Journal of Electronic Testing (2022) 38:217–230
https://doi.org/10.1007/s10836-022-06000-3

Hardware Efficient Approximate Multiplier Architecture for Image Processing Applications

Shravani Chandaka[1] · **Balaji Narayanam**[2]

Received: 4 February 2022 / Accepted: 7 April 2022 / Published online: 9 June 2022

Abstract
In this research paper, approximate multipliers are designed to reduce the computational time and power delay product. However, there is a high possibility to further optimize the area and power using the modified Wallace Tree Multiplier (MWTM). This research paper proposes, two modified approximate 4:2 compressors are used for partial product addition in multipliers. Using the proposed MWTM, it is observed that Normalized Error Distance (NMED), Mean Relative Error Distance (MRED) and Power Delay Product (PDP) are reduced. The proposed architectures are synthesized using 90-nm CMOS standard cells. Modified Wallace tree multipliers of various sizes (8, 16 and 32 bit) are designed and their performance is compared with the existing general multipliers. The synthesis results of 8-bit MWTM shows that on an average the delay and power are reduced in the range of 10%–55.37% and 13.03%–13.78% when compared to existing multipliers. Moreover, for 16-bit MWTM shows that on an average the delay and power are reduced in the range of 0.11%–3.12% and 0.28%–6.59%. And 32-bit MWTM shows that on an average the power is reduced in the range of about 8%–27.99%. The image processing operations image blending, image smoothening and edge detection are implemented using the proposed MWTM. The results proved the efficiency of the MWTM.

Keywords Approximate Circuits · Compressor · Modified Wallace Tree Multiplier · Image Processing · Error Rate · Normalised Error Distance · Peak Signal to Noise Ratio

1 Introduction

Imprecise computing has opened a new era for developing high-performance, low-energy circuits and systems. Approximate circuits are useful in wide range of error resilient applications such as image processing, machine learning, and multimedia applications [26]. These applications can have a trade-off between accuracy, power, and speed. Multipliers and adders are resource hungry components in any digital circuits [1]. Approximate computation is one of these methods that is often used with a trade-off in accuracy [19]. Simplifying the arithmetic unit is the fundamental step for approximate computing [31]. Many previous research works [15-17, 23-25, 28-30, 38, 39] have focused on estimated multipliers, which provide higher speeds and lower power consumption at the expense of lower accuracies. Approximating such circuits will give improved efficiency in terms of power, area and speed [27]. Specifically for image processing applications, multiplier is the key element. When compared to array and exact multipliers parallel multipliers are faster. But the limitation of parallel multipliers is high power consumption [22]. When multipliers are used in large computational applications, there is a need for optimization. Power and delay of any digital circuit are always inversely proportional. However, different methods were proposed in the abstract levels to optimize power and delay specifications [6]. Not all the digital circuits need approximate circuits there are a few general-purpose processors which require exact circuits also [2]. For such systems error correction requires extra clock cycle and hence processing time increases. Approximate circuits are best suited for the

Responsible Editor: S. T. Chakradhar

✉ Shravani Chandaka
shravanichandaka11@gmail.com

Balaji Narayanam
prof.balaji.ece@gmail.com

[1] IT Application Developer, Legato Health Technologies, Bangalore, Karnataka, India

[2] Department of Electronics and Communication Engineering, JNTUK, Kakinada, Andhra pradesh, India

applications where errors are tolerant such as, image processing applications [27].

According Moore's law the size of transistors decreases exponentially [34]. This directs to improve circuit's efficiency and reduce power. This indicates an increasing challenge to improve circuit performance and power efficiency by using conventional technologies. This challenge is addressed with multiprocessor architectures and hybrid integration [33]. Many arithmetic circuits are approximated to speed up the computation. Approximate adders, multipliers, and logarithmic based multiplication are widely used these days in the digital circuits. Furthermore, formal verification approaches have been developed for a given error restriction to reduce circuit space and power dissipation [34, 37]. Three architectures are commonly used to compute Partial Products (PPs): carry save adder [7], the Wallace tree multiplier [38], and the Dadda tree multiplier [8]. Specifically, in Wallace tree, half adders, full adders and 4:2 compressors are the major blocks for fast computation. Partial products are generated in parallel by the adders in each level and the process is continued until two rows of partial products are obtained. Approximate compressors based Dadda multiplier is implemented in [28]. Large multiplier is constructed using the 4X4 multiplier [4]. Akbari et al. [2] proposed a reconfigurable compressor which can operate in both exact and approximate mode when required using dynamic accuracy. Further reduction in delay and energy is achieved using learning-based accelerators in [3, 5, 9, 10, 40]. Bharat et al. [12] proposed LOBA multiplier which is aimed to reduce the number of bits by selecting k-bits from n-bit input. Jothin et al. [20] has proposed an approximate multiplier using significance probability. Garg et al. [13] proposed a dynamic accuracy configurable multiplier (ACM) which offers twin performance improvement. Gorantla et al. [14] proposed approximate subtractors and dividers based on k-map minimization technique.

In this research, two approximate 4:2 compressor architectures are proposed. These compressors can be used in parallel multipliers like Wallace Tree [38] and Dadda [11]. Momeni et al. [28] proposed 4:2 compressor by introducing error in the truth table and finding minimized sum and carry expressions. These are referred as design1 and design2. While the compressors proposed by Akbari et al. [2] were referred as design3 and design4. These are designed by using dynamic switching circuitry for exact and approximate compressor design. Eight approximate 4:2 compressors are proposed in [35] based on majority logic. In this research paper these eight designs are referred as design5, 6, 7, 8, 9, 10, 11, and 12.

Following are the contributions of this research,

i. Design of modified approximate 4:2 compressors.
ii. Design of modified Wallace tree multiplier (MWTM) using modified approximate 4:2 compressors.
iii. Performance comparison of MWTM.
iv. Implementation of image processing operations (image blending, image smoothening and edge detection) using MWTM.

The rest of this paper is organized as follows. In Sect. 2, some prior works on the approximate multipliers are reviewed. The logic circuits of the proposed 4:2 compressors are explained in Sect. 3. Section 4 evaluates the accuracy of the modified Wallace and Conventional Wallace Tree Multiplier utilizing the proposed compressors while the effectiveness and hardware efficiency of the proposed compressors is assessed in Sect. 5. Finally, this paper is concluded in Sect. 6.

2 Proposed Compressor

In this section, firstly the conventional 4:2 compressor operation is discussed and then the proposed approximate compressor design methodology details are presented.

2.1 Conventional 4:2 Compressor

In parallel multipliers compressors are used to reduce the delay in partial product addition. A 4:2 compressor (exact) is also called as (5,3) counter. The most widely used 4:2 compressor circuit is as shown in Fig. 1. It has five inputs a_1, a_2, a3, a4, C_{in} and three outputs sum, carry, and C_{out}. The compressor is a combination of two full adders. Here the output carry C_{out} is independent of input carry C_{in}, this characteristic is used in multipliers to reduce delay. The compressor is designed such that C_{out} from n^{th} column is added to C_{in} of compressor in $(n + 1)^{th}$ column. The expressions

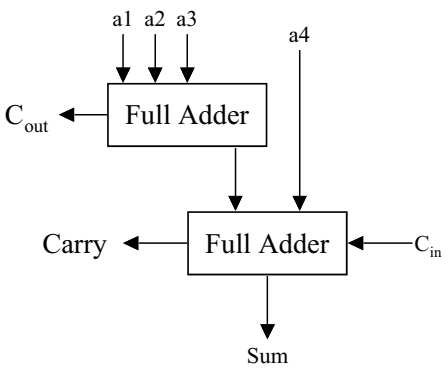

Fig. 1 Exact Compressor

of sum, carry and C_{out} for an exact compressor are given in Eqs. (1), (2), and (3).

$$\text{Sum} = a_1 \oplus a_2 \oplus a_3 \oplus a_4 \oplus C_{in} \tag{1}$$

$$\text{Carry} = (a_1 \oplus a_2 \oplus a_3 \oplus a_4)C_{in} + ((a_1 \oplus a_2 \oplus a_3 \oplus a_4)a_4) \tag{2}$$

$$C_{out} = (a_1 \oplus a_2)a_3 + (a_1 \oplus a_2)a_1 \tag{3}$$

The truth table for exact compressor with 16 input combinations is presented in Table 1.

2.2 Proposed 4:2 Compressor Designs

Two designs are proposed in this section for the approximate 4:2 compressor. The exact full-adder is replaced by an intuitive design of an approximate 4–2 compressor. The proposed approximate 4–2 compressors are designed by simplifying the truth table or Karnaugh Map (k-Map) [21]. The main objective is to minimize the logic function of compressor is by adding a few approximations in the K-map of exact design. Here, in all the proposed designs C_{in} and C_{out} are ignored in order to reduce area.

Minimized expressions for sum and carry are presented in the sub-sections.

2.2.1 Modified Approximate Compressor Design_p1

The K-map for approximate 4:2 compressor consists of four inputs a_1, a_2, a_3 and a_4. The modified logic circuit for design_p1 is as shown in Fig. 2. Basing on the obtained minterm the reduced expressions for sum and carry are given in Eqs. (4) and (5).

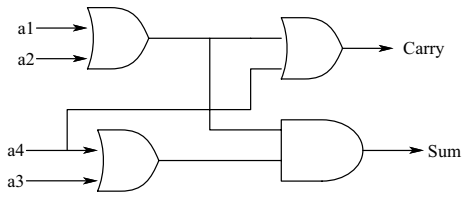

Fig. 2 Design_p_1

$$\text{Sum} = (a_1 + a_2)(a_4 + a_3) \tag{4}$$

$$\text{Carry} = a_1 + a_2 + a_4 \tag{5}$$

The truth table for the approximate compressor design_p_1 shows the error bits marked with '*'. The Table 2 shows exact and approximate sum and carry outputs for the four inputs. From the table it can be observed that out of 16 inputs, 7 inputs are giving wrong outputs because of K-map simplification. As a result, we can say the error rate is 43.75% which is tolerable for image processing applications.

2.2.2 Modified approximate compressor Design_p_2

The approximate compressor introduces 4 errors in sum and carry. The error rate here is 25%. When This error

Table 1 Truth Table for Exact Compressor

INPUTS						Exact Compressor Outputs	
C_{in}	a_1	a_2	a_3	a_4	C_{out}	Carry	Sum
0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	1
0	0	0	1	0	0	1	1
0	0	0	1	1	1	0	0
0	0	1	0	0	0	1	1
0	0	1	0	1	0	0	0
0	0	1	1	0	1	0	0
0	0	1	1	1	1	1	1
1	1	0	0	0	0	1	1
1	1	0	0	1	0	0	0
1	1	0	1	0	0	0	0
1	1	0	1	1	1	1	1
1	1	1	0	0	0	0	0
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1
1	1	1	1	1	1	0	0

Table 2 Truth table for approximate compressor Design_p_1

INUTS				Exact Compressor		Approximate Compressor Design_P_1	
a_1	a_2	a_3	a_4	Carry	Sum	Carry	Sum
0	0	0	0	0	0	0	0
0	0	0	1	0	1	1^*	0^*
0	0	1	0	0	1	0	1
0	0	1	1	1	0	1	0
0	1	0	0	0	1	1^*	0^*
0	1	0	1	1	0	1	1^*
0	1	1	0	1	0	1	0
0	1	1	1	1	1	1	1
1	0	0	0	0	1	1^*	0^*
1	0	0	1	1	0	1	1^*
1	0	1	0	1	0	1	1^*
1	0	1	1	1	1	1	1
1	1	0	0	1	0	1	0
1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1
1	1	1	1	0	0	1^*	1^*
ERROR						9	4

probability has reduced to major extent. The logic equations design_p_2 are given in Eqs. (6) and (7).

$$\begin{aligned}\text{Sum} = {} & a_3{}^1 a_4{}^1 (a_1 \oplus a_2) + a_3{}^1 a_4 (a_1 \oplus a_2)^1 \\ & + a_3 a_4{}^1 (a_1 \oplus a_2)^1 + a_3 a_4 (a_1 \oplus a_2)\end{aligned} \tag{6}$$

$$\begin{aligned}\text{Carry} = {} & a_1 a_3{}^1 (a_2 \oplus a_4) + a_4 (a_1 \oplus a_3) \\ & + a_1 a_3 a_4{}^1 + a_1 a_2{}^1 a_3 a_4{}^1 + a_2{}^1 a_3 a_4\end{aligned} \tag{7}$$

Based on the minimized expression the logic circuit is as shown in Fig. 3. Here the circuit is designed based on multiplexers so the speed of operation increases at the cost of area.

The truth table depicting the sum and carry outputs of the proposed approximate compressor design_p_2 is ass shown in Table 3. The performance of proposed approximate 4:2 compressors is compared with 12 different existing approximate 4:2 compressor designs proposed in [28, 2], and [35].

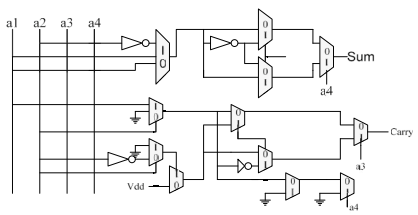

Fig. 3 Design_p_2

The performance of the proposed approximate 4:2 compressors is compared with 12 different approximate 4:2 compressor existing designs proposed in [28, 2], and [35].

Coming to the performance the proposed compressor design gives better error rate. Not all the proposed designs are showing better results but deisgn_p_2 claims to operate at high speed and improve daccuracy than all the existing compressors listed in Table 4.

Design_p_1 claims to be better than design9, design12, design13, design14 and design17.

The next section details about the modified Wallace tree multiplier (MWTM) based on design_p_1 and design_p_2.

3 Approximate Multiplier

The effect of using the proposed approximate compressor in conventional Wallace Tree and modified Wallace Tree [36] is explored in this section. Parallel multipliers are broadly classified into array multipliers and tree multipliers (Wallace and Dadda). Wallace tree multiplier (WTM) is fast compared to array multiplier. But it requires huge number of adders which leads to a complex circuit. Reducing the WTM circuit complexity and number of adders required would definitely be a challenging and cost effective. Development of area efficient multiplier as an alternative to exact multiplier is noteworthy. This is achieved by approximating the multiplier design structure. Mainly, multipliers can be approximated by using three approaches.

Table 3 Truth table for approximate compressor design_p2

INPUTS				Exact Compressor		Approximate Compressor Design_p_2	
a_1	a_2	a_3	a_4	Carry	Sum	Carry	Sum
0	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	1	0
0	1	1	1	1	1	1	1
1	0	0	1	1	0	**1***	1
1	0	1	0	1	0	1	**1***
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	**1***	**1***
ERROR						3	3

i. Approximation in partial products generation.
ii. Applying partial product truncation.
iii. Using approximate adders or compressors to accumulate the partial products.

An exact multiplier comprises of three stages.

i. Partial product generation.
ii. Minimizing the partial products.
iii. Propagating the carry to final stage for obtaining the result.

The second stage of approximate multipliers play vital role in reducing the delay, power consumption, and circuit complexity of any multiplier design. In conventional Wallace tree multiplier (WTM) [36] partial product addition depends on carry propagation and results in delay. In order to reduce the delay modified wallace tree multiplier

Table 4 Error Analysis of proposed approximate 4:2 Compressor and comparison with 12 existing compressor designs

INPUTS				Design1		Design2		Design3		Design4		Design5		Design6		Design7		Design8		Design9		Design10		Design11		Design12		Design_p1		Design_p2	
a1	a2	a3	a4	S	C	S	C	S	C	S	C	S	C	S	C	S	C	S	C	S	C	S	C	S	C	S	C	S	C	S	C
0	0	0	0	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
0	0	0	1	1	1	1	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	**0**	**0**	1	**0**	**1**	**0**	**1**	**1**	**0**
0	0	1	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	1	**0**	**0**	0	**1**	**0**	**0**	**0**	**1**	**0**
0	0	1	1	0	0	1	0	1	0	0	1	0	1	0	1	0	1	1	0	0	1	1	**0**	**0**	1	**0**	**1**	**0**	**1**	**0**	**1**
0	1	0	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	1	**0**	**0**	0	**1**	**0**	**0**	**1**	**1**	**0**
0	1	0	1	0	0	0	0	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	**0**	**0**	1	**0**	**1**	**1**	**1**	**0**	**1**
0	1	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	1	0	0	0	1	**0**	**0**	1	**1**	**0**	**1**	**1**	**0**	**1**
0	1	1	1	1	1	1	0	1	1	0	1	0	1	1	1	0	1	1	0	1	0	0	1	**1**	1	**1**	**0**	**1**	**1**	**1**	**1**
1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	0	0	1	1	0	**0**	0	**1**	**0**	**0**	**1**	**1**	**1**
1	0	0	1	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	0	0	1	1	0	**0**	1	**0**	**1**	**1**	**1**	**1**	**1**
1	0	1	0	0	0	0	0	0	1	1	0	1	0	1	0	1	0	1	0	0	0	1	0	**0**	1	**1**	**0**	**1**	**1**	**0**	**1**
1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	0	0	1	**1**	1	**1**	**0**	**1**	**1**	**1**	**1**
1	1	0	0	0	0	1	0	1	0	1	0	1	0	0	1	1	0	1	0	0	0	1	0	**0**	1	**0**	**1**	**0**	**1**	**0**	**1**
1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	0	0	1	**1**	1	**1**	**0**	**1**	**1**	**1**	**1**
1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	1	1	0	1	0	0	0	1	0	**0**	1	**1**	**0**	**1**	**1**	**1**	**1**
1	1	1	1	0	0	1	0	1	1	1	1	1	1	1	1	1	0	1	0	1	0	0	1	**1**	1	**1**	**0**	1	1	1	1
ERROR				**1**	**9**	**5**	**10**	**4**	**3**	**9**	**6**	**10**	**10**	**5**	**5**	**11**	**9**	**10**	**11**	**6**	**12**	**10**	**8**	**6**	**2**	**6**	**8**	**9**	**4**	**3**	**3**

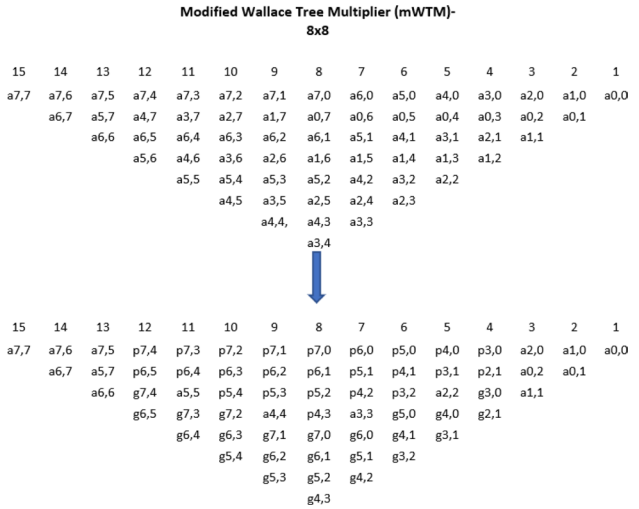

Fig. 4 Alteration of partial products to generate and propagate signals

(MWTM) is proposed. Three different sized Wallace tree multipliers (WTM) and Wallace tree multipliers (WTM) are designed for unsigned multiplication of 8-bit, 16-bit and 32-bit input data. Consider an example of an 8-bit MWTM in which the two 8-bit inputs are α_m and β_n and the partial products are denoted by $a_{m,n}$ and $a_{n,m}$ where m and n varies from 0 to 7. The partial products obtained on multiplication of α_m and β_n for varying values of m and n (0 to 7) is shown in Fig. 4. From Fig. 4 it can be observed that in first row 15 partial products are obtained for different values of m and n. In a tree structure the number of partial products gets reduced to a single partial product in the last row. In the conventional Wallace tree [36] depicted, the column contains varied number of partial product terms. If the number of partial products in a column are more than three then full adders and half adders are used to obtain next stage partial products. The full adders and half adders depend on carry propagation and results in delay. In order to reduce the delay the MWTM is proposed in which if a column has more than three partial products instead of using full adder and half adder for partial product generation, the partial products are combined to form propogate and generate signals as given in Eqs. (8) and (9).

$$P_{m,n} = a_{m,n} + a_{n,m} \tag{8}$$

$$g_{m,n} = a_{m,n} \cdot a_{n,m} \tag{9}$$

In Fig. 4 it can be observed that in column 7 and 9 the m and n values are same for $a_{3,3}$ and $a_{4,4}$. Hence they cannot be used to produce propagate and generate signals. The column 8 contains 4 different propagate signals and 4 generate signals. Hence such columns which contain upto 4 propogate signals is suitable for addition using approximate 4:2 compressor.

The modified MWTM is as shown in Fig. 5.

The propose approximate 4:2 compressor designs are used in the design of MWTM with widths of 8, 16, and 32. Similarly we have designed a conventional Wallace tree multiplier (WTM) using the proposed approximate compressors.

4 Error Metrics

In this section, various error metrics like error distance, Normalized error distance and Mean relative error distance [18, 32] are used to evaluate characteristics of approximate circuits are discussed below.

Following error metrics are considered,

i. Error Distance (ED):

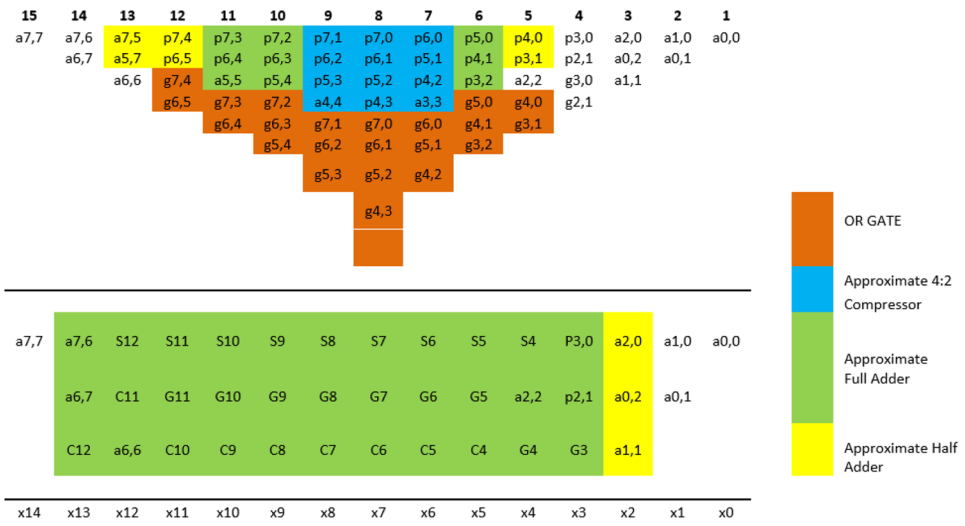

Fig. 5 Modified Wallace tree multiplier (MWTM) by altering partial products

The Error Distance gives the absolute difference between the approximate and exact outputs of multipliers. Let, M and M' be the exact and imprecise multiplier outputs respectively. Then,

$$ED = \left| M - M' \right| \tag{10}$$

ii. Normalized Error Distance (NED):

NED is independent of size of the multiplier. So, it is very useful in characterizing any approximate circuit. It is defined as the average error over normalized with maximum error value.

Table 5 Comparison of error metrics MWTM using proposed compressors and with existing designs

Bit-Width	Metrics	MED	WCE	MRED	NED	ED	AC
8-bit	Design1 [28]	2783.485	29632	3.4E-06	0.094	180996144	81.80
	Design2 [28]	7922.701	22112	5.6E-06	0.358	515173624	48.20
	Design3 [2]	7052.180	16376	5.0E-06	0.431	458567992	53.89
	Design4 [2]	5773.803	24928	6.9E-06	0.232	375441560	62.25
	Design5 [35]	5773.803	24928	6.9E-06	0.232	375441560	62.25
	Design6 [35]	1883.600	14144	2.2E-06	0.133	122481072	87.68
	Design7 [35]	8051.579	16472	5.6E-06	0.489	523553920	47.36
	Design8 [35]	8575.826	22648	2.0E-05	0.379	557643096	43.93
	Design9 [35]	8154.952	28272	5.4E-06	0.288	530275744	46.68
	Design10 [35]	8048.819	17792	5.7E-06	0.452	523374424	47.37
	Design11 [35]	7015.739	16760	5.7E-06	0.419	456198400	54.13
	Design12 [35]	8051.579	16472	5.6E-06	0.489	523553920	47.36
	Design_p1	**7052.180**	**16376**	**5.0E-06**	**0.431**	**458567992**	**53.89**
	Design_p2	**398.209**	**21504**	**4.1E-07**	**0.019**	**25893536**	**97.40**
16-bit	Design1 [28]	2783.49	29632	3.42E-06	0.0939	180996144	81.801
	Design2 [28]	1081608323.20	1081618424	1.54E-05	1.0000	70331581215992	-7071855.904
	Design3 [2]	1081608423.94	1081618424	1.54E-05	1.0000	70331587766584	-7071856.563
	Design4 [2]	5773.80	24928	6.88E-06	0.2316	375441560	62.249
	Design5 [35]	5773.80	24928	6.88E-06	0.2316	375441560	62.249
	Design6 [35]	1883.60	14144	2.16E-06	0.1332	122481072	87.684
	Design7 [35]	1081608669.24	1081618424	1.54E-05	1.0000	70331603717328	-7071858.16
	Design8 [35]	8575.83	22648	1.96E-05	0.3787	557643096	43.928
	Design9 [35]	1082138158.11	1082146816	1.54E-05	1.0000	70366033731136	-7075320.16
	Design10 [35]	1081608660.76	1081618424	1.54E-05	1.0000	70331603165720	-7071858.11
	Design11 [35]	7015.74	16760	5.74E-06	0.4186	456198400	54.129
	Design12 [35]	1081608669.24	1081618424	1.54E-05	1.0000	70331603717328	-7071858.16
	Design_p1	**1081608423.94**	**1081618424**	**1.54E-05**	**1.0000**	**70331587766584**	**-7071856.56**
	Design_p2	**398.21**	**21504**	**4.11E-07**	**0.0934**	**25893536**	**97.396**
32-bit	Design1 [28]	2628.511	37968	2.72E-06	0.06923	170918928	84.95
	Design2 [28]	4294949834.86	4.29E + 09	1.54E-05	1.00000	2.79E + 14	-24598503.6
	Design3 [2]	4294949834.86	4.29E + 09	1.54E-05	1.00000	2.79E + 14	-24598503.6
	Design4 [2]	4361.358	37144	3.83E-06	0.11742	283597336	75.02
	Design5 [35]	4361.358	37144	3.83E-06	0.11742	283597336	75.02
	Design6 [35]	1572.673	16000	1.52E-06	0.09829	102263080	90.99
	Design7 [35]	4294949834.86	4.29E + 09	1.54E-05	1.00000	2.79E + 14	-24598503.6
	Design8 [35]	7221.945	31256	1.08E-05	0.23106	469606968	58.64
	Design9 [35]	4294949834.86	4.29E + 09	1.54E-05	1.00000	2.79E + 14	-24598503.6
	Design10 [35]	4294949834.86	4.29E + 09	1.54E-05	1.00000	2.79E + 14	-24598503.6
	Design11 [35]	7879.937	16760	5.49E-06	0.47016	512392872	54.87
	Design12 [35]	4294949834.86	4.29E + 09	1.54E-05	1.00000	2.79E + 14	-24598503.6
	Design_p1	**4294949834.86**	**4.29E + 09**	**1.54E-05**	**1.00000**	**2.79E + 14**	**-24598503.6**
	Design_p2	**328.275**	**42896**	**2.95E-07**	**0.00765**	**21346096**	**98.12**

$$\mathrm{NED} = \frac{1}{\left(2^{N-1}\right)^2} \sum_{i=1}^{2^N} \frac{|ED_i|}{2^{2N}} \tag{11}$$

where N = Size of the multiplier and ED = Error distance.

iii. Mean Relative Error Distance (MRED):

The Mean Relative Error Distance, defined as the average value of relative error distance. Which is calculated using exact result.

$$\mathrm{MRED} = \frac{1}{\left(2^{N-1}\right)^2} \sum_{i=1}^{2^N} \frac{|ED_i|}{P_i} \tag{12}$$

Here worst-case error (WCE) and mean error distance (MED) are also calculated for better analysis.

5 Results and Discussion

In this section, the proposed approximate 4:2 compressor designs are tested and evaluated. The proposed designs are analysed based on power consumption, area and the speed of operation. The approximate circuits differ exact circuits with certain error distance. The metrics used to evaluate the error are, Error Distance (ED), Mean Error Distance (MED), Normalised Error Distance (NED), Worst Case Error (WCE), and Accuracy (AC). The proposed architectures are implemented in Verilog HDL and synthesized using 90-nm, CMOS technology. Industrial standard cadence compiler is used in the design of approximate multipliers to obtain power, delay and area.

5.1 Comparison of Accuracy for Approximate Multipliers

Scalability has a lot of advantages with respect to area and power. Technology scaling would definitely show a remarkable dip in the power. Error analysis on both modified Wallace Tree multiplier (MWTM) and conventional Wallace tree multiplier (WTM) is performed. The detailed comparison is shown in Tables 5 and 6 respectively. By observing the evaluation metrics MED, NED and ED we can say that MWTM has shown better performance than WTM.

5.2 Comparison of Electrical Performance of Approximate Multipliers

This section, shows the comparison of area, power, and delay of existing compressors compared to the proposed designs for 90 nm. We have extended our proposed approximate MWTM to two more variants 16 bit and 32-bit. Electrical performance of 16-bit and 32-bit MWTM and WTM using proposed approximate compressor deisgn_p1 and design_p2 are also shown in Tables 7 and 8.

From Table 7, it is observed that the 8-bit MWTM with Design_p1 and Design_p2 reduce delay and power compared to that of MWTM with remaining Existing Compressors. Further, 8-bit MWTM with Design_p1 and Design_p2 reduced delay and power on an average of 55.37%-10% and 13.78%-13.03% compared to MWTM with remaining Existing 12 Approximate Compressors.

Moreover, from Table 1, it is observed that the 16-bit MWTM with Design_p1 achieves reduced energy by nearly 6.43%-0.47%, respectively, compared to that of MWTM

Table 6 Comparison of error metrics of 8-bit width conventional Wallace Tree Multiplier (WTM) using proposed compressors and with existing designs

WTM_8bit						
Design No	**MED**	**WCE**	**MRED**	**NED**	**ED**	**Accuracy**
Design1 [28]	3291.372	19296	4.07E-06	0.171	2.1402E + 08	78.48
Design2 [28]	4102.636	18640	4.21E-06	0.220	2.6677E + 08	73.18
Design3 [2]	3428.123	8568	2.87E-06	0.400	2.2291E + 08	77.59
Design4 [2]	3402.944	16600	4.33E-06	0.205	2.2128E + 08	77.75
Design5 [35]	3402.944	16600	4.33E-06	0.205	2.2128E + 08	77.75
Design6 [35]	1401.198	9728	1.55E-06	0.144	9.1113E + 07	90.84
Design7 [35]	3335.504	13864	3.02E-06	0.241	2.1689E + 08	78.19
Design8 [35]	5896.186	17952	9.65E-06	0.328	3.8340E + 08	61.45
Design9 [35]	7301.613	18080	5.27E-06	0.404	4.7479E + 08	52.26
Design10 [35]	3499.809	11584	3.25E-06	0.302	2.2758E + 08	77.12
Design11 [35]	4009.423	10256	5.41E-06	0.391	2.6071E + 08	73.78
Design12 [35]	3335.504	13864	3.02E-06	0.241	2.1689E + 08	78.19
Design_p1	**7969.972**	**18960**	**5.27E-06**	**0.420**	**5.1825E + 08**	**47.89**
Design_p2	**1040.082**	**7168**	**4.04E-08**	**0.106**	**2.6063E + 06**	**98.74**

Table 7 Design Metrics Comparison of MWTM with Existing and Proposed 4–2 Compressors

Bit-Width	Metrics	Area (μm^2)	Power (mW)	Delay (ns)	PDP (fJ)
8-bit	Design1 [28]	1920	13.93	2.78	38.73
	Design2 [28]	1920	13.93	2.78	38.73
	Design3 [2]	2030	16.11	3.35	53.97
	Design4 [2]	1220	10.54	2.92	30.78
	Design5 [35]	1220	10.54	2.92	30.78
	Design6 [35]	2210	16.12	3.08	49.65
	Design7 [35]	2061	16.25	2.94	47.78
	Design8 [35]	1929	16.23	2.78	45.12
	Design9 [35]	2030	13.92	2.91	40.51
	Design10 [35]	2030	13.71	2.91	39.89
	Design11 [35]	2061	16.16	2.89	46.71
	Design12 [35]	2061	16.25	2.94	47.78
	Design_p1	1987	16.14	2.88	46.49
	Design_p2	2422	14.01	6.69	93.73
16-bit	Design1 [28]	8567	15.84	7.36	116.59
	Design2 [28]	8567	15.84	7.36	116.59
	Design3 [2]	9606	18.92	7.89	149.28
	Design4 [2]	5769	12.12	7.37	89.33
	Design5 [35]	5769	12.12	7.37	89.33
	Design6 [35]	9726	19.56	7.41	144.94
	Design7 [35]	9133	19.23	7.39	142.11
	Design8 [35]	8603	18.97	7.37	139.81
	Design9 [35]	9006	15.99	7.36	117.69
	Design10 [35]	9006	15.65	7.36	115.19
	Design11 [35]	9133	19.09	7.35	140.32
	Design12 [35]	9133	19.23	7.39	142.11
	Design_p1	8836	18.95	7.37	139.67
	Design_p2	10574	20.09	10.79	216.78
32-bit	Design1 [28]	36076	68.56	15.26	1046.23
	Design2 [28]	36076	68.56	15.26	1046.23
	Design3 [2]	37832	81.91	15.48	1267.97
	Design4 [2]	24886	53.79	15.27	821.38
	Design5 [35]	24886	53.79	15.27	821.38
	Design6 [35]	40714	84.47	15.29	1291.55
	Design7 [35]	38340	83.45	15.29	1275.96
	Design8 [35]	36121	72.45	15.26	1105.59
	Design9 [35]	37832	68.59	15.25	1045.99
	Design10 [35]	37832	66.83	15.25	1019.16
	Design11 [35]	38340	82.62	15.25	1259.96
	Design12 [35]	38340	83.45	15.25	1272.62
	Design_p1	37153	82.02	15.20	1246.71
	Design_p2	44097	60.83	16.58	1008.57

with Design3, Design6, Design7 and Design11. The 16-bit MWTM with Design_p1 achieves nearly average value of 3.12%-0.11% and 6.59%-0.28% reduced to power and delay, respectively, compared to 16-bit MWTM with Design6, Design7 and Design12.

Similarly, it is observed that the 32-bit MWTM with Design_p2 consumes lesser power of about 27.99%-8.98% compared to that 32-bit MWTM with Design1-Design12 respectively. It is also noted that the MWTM with Design_p1 offer better values of power and delay in minimum input operands. It is also observed that MWTM with Design_p1 offer better values of power in higher order multiplier design.

Table 8 shows the synthesis results in terms of power, area, energy, and delay.

Table 8 Design Metrics Comparison of Wallace Tree Multiplier with Existing and Proposed 4–2 Compressors

Bit-Width	Metrics	Area (μm^2)	Power (mW)	Delay (ns)	PDP (fJ)
8-bit	Design1 [28]	1947	14.36	3.61	51.84
	Design2 [28]	948	99.92	2.86	285.77
	Design3 [2]	948	99.92	3.82	381.69
	Design4 [2]	1970	16.15	3.56	57.49
	Design5 [35]	2148	15.83	3.56	56.35
	Design6 [35]	2160	14.55	3.56	51.80
	Design7 [35]	1798	13.63	3.82	52.07
	Design8 [35]	364	6.00	3.82	22.92
	Design9 [35]	1895	19.61	3.39	66.48
	Design10 [35]	1717	120.85	2.59	313.00
	Design11 [35]	1861	19.85	3.63	72.06
	Design12 [35]	1699	21.94	2.60	57.04
	Design_p1	**1893**	**18.48**	**3.57**	**65.97**
	Design_p2	**1530**	**16.62**	**3.26**	**54.18**
16-bit	Design1 [28]	9366	20.02	7.89	157.96
	Design2 [28]	8076	16.62	7.31	121.49
	Design3 [2]	8673	16.01	7.86	125.84
	Design4 [2]	4676	11.23	7.87	88.39
	Design5 [35]	4676	11.23	7.87	88.39
	Design6 [35]	9527	16.97	7.86	133.39
	Design7 [35]	8080	15.12	7.79	117.79
	Design8 [35]	7753	23.89	7.06	168.67
	Design9 [35]	8331	25.27	7.73	195.34
	Design10 [35]	7683	25.14	7.69	193.33
	Design11 [35]	8458	22.63	7.73	174.93
	Design12 [35]	7819	23.03	7.79	179.41
	Design_p1	8455	15.67	7.87	123.33
	Design_p2	10645	24.25	7.92	192.06
32-bit	Design1 [28]	39273	87.39	15.81	1381.64
	Design2 [28]	34114	83.54	15.21	1270.65
	Design3 [2]	36500	70.02	15.77	1104.22
	Design4 [2]	20514	51.66	15.76	814.17
	Design5 [35]	20514	51.66	15.76	814.17
	Design6 [35]	39915	74.46	15.77	1174.24
	Design7 [35]	34126	66.33	15.69	1040.72
	Design8 [35]	32818	119.61	14.96	1789.37
	Design9 [35]	35131	125.99	15.66	1973.01
	Design10 [35]	32539	123.25	15.59	1921.47
	Design11 [35]	35640	113.26	15.66	1773.66
	Design12 [35]	33085	162.72	15.64	2544.95
	Design_p1	35628	68.61	15.77	1081.98
	Design_p2	44382	58.24	13.86	807.21

From Table 8, it is observed that the 8-bit WTM with Design_p1 and Design_p2 reduce delay and power compared to that of WTM with remaining Existing Compressors. Further, 8-bit WTM with Design_p1 and Design_p2 reduced delay and power on an average of 14.7%-3.83% and 15.24%-5.7% compared to MWTM with remaining existing 12 approximate compressors. The 16-bit WTM with Design_p1 achieves nearly average value of 11.37%-0.01% and 2.12%-0.25% reduced to delay and power, respectively, compared to 16-bit WTM with remaining existing 12 approximate compressors.

Similarly, it is observed that the 32-bit WTM with Design_p2 consumes lesser delay of about 12.35%-7.38% compared to that 32-bit WTM with Design1-Design12 respectively. It is also noted that the WTM with Design_p2

offer better values of power in minimum input operands. It is also observed that WTM with Design_p1 offer better values of power and delay in higher order multiplier design.

5.3 Application in Image Processing

We considered image blending, smoothening in this section to inspect the behaviour of approximate multipliers in typical applications that are resistant to errors. Here we have not reported the design proposed by various authors whose accuracy is very low.

5.3.1 Image Blending

Image blending is the technique used to combine two different images. Here we have taken images of two sizes 256 × 256 and 512 × 512 each of 8-bit width and 16-bit width respectively. Image blending is obtained multiplying two different images pixel by pixel. We have a performed processing using proposed approximate multiplier on standard images and compared with the exact multiplier. Consider G(i,j) is the output image and the input images are denoted as X(m,n) and Y(m,n).

Here m,n are the input pixel values and I,j are the output image pixel values.

$$G(i,j) = X(m,n) \times Y(m,n) \tag{13}$$

Figure 6a and b shows the two input test images and Fig. 6c to g shows the blended images obtained from the exact multiplier, proposed approximate MWTM and the existing approximate designs specified in [28], [2], and [35].

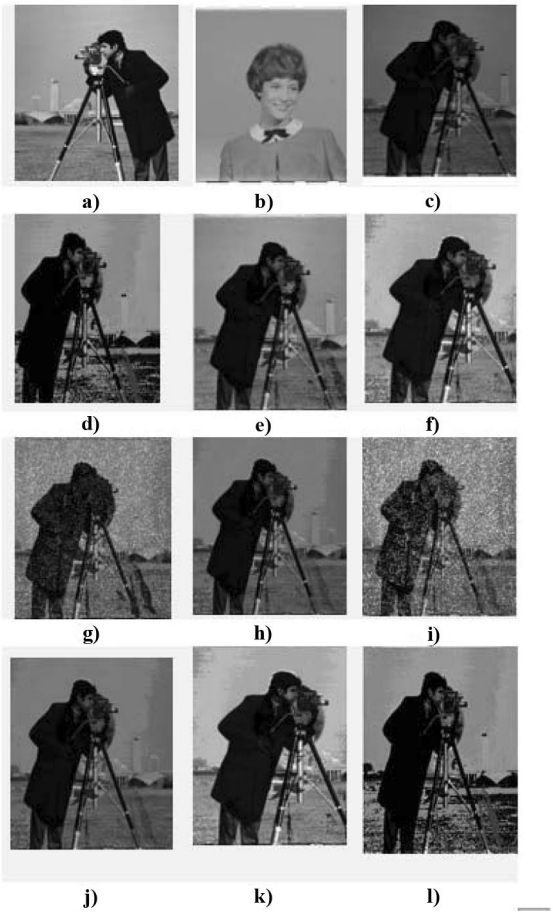

Fig. 6 Image Blending results for sample images **a**) input image1 **b**) input image2 **c**) exact output **d**) Design_p1 **e**) Design_p2 **f**) Design1 [28] **g**) Design4 [2] **h**)Design6 [35] **i**) Design7 [35] **j**) Design8 [35] **k**) Design10 [35] **l**)Design11 [35]

5.3.2 Image Smoothening

Smoothening is process which provides a blur effect and also reduces the noise. We have used the Gaussian mask for kernel convolution. Gaussian blurring is the preprocessing step in computer vision applications. In real time image processing applications convolution and multiplication are the vast resource consuming operations. An efficient hardware design for convolution is definitely remarkable research in the field of real time image processing.

Following is the expression for Gaussian Smoothening.

$$G(x.y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{14}$$

Here x,y are pixel values and σ is the standard deviation.

The output image after smoothening is given by X (i, j) and the expression for smoothening using gaussian mask as denoted by kernel is given in Eq. (15).

$$X(i,j) = \frac{1}{256} \sum_{x=-2}^{2} \sum_{y=-2}^{2} X(i+m, j+n).kernel(m+3, n+3) \tag{15}$$

$$\text{where, kernel} = \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}.$$

Here in Gaussian smoothening, multiplication operation is done using approximate circuit. Whereas, addition and subtraction operations are exact. Smoothening is done using MWTM. Figure 7 shows some results obtained after performing filtering on standard images of 256 × 256 size. Table 9 gives performance analysis based on PSNR and SSIM values for different filtered images. Figure 7 shows the Gaussian Smoothed images obtained using approximate MWTM for standard images.

Table 9 Performance analysis of MWTM using proposed and existing designs for Gaussian Filtering

Image	Lena			Cameraman			Girl		
									
Metrics	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
Design1 [28]	0.099	58.18	0.2138	0.024	64.37	0.6563	0.027	63.75	0.7030
Design2 [28]	0.228	54.55	0.1105	0.257	54.03	0.1043	0.171	55.79	0.3717
Design3 [2]	0.162	56.04	0.0806	0.219	54.73	0.1904	0.116	57.47	0.1368
Design4 [2]	0.120	57.35	0.0087	0.139	56.71	0.0142	0.115	57.52	0.0095
Design5 [35]	0.119	57.35	0.0086	0.139	56.71	0.0142	0.115	57.52	0.0095
Design6 [35]	0.101	58.09	0.2496	0.005	71.00	0.7589	0.006	70.38	0.8397
Design7 [35]	0.299	53.38	0.3031	0.344	52.77	0.3260	0.018	65.68	0.7246
Design8 [35]	0.174	55.73	0.0948	0.207	54.97	0.1650	0.114	57.56	0.4385
Design9 [35]	0.112	57.65	0.2029	0.094	58.42	0.3381	0.174	55.72	0.2447
Design10 [35]	0.278	53.69	0.1784	0.294	53.44	0.2597	0.174	55.72	0.2447
Design11 [35]	0.119	57.37	0.0105	0.166	55.93	0.2535	0.115	57.53	0.0130
Design12 [35]	0.010	53.38	0.3031	0.344	52.77	0.3260	0.018	65.68	0.7246
Design_p1	**0.162**	**56.04**	**0.0806**	**0.219**	**54.73**	**0.1904**	**0.116**	**57.47**	**0.1368**
Design_p2	**0.089**	**58.62**	**0.2731**	**0.003**	**72.89**	**0.7896**	**0.005**	**71.11**	**0.8671**

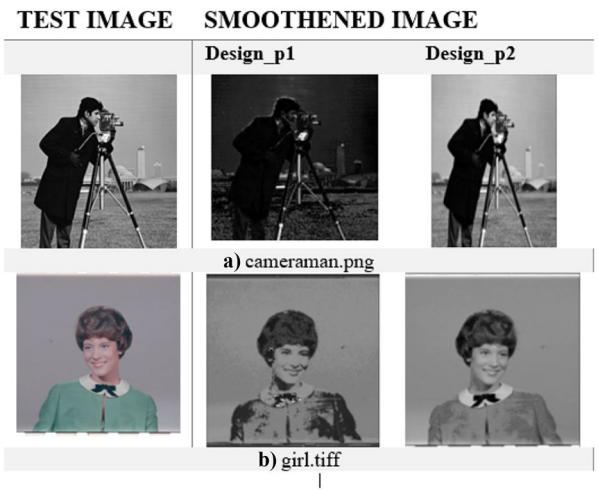

Fig. 7 Image Smoothening for **a**) Cameraman.jpg and smoothened images from design_p1 and design_p2 **b**)

6 Conclusion

This paper discussed the designs of two approximate compressors and their implementation in modified Wallace Tree. The proposed approximate compressors accuracy is high compared to existing approximate compressors. The approximate compressor design_p1 and design_p2, proved to be efficient in terms of power, area, and delay compared to existing and moreover, the MED, WCE, MRED, NED, ED achieves better values for the proposed approximate compressors, and MWTM. The proposed approximate compressors and MWTM are analyzed for image blending, image smoothening and edge detection applications. 8-bit MWTM shows that on an average the delay and power are reduced in the range of 10%–55.37% and 13.03%–13.78% when compared to existing multipliers. Moreover, for 16-bit MWTM shows that on an average the delay and power are reduced in the range of 0.11%–3.12% and 0.28%–6.59%. And 32-bit MWTM shows that on an average the power is reduced in the range of about 8%–27.99%. The architecture proposed can be further extended to real time image processing applications.

Funding No funding was received to assist with the preparation of this manuscript.

Data Availability Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Declarations

Conflict of Interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Competing Interests The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abid Z, El-Razouk H, El-Dib DA (2008) Low power multipliers based on new hybrid full adders. Microelectron J 39(12):1509–1515. https://doi.org/10.1016/j.mejo.2008.04.002 (ISSN 0026–2692)
2. Akbari O, Kamal M, Afzali-Kusha A, Pedram M (2016) RAP-CLA: A reconfigurable approximate carry look-ahead adder. IEEE Trans Circuits Syst II Express Briefs 65(8):1089–1093. https://doi.org/10.1109/TCSII.2016.2633307
3. Akbari O, Kamal M, Afzali-Kusha A, Pedram M, Shafique M (2018) PX-CGRA: Polymorphic approximate coarse-grained reconfigurable architecture. In: Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE), pp 413–418. https://doi.org/10.23919/DATE.2018.8342045
4. Ansari MS, Jiang H, Cockburn BF, Han J (2018) Low-Power Approximate Multipliers Using Encoded Partial Products and Approximate Compressors. IEEE J Emerg Sel Top Circuits Syst 8(3):404–416. https://doi.org/10.1109/JETCAS.2018.2832204
5. Ansari MS, Mrazek V, Cockburn BF, Sekanina L, Vasicek Z, Han J (2020) Improving the Accuracy and Hardware Efficiency of Neural Networks Using Approximate Multipliers. IEEE Trans Very Large Scale Integr Syst 28(2):317–328. https://doi.org/10.1109/TVLSI.2019.2940943
6. Bahadori M, Kamal M, Afzali-Kusha A, Pedram M (2016) High-Speed and Energy-Efficient Carry Skip Adder Operating Under a Wide Range of Supply Voltage Levels. IEEE Trans Very Large Scale Integr Syst 24(2):421–433. https://doi.org/10.1109/TVLSI.2015.2405133
7. Behrooz P (2010) Computer Arithmetic: Algorithms and Hardware Designs, 2nd edn. Oxford University Press, New York
8. Bickerstaff KC, Schulte MJ, Swartzlander EE (1995) Parallel reduced area multipliers. J VLSI Sig Proc 9:181–191. https://doi.org/10.1007/BF02407084
9. Brandalero M, Beck ACS, Carro L, Shafique M (2018) Approximate On-The-Fly Coarse-Grained Reconfigurable Acceleration for General-Purpose Applications. In: Proc. 55th ACM/IEEE Design Automation Conference (DAC), pp 1–6. https://doi.org/10.1109/DAC.2018.8465930
10. Chen T, Du Z, Sun N, Wang J, Wu C, Chen Y, Temam O (2014) Diannao: "A small-footprint high-throughput accelerator for ubiquitous machine-learning." ACM Sigplan Notices 49:269–284. https://doi.org/10.1145/2541940.2541967
11. Dadda L (1983) Some schemes for fast serial input multipliers. In: Proc. IEEE 6th Symposium on Computer Arithmetic (ARITH), pp 52–59. https://doi.org/10.1109/ARITH.1983.6158074
12. Garg B, Patel SK, Dutt S (2020) LoBA: A Leading One Bit Based Imprecise Multiplier for Efficient Image Processing. J Electron Test 36:429–437. https://doi.org/10.1007/s10836-020-05883-4
13. Garg B, Sharma GK (2017) ACM: An Energy-Efficient Accuracy Configurable Multiplier for Error-Resilient Applications. J Electron Test 33:479–489. https://doi.org/10.1007/s10836-017-5667-8
14. Gorantla A, Deepa P (2019) Design of approximate subtractors and dividers for error tolerant image processing applications. J Electron Test 35(6):901–907. https://doi.org/10.1007/s10836-019-05837-5
15. Hashemi S, Bahar RI, Reda S (2015) DRUM: A Dynamic Range Unbiased Multiplier for approximate applications. In: Proc. IEEE/ACM Int Conf Comput Aided Des., pp 418–425. https://doi.org/10.1109/ICCAD.2015.7372600
16. Kulkarni P, Gupta P, Ercegovac M (2011) Trading Accuracy for Power with an Underdesigned Multiplier Architecture. In: Proc. 24th Internatioal Conference on VLSI Design, pp 346–351. https://doi.org/10.1109/VLSID.2011.51
17. Kyaw KY, Goh WL, Yeo KS (2010) Low-power high-speed multiplier for error-tolerant application. In: Proc. IEEE Int Conf Electron Devices Solid-State Circuits (EDSSC), pp 1–4. https://doi.org/10.1109/EDSSC.2010.5713751
18. Jiang H, Liu C, Maheshwari N, Lombardi F, Han J (2016) A comparative evaluation of approximate multipliers. In: Proc. IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), pp 191–196. https://doi.org/10.1145/2950067.2950068
19. Jiang H, Santiago FJH, Mo H, Liu L, Han J (2020) Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications. Proc IEEE 108(12):2108–2135. https://doi.org/10.1109/JPROC.2020.3006451
20. Jothin R, Vasanthanayaki C (2018) High Performance Modified Static Segment Approximate Multiplier based on Significance Probability. J Electron Test 34:607–614. https://doi.org/10.1007/s10836-018-5748-3
21. Kulkarni P, Gupta P, Ercegovac M (2011) Trading Accuracy for Power with an Underdesigned Multiplier Architecture. In: Proc. 24th Internatioal Conference on VLSI Design, pp 346–351. https://doi.org/10.1109/VLSID.2011.51
22. Kuo K-C, Chou C-W (2010) Low power and high-speed multiplier design with row bypassing and parallel architecture. Microelectron J 41:639–650. https://doi.org/10.1016/j.mejo.2010.06.009
23. Lau M, Ling K, Chu Y-C (2009) Energy-aware probabilistic multiplier: Design and analysis. In: Proc. International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES '09). Association for Computing Machinery, New York, NY, USA, pp 281–290. https://doi.org/10.1145/1629395.1629434
24. Liu C, Han J, Lombardi F (2014) A low-power, high-performance approximate multiplier with configurable partial error recovery. In: Proc. Design Automation & Test in Europe Conference & Exhibition (DATE), pp 1–4. https://doi.org/10.7873/DATE.2014.108
25. Lin C, Lin I (2013) High accuracy approximate multiplier with error correction. In: Proc. IEEE 31st International Conference on Computer Design (ICCD), pp 33–38. https://doi.org/10.1109/ICCD.2013.6657022
26. Liu W, Lombardi F, Schulte M (2020) Approximate Computing: From Circuits to Applications. Proc IEEE 108(12):2103–2107. https://doi.org/10.1109/JPROC.2020.3033361
27. Mittal S (2016) A Survey of Techniques for Approximate Computing. ACM Comp Surv 48(4):62. https://doi.org/10.1145/2893356
28. Momeni A, Han J, Montuschi P, Lombardi F (2015) Design and Analysis of Approximate Compressors for Multiplication. IEEE Trans Comput 64(4):984–994. https://doi.org/10.1109/TC.2014.2308214
29. Moore GE (2006) "Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff.," in IEEE Solid-State Circuits Society Newsletter, vol. 11, no. 3, pp. 33-35. https://doi.org/10.1109/N-SSC.2006.4785860
30. Narayanamoorthy S, Moghaddam HA, Liu Z, Park T, Kim NS (2015) Energy-efficient approximate multiplication for digital signal processing and classification applications. IEEE Trans Very Large Scale Integr Syst 23(6):1180–1184. https://doi.org/10.1109/TVLSI.2014.2333366
31. Raha A, Jayakumar H, Raghunathan V (2016) Input-Based Dynamic Reconfiguration of Approximate Arithmetic Units for Video Encoding. IEEE Trans Very Large Scale Integr Syst 24(3):846–857. https://doi.org/10.1109/TVLSI.2015.2424212
32. Rodrigues G, Lima Kastensmidt F, Bosio A (2020) Survey on Approximate Computing and Its Intrinsic Fault Tolerance. Electronics 9(4):557. https://doi.org/10.3390/electronics9040557

33. Shafique M, Garg S, Henkel J, Marculescu D (2014) The EDA challenges in the dark silicon era. In: Proc. 51st ACM/IEEE Design Automation Conference (DAC), pp 1–6. https://doi.org/10.1145/2593069.2593229
34. Shin D, Gupta SK (2010) Approximate logic synthesis for error tolerant applications. In: Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE 2010), pp 957–960. https://doi.org/10.1109/DATE.2010.5456913
35. Strollo AGM, Napoli E, De Caro D, Petra N, Meo GD (2020) Comparison and Extension of Approximate 4–2 Compressors for Low-Power Approximate Multipliers. IEEE Trans Circuits Syst I Regul Pap 67(9):3021–3034. https://doi.org/10.1109/TCSI.2020.2988353
36. Venkatachalam S, Ko S (2017) Design of Power and Area Efficient Approximate Multipliers. IEEE Trans Very Large Scale Integr Syst 25(5):1782–1786. https://doi.org/10.1109/TVLSI.2016.2643639
37. Venkataramani S, Sabne A, Kozhikkottu V, Roy K, Raghunathan A (2012) SALSA: Systematic logic synthesis of approximate circuits. In: Proc. Design Automation Conference (DAC), pp 796–801. https://doi.org/10.1145/2228360.2228504
38. Wallace CS (1964) A Suggestion for a Fast Multiplier. IEEE Trans Electron Comput 13(1):14–17. https://doi.org/10.1109/PGEC.1964.263830
39. Zendegani R, Kamal M, Bahadori M, Afzali-Kusha A, Pedram M (2017) RoBA Multiplier: A Rounding-Based Approximate Multiplier for High-Speed yet Energy-Efficient Digital Signal Processing. IEEE Trans Very Large Scale Integr Syst 25(2):393–401. https://doi.org/10.1109/TVLSI.2016.2587696
40. Zhang Q, Wang T, Tian Y, Yuan F, Xu Q (2015) Approx ANN: An approximate computing framework for artificial neural network. In: Proc. Design Automation & Test in Europe Conference & Exhibition (DATE), pp 701–706

Shravani Chandaka received her M.tech in Digital Electronics and Communication Engineering from Chaitanya Engineering College, Visakhapatnam, Andhra Pradesh and India. She is working as an IT Application Developer in Legato Health Technologies, Bangalore, Karnataka, India. Her research interests are Evolvable Hardware, VLSI Architectures, Image Processing, and Partial Reconfiguration.

Balaji Narayanam received his Ph.D. degree from Andhra University, Visakhapatnam, Andhra Pradesh, India. He is working as a Professor in the Department of Electronics and Communication Engineering, Jawaharlal Nehru Technological University, Kakinada, Andhra Pradesh, India. His research interests are VLSI signal processing, Biomedical Image processing, Neural networks, and Partial Reconfiguration.