Two Efficient Approximate Unsigned Multipliers by Developing New Configuration for Approximate 4:2 Compressors

Ladan Sayadi, Somayeh Timarchi, Akbar Sheikh-Akbari

***Abstract*— Approximate computing is a promising approach for reducing power consumption and design complexity in applications that accuracy is not a crucial factor. Approximate multipliers are commonly used in error-tolerant applications. This paper presents three approximate 4:2 compressors and two approximate multiplier designs, aiming at reducing the area and power consumption, while maintaining acceptable accuracy. The paper seeks to develop approximate compressors that align positive and negative approximations for input patterns that have the same probability. Additionally, the proposed compressors are utilized to construct approximate multipliers for distinct columns of partial products based on the input probabilities of the two compressors in adjacent columns. The proposed approximate multipliers are synthesized using the 28nm technology. Compared to the exact multiplier, the first proposed multiplier improves power×delay and area×power by 91% and 86%, respectively, while the second proposed multiplier improves the two parameters by 90% and 84%, respectively. The performance of the proposed approximate methods was assessed and compared with the existing methods for image multiplication, sharpening, smoothing and edge detection. Also, the performance of the proposed multipliers in the hardware implementation of the neural network was investigated, and the simulation results indicate that the proposed multipliers have appropriate accuracy in these applications.**

***Index Terms*—Approximate Computing, Multiplier, 4:2 Compressor, Low-Power Design, Image Processing**.

I. Introduction

Minimizing power consumption has become a significant challenge for preserving lifetime and reliability due to the fast increase of integrated circuits' density. In addition, power consumption is a major consideration for portable devices [1]. Approximate computing units has demonstrated to be a promising and viable approach in achieving low-power design [5]-[7]. Approximate computing can be employed in error-tolerable applications in which the occurrence of inaccuracy does not affect the results, e.g, multimedia processing [33]-[34], machine learning [2]-[3], and data mining [4]. These applications have intensive multiplication operations. Multiplier is one of the basic circuits in processors and is known as a power-hungry unit [4]. A multiplier involves three steps:

1) partial product generation
2) partial product reduction
3) final addition

Depending on which step or steps are approximated, various categories of approximate multipliers exist. Some schemes are focused on simplification of partial products generation [8], [9], [30], [31]. In [8], the operands are split into two equal-sized parts: multiplication and non-multiplication parts. In the multiplication part, multiplication is performed conventionally, while in non-multiplication part, the operation is performed from the most to the least significant bits. If at least one bit equals '1', all other bits of output are set to '1'. In [9], the authors introduced the so-called dynamic range unbiased multiplier. Their method first identifis the most significant '1' using a leading '1' detector block, and then a subsequent *k*-bit is considered as operand. Ref. [30] presented a truncation-based Booth multiplier with a compensation block, which has a variable truncation factor. By simplification and approximation of the partial product generation, [31] developed an approximate hybrid high-radix encoding. In the method, the most significant bits (MSBs) of the multiplicand are encoded using radix-4 encoding, while the *k* least significant bits (LSBs) are encoded by radix-2^k.

The partial product reduction is known as the step that contributes more area and power consumption than others [11]. Significant research on reducing the hardware costs associated with the second multiplication step has been reported in the literature and it has been shown that using compressors to create more efficient circuitry is one of the viable options [18]. In [10]-[21], approximate compressors for partial products reduction were reported, where the proposed methods achieved different degrees of accuracy and cost reduction.

Other methods for partial product reduction were reported in [22] and [23]. In [22], an approximate multiplier, also known as a broken-array multiplier, was reported. Ref. [29] provided approximate adders that is used for additions required in multiplication for DSP applications

In [23], an approximate multiplier was introduced which designed approximate adders by cutting the carry propagation chain and incorporating a parallel error recovery section,

There are also other approaches for designing approximate multipliers as presented in [24]-[27], and [32]. Approximate booth encoding is another approximation method presented in [24] and [25]. In [26], inexact adders were used to compute the

Ladan Sayadi and Somayeh Timarchi are with the Faculty of Electrical Engineering, Shahid Beheshti University, Tehran 1983963113, Iran (l.sayadi, s_timarchi@sbu.ac.ir). Akbar Sheikh-Akbari is with Leeds Beckett University, Leeds, U.K. (A.Sheikh-Akbati @leedsbeckett.ac.uk).:

summation of the mantissa using logarithmic multipliers. In [32], an approximate logarithmic multiplier was presented by truncating the operation and compensating the error of adders. Performing multiplication using a shifter, adder, and subtractor was proposed in [27]. To further reduce the power consumption and design complexity, this paper proposes three approximate 4:2 compressors and two approximation multipliers. The contributions of this paper are as follow:

- Proposing three approximate 4:2 compressors.
    - Analyzing the probabilities of the compressor's output for stage one of partial product reduction.
    - Presenting a new technique for designing an approximate compressor, which is based on using equal positive and negative approximation for compressor's input patterns that have equal probabilities.
- Proposing two new approximate unsigned multipliers.
    - Describing how the compressors in the adjacent columns of multiplier's structure affects error performance by examining the probabilities.
    - Proposing a new method for utilizing appropriate compressors for different columns of approximate multiplier to reduce the error.

The approximate multipliers were described in VHDL for 8-bit operands and synthesized by design compiler (DC) tool with 28 nm TSMC technology. The accuracy and quality metrics were computed using MATLAB software. The two proposed multipliers were used for image multiplication, image sharpening, smoothing, and edge detection. It is also essential to consider both hardware performance and quality parameters for a fair comparison. So, in this paper, a Figure of Merit (FOM) was used which considers both quality and efficiency factors. The use of the FOM assures that there is a satisfactory trade-off between quality and efficiency.

The rest of the paper is organized as follows: Section II explains exact 4:2 compressors and briefly reviews previous approximate 4:2 compressors. Section III introduces the designing technique and proposed schemes in details, including the proposed compressor and multiplier structures. An evaluation and comparison of the two proposed multipliers with the state-of-the-art methods are presented in Section IV. The evaluation section includes the implementation parameters, accuracy metrics, and results of image multiplication sharpening, smoothing, edge detection and neural network application which includes multi-layer perceptron (MLP) neural network and a convolution neural network (CNN). Ultimately, Section V concludes the paper.

II. Background

Compressors are widely used in multipliers to speed up the partial product reduction process. They come in a variety of topologies, including 7:2, 5:2, and 4:2. However, the 4:2 compressor is more common. A conventional implementation of the exact 4:2 compressor comprises cascaded full adders, as shown in Fig. 1 [11] and its outputs are calculated by (1), (2), and (3).

$$sum = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus c_{in} \tag{1}$$

$$carry = (x_1 \oplus x_2 \oplus x_3 \oplus x_4) \cdot c_{in} + \overline{(x_1 \oplus x_2 \oplus x_3 \oplus x_4)} \cdot x_4 \tag{2}$$

$$C_{out} = (x_1 \oplus x_2) \cdot x_3 + \overline{(x_1 \oplus x_2)} \cdot x_1 \tag{3}$$

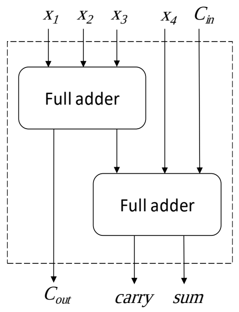


Fig. 1. Structure of The Exact 4:2 Compressor [11].

where C_{in}, C_{out}, and *sum* are input carry, output carry and sum of a 4:2 compressor, respectively. C_{in} is taken from the compressor in the lower significant position, whereas C_{out} goes to the compressor in the higher significant position.

Several efficient approximate 4:2 compressors designs aiming at reducing the complexity of exact compressor have been reported in the literature [10], [12]-[23]. In [12], two approximate compressors were proposed where the first one is based on inserting errors in the truth-table of the exact 4:2 compressor and the second one is based on ignoring the C_{in} and C_{out}. The second method was further expanded by other researchers.

An approximate 4:2 compressor made of a single OR gate was introduced in [10]. In [13], authors presented an approximate 4:2 compressor, which produces a 21% error rate in the truth table. This method was then extended to design a 5:2 compressor. Four reconfigurable dual-quality compressors with the ability to operate in either precise or approximate mode were introduced in [14], where each of their unused circuit modes are turned off using a power gating technique. In [15], partial products are converted into more likely and unlikely terms and then an approximate 4:2 compressor, a full adder and a half adder were proposed based on this concept. In [16], the majority function was used to define *carry*, while the *sum* is constant and equal to '1', and FinFET was used to build the approximate 4:2 compressor in transistor level. In [17], an approximate 4:2 compressor and an error recovery module for compensating generated negative errors was introduced. In

[18], the authors presented two approximate 4:2 compressors, where one of them is a modified dual-stage compressor and its designed truth table contains equal number of +1 and -1 errors. In [19], an approximate compressor based on the stacking circuit concept was reported. In [20], three approximate compressors were developed by designing just *sum* output, which results in generating negative error distances in the truth table. So, an error-correcting module was employed for compensation of the negative approximation. In [21], the authors reported the design of two structure 4×4 multiplier with an approximate 4:2 compressor through preprocessing partial products, where larger multipliers were constructed by the given 4×4 multiplier.

III. PROPOSED SCHEMES

The proposed scheme consists of four sections. The first two sections describe the proposed design technique of compressors and three approximate 4:2 compressors based on the proposed design technique. In the next two sections, a new approach for multiplier structure as well as two new approximate unsigned multiplier structures are presented.

A. Proposed technique for designing approximate compressors

This section attempts to analyze the parameters that influence the accuracy of an approximate compressor and then proposes an approach to explore a new design. The efficiency of an approximate compressor can be analyzed by applying it to a multiplier structure. Fig. 2 is commonly used to represent an 8-bit unsigned multiplier. Two stages are depicted in Fig. 2: stage 1 and stage 2.

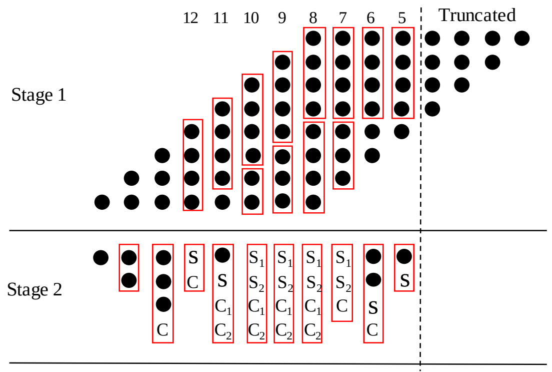


Fig. 2. Overall structure of proposed an 8-bit multiplier.

Assuming uniform distribution for input operands of the multiplier, the probability of compressors input in stage 1 can be calculated. However, it is not clear for compressors in stage 2, because it is contingent on the approximate compressor's structure. So, this section focuses on the probability of the output derived from stage 1.

Let's assume P_0 be the probability that the compressor output is exact also, P_+ and P_- signify the probabilities of compressor output with positive and negative approximation, respectively. In this scenario, placing two compressors in one column, for example in columns 7 through 10, results in three different Error Distance (ED) with absolute values of 0, 1, and 2, where ED is the difference between the correct and incorrect output. Fig. 3 summarizes all conceivable output states' combinations of two compressors in one column, and their corresponding absolute ED.

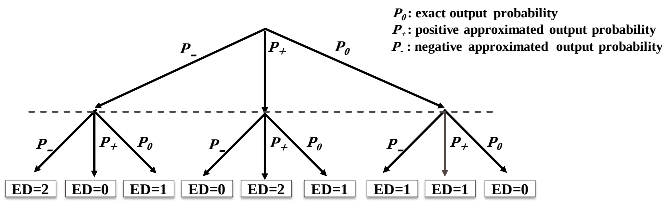

P_0: exact output probability
P_+: positive approximated output probability
P_-: negative approximated output probability
P_-
P_+
P_0
ED=2
ED=0
ED=1
ED=0
ED=2
ED=1
ED=1
ED=1
ED=0

Fig. 3. States of absolute ED resulting of two compressors in one column at stage1.

According to the EDs, shown at the bottom of the Fig. 3, under three conditions, the sum of EDs of two compressors in the same column becomes zero. One of the conditions is when both compressor's outputs are exact, where the probability of this condition equals $P_0 \times P_0 = P_0^2$. Besides, if the probability of the first (and the second) compressor's outputs has opposite signs (P_+ and P_-), the generated ED will be zero with the probability of $2 \times (P_+ \times P_-)$. Here, the probability of the correct output for a column at stage 1 is given by $P\,(\text{ED}=0)$, which equals the sum of all the three conditions' probabilities, where ED is zero. Equation (4) defines $\sum P(\text{ED} = 0)$. According to the probability concepts, (5) is also established.

$$\sum P(\text{ED} = 0) = P_0{}^2 + 2 \times (P_+ \times P_-) \tag{4}$$

$$P_0 + P_+ + P_- = 1 \tag{5}$$

A well-designed approximate multiplier maximizes $\sum P(\text{ED} = 0)$. This means that P_0 must be as high as possible or $P_+ = P_-$ in order to maximize $\sum P(\text{ED} = 0)$.

Let's assume that the probability of a positive or negative outcome is denoted using the notation p_{i+} and p_{i-} for each of the 16 input patterns of 4:2 compressor. The ED for each pattern is represented by ED_i. So, the positive and negative approximation probabilities of the compressor output can be determined using equation (6) and (7), respectively. Equations (6) and (7) must be equal to set the condition $P_+ = P_-$.

$$\sum (p_{i+} \times ED_i) = P_+ \tag{6}$$

$$\sum (p_{i-} \times ED_i) = P_- \tag{7}$$

Consequently, it is necessary to know the probability of the compressor's inputs to reduce the multiplication error. The compressor's inputs at stage 1 are partial products, which are generated by an AND gate. By assuming a uniform distribution of the multiplier's inputs, the probability of the partial product to be '0' or '1' is 3/4 and 1/4, respectively. For example, the probability of '0001' is $(\frac{3}{4} \times \frac{3}{4} \times \frac{3}{4} \times \frac{1}{4} = \frac{27}{256})$. A categorization based on the probability of each of the 16 input patterns labeled as $x_1x_2x_3x_4$ is illustrated in Table I. The input patterns that have the same compressor's output are grouped together, as explained in [20].

TABLE I
PROBABILITY OF OCCURRENCE OF INPUTS AND GROUPING OF INPUTS

Grouping No.	**$x_1x_2x_3x_4$**	**Probability of each input in the group**
first	0000	81/256
second	0001,0010,0100,1000	27/256
third	0011,1100,0101,1010,0110,1001	9/256
fourth	0111,1110,1011,1101	3/256
fifth	1111	1/256

Since each group of inputs has the same probability, $P_+=P_-$ can be established by considering of equal positive and negative ED_i in each input group. So, an equal negative and positive approximations to each input group in the proposed approximate 4:2 compressor will be applied. Further information about the importance of using both positive and negative approximation in the truth table for compressors development can be found in [13] and [18]. However, the AND gates in stage 1 does not generate equal probability for the compressor inputs, as explained in [18]. The probabilities of the input are not equal, as shown in Table I, which is in contrast to [18].

B. *Three proposed approximate compressors*

The proposed schemes do not include the C_{in} and C_{out}, similar to existing approximate compressor. The proposed compressors are categorized according to the number of gates that they have. They are proposed as follows:

1) Approximate Compressors with 6 Gate (AC6G)
2) Approximate Compressors Free Gate I (ACFGI)
3) Approximate Compressor Free Gate II (ACFG II)

I. AC6G design

An important design consideration for this compressor is assuming that $P_+=P_-$ as mentioned in section A. The compressors in the same category of the proposed AC6G design are listed in Table II. Using input permutation, 16 compressors in this category can be created. The compressors are denoted by the notation AC6G-n, where n is the compressor number.

As an example, AC6G-12 Karnaugh map has been shown in Fig. 4, where Fig 4a and b show Karnaugh map of output sum and carry, respectively and Fig. 4c shows the used approximation for input patterns. The first to fifth groups of the inputs are shown in black, green, blue, red, and yellow colour, respectively. Seven input patterns from third and fifth group are approximated. $P_+=P_-$ is set for the third group of the inputs, while the fifth group is approximated due to the absence of the C_{in} and C_{out}. The compressors in AC6G category have the highest accuracy compared with the two other proposed category due to the well-established condition $P_+= P_-$. According to (2), Critical Path Delay (CPD) of exact compressor is "$2t_{XOR}+t_{AND}+t_{OR}$", where t_{XOR}, t_{AND} and t_{OR} is the delay of 2-input XOR, AND and OR gates, respectively. CPD of AC6Gs equals to "$2\times t_{OR}+t_{AND}$".

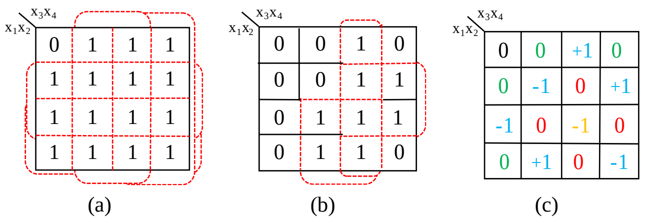


Fig. 4. Karnaugh map for AC6G-12. a) sum output. b) carry output. c) the approximation used for input patterns.

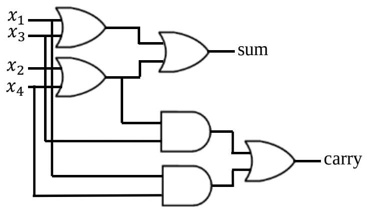


Fig. 5. The gate schematic of the AC6G-12

TABLE II
OUTPUTS OF ALL ACF6G COMPRESSORS

Compressors No.	**sum**	**carry**
AC6G-1	$(x_1+x_2)+(x_3+x_4)$	$(x_1\cdot(x_3+x_4))+(x_2\cdot x_3)$
AC6G-2	$(x_1+x_2)+(x_3+x_4)$	$(x_1\cdot(x_3+x_4))+(x_2\cdot x_4)$
AC6G-3	$(x_1+x_2)+(x_3+x_4)$	$(x_1\cdot(x_3+x_4))+(x_3\cdot x_4)$
AC6G-4	$(x_1+x_2)+(x_3+x_4)$	$(x_2\cdot(x_3+x_4))+(x_1\cdot x_3)$
AC6G-5	$(x_1+x_2)+(x_3+x_4)$	$(x_2\cdot(x_3+x_4))+(x_1\cdot x_4)$
AC6G-6	$(x_1+x_2)+(x_3+x_4)$	$(x_2\cdot(x_3+x_4))+(x_3\cdot x_4)$
AC6G-7	$(x_1+x_2)+(x_3+x_4)$	$(x_3\cdot(x_1+x_2))+(x_1\cdot x_2)$
AC6G-8	$(x_1+x_2)+(x_3+x_4)$	$(x_4\cdot(x_1+x_2))+(x_1\cdot x_2)$
AC6G-9	$(x_1+x_3)+(x_2+x_4)$	$(x_1\cdot(x_2+x_4))+(x_2\cdot x_3)$
AC6G-10	$(x_1+x_3)+(x_2+x_4)$	$(x_1\cdot(x_2+x_4))+(x_3\cdot x_4)$
AC6G-11	$(x_1+x_3)+(x_2+x_4)$	$(x_3\cdot(x_2+x_4))+(x_1\cdot x_2)$
AC6G-12	$(x_1+x_3)+(x_2+x_4)$	$(x_3\cdot(x_2+x_4))+(x_1\cdot x_4)$
AC6G-13	$(x_1+x_4)+(x_2+x_3)$	$(x_1\cdot(x_2+x_3))+(x_2\cdot x_4)$
AC6G-14	$(x_1+x_4)+(x_2+x_3)$	$(x_1\cdot(x_2+x_3))+(x_3\cdot x_4)$
AC6G-15	$(x_1+x_4)+(x_2+x_3)$	$(x_4\cdot(x_2+x_3))+(x_1\cdot x_2)$
AC6G-16	$(x_1+x_4)+(x_2+x_3)$	$(x_4\cdot(x_2+x_3))+(x_1\cdot x_3)$

This paper focuses on the design of compressors with nearly equal positive and negative approximations for the input groups. There is a trade-off between designing an efficient hardware and verifying $P_+=P_-$. Since reducing the cost of the hardware is a fundamental design objective, it is aimed to achieve a free gate compressor, where condition $P_+=P_-$ is met for as many inputs as possible for the two proposed compressors, named: ACFGI and ACFGII.

II. ACFGI design

The ACFGI outputs' equations are tabulated in Table III. *Carry* output corresponds to one of the inputs, whereas its *sum* output is equal to '1'. Four distinct compressor designs can be generated from the information tabulated in Table III by permutation input.

As an example, the Karnaugh map of the ACFGI-4 is shown in Fig. 6.

TABLE II
OUTPUTS OF ALL ACFGI COMPRESSORS

Compressors No.	sum	carry
ACFGI-1	1	x_1
ACFGI-2	1	x_2
ACFGI-3	1	x_3
ACFGI-4	1	x_4

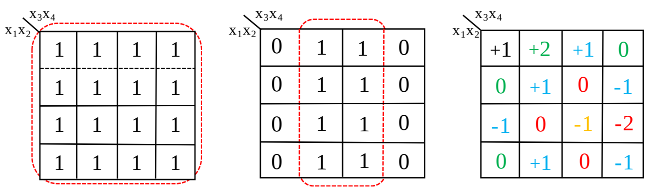


Fig. 6. Karnaugh map of ACFGI-4. a) sum output. b) carry output. c) the approximation used for the input patterns.

In the design of this compressor, achieving low hardware is a priority, so $P_+=P_-$ is confirmed only for the third input group. Despite its lack of high accuracy, this compressor is the most efficient candidate for low-cost implementation. Since *sum* output is a constant value of '1', the final addition step in the multiplier is simplified. This compressor is appropriate for applying to the least significant partial product columns.

III. ACFGII design

The output equations of the ACFGII designs are tabulated in Table IV. From this table, it can be seen that twelve distinct compressors with a slightly different set of input permutations, can be developed. However, one of the possible compressors have been reported in [14]. Hence, this paper explores all possible ACFGII compressors and their use in multipliers.

TABLE IV
OUTPUTS OF ALL ACFGII COMPRESSORS

Compressors No.	sum	carry
ACFGII-1	x_1	x_2
ACFGII-2	x_1	x_3
ACFGII-3	x_1	x_4
ACFGII-4	x_2	x_1
ACFGII-5	x_2	x_3
ACFGII-6	x_2	x_4
ACFGII-7	x_3	x_1
ACFGII-8	x_3	x_2
ACFGII-9	x_3	x_4
ACFGII-10	x_4	x_1
ACFGII-11	x_4	x_2
ACFGII-12	x_4	x_3

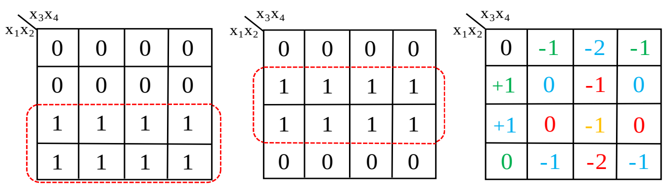


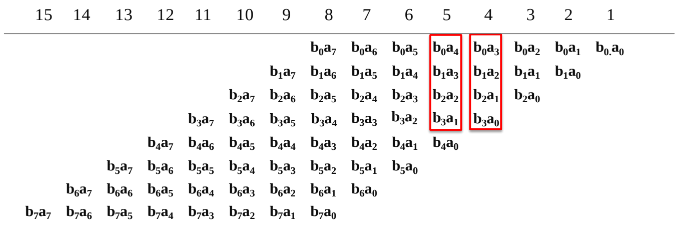
Fig. 7. Karnaugh map for ACFGII-1. a) sum output. b) carry output. c) approximation used for input patterns

ACFGII-1's compressor's Karnaugh map is shown in Fig. 7. From Fig. 7(a) and (b), the *sum* and *carry* have equal ones in their Karnaugh maps. From Fig. 7(c), the ACFGII-1 represents ten inputs' approximations, where the input '0000' has the highest probability values and it is not approximated. Therefore, ACFGII generates higher performance in terms of accuracy in comparison to ACFGI.

C. *Proposed technique for designing approximate multiplier*

In this paper different compressors for adjacent partial product column are used. The input probability of the two compressors in the two columns of the partial products is first studied. The proposed design approach for creating the multiplier's structure is then explained.

Fig. 8 shows the first stage of the 8-bit multiplier with its two operands: A and B. Two compressors in columns 4 and 5 have been highlighted by red color to analyze their behavior as an example of adjacent compressors.



15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
b0a7 b0a6 b0a5 b0a4 b0a3 b0a2 b0a1 b0a0
b1a7 b1a6 b1a5 b1a4 b1a3 b1a2 b1a1 b1a0
b2a7 b2a6 b2a5 b2a4 b2a3 b2a2 b2a1 b2a0
b3a7 b3a6 b3a5 b3a4 b3a3 b3a2 b3a1 b3a0
b4a7 b4a6 b4a5 b4a4 b4a3 b4a2 b4a1 b4a0
b5a7 b5a6 b5a5 b5a4 b5a3 b5a2 b5a1 b5a0
b6a7 b6a6 b6a5 b6a4 b6a3 b6a2 b6a1 b6a0
b7a7 b7a6 b7a5 b7a4 b7a3 b7a2 b7a1 b7a0

Fig. 8. Stage 1 of the 8-bit multiplier.

Regardless of their operand's lengths, there is a relation between two adjacent columns in multipliers, e.g., if both b_0a_3 and b_1a_2 in column 4 are equal to '1', it implies that both a_3 and b_1 have value of 1. So, it is reasonable to assume that b_1a_3 in column 5 must have value of '1'. Hence, it can be concluded that the inputs of the compressor in neighboring columns are dependent.

Let's assume having two compressors, one in column *i* and another in column *i*+1. If one of the 16 input patterns occurs in column *i*, the probability of some inputs occurring in column *i*+1 is zero. Additionally, the probability of some of the input patterns is higher than others due to their mentioned dependencies. As a result, if one of the input patterns occurs in column *i*, the probabilities cannot be obtained using given information in Table I. Hence, the conditional probabilities for patterns occurrences have been calculated from the multiplier simulation using MATLAB software and tabulated in Table V.

For instance, if the compressor input in column *i* is "0001", then P(*i*+1|*i*=0001) gives the probability of all possible input patterns in column i+1. Based on presented information in Table V, if an input pattern from the *n*th ($1\leq n\leq 5$) input group occurs in column *i* of a compressor, the same input pattern is most probably among inputs of the *n*th group in column *i*+1. For example, if the compressor input in column *i* is equal to group belongs to '0001'. It worth to mention that Table I represents probabilities of all input patterns in a group that have the same probability, while Table V represents, conditional occurrence probabilities of the input patterns and if a pattern occurs in column *i*, the groups do not have the same probability in column *i*+1. '0001' (an input from the second group), then in

TABLE V
OCCURRENCE PROBABILITY OF COLUMN I+1 INPUTS ACCORDING TO COLUMN I INPUTS

Probabilities	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
P(i+1|i=0000)	$\frac{178}{324}$	$\frac{42}{324}$	$\frac{32}{324}$	0	$\frac{30}{324}$	$\frac{6}{324}$	0	0	$\frac{26}{324}$	$\frac{6}{324}$	$\frac{4}{324}$	0	0	0	0	0
P(i+1|i=0001)	$\frac{26}{108}$	$\frac{26}{108}$	$\frac{16}{108}$	$\frac{16}{108}$	$\frac{6}{108}$	$\frac{6}{108}$	0	0	$\frac{4}{108}$	$\frac{4}{108}$	0	$\frac{2}{108}$	0	0	0	0
P(i+1|i=0010)	$\frac{30}{108}$	$\frac{10}{108}$	$\frac{22}{108}$	0	$\frac{18}{108}$	$\frac{6}{108}$	$\frac{12}{108}$	0	$\frac{6}{108}$	0	$\frac{4}{108}$	0	0	0	0	0
P(i+1|i=0011)	0	0	$\frac{10}{36}$	$\frac{10}{36}$	0	0	$\frac{6}{36}$	$\frac{6}{36}$	0	0	$\frac{2}{36}$	$\frac{2}{36}$	0	0	0	0
P(i+1|i=0100)	$\frac{32}{108}$	$\frac{8}{108}$	$\frac{8}{108}$	0	$\frac{22}{108}$	$\frac{4}{108}$	0	0	$\frac{16}{108}$	$\frac{4}{108}$	$\frac{4}{108}$	0	$\frac{10}{108}$	0	0	0
P(i+1|i=0101)	$\frac{6}{36}$	$\frac{4}{36}$	$\frac{4}{36}$	$\frac{4}{36}$	$\frac{4}{36}$	$\frac{4}{36}$	0	0	$\frac{2}{36}$	$\frac{2}{36}$	$\frac{2}{36}$	$\frac{2}{36}$	0	$\frac{2}{36}$	0	0
P(i+1|i=0110)	0	0	0	0	$\frac{12}{36}$	$\frac{4}{36}$	$\frac{8}{36}$	0	0	0	0	0	$\frac{6}{36}$	$\frac{2}{36}$	$\frac{4}{36}$	0
P(i+1|i=0111)	0	0	0	0	0	0	$\frac{4}{12}$	$\frac{4}{12}$	0	0	0	0	0	0	$\frac{2}{12}$	$\frac{2}{12}$
P(i+1|i=1000)	$\frac{42}{108}$	$\frac{10}{108}$	$\frac{8}{108}$	0	$\frac{10}{108}$	$\frac{2}{108}$	0	0	$\frac{26}{108}$	$\frac{6}{108}$	$\frac{4}{108}$	0	0	0	0	0
P(i+1|i=1001)	$\frac{6}{36}$	$\frac{6}{36}$	$\frac{4}{36}$	$\frac{4}{36}$	$\frac{2}{36}$	$\frac{2}{36}$	0	0	$\frac{4}{36}$	$\frac{4}{36}$	$\frac{2}{36}$	$\frac{2}{36}$	0	0	0	0
P(i+1|i=1010)	$\frac{6}{36}$	$\frac{2}{36}$	$\frac{4}{36}$	0	$\frac{6}{36}$	$\frac{2}{36}$	$\frac{4}{36}$	0	$\frac{6}{36}$	$\frac{2}{36}$	$\frac{4}{36}$	0	0	0	0	0
P(i+1|i=1011)	0	0	$\frac{2}{12}$	$\frac{2}{12}$	0	0	$\frac{2}{12}$	$\frac{2}{12}$	0	0	$\frac{2}{12}$	$\frac{2}{12}$	0	0	0	0
P(i+1|i=1100)	0	0	0	0	0	0	0	0	$\frac{16}{36}$	$\frac{4}{36}$	$\frac{4}{36}$	0	$\frac{10}{36}$	$\frac{2}{36}$	0	0
P(i+1|i=1101)	0	0	0	0	0	0	0	0	$\frac{2}{12}$	$\frac{2}{12}$	$\frac{2}{12}$	$\frac{2}{12}$	$\frac{2}{12}$	$\frac{2}{12}$	0	0
P(i+1|i=1110)	0	0	0	0	0	0	0	0	0	0	0	0	$\frac{6}{12}$	$\frac{2}{12}$	$\frac{4}{12}$	0
P(i+1|i=1111)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\frac{2}{4}$	$\frac{2}{4}$

column *i*+1, the highest probability input compared to other inputs of second

The design principle deduced from Table V is that if the compressor input in column *i* is approximated, then that input in column *i*+1 must be calculated accurately or approximated by the opposite sign. It is possible to get close to approximations with opposite signs by using proposed compressors with input permutations for columns *i* and *i*+1.

The proposed multiplier structures utilize a range of compressors with different approximations following the result of Table V. Since the proposed design guideline is generic, a trial-and-error method is required to determine the appropriate position for the compressors to reduce the total error of multiplier.

Algorithm 1 describes the proposed method in multipliers implementation. It is also notable that since the process of creating partial products in a multiplier of any size is the same, the probabilities given in Table V for a multiplier of size *n* are true and proposed design can be easily generalized to *n*-bit multipliers. The first four columns of proposed multipliers are truncated. For proposed-mul1 and proposed-mul2, respectively, ACFGIs and ACFGIIs are utilized, while AC6Gs are used in the upper columns for both proposed multipliers. As a result, algorithm is started by figuring out which AC6Gs should be located in the upper columns then taking into account the interdependency between columns, appropriate compressors for the middle columns are chosen. To begin, exact compressors are arranged in a multiplier structure, and then the compressor for column 11 is chosen by swapping in all 16 compressors and comparing their NMED parameters to get the lowest value. The same process is repeated for other upper column. The aforementioned procedure has been also used to establish the best arrangement of compressors for the middle columns.

For general *n*-bit multiplier, similar to the proposed 8-bit multiplier described above, the columns are partitioned into three sections. The truncated section includes column 1 to n/2. The middle columns are ((n/2)+1) to ((3n/2)-1) and the upper columns include column (3n/2) to (2n-3).

D. Two Proposed approximate 8-bit multipliers' structures

This section describes the structures of the two proposed approximate multipliers, called proposed-mul1 and proposed-mul2. In the multipliers, four least significant columns of partial products are eliminated. The AC6Gs are used for eleventh to thirteenth column because of their great precision. The ACFGIs and ACFGIIs are used for columns 5 to 10 in the proposed-mul1 and proposed-mul2.

Algorithm 1: specifying approximate compressors for different columns of approximate multiplier

```
Let's assume
        First, all of compressor (i,j) are exact
        i is number of stage and j is number of column
        k is the type of compressor
        N is the operand length
output: Assigning appropriate approximate compressors for compressor (i,j)
--specifying appropriate AC6G compressors for upper columns
1:  for i in [stage1, stage2, …,stage($\log_2 n - 1$)]
2:      for j in [column ($\frac{2\times(2n-1)}{3} + 1$) to column ($2n - 3$)]
3:          for k in [AC6G_1 to AC6G_16]
4:              compressor(i,j) ← k
5:              if NMED(k) < NMED (k-1)
6:                  compressor(i,j) == k
7:              else
8:                  compressor(i,j) == k-1
9:              end if
10:         end for
11:     end for
12: end for
--specifying appropriate ACFGI and ACFGII compressors for middle columns--
13: for i in [stage1, stage2, …,stage($\log_2 n - 1$)]
14:     for j in [column ($\frac{n}{2} + 1$) to column ($\frac{2\times(2n-1)}{3}$)]
15:         for k in [ACFGI_1 to ACFGI_4]     --proposed-mul1
15:         (for k in [ACFGII_1 to ACFGII_12]) --proposed-mul2
16:             compressor(i,j) ← k
17:             if NMED(k) < NMED (k-1)
18:                 compressor(i,j) == k
19:             else
20:                 compressor(i,j) == k-1
21:             end if
22:         end for
23:     end for
24: end for
```

The proposed-mul1 structure is depicted in Fig. 9. The compressors are numbered in Fig. 9, which are according to Table III for ACFGI and Table II for AC6G compressors.

Since *sum* output of ACFGI compressors is constant and equals to '1' in the middle columns, one of the final output vectors is equal to '1'. Therefore, modified and simpler full-adder and half-adder are used for sum of the two final vectors as shown in Fig. 9. In [16], the structure of full adder and half adder, with a constant '1' as input have been proposed. According to [16], the implementation of the half adder only requires one NOT gate, and the implementation of the full adder only requires one XOR and one OR gate. According to Fig. 9, CPD of proposed-mul1 is "t_{AND}+2t_{AC6G}+5t_{FA}" which t_{AND}, t_{AC6G} and t_{FA} refers to delay of partial product generation step and delay of AC6G compressor and delay of full-adder, respectively.

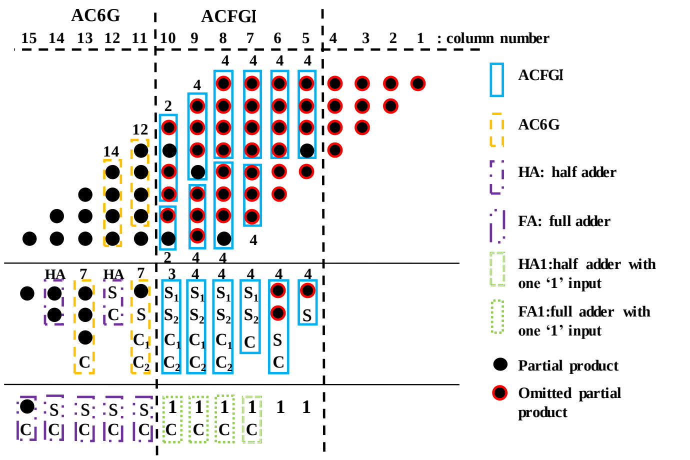

AC6G
ACFGI
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 : column number
ACFGI
AC6G
HA: half adder
FA: full adder
HA1:half adder with one '1' input
FA1:full adder with one '1' input
Partial product
Omitted partial product

Fig. 9. Structure of the proposed-mul1, which consists of ACFGI and ac6g whose numbers according to table III and table II.

From Fig. 9, it can be seen that the partial products are shown with a black dot covered by a red circle symbol, which are omitted in the proposed-mul1 approximate multiplier based on the definition of the ACFGI, see Table III. Elimination of partial products has a significant impact on the hardware optimization. Some of its compressors have less than four inputs. Although earlier work has utilized exact half-adder and full-adder instead of approximate compressors with fewer inputs, the proposed multiplier uses approximate compressors with one or two inputs equal to '0' to achieve an efficient hardware multiplier.

Fig. 10 depicts the proposed-mul2 structure. ACFGII is used for the middle columns, which are numbered according to Table IV. According to Fig. 10, CPD of proposed-mul2 is "t_{AND} + t_{HA} + 9t_{FA}".

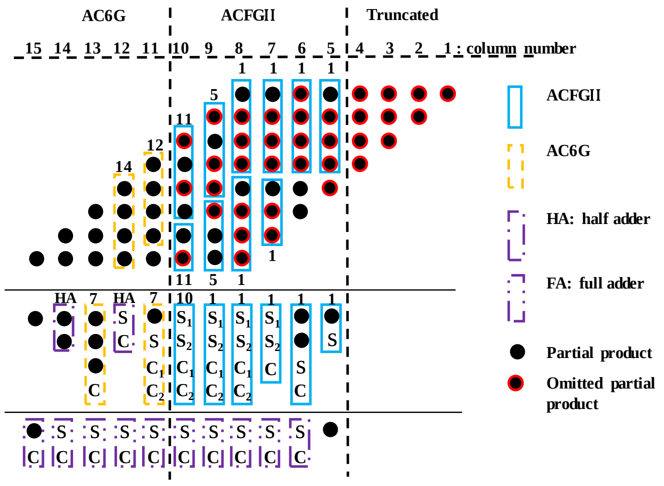


Fig. 10. Structure of Proposed-Mul2 Which Consists of ACFGII and AC6G Whose Numbers According to Table IV And Table II

From Fig. 10, it can be seen that many bits of the partial products have not been used and included in the computations. These bits are highlighted by black dots with a red circle and labeled as the omitted partial product bits. Like proposed-mul1 method, the proposed-mul2 uses AC6G in its columns 11 to 13.

IV. EVALUATION AND COMPARISON

In this section, the proposed-mul1 and proposed-mul2 are evaluated and compared to previous studies. To demonstrate the accuracy and implementation metrics of the proposed-mul1 and 2, a FOM is introduced, which could highlight the tradeoff between precision and hardware efficiency of the designs. Moreover, to assess and verify the efficiency of the proposed designs, they were evaluated via image multiplication, sharpening, smoothing and edge detection.

A. Hardware analysis

Table VI shows the synthesized results for circuit area, critical path delay, power consumption and the power-delay product (PDP) of the proposed-mul1, proposed-mul2, Mul_1[13], Mul_2[13], Multiplier1[15], M8_1[21], M8_2[21], Majority_based[16], C_full[19], M8[18], ALM_SOA[26] and Exact designs. In this table, the delays are the minimum delay at which the circuits can be synthesized. The simulation condition is typical (25˚*C* and supply voltage is 0.9V). All designs were described by VHDL (VHSIC Hardware Description Language). Implementation metrics were synthesized by the Synopsys Design Compiler (DC) using 28nm TSMC technology.

From the results presented in Table VI, it is obvious that both proposed-mul1 and proposed-mul2 designs have superior implementation metrics than other existing designs. This is due to the simplicity of the proposed partial product generation and reduction method that they use.

TABLE VI
HARDWARE RESULTS OF 8-BIT APPROXIMATE MULTIPLIERS

Multipliers	Delay (ns)	Power (uw)	Area (um^2)	PDP (fj)	ADP
Mul_1[13]	0.13	163.4	578.9	21.2	75.3
Mul_2[13]	0.13	155.8	554.0	20.3	72
Multiplier1[15]	0.15	247.7	845.8	37.1	126.9
M8_1[21]	0.18	217.3	688.7	39.1	124
M8_2[21]	0.16	208.9	646.6	33.4	103.5
Majority_based[16]	0.15	148.4	580.1	22.3	87
C_full[19]	0.21	275.2	804.0	57.8	168.8
M8[18]	0.22	283.5	829.8	62.4	182.6
ALM_SOA(M=5)[26]	0.14	158.4	618.1	22.1	86.53
Proposed-mul1	0.09	76.1	331.4	6.8	29.8
Proposed-mul2	0.10	73.4	318.5	7.3	34.4
Exact	0.24	313.1	922.2	75.1	221.3

From Table VI, it can be seen that Mul_2 [13], demonstrates the highest performance amongst existing methods reported in the literature and both proposed -mul1 and mul2 achieve higher performance than Mul_2[13], as follows.

Moreover, the proposed-mul1 outperforms the mul_2[13] in terms of the delay, power, area, PDP, and ADP by 31%, 53%, 43%, 67%, and 59%, respectively. In addition, the proposed-mul2 also demonstrates higher performance to that of Mul_2 [13] in terms of the delay, power, area, PDP, and ADP by 23%, 53%, 42%, 64%, and 52%, respectively.

B. accuracy analysis

To assess and compare the performance of the proposed approximate circuits with the existing methods, some assessment metrics such as: error rate (ER), mean relative error distance (MRED) and normalized mean error distance (NMED) are used [28]. The definition of these metrics are as follows:

Error rate (ED) metric is the difference between the exact and approximate output values, as shown in (9)

$$ED = |M - M'| \tag{9}$$

where *M*, *M'* denote the exact and the approximate output values and | | is the absolute value.

For an *n*-bit multiplier, the NMED can be calculated using (11). To calculate NMED, the Mean ED (MED) is first computed using (10). The NMED is the MED normalized by the maximum output of the exact design. The gap between exact and approximate outputs is more substantial than their relative differences in many approximate applications involving the human senses [16].

$$MED = \frac{1}{2^{2N}} \sum_{i=1}^{2^{2N}} ED_i \tag{10}$$

$$NMED = \frac{1}{2^{2N}(2^N - 1)^2} \sum_{i=1}^{2^{2N}} ED_i \tag{11}$$

The MRED metric represents the average relative error and can be calculated using (12):

$$MRED = \frac{1}{2^{2N}} \sum_{i=1}^{2^{2N}} \frac{ED_i}{M_i} \tag{12}$$

where M_i is the exact value of the multiplication.

The ER represents the percentage of multiplications for which the approximate design differs from its exact design counterpart

MRED, ER, NMED and maximum ED metrics were calculated for 8-bit multipliers by simulating all 65536 input patterns using MATLAB software and tabulated in Table VII.

TABLE VII
ACCURACY RESULTS OF 8-BIT APPROXIMATE MULTIPLIERS

Multipliers	ER(%)	NMED	MRED	Max ED
Mul_1[13]	93.50	0.019	0.088	9620
Mul_2[13]	93.48	0.018	0.082	8056
Multiplier1[15]	81.79	0.025	0.079	4096
M8_1[21]	72.59	0.019	0.060	6936
M8_2[21]	72.60	0.028	0.083	1156
Majority_based [16]	99.81	0.007	0.438	1950
C_full[19]	19.02	0.003	0.007	8264
M8[18]	82.61	0.002	0.041	568
ALM_SOA(M=5)	98.80	0.013	0.055	7670
Proposed-mul1	99.93	0.018	0.509	7120
Proposed-mul2	98.86	0.017	0.151	7148

From Table VII it can be seen that M8 [18] demonstrates the lowest NMED. It can be explained by the fact that the M8 uses exact compressors for most significant columns of its multiplier.

The C_full [19] method's MRED and ER values are the lowest amongst all methods. This can be explained by the fact that there is no truncation in its design.

TABLE IX

THE PSNRS AND SSIM FOR IMAGE MULTIPLICATION OF 8-BIT APPROXIMATE MULTIPLIERS

Multipliers	Lake × bridge		Peppers-color × Peppers-color		Lena × female		Peppers × fruits		Female × tree		Tank × truck		House × tree		mandril × mandril		Lake-color × airplane	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Mul_1[13]	30.48	0.918	34.01	0.866	31.39	0.835	30.24	0.827	29.94	0.927	36.15	0.929	29.97	0.915	28.64	0.930	29.76	0.844
Mul_2[13]	31.68	0.920	34.47	0.891	33.19	0.896	31.11	0.837	31.64	0.865	37.17	0.942	30.55	0.932	29.20	0.935	29.43	0.836
Multiplier1[15]	27.87	0.930	29.47	0.748	30.26	0.828	28.56	0.784	27.67	0.936	35.19	0.940	24.85	0.881	24.60	0.889	23.20	0.869
M8_1[21]	32.20	0.937	34.02	0.853	31.93	0.847	32.03	0.828	31.90	0.852	36.02	0.930	31.19	0.931	30.34	0.917	31.10	0.874
M8_2[21]	29.49	0.929	28.38	0.808	28.72	0.812	28.44	0.788	27.16	0.915	33.29	0.913	28.53	0.854	24.40	0.848	28.42	0.850
Majority_based[16]	41.13	0.977	41.37	0.970	40.17	0.971	40.10	0.967	40.85	0.969	41.10	0.958	40.90	0.970	39.51	0.988	40.42	0.974
C_full[19]	44.90	0.997	38.62	0.980	44.84	0.974	42.96	0.963	42.41	0.997	55.96	0.998	42.51	0.984	38.94	0.984	41.36	0.992
M8[18]	50.15	0.998	49.54	0.997	49.87	0.997	50.54	0.996	49.91	0.997	50.97	0.997	50.20	0.997	46.53	0.999	50.72	0.997
ALM_SOA(M=5) [26]	35.01	0.958	30.42	0.858	31.38	0.859	30.52	0.825	34.60	0.908	35.38	0.905	31.57	0.942	31.52	0.957	32.13	0.886
Proposed-mul1	32.72	0.922	33.44	0.917	32.92	0.904	32.62	0.856	31.90	0.870	36.16	0.913	31.08	0.910	31.49	0.950	30.97	0.871
Proposed-mul2	32.39	0.923	33.80	0.907	32.89	0.884	32.79	0.875	31.78	0.898	35.18	0.915	31.10	0.916	31.03	0.951	30.45	0.874

The proposed_mul1 demonstrates the highest ER and NMED as some of its compressors generate a non-zero output when their inputs are zero. The proposed-mul1 and proposed-mul2 have nearly similar NMED value compared to Mul_2[13].

The proposed Algorithm 1 was used to design the 16-bit approximate multiplier structure. The results of both proposed multipliers for 16-bit input operands are reported in Table VIII. The results are obtained with 10 million operands with uniform distribution. In comparison to 8-bit approximate multiplier, it is found that employing compressors according to Algorithm 1 is effective in reducing the amount of error metrics.

TABLE VIII

ACCURACY RESULTS OF 16-BIT APPROXIMATE MULTIPLIERS

Multipliers	**ER(%)**	**NMED**	**MRED**
Proposed-mul1	100	0.010	0.119
Proposed-mul2	99.98	0.009	0.066

C. Image multiplication, sharpening, smoothing and edge detection

The quality of the proposed multipliers is examined by applying them in fault-tolerant applications. The two proposed multipliers are used to multiply two images, sharpening, smoothing and edge detection as the essential operations in image processing. MATLAB programs is used to perform image applications using the approximate multipliers. The Peak Signal to Noise Ratio (PSNR) and Structural Similarity (SSIM) of the resulting image is used as a measure to assess the performance of the multipliers. The PSNR metric is defined as:

$$PSNR = 10\ log\ \left(\frac{m\times p\times {MAX_I}^2}{\sum_{i=0}^{m-1}\sum_{j=0}^{p-1}(I(i\cdot j)-k(i\cdot j))^2}\right) \qquad (13)$$

Where *m* and *p* are the dimensions of the images. MAX_I is the maximum possible value of image pixels and, *I(i,j)* and *k(i,j)* represents exact and approximate multiplication of two image pixel at location of *i, j*. According to [35], the SSIM is defined by (14).

$$SSIM = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)} \qquad (14)$$

In most image processing applications, 30 dB is considered as an acceptable value for the PSNR of the resulting images. As the difference between the exact and approximate is not distinguishable by the human eyes. The PSNR values are reported for nine image multiplication examples in Table IX.

From Table IX, it can be seen that the resulting PSNRs of the images of the proposed approximation methods are above 30dBs, while the proposed designs have significantly lower hardware cost. The approximate 8-bit multiplier [18] exhibit the highest PSNRs. This can be explained by the fact that this method uses exact compressors for most significant columns of its multiplier bits.

In the [13], [14] and [19] images smoothing and sharpening are commonly used as applications to evaluate the effectiveness of approximate multipliers. Table X displays the minimum, maximum, and average PSNR and SSIM values. Nine images (House, Fruits, Pirate, Blonde Woman, Lena, Boat, Livingroom, Tank, Peppers) are examined to provide these results.

The Sobel operator, utilized for edge detection also implemented using the proposed multipliers. The Sobel operator has several uses in computer vision for extracting basic features [19].

To perform a more comprehensive study and a fair comparison between hardware costs and error measurements, this paper introduces a figure of merit (FOM) by multiplying PSNR by the average of all hardware parameters delay, power and area, as shown in (15):

$$FOM = PSNR \times \left(\frac{delay\ saving\ + area\ saving + power\ saving}{3}\right) \qquad (15)$$

The PSNR value used in the FOM calculation represents the average over all case studies. In [16], authors introduced a FOM, which combines PSNR, delay saving and power saving of the hardware. However, this FOM does not consider the area saving of the hardware. The proposed FOM, defined by (15), of the two proposed designs and references are depicted in Fig. 11.

From Fig. 11, it can be seen that the proposed_Mul1 and 2 methods have the highest FOM value, imply their superiority to other methods.

TABLE X
THE PSNRS AND SSIM ACHIEVED FROM SHARPENING, SMOOTHING AND EDGE DETECTION WITH 8-BIT APPROXIMATE MULTIPLIERS

Multipliers	**Sharpening**						**Smoothing**						**Edge detection**					
	PSNR			SSIM			PSNR			SSIM			PSNR			SSIM		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
Mul-1[13]	29.12	31.23	32.44	0.981	0.987	0.984	29.81	30.62	31.97	0.939	0.941	0.962	42.39	42.92	43.97	0.945	0.948	0.957
Mul-2[13]	32.12	33.95	35.40	0.983	0.988	0.984	34.78	33.95	37.89	0.974	0.984	0.981	41.74	42.67	44.31	0.912	0.947	0.957
Multiplier1[15]	26.37	28.85	30.81	0.973	0.988	0.983	26.07	28.86	30.49	0.916	0.936	0.949	41.20	42.37	44.07	0.910	0.939	0.949
M8-1[21]	29.02	32.10	33.56	0.965	0.991	0.983	27.19	31.36	35.79	0.930	0.950	0.962	43.25	44.05	44.25	0.961	0.962	0.969
M8-2[21]	26.16	30.05	31.46	0.957	0.986	0.981	37.42	39.81	41.86	0.981	0.984	0.989	40.37	41.24	41.96	0.917	0.928	0.940
Majority-based[16]	36.10	37.18	38.66	0.988	0.994	0.992	27.25	29.54	32.85	0.956	0.961	0.969	40.55	41.02	41.34	0.797	0.818	0.861
C-full[19]	39.70	47.26	51.76	0.991	0.999	0.998	42.89	46.64	56.89	0.991	0.993	0.995	50.93	53.21	54.00	0.991	0.994	0.995
M8[18]	47.64	48.13	49.09	0.992	0.999	0.998	39.42	40.41	41.37	0.996	0.996	0.997	58.73	59.19	60.23	0.998	0.998	0.999
ALM-SOA(M=5) [26]	32.14	36.33	38.38	0.969	0.989	0.983	32.09	35.75	37.81	0.953	0.963	0.974	43.49	44.90	45.73	0.946	0.952	0.966
Proposed-mul1	26.74	28.09	29.56	0.945	0.984	0.972	25.19	30.04	31.39	0.929	0.951	0.959	39.44	39.98	40.35	0.796	0.810	0.824
Proposed-mul2	27.51	29.32	31.28	0.967	0.986	0.980	25.60	26.24	27.71	0.951	0.956	0.962	45.62	46.36	47.68	0.965	0.973	0.979

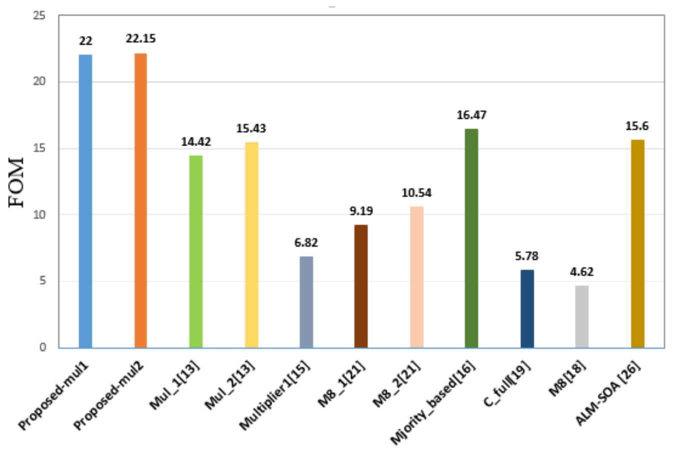


Fig. 11. The Values of FOM for Approximate Multipliers

TABLE XI
ACCURACY OF 8-BIT APPROXIMATE MULTIPLIERS EMPLOYED IN HARDWARE IMPLEMENTATION OF NEURAL NETWORKS

Multipliers	**Accuracy**	
	MLP	**CNN**
Mul_1[13]	91.0%	81.5%
Mul_2[13]	90.9%	82.9%
Multiplier1[15]	91.7%	84.7%
M8_1[21]	91.1%	81.6%
M8_2[21]	91.2%	84.5%
Majority_based [16]	88.7%	82.4%
C_full[19]	91.7%	87.3%
M8[18]	89.9%	83.2%
ALM_SOA(M=5)	81.7%	79.8%
Proposed-mul1	91.1%	87.7%
Proposed-mul2	91.3%	88.1%
Exact	92.1%	88.6%

D. Neural Network Application

Another application of approximate multipliers is the hardware implementation of neural networks. In the following, the efficiency of the proposed multipliers in the hardware implementation of neural networks has been examined and compared with similar works. For this purpose, a multi-layer perceptron (MLP) neural network and a convolution neural network (CNN) have been implemented and the results of the implementations are given in Table XI.

The implemented MLP neural network consists of two layers with 700 neurons in the hidden layer and 10 neurons in the output layer. This network is trained using the MNIST dataset [36]. The implemented CNN also includes two convolution layers and a fully connected layer in the output. This network is trained using the SVHN dataset [37].

As Table XI shows, the proposed multipliers have a less than one percent drop in accuracy compared to the exact multiplier. Although the proposed multipliers are less accurate than the existing multipliers in some cases, it is noteworthy that the proposed multipliers provide high accuracy in the implementation of both networks, while some previous multipliers perform better in implementing one of the two networks and worse in one another.

V. CONCLUSION

In this paper, a six-gate and two gate-free approximate compressors were first introduced and then they were used to developed two approximate multipliers. The probabilities in partial product reduction were studied and used to develop more efficient approximate compressors and multipliers' structure. Simulation results demonstrated that the application of the probability could significantly improve the performance of the approximate methods. The hardware analysis of the proposed approximate methods, called proposed-mul1 and 2 show that the proposed-mul1 outperforms 31%, 53%, 43%, 67%, and 59% the best existing methods in terms of delay, power, area, PDP, and ADP, respectively. These terms are 23%, 53%, 42%, 64%, and 52% in terms of delay, power, area, PDP, and ADP, respectively, for the proposed_mul2. Simulation results also show that the proposed methods have accepted performance in terms of accuracy for human perceptual based applications, e.g., image. A FOM, which combines three hardware parameters called delay saving, area saving and power saving and PSNR, was proposed and used to assess the performance of the proposed_mul1 and 2. The results show the merit of the proposed techniques. The performance of the proposed approximate multipliers in the hardware implementation of the neural network has also been investigated. The simulation results indicate the appropriate accuracy compared to the exact multiplier in these applications. This result indicates that the proposed multipliers can be used for artificial intelligence and machine learning applications.

REFERENCES

[1] A. Wang, B. H. Calhoun, and A. P. Chandrakasan, Sub-threshold design for ultra low-power systems. New York: Springer, 2011.

[2] V. Chippa, S. Chakradhar, K. Roy and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing", Proceedings of the 50th Annual Design Automation Conference on - DAC '13, 2013.

[3] Q. Xu, T. Mytkowicz and N. Kim, "Approximate Computing: A Survey", IEEE Design & Test, vol. 33, no. 1, pp. 8-22, 2016.

[4] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)", 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014.

[5] S. Timarchi and M. Fazlali, "Generalised fault-tolerant stored-unibit-transfer residue number system multiplier for moduli set {2n−1, 2n, 2n+1}", IET Computers & Digital Techniques, vol. 6, no. 5, pp. 269-276, 2012.

[6] M. Fazlali, H. Valikhani, S. Timarchi and H. Malazi, "Fast architecture for decimal digit multiplication", Microprocessors and Microsystems, vol. 39, no. 4-5, pp. 296-301, 2015.

[7] L. Rahimzadeh, M. Eshghi and S. Timarchi, "Radix-4 implementation of redundant interleaved modular multiplication on FPGA", 2014 22nd Iranian Conference on Electrical Engineering (ICEE), 2014.

[8] Khaing Yin Kyaw, Wang Ling Goh and Kiat Seng Yeo, "Low-power high-speed multiplier for error-tolerant application", 2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC), 2010..

[9] S. Hashemi, R. Bahar and S. Reda, "DRUM: A Dynamic Range Unbiased Multiplier for approximate applications", 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2015.

[10] S. Ejtahed and S. Timarchi, "Efficient Approximate Multiplier Based on a New 1-Gate Approximate Compressor", Circuits, Systems, and Signal Processing, vol. 41, no. 5, pp. 2699-2718, 2022.

[11] C.-H. Chang, J. Gu, and M. Zhang, "Ultra Low-Voltage Low-Power CMOS 4-2 and 5-2 Compressors for Fast Arithmetic Circuits," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 10, pp. 1985–1997, 2004.

[12] A. Momeni, J. Han, P. Montuschi and F. Lombardi, "Design and Analysis of Approximate Compressors for Multiplication", IEEE Transactions on Computers, vol. 64, no. 4, pp. 984-994, 2015.

[13] M. Ahmadinejad, M. Moaiyeri and F. Sabetzadeh, "Energy and area efficient imprecise compressors for approximate multiplication at nanoscale", AEU - International Journal of Electronics and Communications, vol. 110, p. 152859, 2019.

[14] O. Akbari, M. Kamal, A. Afzali-Kusha and M. Pedram, "Dual-Quality 4:2 Compressors for Utilizing in Dynamic Accuracy Configurable Multipliers", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 4, pp. 1352-1361, 2017.

[15] S. Venkatachalam and S. Ko, "Design of Power and Area Efficient Approximate Multipliers", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 5, pp. 1782-1786, 2017.

[16] F. Sabetzadeh, M. Moaiyeri and M. Ahmadinejad, "A Majority-Based Imprecise Multiplier for Ultra-Efficient Approximate Image Multiplication", IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 11, pp. 4200-4208, 2019.

[17] M. Ha and S. Lee, "Multipliers With Approximate 4–2 Compressors and Error Recovery Modules", IEEE Embedded Systems Letters, vol. 10, no. 1, pp. 6-9, 2018.

[18] P. Edavoor, S. Raveendran and A. Rahulkar, "Approximate Multiplier Design Using Novel Dual-Stage 4:2 Compressors", IEEE Access, vol. 8, pp. 48337-48351, 2020.

[19] A. Strollo, E. Napoli, D. De Caro, N. Petra and G. Meo, "Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers", IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 9, pp. 3021-3034, 2020.

[20] H. Pei, X. Yi, H. Zhou and Y. He, "Design of Ultra-Low Power Consumption Approximate 4–2 Compressors Based on the Compensation Characteristic", IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, no. 1, pp. 461-465, 2021.

[21] M. Ansari, H. Jiang, B. Cockburn and J. Han, "Low-Power Approximate Multipliers Using Encoded Partial Products and Approximate Compressors", IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 8, no. 3, pp. 404-416, 2018.

[22] H. Mahdiani, A. Ahmadi, S. Fakhraie and C. Lucas, "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications", IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 57, no. 4, pp. 850-862, 2010.

[23] H. Jiang, C. Liu, F. Lombardi and J. Han, "Low-Power Approximate Unsigned Multipliers With Configurable Error Recovery", IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 1, pp. 189-202, 2019.

[24] G. Jain, M. Jain and G. Gupta, "Design of radix-4,16,32 approx booth multiplier using Error Tolerant Application", 2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2017.

[25] Y. Chang, Y. Cheng, S. Liao and C. Hsiao, "A Low Power Radix-4 Booth Multiplier With Pre-Encoded Mechanism", IEEE Access, vol. 8, pp. 114842-114853, 2020.

[26] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi and F. Lombardi, "Design and Evaluation of Approximate Logarithmic Multipliers for Low Power Error-Tolerant Applications", IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 9, pp. 2856-2868, 2018.

[27] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha and M. Pedram, "RoBA Multiplier: A Rounding-Based Approximate Multiplier for High-Speed yet Energy-Efficient Digital Signal Processing", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 2, pp. 393-401, 2017.

[28] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, and J. Han, "A comparative evaluation of approximate multipliers," in Proc. Int. Symp. Nanosc. Archit. (NANOARCH), pp. 191–196, 201

[29] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 32, no. 1, pp. 124–137, 2013.

[30] Z. Aizaz and K. Khare, "Area and power efficient truncated booth multipliers using approximate carry-based error compensation," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 2, pp. 579–583, 2022.

[31] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, "Approximate hybrid high radix encoding for energy-efficient inexact multipliers," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 3, pp. 421–430, 2018.

[32] P. Yin, C. Wang, H. Waris, W. Liu, Y. Han, and F. Lombardi, "Design and analysis of energy-efficient dynamic range approximate logarithmic multipliers for machine learning," IEEE Transactions on Sustainable Computing, vol. 6, no. 4, pp. 612–625, 2021.

[33] A. Amirany, G. Epperson, A. Patooghy, and R. Rajaei, "Accuracy-adaptive spintronic adder for Image Processing Applications," IEEE Transactions on Magnetics, vol. 57, no. 6, pp. 1–10, 2021.

[34] R. Rajaei and A. Amirany, "Nonvolatile low-cost approximate spintronic full adders for computing in memory architectures," IEEE Transactions on Magnetics, vol. 56, no. 4, pp. 1–8, 2020.

[35] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Trans. Image Process., vol. 13, no. 4, pp. 600 600–612, Apr. 2004.

[36] "The mnist database," MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges. [Online]. Available: http://yann.lecun.com/exdb/mnist.

[37] The street view house numbers (SVHN) dataset. [Online]. Available: http://ufldl.stanford.edu/housenumbers.