Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers

Antonio Giuseppe Maria Strollo, *Senior Member, IEEE*, Ettore Napoli, *Senior Member, IEEE*, Davide De Caro, *Senior Member, IEEE*, Nicola Petra, *Member, IEEE*, and Gennaro Di Meo

***Abstract*— Approximate multipliers attract a large interest in the scientific literature that proposes several circuits built with approximate 4-2 compressors. Due to the large number of proposed solutions, the designer who wishes to use an approximate 4-2 compressor is faced with the problem of selecting the right topology. In this paper, we present a comprehensive survey and comparison of approximate 4-2 compressors previously proposed in literature. We present also a novel approximate compressor, so that a total of twelve different approximate 4-2 compressors are analyzed. The investigated circuits are employed to design 8 × 8 and 16 × 16 multipliers, implemented in 28nm CMOS technology. For each operand size we analyze two multiplier configurations, with different levels of approximations, both signed and unsigned. Our study highlights that there is no unique winning approximate compressor topology since the best solution depends on the required precision, on the signedness of the multiplier and on the considered error metric.**

***Index Terms*— Approximate computing, approximate multiplier, approximate compressors, 4-2 compressors, digital arithmetic.**

I. Introduction

IMPROVING the energy efficiency of digital circuits is a major requirement for modern systems on chip. Several important applications such as machine learning, multimedia digital signal processing, data mining, and data recognition are error resilient [1]. For this kind of applications, approximate computing is an effective way to obtain efficiency gain in terms of power, speed, and area [2]–[4].

Hardware-level approximation has been investigated mainly for arithmetic units, such as adders [5]–[10] and multipliers, that often constitute one of the most energy-hungry digital block [11].

Several approaches have been proposed to obtain highly efficient approximate multipliers [12]. Approximate recursive multipliers use elementary 2×2 approximate multiplier modules that are assembled to obtain $n \times n$ multipliers [13]–[15]. The papers [16]–[18] approximate the result of $n \times n$ multipliers by using a small $m \times m$ multiplier (with $m < n$). Ref. [19] exploits approximation in the generation of the partial products by omitting the calculation of some partial products of the multiplier. Logarithmic multipliers [20], [21] sum an approximation of the logarithm of the operands and computes the result as an approximation of the antilogarithm of the sum. In truncated multipliers, [22], [23], some of the partial products are not formed and the resulting truncation error is mitigated with the help of suitable correction functions. Other approaches include runtime configuration of the approximation [24], approximate Booth encoding [25], approximate redundant binary multipliers [26] and cooperation of approximation techniques [27],

The use of approximate compressors has recently emerged as a viable alternative to implement approximate multiplier. Compressors (also known as "one counters") are used to turn the multi-operand sum of the partial products into a two-operand addition, using tree-based logarithmic reduction schemes, such as Wallace [28], Dadda [29], or the Three-Dimensional Method TDM [30]. The most used compressor is the full-adder, that can be defined as a (3,2) compressor because it converts three inputs into a count encoded in two outputs. The half-adder and higher-order compressors (such as 4-2 or 5-3) are also commonly employed in multipliers [31], [32]. Hardware-efficient 6-3 and 7-3 exact compressors are developed in [33] using the concept of the stacking circuit.

Approximate multipliers can be obtained by substituting some of the exact compressors with simpler circuits that introduce some errors but give efficiency gain in terms of power, speed and area. Approximate compressors obtained by truncating the outputs of exact compressors are proposed in [34], while approximate compressors with only two outputs are used in [35], [36]. The paper [37] proposes an approximate 15-4 compressor, built using 5-3 compressor as basic module. The approach in [38] performs lossy compression of the partial product rows, using approximate half-adders to generate a reduced set of product terms. Simple OR gates as approximate counters are employed in [39]. Ref. [40] presents a new family of approximate compressors, where the outputs of compressors have the same weight of the inputs i.e. there are no carry outputs. Two 4×4 approximate multipliers, with different accuracies, are developed in [41], using encoded partial products and approximate compressors; the proposed 4×4 multipliers are used as building blocks for scaling up to larger multipliers.

Manuscript received January 17, 2020; revised March 9, 2020 and March 28, 2020; accepted April 13, 2020. Date of publication May 6, 2020; date of current version September 2, 2020. This work was supported by the Italian Ministry of University, under Project PRIN 20177MEZ7T_004, brain28nm. This article was recommended by Associate Editor M. M. Kermani. *(Corresponding author: Antonio Giuseppe Maria Strollo.)*

The authors are with the Department of Electrical Engineering and Information Technology, University of Napoli Federico II, 80125 Naples, Italy (e-mail: astrollo@unina.it).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2020.2988353

In this paper, we focus our attention on 4-2 compressors, that are commonly used to build exact multipliers [42], owing to the simplified wiring and the efficient transistor-level implementation [43]–[45]. Several approaches have been recently proposed to design approximate 4-2 compressors and to use them to obtain approximate multipliers [46]–[54]. Due to the large number of proposed solutions, the designer who wishes to use approximate 4-2 compressors in a multiplier has a difficult time in selecting the right topology. In this paper we present a comprehensive survey and comparison of previously proposed approximate 4-2 compressors, focusing on the architectures designed to be employed in standard tree-based multipliers.

We show that the stacking circuit technique [33] can be modified to design approximate compressors, we highlight that some of the previously proposed approximate 4-2 compressors can be derived in this way and we present a new approximate 4-2 compressor. Overall, we analyze a total of twelve different approximate 4-2 compressors.

The investigated circuits (plus a hybrid solution using two different approximate 4-2 compressors to reduce different parts of the partial product matrix) are employed to design 8×8 and 16×16 multipliers, implemented in 28nm CMOS technology. For each operand size, we analyze two multiplier configurations, with different levels of approximations, both signed and unsigned.

The paper highlights that the error performance of multipliers using approximate 4-2 compressors depends on the specific connection of each partial product to each input of the approximate compressors. This point is overlooked in previous papers and makes challenging the design of the partial product reduction tree.

Our analysis shows that approximate compressors are well suited for the implementation of unsigned multipliers, while their use in signed multipliers can cause a significant degradation of precision, especially when used in the left-most columns of the partial products matrix.

The comparison between the circuits presented in this paper shows that a significant improvement in electrical performance is provided for certain approximate 4-2 compressors (more than 50% power and delay reduction), while other designs yield a much lower power saving. Considering the power-accuracy tradeoff, our analysis shows that there is no unique winning topology since the best solution depends on the required precision, on the considered error metric and on the signedness of the multiplier. We report tradeoff curves, showing power vs. precision, that can be helpful for the selection of the best suited topology.

We test the approximate multipliers in some examples of image processing. It is shown that some of the investigated approximate multipliers, while performing well for image blending, show less satisfactory behavior in the other test cases.

The main contributions of the paper are:

- a comprehensive survey and comparison of the approximate 4-2 compressors presented in the literature;
- development of a novel approximate compressor, obtained by modifying the technique of the stacking circuit;
- highlight that the partial product reduction tree with approximate compressors should be designed by optimizing the specific connection of each partial product to each input of the approximate compressors, to minimize the error rate;
- show that approximate compressors are well suited for unsigned multipliers, while their use in signed multipliers can result in a significant degradation of precision;
- show power-accuracy tradeoff curves that can be helpful for the selection of the approximate compressor that is best suited for a specific application;

The paper is organized as follows. Section II reviews previously proposed approximate 4-2 compressors and presents a new proposed circuit. Section III analyzes the error characteristics of the multipliers, VLSI implementation results and the trade-off curves between accuracy and power. Applications of the developed multipliers for image processing are presented in Section IV.

II. 4-2 Compressors

A. Exact Compressor

An exact 4-2 compressor should more properly be called a (5,3) counter since it has five inputs of equal weight (denoted as x_1, x_2, x_3, x_4, T_{in}) and three outputs, S, C and T_{out}. The S output has the same weight as the inputs, while C and T_{out} have a double weight. The compressor is designed so that T_{out} is not dependent on T_{in}; this feature is exploited in tree multipliers, where T_{out} produced by a 4-2 compressor in the i-th column is connected to T_{in} of a compressor in column $i+1$, without impacting on delay. The Fig. 1 shows a common implementation of a 4-2 compressor, using two full adders; more efficient designs are proposed in [43]–[45].

B. Previously Proposed Approximate 4-2 Compressors

The T_{in} and T_{out} pins are not used in most of previously proposed approximate 4-2 compressors (with some exceptions, [55]), to simplify both circuit implementation and wiring. The maximum value that can be encoded by using only S and C outputs is three. Having four inputs x_1, ..x_4, it is obvious that at least one error (when all inputs are '1') is unavoidable.

Fig. 2(a) shows the schematic of the approximate 4-2 compressor proposed in [46], while the approximate compressor proposed in [47] is shown in Fig. 2(b). These circuits (referred in the following as "*Momeni*" and "*Venka*") introduce errors in the truth-table of the exact 4-2 compressor to obtain a simpler logic implementation.

Three approximate 4-2 compressors are reported in [48], that will be referred as "*Yang1*", "*Yang2*" and "*Yang3*" in the following. The schematics of the compressors are shown in Fig. 2(c)-(e). The *Yang1* compressor, in Fig. 2(c), is the most accurate version, with a single error in the truth table: when all inputs are '1' the output is $C = 1$, $S = 1$. This behavior corresponds to the so-called "saturating counter" proposed in [49]. The *Yang2* and *Yang3* circuits in Fig. 2(d-e) aim at simplifying the S output circuitry while introducing additional errors in the truth table of the compressor. The Fig. 2(f) shows the schematic of the approximate 4-2 compressor proposed in [50], (named "*Lin*" compressor in the following). An erroneous result is produced when all inputs are '1', as in the *Yang1* compressor, but in this case the outputs are $C = 1$, $S = 0$, resulting in a difference of two between the number of inputs '1' and the approximate count value computed by compressor. The behavior of the *Lin* compressor corresponds to the "reflecting 4:2 counter" proposed in [49]. In [51], the C output of the

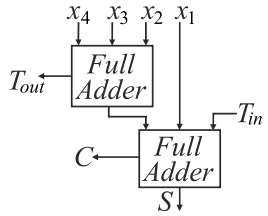

Fig. 1. An exact 4-2 compressor implemented with two full-adders.

approximate compressor of Fig. 2(e) is modified; the proposed design (named "*Ha*" compressor in the following) can be implemented as depicted in Fig. 2(g). The truth table of the *Ha* compressor has four inexact entries (like *Yang3* compressor) but has the characteristic of always underestimating the exact result. Dual-quality 4-2 compressors, having the flexibility of switching between exact and approximate operating modes, are presented in [52]. Fig. 2(h-i) shows the approximate part of the compressors in [52] that will be referred as "*Akbari1*" and "*Akbari2*", respectively. A simple approximate compressor is proposed in [53]. The circuit, named in the following "*Sabetz*" and shown in Fig. 2(l), consists of majority gate, does not use the x_2 input and assumes that S output is constant and equal to '1'. The compressor in [54], referred as "*Ahma*", is also very hardware efficient, using only three NOR and one NAND gates, as shown in Fig. 2(m).

C. Proposed Approximate 4-2 Compressor

The proposed approximate 4-2 compressor uses the concept of the stacking circuit, originally introduced in [33] to design hardware-efficient 6-3 and 7-3 exact compressors and modified here to obtain approximate compressors. Given the four inputs x_1, x_2, x_3, x_4, a 4-bit stacker circuit has four outputs y_1, y_2, y_3, y_4 such that: y_1 will be high if any of the inputs is '1', y_2 will be high if any two of the inputs are '1' and so on. A four-bit stacker is described by the following Boolean equations:

$$y_1 = x_1 + x_2 + x_3 + x_4 \tag{1}$$

$$y_2 = x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 \tag{2}$$

$$y_3 = x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4 \tag{3}$$

$$y_4 = x_1x_2x_3x_4 \tag{4}$$

In designing a compressor, we can count the number of y_i that are '1' instead of counting the x_i, since it easy to see that: $\sum_{i=1}^{4} x_i = \sum_{i=1}^{4} y_i$ (the summation here indicates the counting of the Boolean terms that are '1').

Let us assume that the inputs $x_1..x_4$ are independent of each other and let us indicate as p their probability of being '1'. Analysis of (1)-(4) reveals that y_1 is the term having the highest probability of being '1', followed by y_2, y_3 and by y_4 that is the lowest-probability term (for example, for $p = 0.5$ the probabilities of the y_i terms are: $P(y_1) = 0.9375$, $P(y_2) = 0.6875$, $P(y_3) = 0.3125$ and $P(y_1) = 0.0625$).

Since the maximum count value of an approximate 4-2 compressor is three, we have to neglect one of the y_i terms in (1)-(4); from the above discussion the choice that minimizes the error probability is to neglect y_4. Our aim is now to obtain three new Boolean functions w_1, w_2, w_3, simpler than (1)-(3), such that $\sum_{i=1}^{3} w_i = \sum_{i=1}^{3} y_i$. We proceed by successive approximations. In a first step, the w_i are constructed so as to be as simple as possible, while covering all cases in which y_1 is '1' (remember that y_1 is the highest probability term in (1)-(3)). A possible solution is

$$w_1 = x_1 + x_2 \tag{5}$$

$$w_2 = x_3 \tag{6}$$

$$w_3 = x_4 \tag{7}$$

By counting the number of w_i that are '1' in (5)-(7) we cover not only all the cases in which $y_1 = '1'$, but also all the cases in which $y_2 = '1'$, with the exception of the case in which $x_1x_2 = '1'$ (note that y_2 is the term in (1)-(3) which has the higher probability, after y_1). Furthermore, two of the four cases where y_3 is '1' are also covered by (5)-(7). Thus, an approximate 4-2 compressor can be obtained by counting, with the help of a full-adder, the number w_i that are '1' in (5)-(7), as shown in Fig. 3(a); the truth table of this circuit corresponds to *Ha* compressor, [51].

A more accurate compressor can be obtained by including in (5)-(7) the term x_1x_2 needed to cover all the cases in which y_2 is high. This can be achieved by modifying w_2 as follows:

$$w_2' = x_3 + x_1x_2 \tag{8}$$

The approximate 4-2 compressor obtained from (5),(7), and (8) has never been proposed before and covers also all the cases in which y_3 is high, except for the case in which $x_1x_2x_3 = '1'$. The corresponding circuit, shown in Fig. 3(b) will be named "*Proposed*" compressor in the following.

An even more accurate circuit can finally be obtained by including the term $x_1x_2x_3$ in w_3:

$$w_3' = x_4 + x_1x_2x_3 \tag{9}$$

The equations (5),(8), and (9) satisfy the condition $\sum_{i=1}^{3} w_i = \sum_{i=1}^{3} y_i$. The corresponding circuit is shown in Fig. 3(c) and has the same truth table of the *Yang1* compressor.

D. Approximate Compressors Characteristics

Table I reports the truth tables of the approximate 4-2 compressors of Fig. 2 and Fig. 3(b). The erroneous values in the truth tables are highlighted in bold. The columns denoted as E report the error, defined as the difference between the number of inputs '1' and the approximate count value computed by compressors.

The *Yang1* (Fig. 2(c)) and *Lin* (Fig. 2(f)) compressors have a single error in the truth table. The *Proposed* circuit (Fig. 3(b)) and the *Yang2* compressor (Fig. 2(d)) exhibit two erroneous entries, while the other approximate compressors show a larger number of errors. For the *Yang3*, *Ha* and *Proposed* compressors, the error condition occurs when $x_3x_4 = '1'$, while in the *Proposed* circuit the error takes place for $x_1x_2x_3 = '1'$. In the other compressors, the errors are more uniformly spread across the table.

III. Multiplier Design and Performance

We use the approximate 4-2 compressors to implement $n \times n$ multipliers, with $n = 8$ and $n = 16$, and consider two configurations for each multiplier:

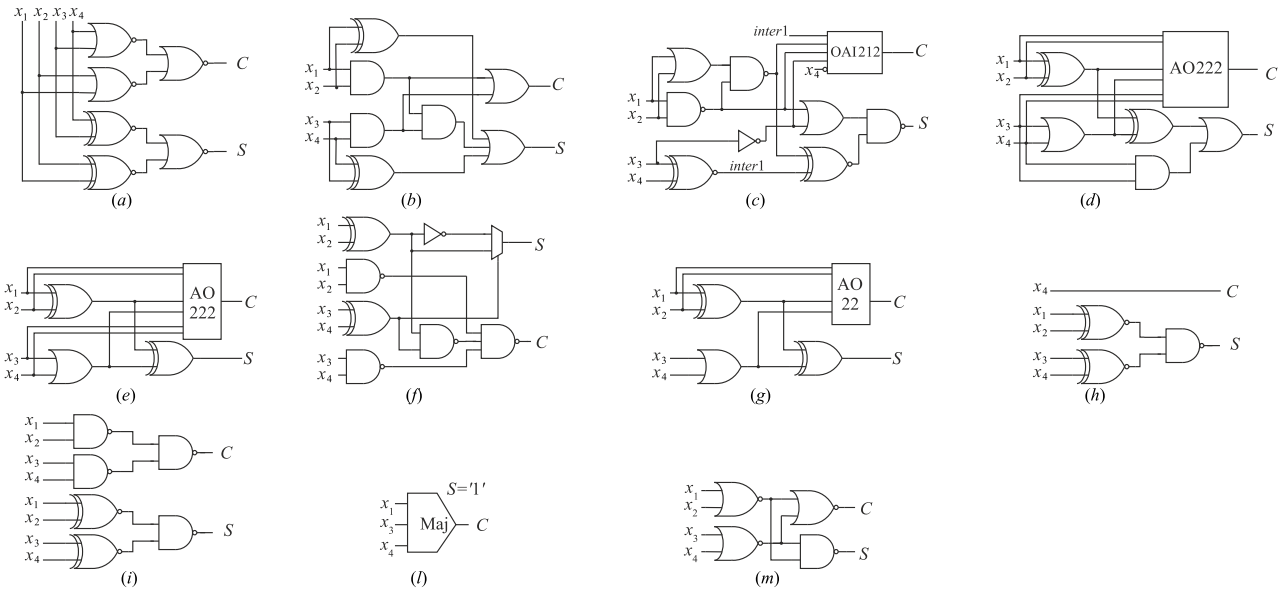

Fig. 2. Schematic of the previously proposed approximate 4-2 compressors. (a) Momeni et. al., [46] (b) Venkatachalam et. al, [47] (c-d-e) Yang et. al [48], (f) Lin et. al [50], (g) Ha et al. [51], (h-i) Akbari et. al [52], (l) Sabetzadeh et. al [53], (m) Ahmadinejad et. al [54]. The modules AO222 and AO22 refer to compound And-Or gates, while the module OAI212 refer to a compound Or-And-Invert gate. Their behavior is the following (where dot "·" stands for "and" and plus "+" for "or"): AO222 (six-inputs gate): $Y = A \cdot B + C \cdot D + E \cdot F$; AO22 (four-inputs gate): $Y = A \cdot B + C \cdot D$; OAI212 (five-inputs gate): $Y = not((A + B) \cdot C \cdot (D + E))$.

TABLE I

TRUTH TABLES OF THE APPROXIMATE 4-2 COMPRESSORS

$x_4..x_1$	*Momeni* [46]		*Venka* [47]		*Yang1* [48]		*Yang2* [48]		*Yang3* [48]		*Lin* [50]		*Ha* [51]		*Akbar1* [52]		*Akbar2* [52]		*Sabetz* [53]		*Ahma* [54]		*Proposed*	
	CS	*E*	*CS*	*E*	*CS*	*E*	*CS*	*E*	*CS*	*E*	*CS*	*E*	*CS*	*E*	*CS*	*E*	*CS*	*E*	*CS*	*E*	*CS*	*E*	*CS*	*E*
0000	01	+1	00		00		00		00		00		00		00		00		01	+1	00		00	
0001	01		01		01		01		01		01		01		01		01		01		01		01	
0010	01		01		01		01		01		01		01		01		01		01		01		01	
0011	01	-1	10		10		10		10		10		10		00	-2	10		01	-1	01	-1	10	
0100	01		01		01		01		01		01		01		01		01		01		01		01	
0101	10		01	1	10		10		10		10		10		01	-1	01	-1	11	+1	11	+1	10	
0110	10		01	-1	10		10		10		10		10		01	-1	01	-1	01	-1	11	+1	10	
0111	11		11		11		11		11		11		11		01	-2	11		11		11		10	-1
1000	01		01		01		01		01		01		01		11	+2	01		01		01		01	
1001	10		01	-1	10		10		10		10		10		11	+1	01	-1	11	+1	11	+1	10	
1010	10		01	-1	10		10		10		10		10		11	+1	01	-1	01	-1	11	+1	10	
1011	11		11		11		11		11		11		11		11		11		11		11		11	
1100	01	-1	10		10		11	+1	11	+1	10		01	-1	10		10		11	+1	01	-1	10	
1101	11		11		11		11		10	-1	11		10	-1	11		11		11		11		11	
1110	11		11		11		11		10	-1	11		10	-1	11		11		11		11		11	
1111	11	-1	11	-1	11	-1	11	-1	11	-1	10	-2	11	-1	10	-2	10	-2	11	-1	11	-1	11	-1

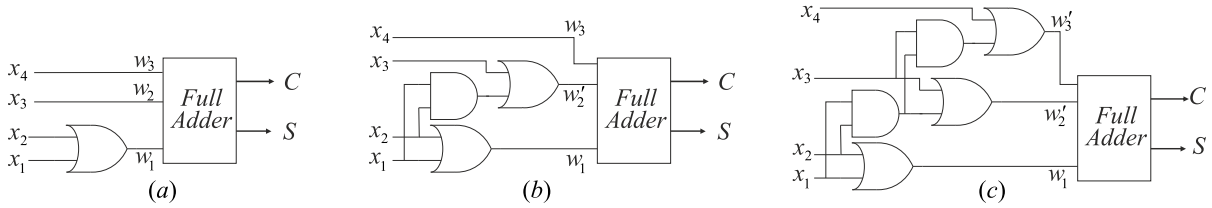

Fig. 3. Derivation of approximate 4-2 compressor using the concept of the stacking circuit. (a) Compressor obtained from (5),(6),(7); its behavior corresponds to the circuit proposed in [51]. (b) Proposed approximate 4-2 compressor obtained from (5),(7),(8). (c) Compressor obtained from (5),(8),(9); its behavior corresponds to the Yang1 circuit shown in Fig. 2(c).

i) *C-N*: uses approximate 4-2 compressors only in the n less significant columns of the partial-product matrix (PPM). Aims at minimizing the errors.
ii) *C-FULL*: uses approximate 4-2 compressors in all the PPM. This is a more aggressive approach, aimed at minimizing the power dissipation.

A. Partial Products Reduction

The PPM is reduced by using an approach like the Dadda multiplier, using several stages of compressors to reduce the maximum height of the PPM to two rows. The maximum height is calculated by working back from the final stage. Owing to the use of 4-2 compressors, this gives 2, 4, 8,

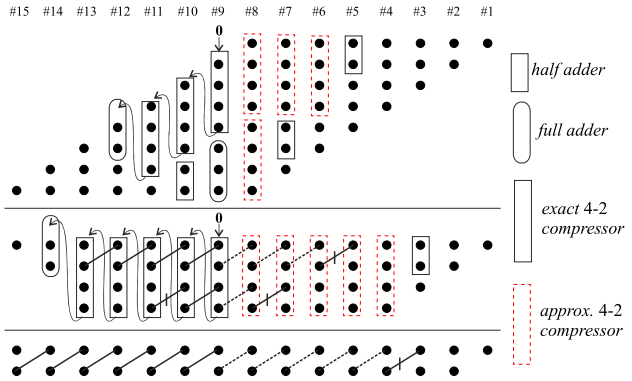

Fig. 4. Reduction scheme for an 8×8 unsigned multiplier with $C - N$ configuration (approximate 4-2 compressors are used only in the 8 less significant columns of the partial-product matrix). Carry and Sum outputs of approximate 4-2 compressors are shown linked by dotted lines, while solid lines are used for full adders and exact 4-2 compressors outputs. Half adder outputs are linked by a solid line with a vertical bar.

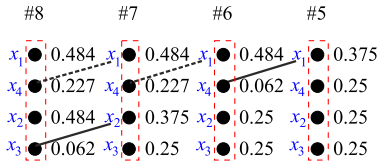

Fig. 5. Partial products probability of the second stage of the multiplier of Fig. 4 using the approximate Ha compressor [51]. The figure shows an input configuration for the approximate 4-2 compressors in the second stage that minimizes the error probability.

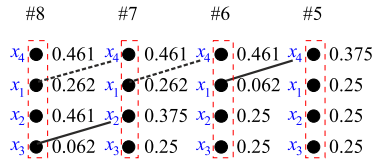

Fig. 6. Partial products probability of the second stage of the multiplier of Fig. 4 using the approximate 4-2 compressor proposed in this paper. The figure shows a possible input configuration for the approximate 4-2 compressors that minimizes the error probability.

16, 32, …as the maximum heights for the various stages. Full and half adders are used only when they are strictly needed to obtain the required reduction of the PPM.

Fig. 4 shows, as an example, the reduction scheme for an unsigned 8×8 multiplier, *C-N* configuration. In this diagram, *Carry* and *Sum* outputs of the approximate 4-2 compressors are shown linked by dotted lines, while solid lines are used for the outputs of exact 4-2 compressors and full adders. Half adder outputs are linked by a solid line with a vertical bar. The arrowed lines in Fig. 4 denote the connections between the T_{in}-T_{out} pins of exact 4-2 compressors. Note the full-adder in column #12 of the first stage and the full-adder in column #14 of the second stage, needed to accept the T_{out} signal of the exact 4-2 compressor in the previous column.

In the *C-N* configuration, the 8×8 multiplier uses 9 approximate 4-2 compressors and 8 exact compressors, while in the 16×16 multiplier the number of approximate and exact compressor is 49 and 48, respectively.

It is important to observe that the investigated approximate compressors (apart from *Yang1* and *Lin*) are not input symmetric, since the error depends on the specific order in which the input bits are connected to the x_1, x_2, x_3, and x_4 inputs. In other words, for the same number of inputs '1', the count value can change if the inputs are permuted (on the contrary, the inputs of exact compressors can be freely permuted without changing system functionality). As an example, the Ha compressor for input $x_4x_3x_2x_1 = 1100$ gives a count value of one, while for input $x_4x_3x_2x_1 = 0011$ yields a count value of two.

Therefore, the error performance of a partial products reduction tree depends on the specific connection of every signal to every input of the approximate compressors. To minimize the error rate, the probability of the signals driving the inputs of the approximate compressors should be considered while designing the partial product compression tree. This point is overlooked in previous papers and makes the design of the partial product reduction tree challenging. Furthermore, not knowing the exact order of connections used in previous papers, it is difficult to regain (and compare) the results presented in the previous art.

In our analysis we proceed as follows (let us consider an unsigned multiplier, for the time being). In the first partial product reduction stage, all partial products are independent of each other and their probability of being '1' (simply indicated as probability in the following) is 1/4. Therefore, there is no preferential connection between these partial products and approximate compressor inputs. Then, the probabilities of the partial products of the second stage are calculated from the compressor truth table [40]. For instance, in the *Ha* compressor, the probability that S is '1' is:

$$\begin{aligned} p(S) &= p(x_4x_3x_2x_1 = 0001) + p(x_4x_3x_2x_1 = 0001) + \ldots \\ &\quad \ldots + p(x_4x_3x_2x_1 = 1100) + p(x_4x_3x_2x_1 = 1111) \\ &= 124/256 \end{aligned}$$

The contribution of each term in the sum is easily obtained from the x_i probability; as an example: $p(x_4x_3x_2x_1 = 1100) = p(x_4)p(x_3)(1 - p(x_2)(1 - p(x_1)) = 9/256$ For the other compressors the output probabilities are obtained in a similar way.

Fig. 5 shows, as an example, the columns 5-8 of the PPM of Fig. 4, using the *Ha* compressor, with the partial products of the second stage annotated with their probability. In the *Ha* circuit an error occurs when $x_3x_4 =$ '1' therefore, to minimize the error probability, x_3 and x_4 should be driven by the two partial products with lower probability, as shown in the same Fig. 5.

The situation is quite like the one of Fig. 5 when *Yang2* or *Yang3* approximate compressor are employed: for *Yang3* compressor the error conditions are the same as the *Ha* compressor, while for *Yang2* the error conditions are a subset of the ones of *Ha* compressor.

In the approximate compressor proposed in this paper, the error condition is $x_1x_2\ x_3 =$ '1' therefore, to minimize the error probability, x_4 should be driven by the partial product with higher probability. This gives a different configuration, as depicted in Fig. 6.

For the other approximate 4-2 compressors it is more difficult to determine the optimal connection between partial products and approximate compressor inputs that minimizes the error probability since the errors are scattered in the truth table (see Table I). Luckily, this scattering of the error conditions also helps in minimizing the influence of the specific connection of every signal to every input of the approximate compressor on the overall multiplier performance. After trying several configurations, we found that using the same scheme

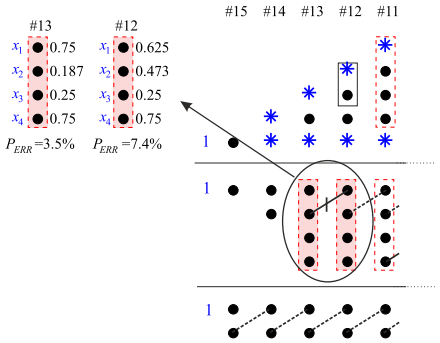

Fig. 7. Left-most columns of the PPM of an 8×8 signed multiplier, *C-FULL* configuration. The star symbols represent the complemented partial products. The probability of the inputs of the two approximate 4-2 compressors in columns #12 and #13 is highlighted (the use of *proposed* approximate 4-2 compressor is assumed).

employed for the *Ha*, *Yang2* and *Yang3* compressors gives the best performances also for the other investigated circuits.

The proposed approach can be applied also to partial products reduction in signed multiplier. In this case, however, the designer should consider that complemented partial products (appearing in the PPM of signed multipliers) have a probability of 3/4 of being '1'. This increases the occurrence of errors, due to the higher probability of partial products.

To better explain this phenomenon, the Fig. 7 shows a portion (left-most columns) of the PPM of an 8×8 signed multiplier, *C-FULL* configuration, where the star symbols represent the complemented partial products. The Figure highlights the probability of the inputs of the two approximate 4-2 compressors in columns #12 and #13. The error probability of these two compressors, using *proposed* circuit, is about 11%. As shown in the following (see Tab. II) this value is larger than the error probability of the entire 8×8 unsigned multiplier *C-N* configuration and not far from the error probability of the 8×8 *C-FULL* unsigned multiplier. A similar behavior is shown by all other approximate compressors, since in all topologies a significant increase in error rate occurs when the probability of partial products increases.

B. Error Metrics

Several metrics have been proposed to assess the error characteristics of approximate arithmetic circuits [12], [40], [56], [57]. Let us indicate as M and M' the results produced by an exact and an approximate multiplier, respectively; the Error Distance is defined as: $ED = |M - M'|$ and the Relative Error Distance as: $RED = ED/|M|$ (for $M \neq 0$). Finally, let us indicate as *MaxOut* the maximum absolute value of the accurate multiplier: $MaxOut = (2^n - 1)^2$ for unsigned multipliers and: $MaxOut = 2^{2n-2}$ for signed multipliers.

The metrics considered in the following are:

- *ER*: The Error Rate, which is the percentage of multiplications for which $ED > 0$.
- *NMED*: The Normalized Mean Error Distance, defined as the average value of *ED* divided by *MaxOut*.
- *MRED*: The Mean Relative Error Distance, defined as the average value of *RED*.
- $NoEB = 2n - \log_2(1 + ERMS)$: The Number of Effective bits, defined as in [40], where *ERMS* is the root mean square of *ED*.
- *PRED* introduced in [57] as the probability of having *RED* higher than 2 percent.

To obtain the error metrics, we have performed exhaustive simulations for 8×8 multipliers, while simulations using 50 million random vectors have been used for the 16×16 multipliers.

1) Unsigned Multipliers: Tables II and III summarize the results obtained for the unsigned multipliers. The *Yang1* compressor gives the best error performance, followed by the *Lin* compressor; both circuits have a single error in the truth table and hence show the same error rate. The *Lin* compressor is less accurate and hence the multipliers using *Lin* compressor exhibit larger *NMED*, *MRED*, *PRED* and lower *NoEB* compared to *Yang1*.

The multipliers using the *Proposed* compressor show good error performances, very close to the one of *Lin*, with a slightly larger *NoEB*. The multipliers using *Yang2*, *Yang3*, and *Ha* (in this order) follow in the ranking. *Yang3* and *Ha* show the same error rate, as expected. The *Momeni*, *Venka*, *Sabetz*, *Ahma* and *Akbari* compressors show worse error performance, with *Akbari2* and *Venka* having the same error rate. As expected, multipliers using *C-FULL* configuration show a sensible degradation in accuracy, compared to *C-N*. Only the multipliers using *Yang1*, *Lin* and *proposed* compressors allow to obtain *MRED*<0.01 for both 8×8 and 16×16 multipliers. *Momeni* and *Sabetz* multipliers show a large *MRED* and *PRED* close to 1; this can be explained observing, from Table I, that these compressors give a non-zero output when their inputs are zero. Let us assume that one of two operands of a multiplier is small: in this case the left-most columns of the partial product matrix are zero, but the *Momeni/Sabetz* approximate compressors placed there produce high *S* outputs, that appear in the most significant bits of the product. Therefore, an erroneous result is computed, with a very large relative error.

The last row in Tables II-V refers to a *Hybrid* configuration. In *Hybrid* multiplier we use two compressor types to reduce the PPM, with the aim of improving the trade-off between error and electrical performance (that will be discussed in the next subsection). To that purpose, in the right-most columns of the PPM (having a lower weight) we employ a simple compressor, characterized by low power dissipation, whereas in the right-most columns of the PPM (that directly contribute to the most significant bits of the result) we use a low-error compressor. Based on the results of Tables II-VI, the choice for right-most columns of the PPM fell on *Ahma* circuit, (we prefer not using *Sabetz/Momeni* compressors due to their undesired characteristic of giving non-zero output when all inputs are zero) and to the *Proposed* circuit for the left-most columns of the PPM (*Lin* compressor, slightly more accurate but also more power hangry, would be another reasonable choice). The subdivision between right-most and left-most PPM columns is somewhat arbitrary. In the following, for the C-N configuration, *Ahma* compressor is used in the less significant $(3/4)n$ columns of the PPM, while the *Proposed* compressor is employed in the following $n/4$ columns. For the C-FULL configuration, *Ahma* is used in the less significant n columns of the PPM, with *Proposed* compressors used in the following columns.

In terms of error metrics, the *Hybrid* configuration shows *NMED* and *NoEB* not far from the ones achievable with the *Proposed* compressor, while the *ER*, *MRED* and *PRED* are higher but still much better than the ones of the *Ahma* compressor.

2) Signed Multipliers: Results for signed multipliers are reported in Tables IV and V. For the *C-N* configuration, error

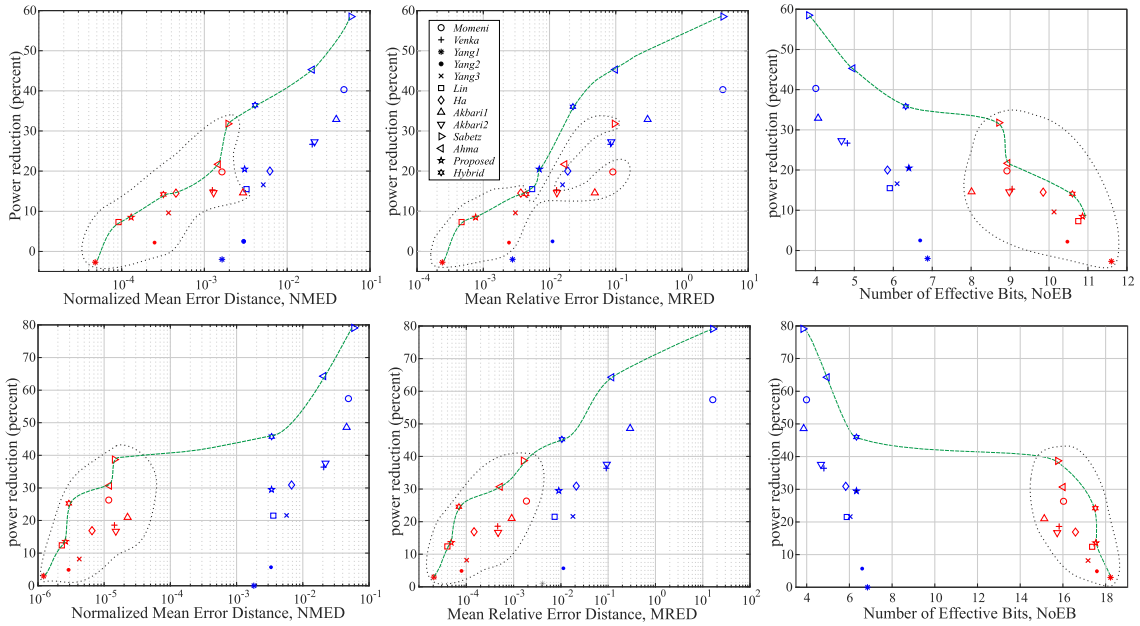

Fig. 8. Trade-off curves for unsigned multipliers. Top: 8 × 8 multipliers; Bottom: 16 × 16 multipliers. The red symbols (circled by the dotted lines) refer to the $C-N$ configuration, while the blue symbols refer to the *C-FULL* multiplier configuration. The dashed green line represents the Pareto-optimal frontier, given by configurations with the best tradeoff.

TABLE II

ERROR PERFORMANCE OF THE 8 × 8 UNSIGNED MULTIPLIERS

Compressor type	*C-N* configuration					*C-FULL* configuration				
	ER	*NMED*	*MRED*	*NoEB*	*PRED*	*ER*	*NMED*	*MRED*	NoEB	*PRED*
Momeni	0.93	1.63×10^{-3}	9.03×10^{-2}	8.92	2.56×10^{-1}	0.99	4.85×10^{-2}	4.14	4.01	8.32×10^{-1}
Venka	0.63	1.25×10^{-3}	1.28×10^{-2}	9.05	1.43×10^{-1}	0.76	2.01×10^{-2}	8.35×10^{-2}	4.82	5.85×10^{-1}
Yang1	0.04	4.79×10^{-5}	2.40×10^{-4}	11.60	2.32×10^{-3}	0.09	1.63×10^{-3}	2.76×10^{-3}	6.88	4.67×10^{-2}
Yang2	0.20	2.50×10^{-4}	2.44×10^{-3}	10.47	2.69×10^{-2}	0.30	2.98×10^{-3}	1.11×10^{-2}	6.69	1.38×10^{-1}
Yang3	0.28	3.68×10^{-4}	3.05×10^{-3}	10.13	3.36×10^{-2}	0.39	5.13×10^{-3}	1.57×10^{-2}	6.10	1.98×10^{-1}
Lin	0.04	9.16×10^{-5}	4.68×10^{-4}	10.75	6.15×10^{-3}	0.09	3.22×10^{-3}	5.46×10^{-3}	5.91	6.08×10^{-2}
Ha	0.28	4.53×10^{-4}	3.68×10^{-3}	9.85	4.41×10^{-2}	0.39	6.20×10^{-3}	1.88×10^{-2}	5.85	2.19×10^{-1}
Akbari1	0.84	2.93×10^{-3}	4.82×10^{-2}	8.01	3.52×10^{-1}	0.92	3.92×10^{-2}	3.03×10^{-1}	4.07	8.37×10^{-1}
Akbari2	0.63	1.29×10^{-3}	1.29×10^{-2}	8.98	1.47×10^{-1}	0.76	2.14×10^{-2}	8.55×10^{-2}	4.67	5.86×10^{-1}
Sabetz	0.98	1.93×10^{-3}	9.52×10^{-2}	8.71	2.83×10^{-1}	0.99	5.91×10^{-2}	4.18	3.83	9.21×10^{-1}
Ahma	0.77	1.47×10^{-3}	1.70×10^{-2}	8.94	1.86×10^{-1}	0.87	2.01×10^{-2}	9.88×10^{-2}	4.96	7.03×10^{-1}
Proposed	0.09	1.30×10^{-4}	7.65×10^{-4}	10.86	8.06×10^{-3}	0.18	3.06×10^{-3}	7.01×10^{-3}	6.40	9.82×10^{-2}
Hybrid	0.56	3.19×10^{-4}	4.21×10^{-3}	10.67	4.27×10^{-2}	0.78	4.08×10^{-3}	2.25×10^{-2}	6.39	2.65×10^{-1}

TABLE III

ERROR PERFORMANCE OF THE 16 × 16 UNSIGNED MULTIPLIERS

Compressor type	*C-N* configuration					*C-FULL* configuration				
	ER	*NMED*	*MRED*	*NoEB*	*PRED*	*ER*	*NMED*	*MRED*	NoEB	*PRED*
Momeni	1.00	1.17×10^{-5}	1.87×10^{-3}	16.03	3.67×10^{-3}	1.00	4.89×10^{-2}	16.3	3.99	8.45×10^{-1}
Venka	0.97	1.43×10^{-5}	4.61×10^{-4}	15.82	2.78×10^{-3}	0.99	2.07×10^{-2}	9.11×10^{-2}	4.81	6.56×10^{-1}
Yang1	0.29	1.22×10^{-6}	2.07×10^{-5}	18.23	3.25×10^{-5}	0.50	1.83×10^{-3}	3.75×10^{-3}	6.85	5.69×10^{-2}
Yang2	0.65	2.90×10^{-6}	7.91×10^{-5}	17.59	3.35×10^{-4}	0.83	3.32×10^{-3}	1.13×10^{-2}	6.60	1.52×10^{-1}
Yang3	0.74	4.20×10^{-6}	1.02×10^{-4}	17.18	3.93×10^{-4}	0.89	5.71×10^{-3}	1.79×10^{-2}	6.04	2.34×10^{-1}
Lin	0.29	2.29×10^{-6}	3.99×10^{-5}	17.37	1.05×10^{-4}	0.50	3.59×10^{-3}	7.37×10^{-3}	5.88	8.37×10^{-2}
Ha	0.74	6.55×10^{-6}	1.46×10^{-4}	16.59	5.85×10^{-4}	0.89	6.76×10^{-3}	2.11×10^{-2}	5.83	2.66×10^{-1}
Akbari1	1.00	2.26×10^{-5}	9.10×10^{-4}	15.13	5.17×10^{-3}	1.00	4.59×10^{-2}	2.91×10^{-1}	3.86	9.05×10^{-1}
Akbari2	0.97	1.50×10^{-5}	4.71×10^{-4}	15.73	2.82×10^{-3}	0.99	2.20×10^{-2}	9.32×10^{-2}	4.67	6.58×10^{-1}
Sabetz	1.00	1.43×10^{-5}	1.62×10^{-3}	15.76	3.98×10^{-3}	1.00	5.90×10^{-2}	16.3	3.83	9.30×10^{-1}
Ahma	0.99	1.20×10^{-5}	5.21×10^{-4}	16.01	3.28×10^{-3}	1.00	2.06×10^{-2}	1.19×10^{-1}	4.96	7.74×10^{-1}
Proposed	0.47	2.61×10^{-6}	4.80×10^{-5}	17.54	1.15×10^{-4}	0.69	3.38×10^{-3}	9.02×10^{-3}	6.33	1.16×10^{-1}
Hybrid	0.96	2.85×10^{-6}	7.44×10^{-5}	17.53	2.52×10^{-4}	0.99	3.39×10^{-3}	9.61×10^{-3}	6.33	1.19×10^{-1}

rate and *NoEB* are close to the ones of unsigned multipliers, while a performance degradation is observed for *NMED*. On the other hand, the *MRED* and *PRED* show an evident deterioration. The situation is much worse for *C-FULL*: here all performance parameters decrease significantly compared to unsigned multiplier. The *MRED* becomes very large (higher than 1 in all cases except for 8 × 8 multipliers that use *Yang1* or *Yang2* compressors), highlighting the presence of frequent and large relative errors. This phenomenon is due to two factors. The first one is the increase in error rate, due to the approximate compressors placed in the left columns of the PPM, where high probability complemented partial products

TABLE IV
ERROR PERFORMANCE OF THE 8 × 8 SIGNED MULTIPLIERS

Compressor type	*C-N* configuration					*C-FULL* configuration				
	ER	*NMED*	*MRED*	*NoEB*	*PRED*	*ER*	*NMED*	*MRED*	NoEB	*PRED*
Momeni	0.94	6.33×10^{-3}	1.72×10^{-1}	8.92	5.77×10^{-1}	1.00	1.91×10^{-1}	8.08	4.07	9.60×10^{-1}
Venka	0.74	6.37×10^{-3}	1.74×10^{-1}	8.84	5.25×10^{-1}	0.93	1.17×10^{-1}	2.33	4.58	8.95×10^{-1}
Yang1	0.05	3.26×10^{-4}	3.13×10^{-2}	11.2	2.99×10^{-2}	0.31	2.99×10^{-2}	0.64	5.72	3.02×10^{-1}
Yang2	0.18	7.47×10^{-4}	4.14×10^{-2}	10.8	8.52×10^{-2}	0.45	3.53×10^{-2}	0.87	5.60	3.99×10^{-1}
Yang3	0.28	1.44×10^{-3}	7.09×10^{-2}	10.1	1.46×10^{-1}	0.59	6.11×10^{-2}	1.93	5.05	5.41×10^{-1}
Lin	0.05	6.07×10^{-4}	5.19×10^{-2}	10.3	4.01×10^{-2}	0.31	5.85×10^{-2}	1.20	4.76	3.05×10^{-1}
Ha	0.28	1.80×10^{-3}	1.01×10^{-1}	9.84	1.63×10^{-1}	0.59	6.88×10^{-2}	2.23	4.90	5.49×10^{-1}
Akbari1	0.95	1.57×10^{-2}	4.43×10^{-1}	7.72	8.10×10^{-1}	1.00	5.15×10^{-1}	18.7	2.72	9.84×10^{-1}
Akbari2	0.74	6.71×10^{-3}	1.98×10^{-1}	8.72	5.29×10^{-1}	0.93	1.39×10^{-1}	2.79	4.23	8.95×10^{-1}
Sabetz	0.98	7.31×10^{-3}	1.99×10^{-1}	8.76	6.25×10^{-1}	1.00	1.99×10^{-1}	7.75	4.05	9.74×10^{-1}
Ahma	0.87	7.13×10^{-3}	2.01×10^{-1}	8.77	5.96×10^{-1}	1.00	1.96×10^{-1}	8.29	4.05	9.71×10^{-1}
Proposed	0.09	5.43×10^{-4}	5.36×10^{-2}	10.8	5.13×10^{-2}	0.40	4.14×10^{-2}	1.40	5.44	3.82×10^{-1}
Hybrid	0.56	1.31×10^{-3}	6.67×10^{-2}	10.6	1.77×10^{-1}	0.90	4.48×10^{-2}	1.48	5.43	7.23×10^{-1}

TABLE V
ERROR PERFORMANCE OF THE 16 × 16 SIGNED MULTIPLIERS

Compressor type	*C-N* configuration					*C-FULL* configuration				
	ER	*NMED*	*MRED*	*NoEB*	*PRED*	*ER*	*NMED*	*MRED*	NoEB	*PRED*
Momeni	1.00	1.31×10^{-5}	6.88×10^{-3}	15.89	1.72×10^{-2}	1.00	5.07×10^{-2}	3.6	3.97	9.60×10^{-1}
Venka	0.98	1.57×10^{-5}	7.30×10^{-3}	15.72	1.89×10^{-2}	1.00	2.89×10^{-2}	6.22	4.59	9.53×10^{-1}
Yang1	0.30	1.36×10^{-6}	2.13×10^{-3}	18.11	3.05×10^{-3}	0.74	7.19×10^{-2}	1.70	5.75	3.77×10^{-1}
Yang2	0.64	2.71×10^{-6}	2.38×10^{-3}	17.66	4.44×10^{-3}	0.92	8.37×10^{-2}	2.51	5.65	4.73×10^{-1}
Yang3	0.74	4.32×10^{-6}	3.54×10^{-3}	17.13	6.53×10^{-3}	0.98	1.55×10^{-2}	5.95	5.04	6.70×10^{-1}
Lin	0.30	2.53×10^{-6}	3.13×10^{-3}	17.27	4.73×10^{-3}	0.74	1.41×10^{-2}	3.07	4.78	4.27×10^{-1}
Ha	0.74	6.55×10^{-6}	5.43×10^{-3}	16.58	9.89×10^{-3}	0.98	1.83×10^{-2}	7.16	4.85	7.02×10^{-1}
Akbari1	1.00	2.83×10^{-5}	1.43×10^{-2}	14.87	3.29×10^{-2}	1.00	1.24×10^{-2}	105	2.76	9.95×10^{-1}
Akbari2	0.98	1.65×10^{-5}	8.54×10^{-3}	15.62	2.00×10^{-2}	1.00	3.44×10^{-2}	7.31	4.23	9.54×10^{-1}
Sabetz	1.00	1.53×10^{-5}	7.39×10^{-3}	15.68	1.97×10^{-2}	1.00	5.66×10^{-2}	35.7	3.87	9.84×10^{-1}
Ahma	1.00	1.32×10^{-5}	6.64×10^{-3}	15.89	1.71×10^{-2}	1.00	4.98×10^{-2}	33.7	4.01	9.79×10^{-1}
Proposed	0.47	2.63×10^{-6}	3.11×10^{-3}	17.53	5.01×10^{-3}	0.88	1.04×10^{-2}	7.38	5.44	5.06×10^{-1}
Hybrid	0.96	2.87×10^{-6}	3.37×10^{-3}	17.51	5.35×10^{-3}	1.00	1.14×10^{-2}	8.78	5.42	5.12×10^{-1}

are present. The second one is due to the fact that errors are likely to occur by multiplying small negative numbers (with many '1' bits); in this condition the exact product M is small in magnitude and the Relative Error Distance $ED/|M|$ is very large (this is similar to what occurs in approximate adders with signed operands [10]).

C. VLSI Implementation

In order to investigate the electrical performance of the approximate 4-2 compressors, the multipliers are described in HDL and synthesized by targeting a commercial standard-cell library in 28nm CMOS, from TSMC. Physical synthesis is performed by using Cadence Genus; no special cells are designed to implement the approximate compressors, that are automatically synthesized according to timing constraints, as this allows to perform a fair comparison between the various compressors. Please note also that the employed standard-cell library does not include any special 4-2 compressor cell. Again, this allows a fair comparison between exact and approximate compressors. We assume typical corner, standard-V_T devices, with nominal supply voltage of 0.9V; the output loading corresponds to a fan-out of four inverters 2×. Power dissipation is computed by simulating the final netlist with 100,000 random vectors to obtain the switching activity of each node; the toggle frequency is 1GHz.

The Table VI reports the electrical performance of unsigned multipliers; the signed multipliers exhibit very similar characteristics. In this Table, "min delay" is the minimum delay at which the circuits can be synthesized, while area and power are obtained by synthesizing all the circuits with the same timing constraint (to make a fair comparison): 500ps, and 750ps for 8×8 and 16×16 multipliers, respectively. In addition to the configurations using the approximate 4-2 compressors, multipliers using exact 4-2 compressors are also implemented, for comparison (corresponding results are reported in the column labeled "exact" in Table VI). The Table VI also shows the variations with respect to the exact multiplier, reported as a percentage, with a negative percentage that indicates an improvement.

1) 8 × 8 *Multipliers, C-FULL Configuration:* The best performance is achieved by *Sabetz* compressor, owing to its simplicity, followed by *Ahma*, *Momeni* and *Akbari1* compressors. The *Yang1* design shows a modest 12% improvement in speed, as expected due to the complexity of the compressor cell. This behavior is confirmed by area and power results, that for *Yang1* are higher that exact multiplier; the improvement in area and power is marginal also for *Yang2*. The *Hybrid* circuit shows a remarkable 36% reduction in power dissipation.

2) 8 × 8 *Multipliers, C-N Configuration:* Speed improvement is limited and is similar for all the considered multipliers, because the critical path traverses just one approximate compressor. Again, the best power reduction (more than 30%) is obtained with the *Sabetz* compressor, while *Ahma* and *Momeni* give a power saving of about 20%. The *Proposed* compressor allows 8.5% power reduction (like *Yang3* and *Lin*), while the *Hybrid* compressor yields about 14% power improvement, in line with *Venka*, *Ha*, and *Akbari* topologies.

3) 16 × 16 *Multipliers:* The behavior is like the one described for 8 × 8 multipliers. The delay improvement for *C-N* configuration is limited (lower than 5%), while for *C-FULL* configuration a delay reduction up to about 50% is obtained with *Sabetz* compressor.

TABLE VI
AREA, DELAY AND POWER OF EXACT AND APPROXIMATE MULTIPLIERS

		Exact	*Momeni*	*Venka*	*Yang1*	*Yang2*	*Yang3*	*Lin*	*Ha*	*Akbari1*	*Akbari2*	*Sabetz*	*Ahma*	*Proposed*	*Hybrid*
8× 8 *C-FULL*	Min delay (*ps*)	260	195 -24.9%	195 -24.9%	219 -15.7%	212 -18.2%	207 -20.3%	219 -15.7%	207 -20.3%	187 -28.0%	197 -24.2%	129 -50.4%	175 -32.6%	216 -16.9%	202 -22.0%
	Area (μm^2)	196	133 -32.0%	159 -18.7%	202 3.3%	181 -7.4%	169 -13.6%	169 -13.6%	149 -23.8%	135 -31.0%	150 -23.3%	84 -57.0%	108 -44.8%	155 -20.7%	126 -35.6%
	Power (μW)	177	106 -40.3%	130 -26.7%	181 2.0%	173 -2.5%	148 -16.6%	150 -15.5%	142 -20.0%	119 -32.9%	129 -27.3%	73 -58.5%	97 -45.3%	141 -20.5%	113 -36.3%
8× 8 *C-N*	Min delay (*ps*)	260	239 -7.9%	244 -6.0%	251 -3.3%	248 -4.5%	242 -6.8%	245 -5.6%	244 -6.0%	242 -6.8%	249 -4.1%	231 -11.0%	240 -7.6%	248.6 -4.2%	244 -5.9%
	Area (μm^2)	196	165 -15.5%	177 -9.3%	201 2.9%	188 -3.7%	183 -6.4%	181 -7.5%	167 -14.6%	165 -15.7%	169 -13.4%	133 -32.0%	149 -23.8%	175 -10.6%	163 -16.6%
	Power (μW)	177	142 -19.8%	150 -15.3%	182 2.7%	173 -2.2%	160 -9.6%	165 -7.3%	152 -14.5%	151 -14.6%	152 -14.6%	121 -31.8%	139 -21.7%	162 -8.5%	152 -14.3%
16× 16 *C-FULL*	Min delay (*ps*)	375	261 -30.5%	270 -28.2%	321 -14.4%	307 -18.2%	293 -22.0%	308 -17.9%	287 -23.5%	248 -34.0%	262 -30.2%	174 -53.7%	233 -37.8%	318 -15.2%	301 -19.6%
	Area (μm^2)	920	597 -35.1%	757 -17.8%	1003 9.0%	894 -2.9%	794 -13.7%	818 -11.1%	716 -22.2%	599 -34.9%	729 -20.8%	319 -65.3%	434 -52.8%	727 -21.0%	577 -37.3%
	Power (μW)	938	400 -57.4%	597 -36.4%	938 0.0%	885 -5.7%	736 -21.6%	736 -21.5%	648 -30.9%	482 -48.6%	585 -37.6%	196 -79.1%	335 -64.3%	662 -29.5%	513 -45.3%
16× 16 *C-N*	Min delay (*ps*)	375	367 -2.1%	363 -3.2%	371 -1.2%	367 -2.1%	363 -3.2%	366 -2.3%	361 -3.8%	366 -2.3%	362 -3.5%	356 -5.0%	360 -4.1%	363 -3.2%	362 -3.5%
	Area (μm^2)	920	798 -13.3%	841 -8.6%	925 0.5%	874 -5.1%	880 -4.4%	827 -10.1%	792 -14.0%	758 -17.6%	831 -9.7%	633 -31.2%	676 -26.5%	821 -10.8%	713 -22.5%
	Power (μW)	938	692 -26.3%	763 -18.6%	910 -3.0%	892 -4.9%	861 -8.2%	822 -12.4%	780 -16.9%	741 -21.0%	781 -16.7%	575 -38.7%	650 -30.7%	811 -13.6%	704 -25.0%

TABLE VII
IMAGE BLENDING RESULTS

	Cameraman Lena		*Boat House*		*House Moon*		*AVERAGE*	
	SSIM	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*
Momeni	0.65	23.15	0.82	27.37	0.78	28.03	0.75	26.18
Venka	0.95	32.59	0.91	28.74	0.92	32.08	0.93	31.14
Yang1	1.00	58.33	1.00	54.15	1.00	54.25	1.00	55.58
Yang2	1.00	47.78	1.00	46.42	0.99	44.11	0.99	46.10
Yang3	0.99	44.42	0.98	40.83	0.97	40.50	0.98	41.92
Lin	1.00	52.45	1.00	48.20	1.00	48.63	1.00	49.76
Ha	0.99	42.66	0.98	38.97	0.97	37.98	0.98	39.87
Akbari1	0.75	27.12	0.64	24.57	0.78	27.79	0.72	26.49
Akbari2	0.95	32.57	0.91	28.66	0.92	31.95	0.93	31.06
Sabetz	0.63	22.43	0.75	24.46	0.77	27.10	0.72	24.66
Ahma	0.86	31.05	0.85	27.61	0.84	31.16	0.85	29.94
Proposed	1.00	50.99	1.00	50.30	1.00	50.22	1.00	50.50
Hybrid	1.00	48.90	1.00	48.12	0.99	47.68	1.00	48.23

The largest power reduction is obtained again by *Sabetz* compressor (ranging from about 40% for *C-N* configuration to about 80% for *C-FULL* configuration) followed by *Ahma, Momeni*, *Akbari1* and *Hybrid* compressors (the *Hybrid* circuit exhibits about 45% power reduction for 16 × 16 multiplier, *C-FULL* configuration).

D. Tradeoffs

1) Unsigned Multipliers: Fig. 8 shows the tradeoff curves (power reduction vs. NMED, MRED, and NoEB) for 8 × 8 and 16 × 16 multipliers. In this Figure the red symbols (circled by the dotted lines) refer to the *C-N* configuration, while the blue symbols refer to the *C-FULL* multipliers configuration. The green dashed line represents the Pareto-optimal frontier, given by configurations with the best power-accuracy tradeoff.

As expected, there is no unique winning topology since the best solution depends on the required precision and on the considered metric. The plots displaying power reduction vs NMED, *C-N* configuration, show that *Lin*, *Proposed, Ha* and *Hybrid* compressors lie on the Pareto-optimal frontier, followed by *Ahma* and *Sabetz* (these two latter compressors characterized by larger NMED).

For the *C-FULL* configuration only *Hybrid*, *Ahma* and *Sabetz* lie on the Pareto frontier; the power saving gained by the *C-FULL* configuration is paid with a substantial increase of NMED especially for 16 × 16 multipliers. A similar picture is observed when MRED is considered as accuracy metric. It is worthwhile to note that in the *C-FULL* configuration *Ahma* and *Sabetz* allow to reach a remarkable power saving but the corresponding MRED becomes very large (higher than 0.1). By analyzing the power reduction vs NoEB plots, it can be observed that 16 × 16 multipliers show two well distinct clusters, corresponding *C-FULL* and *C-N* configurations. Again, *Lin*, *Proposed, Ha* and *Hybrid* compressors lie on the high-precision side of the Pareto-optimal frontier for the *C-N* configuration, while *Ahma* and *Sabetz* give the best trade-off when a larger decrease in the precision can be tolerated.

In all considered cases, *Momeni*, *Venka*, *Akbari* and *Yang* topologies reveal sub-optimal.

2) Signed Multipliers: Results for signed multipliers are shown in Fig. 9. The topologies on the Pareto-optimal frontier are the same found for unsigned multipliers. It is worth noting, however, that the two clusters, corresponding *C-FULL* and *C-N* configurations, are more clearly separated, compared to Fig. 8, especially for 16 × 16 multipliers. This implies that any power improvement in *C-FULL* configuration (compared to *C-N*) results in a drastic worsening of the error; therefore, the use of *C-FULL* multipliers in signed multipliers is not recommended.

For example, the 16bit *C-N* multiplier with *Sabetz* compressors achieves an NMED of 1.5×10^{-5} with a power saving of about 39%; with *C-FULL* configuration the compressor giving lower NMED and better power saving is *Hybrid*, with 45.3% power saving but a three orders of magnitude larger NMED, about 1.1×10^{-2}.

IV. APPLICATIONS

In order to investigate the behavior of the approximate multipliers in typical error-resilient applications, we considered

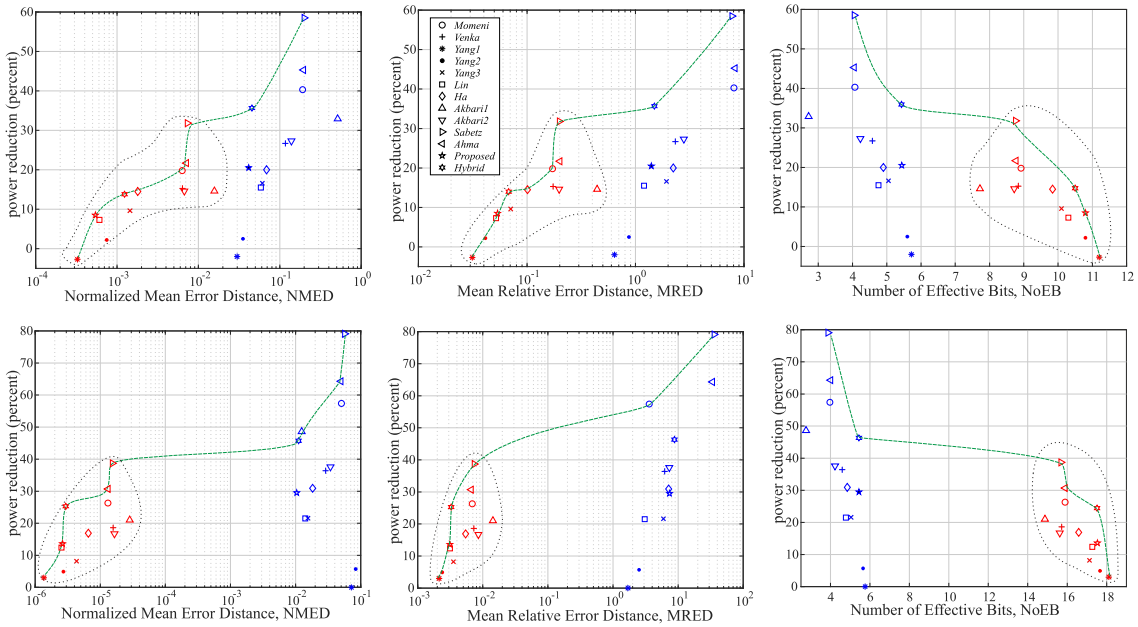

Fig. 9. Trade-off curves for the signed multipliers. Top: 8×8 multipliers; Bottom: 16×16 multipliers. The red symbols (circled by the dotted lines) refer to the $C - N$ configuration, while the blue symbols refer to the *C-FULL* multiplier configuration. The dashed green line represents the Pareto-optimal frontier, given by configurations with the best tradeoff.

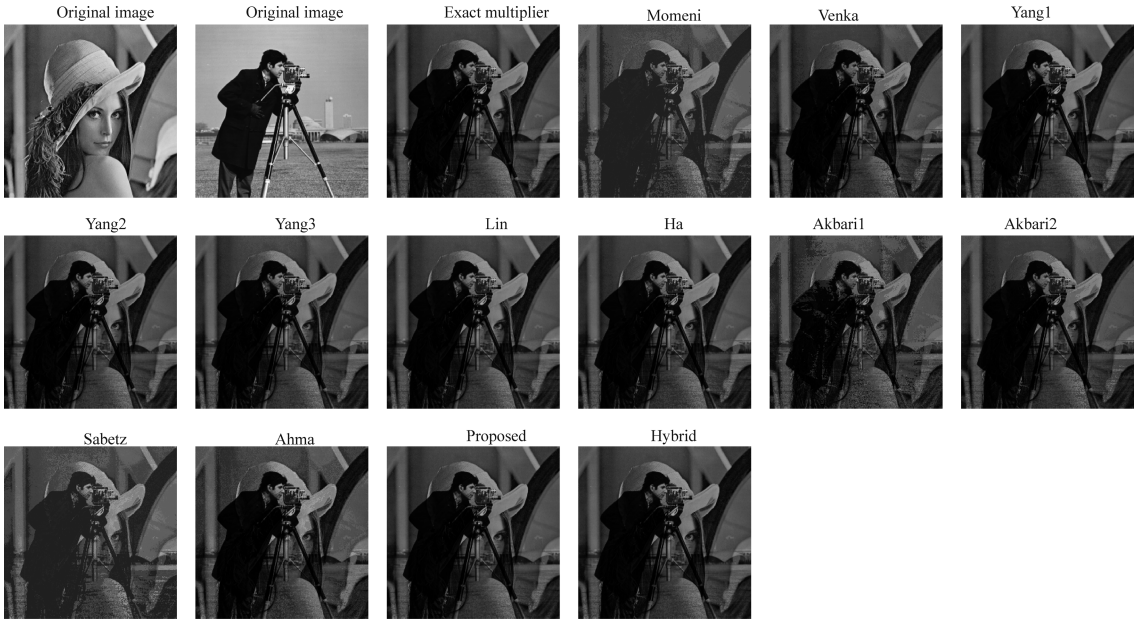

Fig. 10. Image blending results. A multiplier is used to multiply "Lena" and "Cameraman" images on a pixel by pixel basis. Top left: original "Lena" image. Top, second left: original "Cameraman" image. Top, third left: reference result, obtained by using an exact 8×8 multiplier to blend the two original images. Other images: results obtained with approximate multipliers (*C-FULL* configuration) using the investigated 4-2 compressors.

some examples of image processing using either unsigned or signed multipliers.

A. Unsigned Multipliers

1) Image Blending: In this application [20], [46], [53], [54], two 8-bit grayscale images are multiplied on a pixel-by-pixel basis; the product is scaled back to 8-bit, to obtain a final 8bit grayscale image that blends the two input images. We have performed the processing using *C-FULL* multipliers and a set of test images. To assess the quality of the generated outputs, they have been compared with a reference image obtained with an exact 8×8 multiplier. The peak signal-to-noise ratio (PSNR) and the mean structural similarity index (SSIM) are used as metrics to compare the quality of the output images with the reference one.

The Fig. 10 shows an example of image blending, while Tab. VII reports PSNR and SSIM for some image combinations. Most of the investigated compressors work well in this application, with SSIM close to 1 and PSNR larger than 45dB; the *Hybrid* circuit performs very well. The multipliers using *Momeni*, *Sabetz* and *Akbari1* compressors, while having a good power reduction, show relatively low PSNR and SSIM; the use of *Ahma* compressor also results in a visible degradation of the image quality.

TABLE VIII
RESULTS FOR IMAGE SMOOTHING AND SHARPENING

	Smoothing								*Sharpening*							
	Cameraman		*Boat*		*House*		*AVERAGE*		*Cameraman*		*Boat*		*House*		*AVERAGE*	
	SSIM	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*
Venka	0.97	36.05	0.97	34.82	0.96	31.32	0.97	34.06	0.99	27.80	0.98	27.66	0.96	23.56	0.98	26.34
Yang1	1.00	∞	1.00	∞	1.00	∞	1.00	∞	1.00	∞	1.00	∞	1.00	∞	1.00	∞
Yang2	0.99	44.85	1.00	45.88	0.99	39.64	0.99	43.46	1.00	49.44	1.00	48.84	1.00	43.11	1.00	47.13
Yang3	0.99	44.85	1.00	45.88	0.99	39.64	0.99	43.46	1.00	44.78	1.00	47.35	1.00	39.56	1.00	43.90
Lin	1.00	∞	1.00	∞	1.00	∞	1.00	∞	1.00	∞	1.00	∞	1.00	∞	1.00	∞
Ha	0.99	44.86	1.00	45.88	0.99	39.67	0.99	43.47	1.00	44.63	1.00	47.17	1.00	39.46	1.00	43.75
Akbari1	0.90	28.09	0.87	26.77	0.92	28.14	0.90	27.67	0.80	20.54	0.74	18.41	0.86	19.88	0.80	19.61
Akbari2	0.97	36.05	0.97	34.82	0.96	31.32	0.97	34.06	0.99	27.80	0.98	27.66	0.96	23.56	0.98	26.34
Ahma	0.96	35.44	0.96	35.43	0.95	31.32	0.96	34.06	0.97	24.14	0.95	26.21	0.92	20.96	0.95	23.77
Proposed	1.00	∞	1.00	∞	1.00	∞	1.00	∞	1.00	57.17	1.00	57.74	1.00	50.74	1.00	55.22
Hybrid	0.99	45.51	0.99	46.09	0.99	40.22	0.99	43.94	0.99	38.44	0.99	40.49	0.98	33.66	0.99	37.53

2) Image Smoothing and Sharpening: Image smoothing and image sharpening are commonly used benchmarks to analyze the performance of approximate multipliers in the scientific literature. In image smoothing [52], [58], [54], each output pixel is computed as:

$$Y(i,j) = \frac{1}{60}\sum_{m=-2}^{2}\sum_{n=-2}^{2} I(i+m, j+n)h_1(m+3, n+3) \tag{10}$$

while in the sharpening application [18], [36], [41], [48], [51], [52], [54]:

$$Y(i,j) = 2I(i,j) - \frac{1}{273}\sum_{m=-2}^{2}\sum_{n=-2}^{2} I(i+m, j+n)h_2(m+3, n+3) \tag{11}$$

Following the previous papers, in (10),(11) the multiplications are approximate while other operations (addition, subtraction and division) are exact. We have performed the processing using the *C-FULL* multiplier configuration; the Fig. 11 displays some examples of results obtained with the "boat" test image and shows the 5×5 mask matrices h_1 and h_2. The Tab. VIII reports PSNR and SSIM for different images and filtering types.

The *C-FULL* multipliers built with the *Momeni* and *Sabetz* compressors fail in this application (the pixel computation always produces a faulty result) due to the error mechanism discussed in Sect. III-B-1 that reveals very critical when using mask matrices like h_1 and h_2 in Fig. 11, with several coefficients small in size (please note that *Momeni* and *Sabetz* compressors can be used with some success if employed in less aggressive approximate multiplier configurations, like *C-N*).

In the considered image smoothing and sharpening applications, the *Yang1* and the *Lin* compressors give the same result as the exact multiplier (i.e. no errors are introduced with the mask matrices h_1 and h_2, owing to the very low error rate reported in Table II), while the *Proposed* compressor produces an exact result for image smoothing and a result very close to the exact one (but not exactly the same) for image sharpening. The *Hybrid* multiplier gives an SSIM of about 0.99, with a very good PSNR of about 40dB.

The image obtained with the *Akbari1* multiplier show several artifacts, especially for image sharpening; *Akbari2, Ahma* and *Venka* show a PSNR in the range 25-35dB, while better results are achieved with the *Yang2*, *Yang3* and *Ha* compressors.

TABLE IX
EDGE DETECTION RESULTS

	Cameraman		*Lena*		*House*		*AVERAGE*	
	SSIM	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*	*SSIM*	*PSNR*
Momeni	0.19	12.4	0.18	14.3	0.16	12.6	0.18	13.1
Venka	0.20	14.0	0.11	11.3	0.06	9.9	0.13	11.7
Yang1	0.91	33.5	0.86	32.5	0.68	29.1	0.82	31.7
Yang2	0.83	27.3	0.80	29.8	0.68	28.2	0.77	28.4
Yang3	0.68	26.9	0.64	25.6	0.46	23.5	0.59	25.3
Lin	0.82	29.7	0.71	27.5	0.50	22.2	0.68	26.4
Ha	0.57	22.4	0.41	19.2	0.28	15.9	0.42	19.1
Akbari1	0.02	1.8	0.02	1.6	0.02	2.5	0.02	2.0
Akbari2	0.19	13.2	0.09	10.3	0.02	8.2	0.10	10.6
Sabetz	0.24	14.9	0.27	16.8	0.29	17.3	0.27	16.3
Ahma	0.03	6.6	0.02	7.9	0.06	9.3	0.04	7.9
Proposed	0.83	30.6	0.74	28.3	0.56	25.2	0.71	28.0
Hybrid	0.78	30.2	0.73	28.1	0.53	24.1	0.68	27.5

B. Signed Multipliers

To test the behavior of approximate signed multipliers, we considered edge detection using the Sobel operator, that finds several applications in computer vision for low-level feature extraction. In this application the x and y component of the gradient of the image I (indicated as G_X and G_Y) are computed by convolving the image with the two kernels S_X and S_Y [59], [60]:

$$G_X = S_X * I; \quad G_Y = S_Y * I \tag{12}$$

The kernel S_X (corresponding to a 5×5 Sobel template), is shown in Fig. 12, and S_Y is the transpose of S_X. The edges of the image are extracted from the gradient magnitude, given by:

$$G = \sqrt{G_X^2 + G_Y^2} \tag{13}$$

In our tests, the convolutions in (12) are realized with approximate *C-N* signed multiplier, while square and square root in (13) are exact. The Fig. 12 shows some results obtained with the "Lena" test image, while Tab. IX reports PSNR and SSIM obtained for some test images.

As it can be observed, in this application only *Yang*, *Lin*, *Proposed* and *Hybrid* multipliers give acceptable results, with PSNR larger than 25dB. The other compressors show evident artifacts, related to the error mechanisms discussed in Sect. III-B2, and are inadequate for this application.

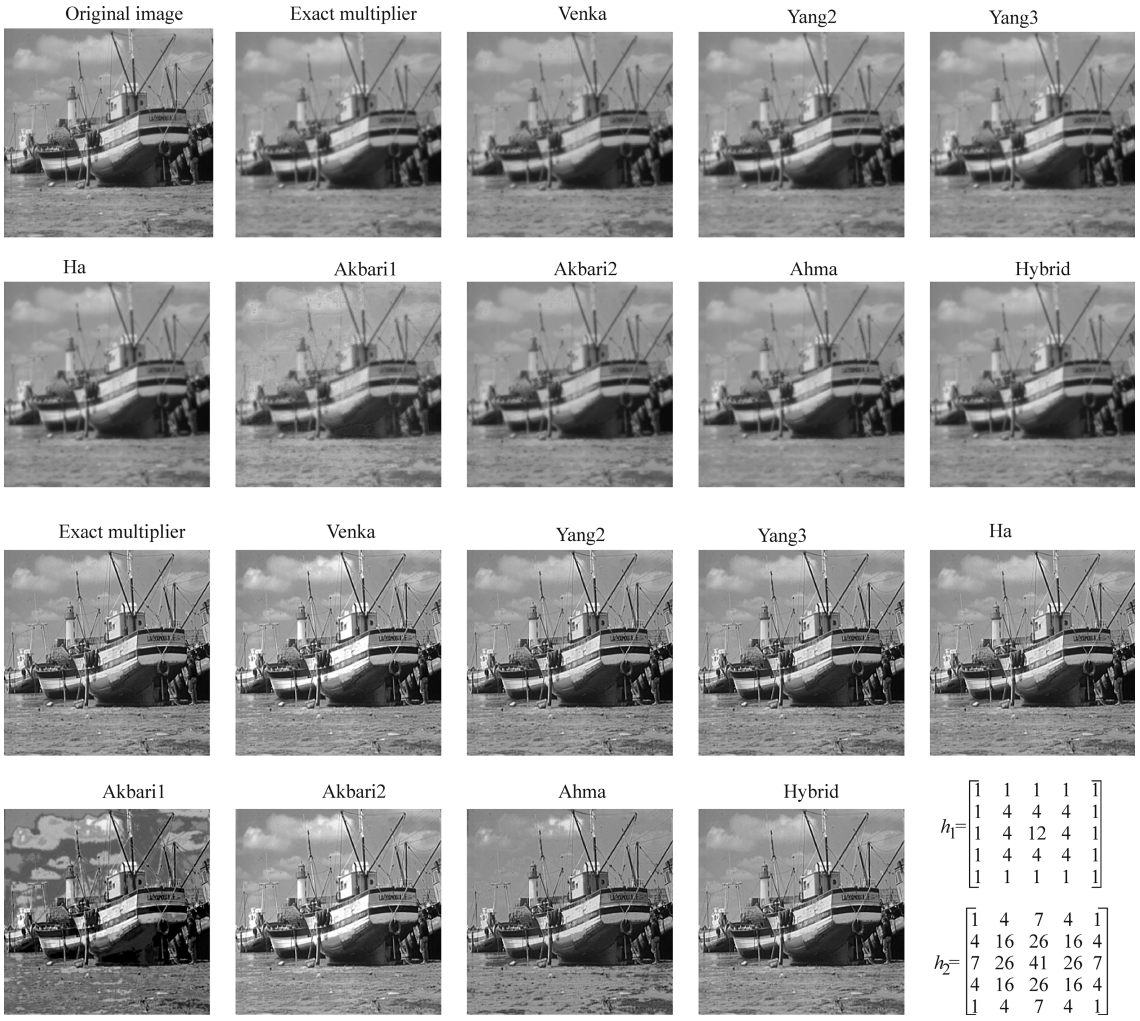

Fig. 11. Filtered "boat" image. First two rows: original image and smoothed images obtained with different multipliers; last two rows: sharpened images. Bottom right: mask matrices used in (10), (11).

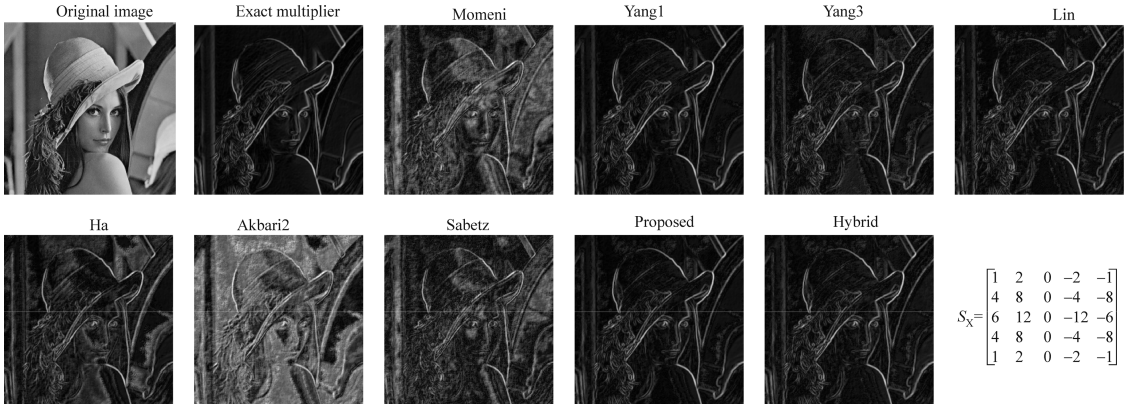

Fig. 12. Edge detection using the Sobel operator, Top left: original "Lena" image. Top, second left: reference result, obtained by using an exact multiplier. Other images: results obtained with approximate $C - N$ multiplier. Bottom right: kernel matrix S_X in (12).

V. Conclusion

In this paper, we have presented a comprehensive survey and comparison of previously proposed approximate 4-2 compressors. We have shown that the stacking circuit technique [33] can be modified to design approximate compressors, we have highlight that some of the previously proposed approximate 4-2 compressors can be derived in this way and we have developed a new approximate 4-2 compressor. A total of twelve different approximate 4-2 compressors have been investigated and several multipliers, with different signedness and level of approximations, have been implemented in a 28nm CMOS technology.

The power-accuracy tradeoff shows that there is no unique winning topology since the best solution depends on the required precision, the considered error metric and the signedness of the multipliers. We have reported power-error tradeoff

curves and examples of image processing, that can be helpful for the selection of the topology best suited for a given application. Based on our analysis, some final considerations can be drawn:

- approximate compressors that give a non-zero result for zero inputs (like *Momeni* and *Sabetz*) are not recommended, as their use results in large relative errors for small operands;
- the compressors with best trade-off between energy saving and accuracy are found to be *Lin*, *Proposed*, *Ha*, *Ahma*. The *Lin* compressor has the additional advantage of being input commutative (its behavior does not depend on the specific order in which the signals are connected to the input pins) and this simplifies the optimal design of the partial product reduction tree;
- using different approximate compressors in different columns of the partial products matrix (as in the *Hybrid* configuration) can be a viable solution to improve the energy-accuracy trade-off;
- for signed multipliers, using approximate compressors in the left-most significant columns of the partial products matrix should be avoided, as this largely increases the error rate and increases and the relative error, especially when multiplying negative numbers small in magnitude;

Possible further investigations include: the use of approximate 4-2 compressors to build recursive multipliers, as in [41], and the use of modified stacking circuits to design approximate compressors of different sizes.

REFERENCES

[1] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proc. 50th Annu. Design Autom. Conf. (DAC)*, 2013, pp. 1–9.

[2] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in *Proc. 52nd Annu. Design Autom. Conf. (DAC)*, 2015, pp. 1–6.

[3] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Des. Test.*, vol. 33, no. 1, pp. 8–22, Feb. 2016.

[4] W. Liu, F. Lombardi, and M. Shulte, "A retrospective and prospective view of approximate computing [point of view]," *Proc. IEEE*, vol. 108, no. 3, pp. 394–399, Mar. 2020.

[5] L. B. Soares, M. M. A. da Rosa, C. M. Diniz, E. A. C. da Costa, and S. Bampi, "Design methodology to explore hybrid approximate adders for energy-efficient image and video processing accelerators," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 6, pp. 2137–2150, Jun. 2019.

[6] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 124–137, Jan. 2013.

[7] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proc. 52nd Annu. Design Autom. Conf. (DAC)*, 2015, pp. 1–6.

[8] H. Jiang, J. Han, and F. Lombardi, "A comparative review and evaluation of approximate adders," in *Proc. Conf. GLSVLSI*, 2015, pp. 343–348.

[9] D. Esposito, D. De Caro, M. De Martino, and A. G. M. Strollo, "Variable latency speculative Han–Carlson adders topologies," in *Proc. 11th PRIME Conf.*, Jun. 2015, pp. 45–48.

[10] D. Esposito, D. De Caro, and A. G. M. Strollo, "Variable latency speculative parallel prefix adders for unsigned and signed operands," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 8, pp. 1200–1209, Aug. 2016.

[11] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2014, pp. 10–14.

[12] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, and J. Han, "A comparative evaluation of approximate multipliers," in *Proc. Int. Symp. Nanosc. Archit. (NANOARCH)*, 2016, pp. 191–196.

[13] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *Proc. 24th Int. Conf. VLSI Design*, Jan. 2011, pp. 346–351.

[14] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, and J. Henkel, "Architectural-space exploration of approximate multipliers," in *Proc. IEEE/ACM/Design Automat. Conf. (DAC)*, Nov. 2016, pp. 1–8.

[15] G. A. Gillani, M. A. Hanif, B. Verstoep, S. H. Gerez, M. Shafique, and A. B. J. Kokkeler, "MACISH: Designing approximate MAC accelerators with internal-self-healing," *IEEE Access*, vol. 7, pp. 77142–77160, 2019.

[16] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 6, pp. 1180–1184, Jun. 2015.

[17] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2015, pp. 418–425.

[18] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "TOSAM: An energy-efficient truncation- and rounding-based scalable approximate multiplier," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 5, pp. 1161–1173, May 2019.

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

[20] 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 Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 9, pp. 2856–2868, Sep. 2018.

[21] M. S. Ansari, B. F. Cockburn, and J. Han, "A hardware-efficient logarithmic multiplier with improved accuracy," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, no. 1, Mar. 2019, pp. 928–931.

[22] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, "Design of fixed-width multipliers with linear compensation function," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 5, pp. 947–960, May 2011.

[23] D. De Caro, N. Petra, A. G. M. Strollo, F. Tessitore, and E. Napoli, "Fixed-width multipliers and multipliers-accumulators with min-max approximation error," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 9, pp. 2375–2388, Sep. 2013.

[24] M. Imani, D. Peroni, and T. Rosing, "CFPU: Configurable floating point multiplier for energy-efficient computing," in *Proc. 54th Annu. Design Autom. Conf.*, Austin, TX, USA, Jun. 2017, pp. 1–6.

[25] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate Radix-4 booth multipliers for error-tolerant computing," *IEEE Trans. Comput.*, vol. 66, no. 8, pp. 1435–1441, Aug. 2017.

[26] W. Liu *et al.*, "Design and analysis of approximate redundant binary multipliers," *IEEE Trans. Comput.*, vol. 68, no. 6, pp. 804–819, Jun. 2019.

[27] V. Leon, K. Asimakopoulos, S. Xydis, D. Soudris, and K. Pekmestzi, "Cooperative arithmetic-aware approximation techniques for energy-efficient multipliers," in *Proc. 56th Annu. Design Autom. Conf. (DAC)*, Las Vegas, NV, USA, 2019, pp. 1–6.

[28] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron. Comput.*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[29] L. Dadda, "Some schemes for fast serial input multipliers," in *Proc. IEEE 6th Symp. Comput. Arithmetic (ARITH)*, vol. 34, Jun. 1983, pp. 349–356.

[30] V. G. Oklobdzija, D. Villeger, and S. S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," *IEEE Trans. Comput.*, vol. 45, no. 3, pp. 294–306, Mar. 1996.

[31] D. Baran, M. Aktan, and V. G. Oklobdzija, "Energy efficient implementation of parallel CMOS multipliers with improved compressors," in *Proc. ISLPED Conf.*, 2010, pp. 147–152.

[32] A. K. Verma and P. Ienne, "Automatic synthesis of compressor trees: Reevaluating large counters," in *Proc. DATE Conf.*, Apr. 2007, pp. 443–448.

[33] C. Fritz and A. T. Fam, "Fast binary counters based on symmetric stacking," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2971–2975, Oct. 2017.

[34] D. R. Kelly, B. J. Phillips, and S. Al-Sarawi, "Approximate signed binary integer multipliers for arithmetic data value speculation," in *Proc. DASIP Conf.*, 2009, pp. 97–104.

[35] A. Cilardo *et al.*, "High speed speculative multipliers based on speculative carry-save tree," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 12, pp. 3426–3435, Dec. 2014.

[36] Y. Guo, H. Sun, L. Guo, and S. Kimura, "Low-cost approximate multiplier design using probability-driven inexact compressors," in *Proc. APCCAS Conf.*, Oct. 2018, pp. 291–294.
[37] R. Marimuthu, Y. E. Rezinold, and P. S. Mallick, "Design and analysis of multiplier using approximate 15-4 compressor," *IEEE Access*, vol. 5, pp. 1027–1036, 2017.
[38] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, and A. Yakovlev, "Energy-efficient approximate multiplier design using bit significance-driven logic compression," in *Proc. DATE Conf.*, Mar. 2017, pp. 7–12.
[39] D. Esposito, A. G. M. Strollo, and M. Alioto, "Low-power approximate MAC unit," in *Proc. 13th Conf. Ph.D. Res. Microelectron. Electron. (PRIME)*, Jun. 2017, pp. 81–84.
[40] D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro, and N. Petra, "Approximate multipliers based on new approximate compressors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 12, pp. 4169–4182, Dec. 2018.
[41] M. S. Ansari, H. Jiang, B. F. Cockburn, and J. Han, "Low-power approximate multipliers using encoded partial products and approximate compressors," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 404–416, Sep. 2018.
[42] N. H. E. Weste and M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. London, U.K.: Pearson, 2011.
[43] C. Chang, J. Gu, and M. Zhang, "Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 10, pp. 1985–1997, Oct. 2004.
[44] S. Veeramachaneni, K. Krishna, L. Avinash, S. Puppala, and M. B. Srinivas, "Novel architectures for high-speed and low-power 3-2, 4-2 and 5-2 compressors," in *Proc. IEEE Int. Conf. VLSI Design*, vol. 1, Jan. 2007, pp. 324–329.
[45] A. Pishvaie, G. Jaberipur, and A. Jahanian, "Improved CMOS (4;2) compressor designs for parallel multipliers," *Comput. Electr. Eng.*, vol. 38, no. 6, pp. 1703–1716, Nov. 2012.
[46] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 984–994, Apr. 2015.
[47] S. Venkatachalam and S.-B. Ko, "Design of power and area efficient approximate multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 5, pp. 1782–1786, May 2017.
[48] Z. Yang, J. Han, and F. Lombardi, "Approximate compressors for error-resilient multiplier design," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, Oct. 2015, pp. 183–186.
[49] B. J. Phillips, D. R. Kelly, and B. W. Ng, "Estimating adders for a low density parity check decoder," *Proc. SPIE*, vol. 6313, Aug. 2006, Art. no. 631302.
[50] C.-H. Lin and I.-C. Lin, "High accuracy approximate multiplier with error correction," in *Proc. 31st ICCD Conf.*, Oct. 2013, pp. 33–38.
[51] M. Ha and S. Lee, "Multipliers with approximate 4–2 compressors and error recovery modules," *IEEE Embedded Syst. Lett.*, vol. 10, no. 1, pp. 6–9, Mar. 2018.
[52] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 4, pp. 1352–1361, Apr. 2017.
[53] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, "A majority-based imprecise multiplier for ultra-efficient approximate image multiplication," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 11, pp. 4200–4208, Nov. 2019.
[54] M. Ahmadinejad, M. H. Moaiyeri, and F. Sabetzadeh, "Energy and area efficient imprecise compressors for approximate multiplication at nanoscale," *AEU-Int. J. Electron. Commun.*, vol. 110, Oct. 2019, Art. no. 152859.
[55] M. H. Moaiyeri, F. Sabetzadeh, and S. Angizi, "An efficient majority-based compressor for approximate computing in the nano era," *Microsyst. Technol.*, vol. 24, no. 3, pp. 1589–1601, Mar. 2018.
[56] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Trans. Comput.*, vol. 62, no. 9, pp. 1760–1771, Sep. 2013.
[57] V. Leon, G. Zervakis, S. Xydis, D. Soudris, and K. Pekmestzi, "Walking through the energy-error Pareto frontier of approximate multipliers," *IEEE Micro*, vol. 38, no. 4, pp. 40–49, Jul./Aug. 2018.
[58] H. R. Myler and A. R. Weeks, *The Pocket Handbook of Image Processing Algorithm in C*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2009.
[59] M. S. Nixon and A. S. Aguado, *Feature Extraction & Image Processing for Computer Vision*, 3rd ed. Amsterdam, The Netherlands: Elsevier, 2012.
[60] *Developer Reference for Intel Integrated Performance Primitives—FilterSobel*. Accessed: Jan. 13, 2020. [Online]. Available: https://software.intel.com/en-us/ipp-dev-reference-filtersobel

Antonio Giuseppe Maria Strollo (Senior Member, IEEE) received the M.S. degree *(cum laude)* and the Ph.D. degree in electronic engineering from the University of Napoli Federico II, Italy. Since 2002, he has been a Full Professor with the University of Napoli Federico II, where he has been the Head of the Department of Electronic and Telecommunication Engineering from 2005 to 2008. He has published more than 140 articles on international journals and conferences. His current research interests are design and analysis of digital VLSI circuits. From 2009 to 2012, he served as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: REGULAR PAPERS. He is currently an Associate Editor of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.

Ettore Napoli (Senior Member, IEEE) received the Ph.D. degree in electronic engineering in 1999, the degree (Hons.) in electronic engineering in 1995, and the Physics degree (Hons.) in 2009. He was a Research Associate with the Engineering Department, University of Cambridge, U.K., in 2004. He has been an Associate Professor with the University of Napoli since 2005. He has authored or coauthored more than hundred articles published in international journals and conferences. His scientific research interests include the modeling and design of power semiconductor devices and VLSI circuit design.

Davide De Caro (Senior Member, IEEE) received the M.S. degree (Hons.) in electronic engineering and the Ph.D. degree in electronic engineering and computer science from the University of Naples "Federico II", Italy, in 1999 and 2003, respectively. He has worked in the area of digital integrated VLSI circuit design for the last 14 years. He is currently an Associate Professor with the Department of Electrical Engineering and Information Technology, University of Naples. He is author of more than 80 technical papers in international journals and refereed international conferences.

Nicola Petra (Member, IEEE) received the M.S. and Ph.D. degrees from the University of Napoli "Federico II" in 2002 and in 2007, respectively. He is currently working as an Associate Professor with the Department of Electrical Engineering and Information Technology, University of Napoli "Federico II." He authored or coauthored more than 60 articles on scientific journals and international conferences. His research interests include design of digital VLSI circuits for telecommunications and high-performance arithmetic circuits.

Gennaro Di Meo received the M.S. degree *(cum laude)* from the University of Napoli Federico II, Italy, in 2018, where he is currently pursuing the Ph.D. degree in electrical engineering and information technology. His research interests include design of digital VLSI circuits for telecommunications, LMS filters, and approximate computing.