with the maximum negative input of magnitude -2^N left out from the computation, and the resulting simplified architecture of proposed signed approximate multiplier is shown in Fig. 3.

The signed multiplier comprises a sign detect and two's complement block at the input, sign-set block at the output, MSD logic block, right- and left-barrel shifters, unsigned exact multiplier block of $N/2$ -bit wordlength. The multiplier receives the inputs in signed two's complement format. The sign of the input operand is determined, and if the sign bit is set, the two's complement block determines the absolute value

TRUTH TABLE

Inputs				Outputs	
D ₃	D ₂	D ₁	D ₀	Q ₁	Q ₀
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

X: Don't care

$$Q_0 = \overline{D_2} \cdot D_1 + D_3$$

$$Q_1 = D_2 + D_3$$

(a)

Inputs								Outputs		
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	Q ₂	Q ₁	Q ₀
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	1	0	0
0	0	1	X	X	X	X	X	1	0	1
0	1	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1

$$Q_0 = \overline{D_6} \cdot (\overline{D_4} \cdot \overline{D_2} \cdot D_1 + \overline{D_4} \cdot D_3 + D_5) + D_7$$

$$Q_1 = \overline{D_5} \cdot \overline{D_4} \cdot (D_2 + D_3) + D_6 + D_7 ; Q_2 = D_4 + D_5 + D_6 + D_7$$

(b)

Fig. 2 Truth table and Boolean expression of **a** 4:2 priority encoder and **b** 8:3 priority encoder

of the negative operand. MSD logic comprises a modified priority encoder named as sign-extension encoder and an inverter-based control logic block. The sign-extension encoder is used to calculate the effective size of the positive operands. The effective size of the positive operands is computed by finding the number of sign-extension bits (zeros) immediately to the right of the most significant bit. For an N -bit multiplier, $N/2$ to $\log_2(N/2)$ sign-extension encoder is required since the multiplier block is of $N/2$ -bit wordlength. The truth table and Boolean expression of a 4:2 and 8:3 sign-extension encoders for an 8-bit and 16-bit multiplier are shown in Fig. 4a, b. The count of the sign-extension bits is passed to the control logic block, which is used to calculate the amount of shift needed to limit the wordlength of the inputs. The control logic derives the shift amount by concatenating a '0' with the inverted output of the sign-extension encoder. If the size of the input operand is below $N/2$ bits, then

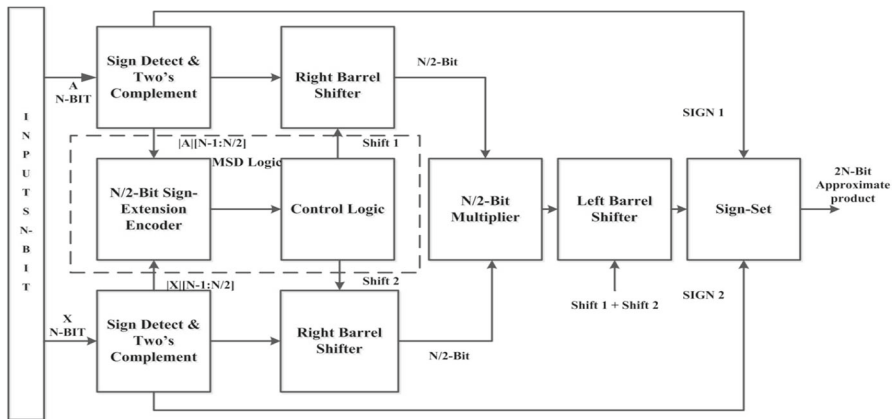


Fig. 3 Block diagram of signed approximate multiplier

shift operation need not be performed. In case of operand wordlength above $N/2$ bits, the right-barrel shifter right-shifts the input operands to the required number of places as decided by the control logic to obtain the $N/2$ -bit input operand. Now the effective size of the input operands is limited to wordlength of size $N/2$ -bit instead of N -bits, and the multiplication is performed as an $N/2 \times N/2$ -bit multiplication. Unsigned exact Wallace tree multiplier is adopted for the $N/2$ -bit multiplier block. In order to compensate for the $N/2$ -bit truncated multiplication, the correction logic block is used. The obtained N -bit product is left-shifted appropriately to compensate for the reduction in the wordlength of the input operand. Depending on the sign of the input operands, the unsigned product is negated in the sign-set block to obtain the final signed approximate product as the output.

The error metrics such as error rate, normalized mean error distance (NMED), absolute value of mean relative error distance (MRED) and percentage mean accuracy are used to evaluate the error characterization of approximate multipliers [8]. For adders and multipliers, error distance (ED) is the absolute difference between the approximated result (M') and the actual result (M). Relative error distance (RED) is defined as ED/M . Mean error distance (MED) is the average value of all possible EDs. Normalized mean error distance (NMED) is the normalization of mean error distance (MED) by the maximum output of the accurate multiplier. Mean relative error distance (MRED) is the average value of all possible REDs. Mean accuracy is defined as $100 - \text{MRED}$. Error rate (ER) is defined as the probability of producing an incorrect result.

In order to evaluate the error performance of the proposed multipliers, two sets of hundred thousand random numbers are generated with uniform probability and the multiplication is performed. The error metrics for a few 16-bit approximate multipliers were simulated and are compared with our design in Tables 1 and 2.

The results tabulated in Tables 1, 2 show that the current design of the approximate multiplier provides highest accuracy in terms of various error metrics. The proposed unsigned multiplier achieves the same error performance as that of the DSM8. This is due to the same functionality of the wordlength reduction logic even though the hard-

TRUTH TABLE					
Inputs				Outputs	
D ₃	D ₂	D ₁	D ₀	Q ₁	Q ₀
0	0	0	0	1	1
0	0	0	1	1	0
0	0	1	X	0	1
0	1	X	X	0	0

X: Don't care

$$Q_1 = \overline{D_3} \cdot \overline{D_2} \cdot \overline{D_1}$$

$$Q_0 = \overline{D_3} \cdot \overline{D_2} \cdot \overline{D_0} + \overline{D_3} \cdot \overline{D_2} \cdot D_1$$

(a)

Inputs								Outputs		
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	Q ₂	Q ₁	Q ₀
0	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	1	X	1	0	1
0	0	0	0	0	1	X	X	1	0	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	0	1	0
0	0	1	X	X	X	X	X	0	0	1
0	1	X	X	X	X	X	X	0	0	0

$$Q_0 = \overline{D_7} \cdot \overline{D_6} \cdot D_5 + \overline{D_7} \cdot \overline{D_6} \cdot \overline{D_4} \cdot D_3 + \overline{D_7} \cdot \overline{D_6} \cdot \overline{D_4} \cdot \overline{D_2} \cdot \overline{D_0} + \overline{D_4} \cdot \overline{D_2} \cdot D_1$$

$$Q_1 = \overline{D_7} \cdot \overline{D_6} \cdot \overline{D_5} \cdot (D_3 + D_4 + \overline{D_2} \cdot \overline{D_1})$$

$$Q_2 = \overline{D_7} \cdot \overline{D_6} \cdot \overline{D_5} \cdot \overline{D_4} \cdot \overline{D_3}$$

(b)

Fig. 4 Truth table and Boolean expression of **a** 4:2 sign-extension encoder and **b** 8:3 sign-extension encoder

were implementation differs in them. Due to the unbiased nature of error distribution of the DRUM8 design, it shows a better error performance than the proposed multiplier. Comparing with the unsigned RoBA design, the error performance of DSM8 is shown to be better [21]. The proposed design also shows better error performance comparing with RoBA designs. The other two designs show poor error performance compared to the proposed design. The relative error is also calculated for inputs of various wordlengths and is tabulated in Table 3.

Table 3 gives the percentage of outputs with the relative error less than the specified percentage for various wordlengths such as 8-bit, 16-bit, and 32-bit proposed approximate designs. It must also be noted that the relative error reduces as wordlength of the operand increases. For a 16-bit design, the relative error is <2% and that for a 32-bit

Table 1 Arithmetic accuracy comparison of proposed 16-bit unsigned multiplier with state-of-the-art designs

Unsigned designs	Error rate (%)	NMED (%)	MRED (%)	Mean accuracy (%)
Proposed unsigned multiplier	99.95	0.13	0.53	99.47
Kulkarni [11]	80.85	1.37	3.33	96.67
U-RoBA [21]	99.96	0.69	2.93	97.07
Venkatachalam Design1 [18]	99.80	1.78	7.63	92.37
DSM8 [16]	99.95	0.13	0.53	99.47
DRUM8 [5]	99.98	0.09	0.36	99.64

Table 2 Arithmetic accuracy comparison of proposed 16-bit signed multiplier with state-of-the-art designs

Signed designs	Error rate (%)	NMED (%)	MRED (%)	Mean accuracy (%)
Proposed signed	99.87	0.032	0.52	99.48
S- RoBA [21]	99.90	0.172	2.88	97.12
AS-RoBA [21]	99.94	0.173	2.89	97.11

Table 3 Variation of relative error (RE) of outputs in percentage for various wordlength for the proposed approximate multiplier

Type of multiplier	Bit width	RE<0.5%	RE<1%	RE<2%	RE<5%	RE<10%	RE<20%
Unsigned	8-bit	3.79%	5.4%	9.6%	31.7%	79.7%	100%
	16-bit	47%	97%	100%			
	32-bit	100%					
Signed	8-bit	9.3%	10.7%	16.1%	41.7%	85.6%	100%
	16-bit	48.6%	97.2%	100%			
	32-bit	100%					

multiplier, it is $<0.5\%$. The difference between the exact and approximate multipliers almost vanishes for 32-bit signal processing applications, and it is negligible for 16-bit applications.

4 Hardware Implementation of the Proposed Multiplier

The proposed approximate multiplier architectures are modeled using Verilog HDL and synthesized in Cadence RTL Compiler using generic Process Design Kit (gPDK) 90-nm CMOS technology with typical library settings. The functionality of the proposed multipliers is verified using Cadence NCSIM, and all the designs are synthesized with proper timing constraints. The post-synthesis circuit performance characteristics such as power, area and critical-path delay are measured with respect to the maximum achievable frequency of the exact multiplier as reference, and are tabulated

Table 4 Post-synthesis performance characteristics of various 8-bit multipliers

	Power (μW)	Delay (ns)	PDP (pJ)	EDP (pJ ns)	Area (μm^2)	PDA (pJ μm^2)
Unsigned designs						
Proposed unsigned	161.05	2.559	0.412	1.054	1147	473
Kulkarni [11]	293.95	2.527	0.742	1.875	1361	1010
U-RoBA [21]	166.04	2.963	0.492	1.457	1902	936
Venkatachalam Design1 [18]	190.42	2.658	0.506	1.345	1244	629
DSM4 [16]	251.45	2.714	0.682	1.851	1191	812
DRUM4 [5]	210.20	2.629	0.552	1.451	1179	651
Exact unsigned (Wallace)	313.27	2.802	0.877	2.457	1672	1466
Signed designs						
Proposed signed	225.94	3.325	0.751	2.497	1483	1113
S-RoBA [21]	264.65	3.442	0.912	3.139	2427	2213
AS-RoBA [21]	254.04	3.331	0.846	2.818	2215	1874
Exact signed (Baugh-Wooley)	289.96	2.811	0.815	2.291	1651	1345

in Tables 4, 5 for various approximate multiplier designs. The circuit characteristics of the proposed unsigned and signed approximate multiplier implementations were compared against exact multiplier architectures like Wallace tree (exact unsigned), and Baugh-Wooley multiplier (exact signed) architectures. For comparison, a few of the state-of-the-art approximate multiplier architectures are considered and implemented using gPDK 90-nm CMOS technology with the same timing constraints. Since leakage currents are also contributing to the total power, better parameters for characterizing circuit performance are energy or power–delay product (PDP), energy–delay product (EDP), and power–delay–area (PDA) [8, 21]. Hence, the compound metrics such as PDP, EDP, and PDA were also computed and are tabulated in Tables 4, 5 for performance comparison. Table 4 gives the circuit characteristics of various 8-bit multipliers.

The synthesis results presented in Table 4 show that the proposed approximate multiplier achieves minimum power–delay product and power–delay–area in comparison with exact as well as other approximate designs. The area requirement of the proposed approximate unsigned (signed) multiplier is 31% (10%) lower than that of the exact Wallace (Baugh-Wooley) multiplier. The proposed 8-bit unsigned (signed) multiplier consumes 48% (22%) less power than the total power consumed by the respective exact multiplier. The energy or power–delay product (PDP) and PDA of the proposed unsigned (signed) approximate multiplier are about 53% (8%) and 67% (17%), respectively, lower than those of the respective exact multiplier. Table 5 gives a comparison of various 16-bit multipliers in terms of power, delay, area, PDP, EDP and PDA.

The synthesis result tabulated in Table 5 shows that the proposed multiplier architecture is capable of delivering low power and good speed performance compared to the existing state-of-the-art multipliers. The area requirement of the proposed approximate unsigned (signed) multiplier is 50% (40%) lower than that of the exact Wallace (Baugh-

Table 5 Post-synthesis performance characteristics of various 16-bit multipliers

	Power (μ W)	Delay (ns)	PDP (pJ)	EDP (pJ ns)	Area (μm^2)	PDA (pJ μm^2)
Unsigned designs						
Proposed unsigned	303.47	3.97	1.21	4.803	3533	4275
Kulkarni [11]	760.49	4.05	3.08	12.474	6241	19,222
U-RoBA [21]	235.97	4.84	1.14	5.518	4522	5155
Venkatachalam Design1 [18]	425.75	4.38	1.86	8.147	4527	8420
DSM8 [16]	402.71	4.44	1.78	7.903	3548	6315
DRUM8 [5]	424.83	4.64	1.96	9.094	3806	7460
Exact unsigned (Wallace)	871.72	4.08	3.56	14.524	7012	24,963
Signed designs						
Proposed signed	414.02	5.34	2.21	11.801	4012	8867
S-RoBA [21]	541.17	5.24	2.83	14.829	5640	15,961
AS-RoBA [21]	537.40	5.15	2.76	14.214	5210	14,380
Exact signed (Baugh-Wooley)	769.07	4.56	3.51	16.005	6679	23,443

Wooley) multiplier. The proposed 16-bit unsigned (signed) multiplier consumes 65% (46%) less power than the total power consumed by the respective exact multiplier. The above results show the efficiency of the multiplier in terms of area and power compared to standard designs. The energy or power–delay product (PDP), EDP, and PDA of the proposed unsigned (signed) approximate multiplier are about 66% (37%), 67% (26%), and 82% (62%), respectively, lower than those of the respective exact multiplier.

It is evident from the error characteristics and circuit characteristics that the proposed approximate multiplier circuit achieves the same accuracy as that of DSM and better circuit characteristic than both DSM and DRUM. The area and power savings achieved in comparison with the exact multiplier for various 16-bit approximate multipliers are plotted in Fig. 5. The results show that the proposed design has better circuit characteristics compared to other designs except U-RoBA design which has better power saving in the case of unsigned multiplier.

The performance comparison of the various 16-bit approximate multipliers in terms of PDP and MRED (%) is plotted in Fig. 6. The error characteristic is represented by MRED, and circuit characteristics are represented by PDP and area. The size of the disk represents the area.

The performance comparison result shows the advantage of the proposed multiplier design over other approximate designs. The lower left corner is the preferred values for better performance. Considering all the circuit parameters such as power, area, and delay (PDA) together with MRED may provide an overall comparison of various performance parameters considered in the design. Area is utilized as a third parameter to get more visibility of performance of different multipliers. The performance comparison in terms of PDA and MRED (%) is given in Fig. 7.

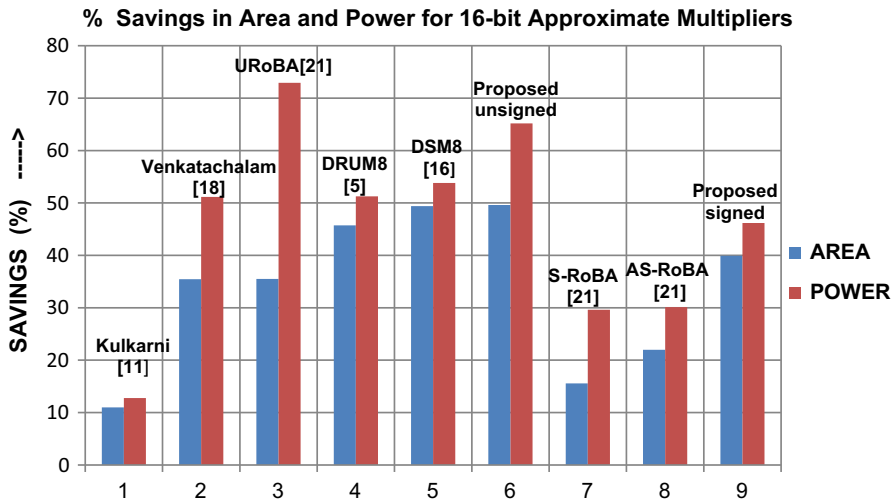


Fig. 5 Area and power savings (%) for various 16-bit approximate multipliers

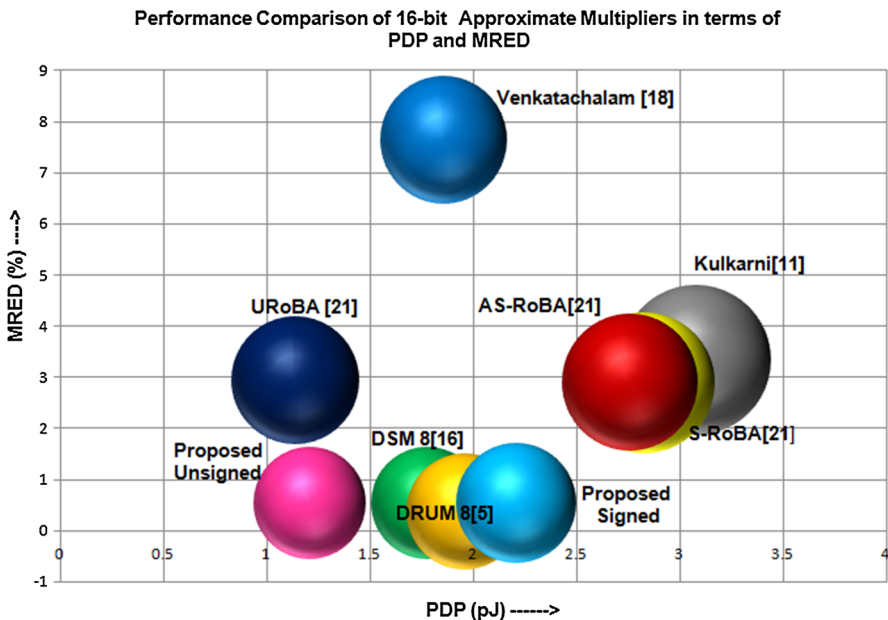


Fig. 6 Performance comparison of various 16-bit approximate multipliers based on PDP and MRED

The results presented in Figs. 6, 7 indicate that the proposed unsigned and signed multiplier architectures are able to achieve better performance in improving the circuit characteristics along with good accuracy compared to other approximate multiplier designs.

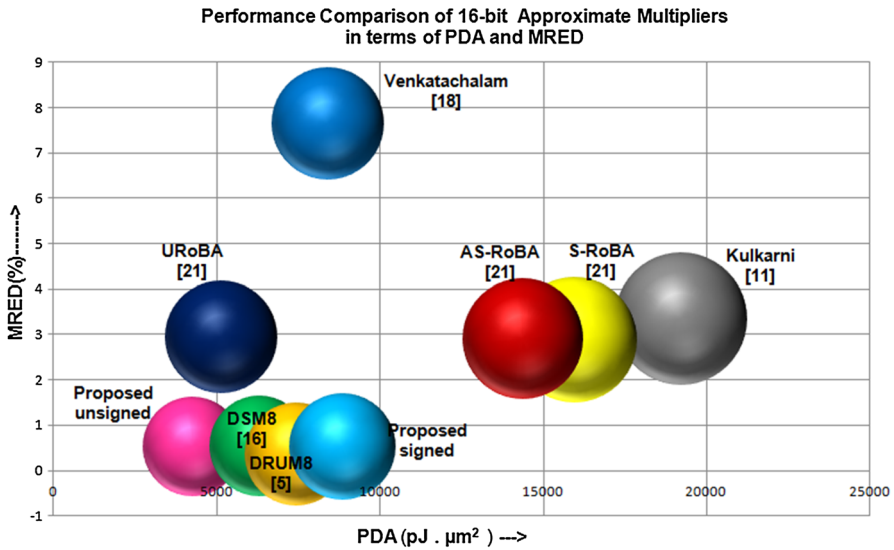


Fig. 7 Performance comparison of various 16-bit approximate multipliers based on PDA and MRED

5 Image Processing Application

5.1 Low-Pass Image Filtering

For demonstrating the efficacy of the proposed design, images were filtered with a low-pass filter (LPF) in order to smoothen a given image. The approximate multipliers are used to implement the multiplication in the convolution sum and the division is accurate.

The low-pass image filtering is defined by the equation:

$$f_{LP}(x, y) = \frac{1}{4368} \sum_{m=-2}^2 \sum_{n=-2}^2 g(m+3, n+3) f(x-m, y-n) \quad (1)$$

where g is a 5×5 low-pass kernel [8] given as:

$$\begin{bmatrix} 16 & 64 & 112 & 64 & 16 \\ 64 & 256 & 416 & 256 & 64 \\ 112 & 416 & 656 & 416 & 112 \\ 64 & 256 & 416 & 256 & 64 \\ 16 & 64 & 112 & 64 & 16 \end{bmatrix}$$

Peak-signal-to-noise ratio (PSNR) and Structural SIMilarity (SSIM) index [6, 20] were calculated to have a quantitative assessment of image quality of the filtered image. The computed values of PSNR and SSIM are tabulated for few 16-bit images in Table 6. It should be noted that the PSNR and SSIM values are calculated by comparing the smoothened image generated by the exact multiplier and the smoothened image

Table 6 Comparison of image quality for LPF and HPF images. *Courtesy: Image compression.info/test images)*

Filter type Image name/resolution	LPF		HPF	
	PSNR (dB)	SSIM	PSNR (dB)	SSIM
Big_tree (4550 × 6088)	64.1607	0.9999	60.6336	0.9995
Big_building (5412 × 7216)	56.3075	0.9999	52.9079	0.9990
Bridge (4049 × 2479)	62.4376	1.0000	58.7950	0.9998
Cathedral (3008 × 2000)	59.6079	0.9984	56.9313	0.9971
Deer (2641 × 4043)	61.8761	0.9996	59.1464	0.9993
Spider_web (2848 × 4256)	57.2609	0.9981	53.9152	0.9765
Zone_plate (2000 × 2000)	57.3866	1.0000	52.4632	0.9999
Nightshot_iso_100 (2352 × 3136)	66.8873	0.9989	63.2094	0.9958
Nightshot_iso_1600 (2352 × 3136)	68.6954	1.0000	64.8583	0.9999
Leaves_iso_200 (2000 × 3008)	59.0983	0.9996	55.8468	0.9989
Leaves_iso_1600 (2000 × 3008)	58.8377	0.9998	55.6196	0.9993
Average	61.141	0.9994	57.661	0.9968

generated using the proposed approximate multiplier. PSNR approximates human perception of reconstruction quality. Although a higher PSNR generally indicates that the reconstruction is of higher quality, it may not be true in all cases. SSIM may be used as an additional quantitative quality assessment parameter to confirm the quality of an image since it measures the perceptual difference between two similar images. The value of SSIM very close to unity indicates that the quality of the compared images is same.

5.2 High-Pass Image Filtering

For further demonstration of the efficacy of the proposed design, images were filtered with high-pass filters (HPFs) in order to sharpen the given image. The high-pass image filtering is defined by the equation:

$$f_{HP}(x, y) = \frac{1}{256} \sum_{m=-2}^2 \sum_{n=-2}^2 g(m+3, n+3) f(x-m, y-n) \quad (2)$$

The 5×5 high-pass kernel g used for image sharpening is [21]

$$\begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -4 & -16 & -24 & -16 & -4 \\ -6 & -24 & 476 & -24 & -6 \\ -4 & -16 & -24 & -16 & -4 \\ -1 & -4 & -6 & -4 & -1 \end{bmatrix}.$$





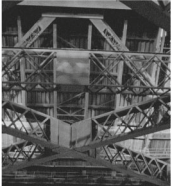







Image	Original Image	Low Pass Filtered Image	High Pass Filtered Image
Big_building			
			
			
			

Fig. 8 Illustrative examples of low-pass- and high-pass-filtered images; a selected part of the actual image is displayed for demonstration

The computed values of PSNR and SSIM are tabulated for few 16-bit images in Table 6. Figure 8 illustrates the low-pass- and high-pass-filtered images obtained using proposed approximate multiplier.

The high PSNR and SSIM values show that the degradation of image quality and the loss of structural information are negligible. This indicates that the proposed approximate multiplier induces negligible degradation of image quality compared to an exact multiplier.

5.3 High-Boost Filtering

Image quality assessment is done also for 8-bit and 16-bit high-boost-filtered images. The high-boost filtering is defined by the equation [3]:

$$g(x, y) = f(x, y) + k * g_{\text{mask}}(x, y) \quad (3)$$

where

$$g_{\text{mask}}(x, y) = f(x, y) - f_{LP}(x, y) \quad (4)$$

Thus, the image $f(x, y)$ is modified to image $g(x, y)$ by adding a scaled high-pass-filtered version of $f(x, y)$. If the constant $k = 1$, then substituting (4) in (3), we get

$$g(x, y) = f(x, y) - f_{LP}(x, y) \quad (5)$$

where $f_{LP}(x, y) = \frac{1}{273} \sum_{m=-2}^2 \sum_{n=-2}^2 g(m+3, n+3) f(x-m, y-n)$.

A 5×5 Gaussian low-pass kernel g [13] is used for filtering, and the kernel g is selected as:

$$\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

The approximate multipliers implement the multiplication in the low-pass filtering part, while the division and subtraction are accurate. PSNR and SSIM were calculated to have a quantitative assessment of image quality of the filtered image. The computed values of PSNR and SSIM are tabulated for few standard 8-bit and 16-bit images in Table 7.

Simulation results show that the structural similarity of images filtered with exact and approximate methods using 16-bit multiplier is almost equal to unity, and the high PSNR value of the 16-bit images testifies the fact that the error due to wordlength reduction of the input operands is negligible. The large PSNR values of the 16-bit multipliers also indicate that the error tolerance of large wordlength multipliers is much better than smaller wordlength multipliers. The structural similarity indices calculated also show that as word size increases, the SSIM increases and in turn the mean square error decreases.

6 Conclusions

In this paper, an area and power efficient approximate multiplier architecture based on data wordlength reduction suitable for multimedia applications has been proposed.

Table 7 Comparison of image quality for 8-bit and 16-bit images

Image resolution IMAGE NAME	16-bit		8-bit	
	PSNR (dB)	SSIM	PSNR (dB)	SSIM
Big_tree	63.7949	1.0000	37.8900	0.9427
Big_building	56.1738	1.0000	29.5744	0.8576
Bridge	61.9097	1.0000	36.0095	0.9600
Cathedral	59.4200	0.9996	34.0250	0.9097
Deer	61.6191	0.9999	36.3034	0.9564
Spider_web	57.2142	0.9996	31.9054	0.3945
Zone_plate	52.5099	1.0000	28.7090	0.9928
Nightshot_iso_100	65.8943	0.9999	41.0576	0.7281
Nightshot_iso_1600	67.4638	1.0000	42.3350	0.9867
Leaves_iso_200	58.3139	0.9999	32.8748	0.9148
Leaves_iso_1600	58.0538	0.9999	32.5468	0.9419
Average	60.215	0.9998	34.839	0.8713

The proposed multiplier architecture offers significant advantages in terms of error performance and circuit performance essential for mobile multimedia applications. The error performance demonstrated by the architectures is good as exemplified by the high values of percentage mean accuracy. The mean error decreases and accuracy increases as the wordlength increases. The suitability of the circuits for low-power multimedia processing is demonstrated by computing the area, power, delay, and the energy of the proposed multiplier. The results show that the proposed architecture is well suited for low-power applications, especially when the data paths are wide. The efficiency of the proposed approximate multiplier was evaluated in image processing applications such as smoothing, sharpening and high-boost filtering. The assessment of image quality also shows that the degradation in the image quality or degradation of the structural information is very minimal or almost nil for large wordlength multipliers. The good accuracy of computation achieved with reduced area and power by the proposed approximate multiplier shows its potential for low-power applications to process massive multimedia signals. A prospect of the proposed approximate multiplier is in error-tolerant floating point multiplication architectures which are based on large unsigned multipliers.

References

1. K. Bhardwaj, P.S. Mane, ACMA: accuracy-configurable multiplier architecture for error-resilient system-on-chip, in *2013 8th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC) (2013)*, pp. 1–6. <https://doi.org/10.1109/ReCoSoC.2013.6581532>
2. V.K. Chippa, S.T. Chakradhar, K. Roy, A. Raghunathan, Analysis and characterization of inherent application resilience for approximate computing, in *Proceedings of the 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC) (2013)*, pp. 1–9. <https://doi.org/10.1145/2463209.2488873>
3. R.C. Gonzalez, R.E. Woods, *Digital Image Processing*, 3rd edn. (Prentice Hall, New Jersey, 2008)
4. J. Han, M. Orshansky, Approximate computing: an emerging paradigm for energy-efficient design, in *Proceedings of the 2013 18th IEEE Test Symposium (ETS) (2013)*, pp. 1–6. <https://doi.org/10.1109/ETS.2013.6569370>

5. 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*, (2015), pp. 418–425. <https://doi.org/10.1109/ICCAD.2015.7372600>
6. A. Hore, D. Ziou, Image quality metrics: PSNR vs. SSIM, in *2010 20th International Conference on Pattern Recognition* (2010), pp. 2366–2369. <https://doi.org/10.1109/icpr.2010.579>
7. K. Itoh, M. Yamaoka, T. Oshima, Adaptive circuits for the 0.5-V nanoscale CMOS era. *IEICE Trans. Electron.* **93**(3), 216–233 (2010). <https://doi.org/10.1587/transele.E93.C.216>
8. H. Jiang, C. Liu, L. Liu, F. Lombardi, J. Han, A review, classification, and comparative evaluation of approximate arithmetic circuits. *ACM J. Emerg. Technol. Comput. Syst. (JETC)* **13**(4), 60:1–60:34 (2017). <https://doi.org/10.1145/3094124>
9. H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, J. Han, A comparative evaluation of approximate multipliers in *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)* (2016), pp. 191–196. <https://doi.org/10.1145/2950067.2950068>
10. G. Karakostas, D. Mohapatra, K. Roy, System level DSP synthesis using voltage overscaling, unequal error protection & adaptive quality tuning, in *2009 IEEE Workshop on Signal Processing Systems* (2009), pp. 133–138. <https://doi.org/10.1109/sips.2009.5336238>
11. P. Kulkarni, P. Gupta, M. Ercegovac, Trading accuracy for power with an underdesigned multiplier architecture, in *Proceedings of the 24th International Conference on VLSI Design (2011)*, pp. 346–351. <https://doi.org/10.1109/vlsid.2011.51>
12. K.Y. Kyaw, W.L. Goh, K.S. Yeo, Low-power high-speed multiplier for error-tolerant application, in *2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)* (2010), pp. 1–4. <https://doi.org/10.1109/EDSSC.2010.5713751>
13. M.S. Lau, K.V. Ling, Y.C. Chu, Energy-aware probabilistic multiplier: design and analysis, in *Proceedings of the 2009 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems* (2009), pp. 281–290. <https://doi.org/10.1145/1629395.1629434>
14. S. Mittal, A survey of techniques for approximate computing. *ACM Comput. Surv. (CSUR)* **48**(4), 1–33 (2016). <https://doi.org/10.1145/2893356>
15. A. Momeni, J. Han, P. Montuschi, F. Lombardi, Design and analysis of approximate compressors for multiplication. *IEEE Trans. Comput.* **64**(4), 984–994 (2015). <https://doi.org/10.1109/TC.2014.2308214>
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. (VLSI) Syst.* **23**(6), 1180–1184 (2015). <https://doi.org/10.1109/TVLSI.2014.2333366>
17. R. Ragavan, B. Barrois, C. Killian, O. Sentieys, Pushing the limits of voltage over-scaling for error-resilient applications, in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2017), pp. 476–481. <https://doi.org/10.23919/DATE.2017.7927036>
18. S. Venkatachalam, S.B. Ko, Design of power and area efficient approximate multipliers. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **25**(5), 1782–1786 (2017)
19. R. Venkatesan, A. Agarwal, K. Roy, A. Raghunathan, MACACO: modeling and analysis of circuits for approximate computing. In *proceedings of the 2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2011), pp. 667–673. <https://doi.org/10.1109/ICCAD.2011.6105401>
20. 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), 600–612 (2004). <https://doi.org/10.1109/TIP.2003.819861>
21. 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. (VLSI) Syst.* **25**(2), 393–401 (2017). <https://doi.org/10.1109/TVLSI.2016.2587696>