Approximate Multipliers Based on New Approximate Compressors

Darjn Esposito, *Member, IEEE*, Antonio Giuseppe Maria Strollo, *Senior Member, IEEE*, Ettore Napoli, *Senior Member, IEEE*, Davide De Caro, *Senior Member, IEEE*, and Nicola Petra, *Member, IEEE*

***Abstract*— Approximate computing is an emerging trend in digital design that trades off the requirement of exact computation for improved speed and power performance. This paper proposes novel approximate compressors and an algorithm to exploit them for the design of efficient approximate multipliers. By using the proposed approach, we have synthesized approximate multipliers for several operand lengths using a 40-nm library. Comparison with previously presented approximated multipliers shows that the proposed circuits provide better power or speed for a target precision. Applications to image filtering and to adaptive least mean squares filtering are also presented in the paper.**

***Index Terms*— Approximate computing, approximate multiplier, digital arithmetic.**

I. Introduction

APPROXIMATE computing is an emerging trend in digital design [1], [2], relaxing the requisite of exact computation to gain substantial performance improvement in terms of power, speed and area. This approach is becoming more and more important for embedded and mobile systems, characterized by severe energy and speed constraints. Approximate computing can be fruitfully applied in several error-resilient applications. Examples are multimedia processing [3], data mining and recognition [2], machine learning.

Multipliers are fundamental subsystems for microprocessors, digital signal processors, and embedded systems with applications ranging from filtering to convolutional neural networks. Unfortunately, multipliers are characterized by complex logic design [4] and constitute one of the most energy-hungry digital blocks [5]. Therefore, approximate multiplier design has become an important research subject in recent years [6].

A multiplier includes a few basic blocks: partial products generation, partial products reduction and carry-propagate addition. Approximations can be introduced in any of these blocks [6]. For example, truncation of the partial products is a well stablished approximation technique in which some of the partial products are not formed and the truncation error is reduced with the help of suitable correction functions [7]–[9]. Other approaches simplify the partial-product matrix by using approximate 2×2 or 4×4 sub-multipliers [10], [11].

Most of the proposed approximate multipliers rely on approximations in the partial product reduction step. Partial products reduction uses compressors to turn a multi-operand sum into a two-operand addition, using logarithmic schemes, such as Wallace [12], Dadda [13], or the Three-Dimensional Method TDM [14], [15]. A compressor is a logic circuit that counts the number of "ones" in the input. The simple and most used compressor is the full-adder (acting as a 3/2 compressor), but higher-order compressors (such as 4/2 or 5/3 [16]–[20]) are also employed. Compressors are XOR-rich circuits and partial products reduction is a critical multiplier block in terms of speed, power and area. Therefore, several approaches have been recently proposed that use approximate compressors for partial products reduction in approximate multipliers.

Kelly *et al.* [21] propose approximate compressors obtained by discarding (truncating) the outputs of exact compressors; a similar approach is used in [22] where approximate compressor with only two outputs are considered. In [23], Momeni *et al.* propose two approximate 4/2 compressors and investigate the performance of approximate multipliers using developed circuits. Dual-quality 4/2 compressors, having the flexibility of switching between exact and approximate operating modes, are presented in [24]. Approximate half-adders, full-adders and 4/2 compressors are presented in [25], where the proposed circuits are utilized in two variants of 16-bit multipliers. The paper [26] proposes an approximate 15/4 compressor, built using 5/3 compressor as basic module. In [27] a lossy compression of the partial-product rows, based on their bit significance, is proposed; this technique uses approximate half-adders (realized with simple OR gates) to generate a reduced set of product terms. A similar approach, using simple OR gates as approximate counters, is used in [28]. Architectural-space exploration of approximate arithmetic units is investigated in [29] where the authors propose four novels approximate multipliers, based on [10].

In this paper, we present a new family of approximate compressors, obtained in a systematic way, aimed to minimize the error probability and the average error. The proposed approximate compressors are implemented by using simple AND-OR gates (no XOR gates are required) and outperform previously proposed circuits in terms of both precision and hardware complexity. We then investigate the use of the proposed compressors to build approximate multipliers. To that purpose, a simple algorithm is proposed, that uses the approximate compressors in the first steps of the partial product reduction tree. Approximate compressors are introduced in the less-significant part of the partial product matrix and added to

Manuscript received January 10, 2018; revised March 12, 2018; accepted May 9, 2018. Date of publication June 12, 2018; date of current version October 23, 2018. This work was supported by the Italian PRIN Project Advanced Nanometer IC Technologies for Next Generation Transceivers under Grant 2015ABZ44K. This paper was recommended by Associate Editor Y. Yu. *(Corresponding author: Darjn Esposito.)*

The authors are with the Department of Electrical Engineering and Information Technology, University of Napoli Federico II, 80138 Naples, Italy (e-mail: darjn.esposito@unina.it).

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

Digital Object Identifier 10.1109/TCSI.2018.2839266

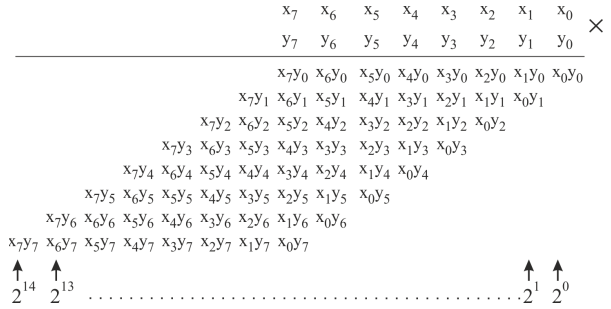

Fig. 1. Partial product matrix of a 8 × 8 bit Multiplier.

the remaining portion of the matrix only if needed, to minimize the overall approximation error. For each operand size, we propose four approximate multiplier versions, with different precision vs electrical performance trade-off.

We have synthesized the circuits developed in this paper and previously proposed approximated multipliers, using a 40nm library, for several operand lengths. Syntheses show that our circuits provide better error-electrical performance trade-off, compared to previously proposed approximate multipliers. We have also investigated the use of approximate multipliers in two applications: image filtering and LMS system identification. For image filtering, the proposed circuits provide excellent compromise between Structural Similarity (SSIM) index and power saving (either 2% SSIM degradation with 64% power saving or 8% SSIM degradation with 77% power saving). In the LMS system identification example, the two most accurate versions of the proposed circuit reach convergence with accuracy comparable to that obtained by using exact multipliers. The other approximate multipliers fail to converge.

The paper is organized as follows. Section II briefly reviews exact compressors. The proposed approximate compressors are described in Section III where a comparison with previous designs is also presented. The algorithm for the design of approximated multipliers using the proposed approximate compressors is described in Section IV. Sections V provide synthesis results and the comparison with previously proposed approximated multipliers. Image filtering and LMS system identification applications are investigated in Section VI, while conclusions are drawn in Section VII.

II. Exact Compressors

Let us consider two unsigned n bit inputs $X = x_{n-1}2^{n-1} + \ldots + x_0$, $Y = y_{n-1}2^{n-1} + \ldots + y_0$. The product Z, between X and Y is:

$$Z = X \cdot Y = p_{2n-1}2^{2n-1} + \ldots + p_0 = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} x_i y_j 2^{i+j} \quad (1)$$

The computation of Z requires the summation of the partial products $x_i y_j$ according to their weights 2^{i+j}. Fig. 1 shows, as an example, the partial products matrix (PPM) for an 8 × 8 multiplier.

Let us consider the partial products belonging to the j-th column of the PPM: $p_0 = x_{j-1}y_0$, $p_1 = x_{j-2}y_1$, $p_2 = x_{j-3}y_3$, …, $p_{j-1} = x_0 y_{j-1}$. The arithmetic sum of the partial products of this column, S, ranges between 0 (when all partial products are low) up to j (when all partial products are 1), and will be indicated in the following as:

$$S = \sum \{p_0, p_1, p_2, \ldots p_{j-1}\} \quad (2)$$

A compressor computes the arithmetic sum in (2), encoding the results in a binary format. The most common compressor is the full-adder having 3 inputs and encoding the results S, into two outputs: *sum* (having the same weight as the inputs) and *carry* (having double weight).

Note that a compressor is fed by signals having the same weight, and some inputs can be carries coming from the column to the right. As an example, the widely used 4/2 compressor [17]–[19] has five inputs (one being a carry from a column to the right) and encodes the result on three outputs: *sum* (having the same weight as the inputs) and *carry*1, *carry*2 (having double weight). The 5/2 compressor [17], [18] has seven inputs (two being carries from a column to the right) and encodes the result on four outputs: *sum* (having the same weight as the inputs) and *carry*1, *carry*2, *carry*3 (having double weight).

III. Proposed Approximate Compressors

The compressors described in the following approximate the arithmetic sum in (2). For most of the combinations of p_i they compute the same value as (2), while in some cases they give an error.

The proposed approximate compressors have j inputs $p_0, p_1, \ldots, p_{j-1}$ and compute $\lceil j/2 \rceil$ outputs by using a novel approach aimed to minimize the error probability and the average error. Please note that the outputs of proposed approximate compressors have the same weight of the inputs (i.e. there are no *carry* outputs). This is different from standard compressors, where the weight of the carry output is two times the weight of the inputs.

In the following, we assume that compressor inputs are partial products belonging to a PPM column. We assume, moreover, that the input bits x_i and y_j are uniformly and independently distributed, so that the probability that each partial product is high can be expressed as:

$$P(p_i) = 1/4 \quad (3)$$

A. Approximate 2/1 Compressor

Let us consider the problem of summing two partial products belonging to the same column. As observed in [22], [25], and [28]:

$$S = \sum \{p_0, p_1\} = \sum \{p_0 p_1, p_0 + p_1\} \quad (4)$$

Eq. (4) shows that we can sum a recoded version of the two partial products, given by the logical AND and the logical OR of the partial products. The probability of the two terms in the right-hand side of (4) is:

$$P(p_0 p_1) = 1/16$$
$$P(p_0 + p_1) = 7/16 \quad (5)$$

From (5), we foresee the possibility of simplifying the hardware, with a reduced error probability, by neglecting the low-probability term $p_0 p_1$. In this way, the arithmetic sum of two partial products belonging to the same column can be approximated as:

$$S_{APP} = \sum \{p_0 + p_1\} \simeq S \quad (6)$$

With (6) we approximate a 2/1 compressor (half-adder) with an OR gate (with consequent area, delay and power improvement), at a price of an error when both partial products

TABLE I
APPROXIMATE 2/1 COMPRESSOR

p_1	p_0	w_1	S	S_{APP}	Err_i	$P(Err_i)$	S_{APP} [25]	Err_i [25]
0	0	0	0	0	0	-	0	0
0	1	1	1	1	0	-	1	0
1	0	1	1	1	0	-	1	0
1	1	1	2	1	1	1/16	3	-1

are high. Table I shows the behavior of the approximate 2/1 compressor. Here, the output of the approximate compressor is indicated as w_1, while Err_i indicates the difference (error) between exact and approximate compressor:

$$Err_i = S - S_{APP} \tag{7}$$

From Table I, it can be observed that the approximate compressor described by (6) under-estimates the exact sum S. As it will be shown in the following, this behavior is common to all the approximate compressors proposed in this paper.

Let us define the error probability, P_E, and the mean error, *Emean*, as:

$$P_E = \sum_i P\left(Err_i\right)$$
$$Emean = \sum_i P\left(Err_i\right) Err_i \tag{8}$$

where $p(Err_i)$ is the probability of having an error for the i-th partial products combination. From Table I, $Err_i = 1$ when $p_1 = p_0 = 1$, which occurs with probability 1/16. Therefore:

$$\text{(approx. 2/1 compressor)} \begin{cases} P_E = 1/16 \\ Emean = 1/16 \end{cases} \tag{9}$$

It is worth noting that (6) was already used in [27] and [28]; it is also worth observing that the approximate 2×2 sub-multiplier of [10] can be obtained by using this approximate 2/1 compressor. In [25] an approximate half-adder is proposed, where an OR gate is used as sum output and an additional AND gate produces the carry output. The last column in Table I shows the error between exact and approximate half-adder of [25]. As it can be observed, design [25], while using an additional gate, gives the same error probability (1/16) and absolute mean error (1/16) as the proposed 2/1 approximate compressor.

B. Approximate 3/2 Compressor

Using (4), the sum of three partial products is written as:

$$S = \sum \{p_0, p_1, p_2\} = \sum \{p_0 p_1, p_0 + p_1, p_2\} \tag{10}$$

We can apply the AND, OR recoding of (4) again, to the first and last term at the right-hand side of (10):

$$S = \sum \{p_0 p_1 p_2, p_0 p_1 + p_2, p_0 + p_1\} \tag{11}$$

The probability of the $p_0 p_1 p_2$ term in (4), involving the AND of three partial products, is very low:

$$P\left(p_0 p_1 p_2\right) = 1/64 \tag{12}$$

Thus, we can neglect this low-probability term obtaining:

$$S_{APP} = \sum \{p_0 p_1 + p_2, p_0 + p_1\} \tag{13}$$

TABLE II
APPROXIMATE 3/2 COMPRESSOR

p_2	p_1	p_0	w_2	w_1	S	S_{APP}	Err_i	$P(Err_i)$	S_{APP} [25]
0	0	0	0	0	0	0	0	-	0
0	0	1	0	1	1	1	0	-	1
0	1	0	0	1	1	1	0	-	1
0	1	1	1	1	2	2	0	-	2
1	0	0	1	0	1	1	0	-	1
1	0	1	1	1	2	2	0	-	2
1	1	0	1	1	2	2	0	-	1
1	1	1	1	1	3	2	1	1/64	2

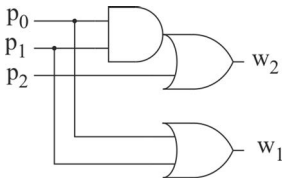

Fig. 2. Schematic of the proposed approximate 3/2 compressor.

With (13) we approximate a 3/2 compressor (full-adder) with an OR gate and an AND-OR gate. Table II shows the behavior of the approximate 3/2 compressor. We have $Err = 1$ only when $p_2 = p_1 = p_0 = 1$, which occurs with probability 1/64. Therefore:

$$\text{(approx. 3/2 compressor)} \begin{cases} P_E = 1/64 \\ Emean = 1/64 \end{cases} \tag{14}$$

The schematic of the proposed approximate 3/2 compressor is shown in Fig. 2. Compared with the approximate full-adder proposed in [25], our circuit is both simpler ([25] requires an XOR gate) and more precise. The last column in Table II reports the arithmetic result computed by the approximate 3/2 compressor in [25]. As it can be observed, design [25] gives an error for two inputs combinations. The first input combination, "110", occurs with probability: (1/4)(1/4)(3/4) = 3/64 (please, recall that from (3) the probability that each partial product is high is 1/4 and hence the probability that each partial product is low is 3/4). The second input combination, "111", occurs with probability: (1/4)(1/4)(1/4) = 1/64. From (8), the error probability of the approximate 3/2 compressor [25] is: $P_E = 3/64 + 1/64 = 4/64$. The mean error is obtained by observing that the error is +1 in both input combinations (the result is under-estimated by 1). Thus, we have: $Emean = 3/64 + 1/64 = 4/64$.

C. Approximate 4/2 Compressor

The sum of four partial products can be written, with the help of (4), as:

$$S = \sum \{p_0 p_1, p_0 + p_1, p_2 p_3, p_2 + p_3\} \tag{15}$$

We can apply the AND, OR recoding again, obtaining:

$$S = \sum \{(p_0 p_1)(p_2 + p_3), (p_2 p_3)(p_0 + p_1), (p_0 p_1) + (p_2 + p_3), (p_2 p_3) + (p_0 + p_1)\} \tag{16}$$

The probability of the first two terms in (16), involving the AND of three partial products, is very low (see (12)). Thus, we can obtain an approximate 4/2 compressor by neglecting these low-probability terms, as follows:

$$S_{APP} = \sum \{p_0 p_1 + p_2 + p_3, p_2 p_3 + p_0 + p_1\} \tag{17}$$

TABLE III
APPROXIMATE 4/2 COMPRESSOR

p_3	p_2	p_1	p_0	w_2	w_1	S	S_{APP}	Err	$P(Err_i)$	S_{APP} [23]	S_{APP} [25]	S_{APP} [24]
0	0	0	0	0	0	0	0	0	-	1	0	0
0	0	0	1	0	1	1	1	0	-	1	1	1
0	0	1	0	0	1	1	1	0	-	1	1	1
0	0	1	1	1	1	2	2	0	-	1	2	2
0	1	0	0	1	0	1	1	0	-	1	1	1
0	1	0	1	1	1	2	2	0	-	2	1	1
0	1	1	0	1	1	2	2	0	-	2	1	1
0	1	1	1	1	1	3	2	1	3/256	3	3	3
1	0	0	0	1	0	1	1	0	-	1	1	1
1	0	0	1	1	1	2	2	0	-	2	1	1
1	0	1	0	1	1	2	2	0	-	2	1	1
1	0	1	1	1	1	3	2	1	3/256	3	3	3
1	1	0	0	1	1	2	2	0	-	1	2	2
1	1	0	1	1	1	3	2	1	3/256	3	3	3
1	1	1	0	1	1	3	2	1	3/256	3	3	3
1	1	1	1	1	1	4	2	2	1/256	3	3	2

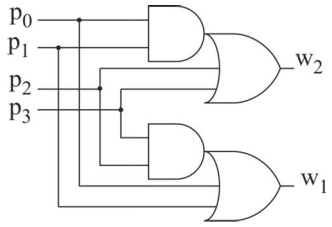

Fig. 3. Schematic of the proposed approximate 4/2 compressor.

Table III shows the behavior of the proposed approximate 4/2 compressor. For four p_i combinations, we have an error of 1, while there is a unique partial products combination resulting in an error of 2. The error probability and mean error are easily obtained as:

$$\text{(approx. 4/2 compressor)} \begin{cases} P_E = 13/256 \\ Emean = 14/256 \end{cases} \tag{18}$$

Fig. 3 shows the schematic of the proposed approximate 4/2 compressor, using two AND-OR gates.

The last three columns in Table III report the arithmetic result computed by previously proposed approximate 4/2 compressor. In every case the compressor proposed in this paper requires simpler hardware and provides smaller error probability and mean error. The circuit proposed in [23] (design 2) uses two XOR and four NOR gates, giving an error in only four entries of the table. It exhibits a rather large error probability $P_E = 100/256$ (mainly due to the error in the first row of the table occurring with probability 81/256), with a mean error $Emean = -62/256$. The approximate compressor in [25] (requiring two XOR and several AND-OR gates) gives error in five entries of the table with $P_E = Emean = 37/256$. The approximate 4/2 compressor proposed in [24] (structure 4) still requires two XOR gates, is slightly simpler than [25], and exhibits $P_E = 37/256$, $Emean = 38/256$.

Data in Table III highlight the main feature of the proposed approach: errors in our approximate compressor are present only in the low-probability cases in which three or more inputs are '1'. In [23]–[25], instead, errors are more randomly spread across the table, resulting in larger overall error probability and mean error.

TABLE IV
APPROXIMATE 5/3 COMPRESSOR

p_4	p_3	p_2	p_1	p_0	w_3	w_2	w_1	S	S_{APP}	Err	$P(Err_i)$	S_{APP} [26] #1	S_{APP} [26] #2	S_{APP} [26] #3	S_{APP} [26] #4
0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
0	0	0	0	1	1	0	0	1	1	0	-	1	1	1	1
0	0	0	1	0	1	0	0	1	1	0	-	1	1	1	1
0	0	0	1	1	1	0	1	2	2	0	-	2	2	2	0
0	0	1	0	0	0	0	1	1	1	0	-	1	1	1	3
0	0	1	0	1	1	0	1	2	2	0	-	2	2	2	2
0	0	1	1	0	1	0	1	2	2	0	-	2	2	2	2
0	0	1	1	1	1	0	1	3	2	1	9/1024	3	3	3	3
0	1	0	0	0	0	0	1	1	1	0	-	1	1	1	3
0	1	0	0	1	1	0	1	2	2	0	-	2	0	0	2
0	1	0	1	0	1	0	1	2	2	0	-	2	0	0	2
0	1	0	1	1	1	0	1	3	2	1	9/1024	3	3	3	3
0	1	1	0	0	0	1	1	2	2	0	-	6	0	0	0
0	1	1	0	1	1	1	1	3	3	0	-	7	3	3	1
0	1	1	1	0	1	1	1	3	3	0	-	7	3	3	1
0	1	1	1	1	1	1	1	4	3	1	3/1024	4	2	6	4
1	0	0	0	0	0	1	0	1	1	0	-	1	3	3	1
1	0	0	0	1	1	1	0	2	2	0	-	2	2	2	0
1	0	0	1	0	1	1	0	2	2	0	-	2	2	2	0
1	0	0	1	1	1	1	1	3	3	0	-	3	5	1	1
1	0	1	0	0	0	1	1	2	2	0	-	2	2	2	2
1	0	1	0	1	1	1	1	3	3	0	-	3	5	1	3
1	0	1	1	0	1	1	1	3	3	0	-	3	5	1	3
1	0	1	1	1	1	1	1	4	3	1	3/1024	0	4	4	6
1	1	0	0	0	0	1	1	2	2	0	-	2	2	2	2
1	1	0	0	1	1	1	1	3	3	0	-	3	3	3	3
1	1	0	1	0	1	1	1	3	3	0	-	3	3	3	3
1	1	0	1	1	1	1	1	4	3	1	3/1024	0	4	4	6
1	1	1	0	0	0	1	1	3	2	1	9/1024	7	3	3	1
1	1	1	0	1	1	1	1	4	3	1	3/1024	4	4	4	4
1	1	1	1	0	1	1	1	4	3	1	3/1024	4	4	4	4
1	1	1	1	1	1	1	1	5	3	2	1/1024	5	5	5	5

D. Approximate 5/3 Compressor

The design of approximate 5/3 compressor follows the same approach outlined for 3/2 and 4/2 approximate compressors. In a first step, we use (4) for all but the last partial products:

$$S = \sum \{p_0 p_1,\, p_0 + p_1,\, p_2 p_3,\, p_2 + p_3,\, p_4\} \tag{19}$$

In the second step, we apply the AND, OR recoding of (4) again, to the first and the fourth terms and to the third and the fifth terms at the right-hand side of (19):

$$S = \sum \{p_0 p_1 (p_2 + p_3),\, p_0 p_1 + p_2 + p_3,\, p_0 + p_1,\, p_2 p_3 p_4,\, p_2 p_3 + p_4\} \tag{20}$$

In this way, we obtain two terms where the AND of three partial products appears. The probability of these terms is very low; hence we neglect these terms and obtain:

$$S_{APP} = \sum \{p_0 p_1 + p_2 + p_3,\, p_0 + p_1,\, p_2 p_3 + p_4\} \tag{21}$$

Table IV shows the behavior of the proposed approximate 5/3 compressor. Again, there is a unique partial products combination resulting in an error of 2, having a very low probability of 1/1024. From Table IV, the error probability and mean error are obtained as:

$$\text{(approx. 5/3 compressor)} \begin{cases} P_E = 43/1024 \\ Emean = 44/1024 \end{cases} \tag{22}$$

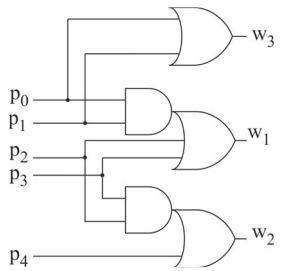

Fig. 4. Schematic of the proposed approximate 5/3 compressor.

The schematic of the proposed 5/3 compressor is shown in Fig. 4.

It is worth noting that an approximate 5/3 compressor can also be obtained with an approximate 2/1 and an approximate 3/2 compressor. This results in a slightly simpler circuit than Fig. 4 (output w_1 is the OR of p_2, p_3) having, however, larger error probability ($P_E = 52/1024$) and mean error ($Emean = 56/1024$) compared to (22).

The last four columns in Table IV report the arithmetic result computed by the approximate 5/3 compressors in [26]. The circuits in [26] are much more complicated compared to Fig. 4, using several XOR gates and multiplexers. The designs in [26] exhibit also large error probability, with errors scattered in several entries of the table.

E. Approximate 6/3 Compressor

As shown for the 4/2 approximate compressors, we use (4) for all the partial products of the 6/3 compressor:

$$S = \sum \{p_0p_1,\ p_0+p_1,\ p_2p_3,\ p_2+p_3,\ p_4p_5,\ p_4+p_5\} \quad (23)$$

Then, we apply the AND, OR recoding of (4) again, to the first and the fourth terms, to the third and the sixth terms and to the fifth and the second terms at the right-hand side of (23):

$$\begin{aligned} S = \sum \{&p_0p_1\,(p_2+p_3),\ p_0p_1+p_2+p_3,\\ &p_2p_3\,(p_4+p_5),\ p_2p_3+p_4+p_5,\\ &p_4p_5\,(p_0+p_1),\ p_4p_5+p_0+p_1\} \end{aligned} \quad (24)$$

By neglecting the low-probability terms where the AND of three partial products appears, we have:

$$\begin{aligned} S_{APP} = \sum \{&p_0p_1+p_2+p_3,\ p_2p_3+p_4+p_5,\\ &p_4p_5+p_0+p_1\} \end{aligned} \quad (25)$$

Table V shows the behavior of the proposed approximate 6/3 compressor. Please note that this table reports only the partial products combinations resulting in an output error (in the other cases the error is zero). From Table V, we have:

$$\text{(approx. 6/3 compressor)} \begin{cases} P_E = 316/4096 \\ Emean = 336/4096 \end{cases} \quad (26)$$

The schematic of the proposed approximate 6/3 compressor is shown in Fig. 5.

F. Higher-Order Approximate Compressors

Higher order approximate compressors (8/4, 10/5 etc.), can be designed following an approach like the one shown in the previous subsections. As an example, an approximate 8/4 compressor is obtained as:

$$\begin{aligned} S_{APP} = \sum \{&(p_0p_1)+(p_2+p_3),\ (p_2p_3)+(p_0+p_1),\\ &(p_4p_5)+(p_6+p_7),\ (p_6p_7)+(p_4+p_5)\} \end{aligned} \quad (27)$$

TABLE V
APPROXIMATE 6/3 COMPRESSOR

p_5	p_4	p_3	p_2	p_1	p_0	w_3	w_2	w_1	S	S_{APP}	Err	$P(Err_i)$
0	0	0	1	1	1	0	1	1	3	2	1	27/4096
0	0	1	0	1	1	0	1	1	3	2	1	27/4096
0	0	1	1	1	1	1	1	1	4	3	1	9/4096
0	1	0	1	1	1	1	1	1	4	3	1	9/4096
0	1	1	0	1	1	1	1	1	4	3	1	9/4096
0	1	1	1	0	0	1	0	1	3	1	1	27/4096
0	1	1	1	0	1	1	1	1	4	3	1	9/4096
0	1	1	1	1	0	1	1	1	4	3	1	9/4096
0	1	1	1	1	1	1	1	1	5	3	2	3/4096
1	0	0	1	1	1	1	1	1	4	3	1	9/4096
1	0	1	0	1	1	1	1	1	4	3	1	9/4096
1	0	1	1	0	0	1	0	1	3	2	1	27/4096
1	0	1	1	0	1	1	1	1	4	3	1	9/4096
1	0	1	1	1	0	1	1	1	4	3	1	9/4096
1	0	1	1	1	1	1	1	1	5	3	2	3/4096
1	1	0	0	0	1	1	1	0	3	2	1	27/4096
1	1	0	0	1	0	1	1	0	3	2	1	27/4096
1	1	0	0	1	1	1	1	1	4	3	1	9/4096
1	1	0	1	0	1	1	1	1	4	3	1	9/4096
1	1	0	1	1	0	1	1	1	4	3	1	9/4096
1	1	0	1	1	1	1	1	1	5	3	2	3/4096
1	1	1	0	0	1	1	1	1	4	3	1	9/4096
1	1	1	0	1	0	1	1	1	4	3	1	9/4096
1	1	1	0	1	1	1	1	1	5	3	2	3/4096
1	1	1	1	0	0	1	1	1	4	3	1	9/4096
1	1	1	1	0	1	1	1	1	5	3	2	3/4096
1	1	1	1	1	0	1	1	1	5	3	2	3/4096
1	1	1	1	1	1	1	1	1	6	3	3	1/4096

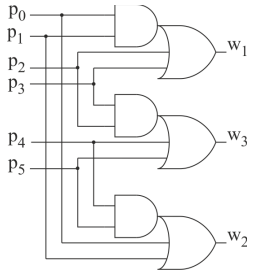

Fig. 5. Schematic of the proposed approximate 6/3 compressor.

TABLE VI
RULES TO COMPOSE HIGH-ORDER ($n > 6$) COMPRESSORS

n mod 4	# 3/2 compr.	# 4/2 compr.	# 5/3 compr.	# 6/3 compr.	*Emean*
0	-	$n/4$	-	-	$(14/256)\cdot(n/4)$
1	-	$\lfloor n/4 \rfloor - 1$	1	-	$(14/256)\cdot\lfloor n/4 \rfloor - 3/256$
2	-	$\lfloor n/4 \rfloor - 1$	-	1	$(14/256)\cdot\lfloor n/4 \rfloor + 7/256$
3	1	$\lfloor n/4 \rfloor$	-	-	$(14/256)\cdot\lfloor n/4 \rfloor + 1/64$

This corresponds to using two approximate 4/2 compressors. Similar results are obtained in other cases: an approximate 7/4 compressor includes a 4/2 and a 3/2 approximate compressor, while a 10/5 approximate compressor uses a 6/3 and a 4/2 approximate compressor.

The general rule to compose approximate compressors with $n > 6$ input bits and $\lceil n/2 \rceil$ output bits by using smaller approximate sub-compressors is summarized in Table VI.

We can easily compute the mean error by summing the contribution of each sub-compressor, since the sub-compressors

TABLE VII
HIGHER ORDER COMPRESSOR

Order	*Composition*	P_E	*Emean*
7/4	4/2 3/2	6.5%	0.07
8/4	4/2 4/2	9.9%	0.11
9/5	4/2 5/3	9.1%	0.10
10/5	4/2 6/3	12.4%	0.14
11/6	4/2 4/2 3/2	11.3%	0.13
12/6	4/2 4/2 4/2	14.5%	0.16
13/7	4/2 4/2 5/3	13.7%	0.15
14/7	4/2 4/2 6/3	16.8%	0.19
15/8	4/2 4/2 4/2 3/2	15.8%	0.18
16/8	4/2 4/2 4/2 4/2	18.8%	0.22
17/9	4/2 4/2 4/2 5/3	18.1%	0.21
18/9	4/2 4/2 4/2 6/3	21.1%	0.25
19/10	4/2 4/2 4/2 4/2 3/2	20.1%	0.23
20/10	4/2 4/2 4/2 4/2 4/2	22.9%	0.27

inputs are independent from each other. The result is reported in the last column of Table VI.

The error probability P_E can also be calculated in closed form. When two sub-compressors (A and B) are exploited, the error probability of the resulting compressor is:

$$P_E = P_E(A) + P_E(B) - P_E(A)\,P_E(B) \tag{28}$$

If three compressors are exploited (A, B, and C) we have:

$$\begin{aligned} P_E = {} & P_E(A) + P_E(B) + P_E(C) \\ & - P_E(A)\,P_E(B) - P_E(A)\,P_E(C) - P_E(B)\,P_E(C) \\ & + P_E(A)\,P_E(B)\,P_E(C) \end{aligned} \tag{29}$$

The equation can be extended to the general case and is known as the inclusion-exclusion principle, [30]. Table VII reports decomposition, error probability and mean error for approximate compressors from 7/4 through 20/10.

The results reported in Table VII show that i) the error probability and the mean error tend to increase with the compressor size and ii) approximate compressors having an odd number of inputs tend to perform better compared to compressors having an even number of input bits. The first characteristic is due to the error contribution of the sub-compressors: the 20/10 compressor uses five 4/2 sub-compressors and hence has a larger error compared, say, to the 16/8 compressor that includes four 4/2 sub-compressors. The second characteristic is related to the fact that we considered approximated compressors with j inputs and $\lceil j/2 \rceil$ outputs. Therefore, when j is even the "compression ratio" (the ratio between input and output bits) is 2, while the compression ratio is lower, $2j/(j+1)$, when j is an odd number. Lower compression ratio usually provides lower error rate.

G. Using Approximate Compressors in Signed Multipliers

Fig. 6 shows the partial product matrix of a signed multiplier. Some of the partial products (indicated as q_i or as complemented partial products in the following) are obtained as the NAND of the operands bits. Assuming, as before, that the input bits x_i and y_j are uniformly and independently distributed, the probability that a complemented partial product is high is not given by (3) and is instead:

$$P(q_i) = 3/4 \tag{30}$$

Consequently, the error probability of approximate compressors having a complemented partial product as input is

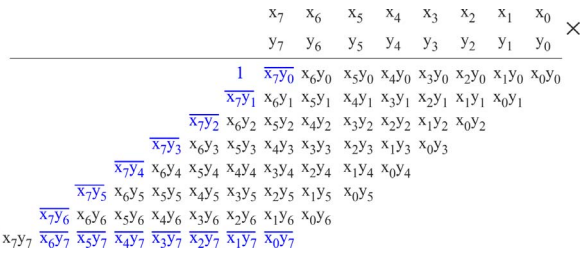

Fig. 6. Partial product matrix of a signed multiplier. Complemented partial products are highlighted in blue.

TABLE VIII
APPROXIMATE 4/2 COMPRESSOR WITH A COMPLEMENTED PARTIAL PRODUCT

q_3	p_2	p_1	p_0	w_2	w_1	S	S_{APP}	*Err*	$P(Err_i)$
0	1	1	1	1	1	3	2	1	1/256
1	0	1	1	1	1	3	2	1	9/256
1	1	0	1	1	1	3	2	1	9/256
1	1	1	0	1	1	3	2	1	9/256
1	1	1	1	1	1	4	2	2	3/256

TABLE IX
ERROR PERFORMANCE OF PROPOSED APPROXIMATE COMPRESSORS WITH A COMPLEMENTED PARTIAL PRODUCT

Order	P_E	*Emean*
2/1	18.7%	0.19
3/2	4.7%	0.047
4/2	12.1%	0.13
5/3	7.13%	0.074
6/3	14.7%	0.16

different with respect to the values computed in previous subsections.

As an example, let us consider the proposed approximate 4/2 compressor driven by a complemented partial product. Table VIII shows the partial products combinations resulting in an output error and the corresponding probabilities. We obtain: $P_E = 31/256$ and $Emean = 34/256$. While larger than (18), these values remain better than [25] (having $P_E = Emean = 63/256$ in presence of a complemented partial product) and [24] ($P_E = 63/256$, $Emean = 64/256$). Also the circuit proposed in [23] has larger error probability when a complemented partial product is an input ($P_E = 60/256$), while the mean error is $Emean = 6/256$

Table IX reports error probability and mean error of the proposed approximate compressors having a complemented partial product as input. Higher-order compressors are constructed, as shown previously, by using smaller sub-compressors.

As it can be observed in Fig. 6, the complemented partial products are in the most significant columns of the PPM. As it will be shown in the next Section, the proposed algorithm for the allocation of approximate compressors minimizes the insertion of approximate compressors in this region of the PPM. Thus, only a few approximate compressors will be driven by complemented partial products. It is worth noting that the proposed algorithm inserts an approximate compressor in the n-th column of the PPM (the one that includes the partial products $\overline{x_7 y_0}$ and $\overline{x_0 y_7}$ in the example of Fig. 6). This is the only approximate compressor driven by two complemented

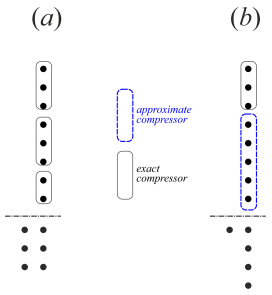

Fig. 7. (a) Exact compressors (two full-adders and one half-adder) reduce a partial products column of height $h = 8$ to $h_{new} = 3$ (b) Example of reduction using one exact full-adder and one approximate 5/3 compressor.

partial products and is implemented by using smaller sub-compressors.

IV. Algorithm for the Allocation of Approximate Compressors in Binary Multipliers

In an approximate multiplier, approximate compressors substitute some of the exact compressors. The allocation of the approximate compressors is a critical factor as it affects both the electrical performance and the arithmetic accuracy of the binary multipliers they compose.

In order to explain the proposed algorithm, let us consider the idealized condition in which there are only h equal-weight partial products to be summed. By using exact half-adders and full-adders these partial products can be reduced to a column of height h_{new}, with:

$$h_{new} \geq \lceil h/3 \rceil \tag{31}$$

The number of full-adders needed to achieve this reduction is given by:

$$FA = \left\lfloor \frac{h - h_{new}}{2} \right\rfloor \tag{32}$$

while the number of required half adders is:

$$HA = \begin{cases} 1 & \text{if: } \dfrac{h - h_{new}}{2} > FA \\ 0 & \text{otherwise} \end{cases} \tag{33}$$

The total number of compressors (full-adders plus half-adders) can be written as:

$$C = FA + HA = \left\lceil \frac{h - h_{new}}{2} \right\rceil \tag{34}$$

As an example, Fig. 7(a) shows the reduction of a column of $h = 8$ partial products to $h_{new} = 3$, with $FA = 2$ and $HA = 1$.

The Dadda multiplier uses this approach to reduce the multiplier PPM. The maximum height of the PPM, h_{max}, follows the series $\{2, 3, 4, 6, 9, 13, 19, 28, \ldots\}$. For instance, in the case of a 12-bit multiplier, the PPM starts with maximum height $h_{max} = 12$ and evolves as: $\{12 \rightarrow 9 \rightarrow 6 \rightarrow 4 \rightarrow 3 \rightarrow 2\}$ so that, after five reduction steps, it is reduced to the two operands summed by the vector merging adder.

In the proposed approach we use not only exact compressors, but also approximate compressors (as shown in the example of Fig. 7(b)). In each reduction step, using approximate compressors, the maximum PPM height is reduced from h_{max} to $h_{max,next}$, where $h_{max,next} \geq \lceil h_{max}/2 \rceil$ is chosen so as to skip one level in the Dadda series, thereby obtaining a faster multiplier. Let us consider, as an example, a 12×12 multiplier using approximate compressor in the first reduction step only. Starting from $h_{max} = 12$ we choose $h_{max,next} = 6$, so that the maximum PPM height evolves as: $\{12 \rightarrow 6 \rightarrow 4 \rightarrow 3 \rightarrow 2\}$, saving one compression step. By exploiting approximate compressors also in the second reduction step, the maximum PPM height evolves as: $\{12 \rightarrow 6 \rightarrow 3 \rightarrow 2\}$, saving two compression steps (at a price in terms of approximation error).

The targets of the proposed algorithm are:

a) Compress as much as possible the n least significant columns of the PPM, using approximate compressors. This maximizes the power saving while providing small error.
b) Insert, in the most significant columns of the PPM, the minimum number of approximate compressors needed to meet the target height $h_{max,next}$. This minimizes the error.
c) Avoid, if possible, the use of the approximate 2/1 compressor, due to its large *Emean* (9) error.

Let us indicate as $k = 1$ the rightmost column of the PPM, as LSP the least significant part of the PPM (composed by columns $k \in [1, n]$) and as MSP the most significant part of the PPM (composed by columns $k \in [n + 1, 2n - 1]$). Let us also indicate as $h(k)$ the height of the k-th column and as $j(k)$ the size of the approximate compressor inserted in column k (e.g. $j(18) = 12$ indicates that a 12/6 approximate compressor is inserted in column 18).

Approximate compressors are inserted in the LSP as follows:

$$j(k) = \begin{cases} 0 & \text{if: } h(k) \leq 2 \\ h(k) & \text{if: } h(k) > 2 \end{cases} \tag{35}$$

This equation states that the columns of the LSP of height up to two are not compressed while the remaining columns are compressed as much as possible.

The MSP is compressed using both approximate and exact compressors. To obtain the number of exact compressors to be used in the k-th column, there are two cases to be considered: i) the column can be compressed by using only exact compressors or: ii) column compression requires some approximate compressors.

In case i), the number of exact compressors to be used is obtained from (34) considering that every exact compressor in column $k-1$ generates a carry for the next column:

$$C^*(k) = \left\lceil \frac{h(k) - \left(h_{max,next} - C(k-1)\right)}{2} \right\rceil \tag{36}$$

where $C(n) = 0$, since column n belongs to LSP and is reduced with approximate compressors that do not generate any carry.

In case ii), the number of exact compressors to be used in column k is obtained from the analysis of column $k+1$. By substituting k with $k+1$ in (36) and with simple algebra one obtains:

$$C^{**}(k) = 2C(k+1) - h(k+1) + h_{max,next} \tag{37}$$

The number of exact compressors that can be placed in column $k+1$ should be compatible to (34). Thus, one can write:

$$C(k+1) = C_{max} \tag{38}$$

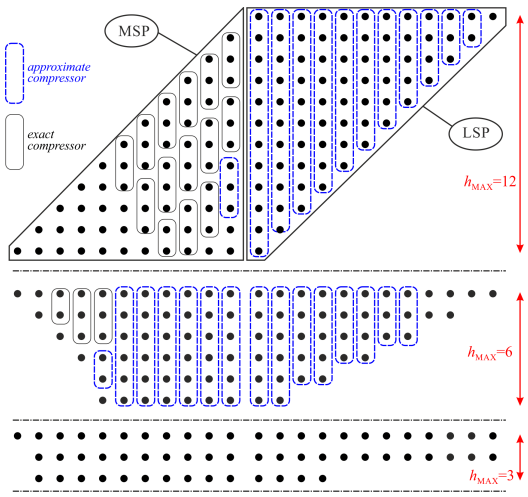

Fig. 8. Reduction of the PPM of a 12 × 12 multiplier, using the proposed algorithm. Each dot represents a partial product. In the first reduction step $h_{max,next} = 6$ is assumed, while the second reduction step uses $h_{max,next} = 3$.

with:

$$C_{max} = h_{max,next} - \left\lceil \frac{h\,(k+2)}{3} \right\rceil \tag{39}$$

Let us consider, as an example, the 12×12 multiplier shown in Fig. 8 where in the first reduction step $h_{max,next} = 6$ is assumed. In column 13, (36) yields: $C^*(13) = 3$, while (37)-(39) yield: $C^{**}(13) = 2$. Since $C^{**} < C^*$ the case ii) holds and the considered column cannot be reduced by using exact compressors only. Therefore, only two exact full-adders ($C^{**} = 2$) are allocated in this column. By indicating as $FA(k)$ the number of exact full-adders allocated in column k, the size of the approximate compressor is obtained from the condition:

$$h\,(k) - 2FA\,(k) - \lfloor j\,(k)\,/2 \rfloor + C\,(k-1) = h_{max,next} \tag{40}$$

that yields:

$$j\,(k) = 2\left(h\,(k) - h_{max,next} - 2FA\,(k) + C\,(k-1)\right) \tag{41}$$

When $j\,(k) = 2$ and there is room enough in column k, an approximate 3/2 compressor is inserted in the PPM, to limit the use of approximate 2/1 compressors. In the considered example, $j\,(13) = 3$ is obtained.

The pseudo-code of Algorithm 1 details the algorithm used to allocate exact and approximate compressors in each column of the PPM.

Fig. 8 shows the transformed PPM when the proposed algorithm is used in a 12 × 12 multiplier for the first reduction step (with $h_{max,next} = 6$) and for the second reduction step ($h_{max,next} = 3$).

V. Implementation Results

The approximate compressors described in Section III and the algorithm of Section IV have been exploited to design 8 × 8, 12 × 12, 16 × 16, and 20 × 20, binary multipliers. To investigate the trade-off between error and electrical performance, each multiplier is designed by using approximate compressors either in the first reduction step only or in the first and in the second reduction steps of the PPM. Moreover, we analyze the case in which the $n-1$ less significant columns of the PPM are truncated (not formed), since columns truncation also represents a straightforward way to trade speed and power for precision in a binary multiplier. In summary four different versions of each multiplier have been implemented, as summarized in Fig. 9.

Algorithm 1 Allocates Exact and Approximate Compressors to Reduce Multiplier PPM

Input: Partial Product Matrix, given by $h(k)$ for $k = 1..2n-1$; Maximum height for the transformed PPM: $h_{max,next}$

Output (for each PPM column): allocated approximate compressors $j(k)$; allocated exact full-adders $FA(k)$ and half-adders $HA(k)$; transformed PPM $h_{next}(k)$

Steps:

1. for $k = 1{:}n$ /* *LSP* */
2. $FA(k) = 0$; $HA(k) = 0$; $C\,(k) = 0$
3. if $h(k) <= 2$
4. $j(k) = 0$; $h_{next}(k) = h(k)$;
5. else
6. $j(k) = h(k)$; $h_{next}(k) = \lceil k/2 \rceil$;
7. end if;
8. end for;
9. for $k = n{+}1{:}2n{-}3$ /* *MSP* */
10. $FA(k) = 0$; $HA(k) = 0$; $j(k) = 0$; $C(k) = 0$; $h_{next}(k) = h(k)$;
11. compute $C*$ from (36);
12. compute C^{**} from (37)-(39);
13. if $C^{**} < C^*$ /* *approx. compressors required* */
14. $C\,(k) = C^{**}$; $FA(k) = C^{**}$;
15. $j(k) = 2(h(k) - h_{max,next} - 2FA(k) + C(k-1))$;
16. if $(j(k) == 2)$ and $(h(k) - 3FA(k) >= 3)$
17. $j(k) = 3$;
18. end if;
19. else /* *only exact compressors* */
20. $C(k) = C^*$;
21. if $C(k) > 0$ then
22. $FA(k) = \lceil (h(k) - h_{max,next} + C(k-1))/2 \rceil$;
23. $HA(k) = C(k) - FA(k)$;
24. end if;
25. end if;
26. $h_{next}(k) = h(k) - 2FA(k) - HA(k) - \lceil j(k)/2 \rceil + C(k-1)$;
27. end for;

	PPM not truncated	PPM truncated
approximate compressors in the first reduction step	*1StepFull*	*1StepTrunc*
approximate compressors in both first and second reduction steps	*2StepsFull*	*2StepsTrunc*

Fig. 9. Nomenclature used for the four different versions of each proposed approximate multiplier.

The proposed approximate multipliers are compared against exact multipliers and against the approximate multipliers proposed in [10], in [29] (in the four versions AM2, AM3, AM4, and AM5) and in [27] using 2, 3 or 4-inputs OR gates as approximate compressor (these circuits are named in the following as: L = 2, L = 3 and L = 4, respectively).

A. Electrical Performance

All the investigated circuits have been described in HDL language and synthesized in TSMC 40 nm technology, impos-

TABLE X
ELECTRICAL PERFORMANCE OF EXACT AND APPROXIMATE MULTIPLIERS

		exact	This Paper				[27]			[29]				[10]
			1Step Full	1Step Trunc	2Steps Full	2Steps Trunc	(L=2)	(L=3)	(L=4)	AM2	AM3	AM4	AM5	
8×8	min delay (ps)	564	500 (-11%)	500 (-11%)	419 (-26%)	375 (-34%)	491 (-13%)	400 (-29%)	311 (-45%)	596 (-6%)	543 (-4%)	550 (-2%)	560 (-1%)	522 (-7%)
	Area (μm^2) delay=564ps	986	524 (-47%)	310 (-69%)	428 (-57%)	171 (-83%)	296 (-70%)	279 (-72%)	144 (-85%)	1033(*) (+5%)	710 (-28%)	563 (-43%)	618 (-37%)	493 (-50%)
	Power ($\mu W/MHz$) delay=564ps	1.29	0.81 (-37%)	0.47 (-64%)	0.72 (-44%)	0.30 (-77%)	0.54 (-58%)	0.50 (-61%)	0.34 (-74%)	1.11(*) (-14%)	0.95 (-26%)	0.82 (-36%)	0.99 (-23%)	0.76 (-41%)
12×12	min delay (ps)	722	642 (-11%)	618 (-14%)	550 (-24%)	538 (-25%)	600 (-17%)	538 (-25%)	453 (-37%)	760 (+5%)	675 (-7%)	687 (-5%)	675 (-7%)	654 (-9%)
	Area (μm^2) delay=722ps	1379	1041 (-24%)	589 (-57%)	681 (-51%)	365 (-74%)	699 (-49%)	428 (-69%)	327 (-76%)	1904 (+38%)	1214 (-12%)	1069 (-22%)	1120 (-19%)	949 (-31%)
	Power ($\mu W/MHz$) delay=722ps	1.94	1.65 (-15%)	0.91 (-53%)	1.22 (-37%)	0.66 (-66%)	1.25 (-36%)	0.85 (-56%)	0.71 (-63%)	2.22 (+14%)	1.76 (-9%)	1.61 (-17%)	1.78 (-8%)	1.51 (-22%)
16×16	min delay (ps)	800	746 (-7%)	730 (-9%)	667 (-17%)	650 (-19%)	695 (-13%)	618 (-23%)	553 -31%)	860 (+8%)	800 (0%)	769 (-4%)	800 (0%)	733 (-8%)
	Area (μm^2) delay=800ps	2595	1859 (-28%)	1002 (-61%)	1147 (-56%)	700 (-73%)	1295 (-50%)	934 (-64%)	656 (-75%)	3103 (*) (+20%)	2266 (-13%)	2060 (-21%)	2173 (-16%)	1716 (-34%)
	Power ($\mu W/MHz$) delay=800ps	3.58	2.94 (-18%)	1.56 (-57%)	2.01 (-56%)	1.29 (-64%)	2.2 (-39%)	1.66 (-54%)	1.32 (-63%)	3.76 (*) (+5%)	3.37 (-6%)	3.15 (-12%)	3.51 (-2%)	2.67 (-25%)
20×20	min delay (ps)	883	813 (-8%)	788 (-11%)	775 (-12%)	731 (-17%)	782 (-11%)	700 (-21%)	615 (-30%)	923 (+5%)	850 (-4%)	850 (-4%)	845 (-4%)	800 (-9%)
	Area (μm^2) delay=883ps	4049	2853 (-30%)	1482 (-63%)	1760 (-57%)	1022 (-75%)	1996 (-51%)	1499 (-63%)	1042 (-74%)	4733 (*) (+17%)	3535 (-13%)	3206 (-21%)	3362 (-17%)	2713 (-33%)
	Power ($\mu W/MHz$) delay=883ps	5.82	4.51 (-22%)	2.35 (-63%)	3.18 (-45%)	1.90 (-67%)	3.34 (-43%)	2.63 (-55%)	2.13 (-63%)	6.01 (*) (+3%)	5.26 (-10%)	5.00 (-14%)	5.53 (-5%)	4.24 (-27%)
Average Improvem.	delay	-	-9.3%	-11.3%	-19.6%	-23.7%	-13.6%	-24.5%	-35.8%	5.7%	-3.5%	-3.7%	-2.9%	-8.7%
	Area	-	-32.3%	-62.7%	-54.9%	-76.0%	-55.0%	-66.9%	-77.7%	19.8%	-16.3%	-26.7%	-22.3%	-37.0%
	Power	-	-23.1%	-58.2%	-42.6%	-68.5%	-43.7%	-56.5%	-65.9%	2.2%	-12.8%	-19.9%	-9.6%	-29.0%

(*) The circuit does not reach the timing of the exact multiplier. The Area and Power data refer to the circuit synthesized at its minimum delay.

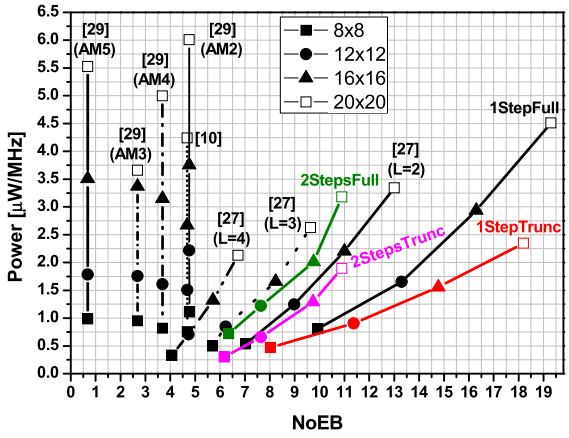

Fig. 10. Approximate multipliers compared in terms of power dissipation and NoEB.

ing proper timing constraints. In all the investigated circuits, the final carry-propagate adder (CPA) was implemented by using a fast parallel-prefix topology (Kogge-Stone). Table X summarizes the results. In this Table "min delay" is the minimum delay at which the circuits can be synthesized. To make the comparison meaningful, area and power are obtained by synthesizing all the circuits with the same timing constraint, equal to the delay of the exact multiplier. For all the approximate multipliers we have also calculated the variations with respect to the exact multiplier (reported as a percentage in Table X, where a negative percentage means improvement, while a positive one means worsening). The last three rows in Table X report, for each approximate multiplier, the improvements in terms of delay, area and power, averaged among the four considered sizes (8×8, 12×12, 16×16, and 20×20). As it can be observed, proposed approximate multipliers give evident advantages with respect to exact multipliers. The delay improvement ranges from about 9% for the 1StepFull topology, up to about 24% for the 2StepsTrunc. Area improvements is even larger (about 32% for 1StepFull, about 76% for 2StepsTrunc); similarly, power reduction is quite evident (about 23% for 1StepFull topology, about 68% for 2StepsTrunc). As expected, 2StepsTrunc gives the best electrical performance, while the smallest improvements are found for 1StepFull. The other two topologies are in between, with 2StepsFull slightly better in terms of speed and 1StepTrunc in terms of power.

To explain the improvements of proposed multipliers compared to the conventional design, let us observe that the critical path includes both the partial-products reduction tree and the final CPA.

In the 1StepFull and 2StepsFull topologies, the CPA has the same size as the conventional design, and the improvements are related only to the use of approximate compressors in place of standard compressors. These improvements are due to two factors. i) approximate compressors allow to reduce the number of stages (and hence of logic levels) compared to standard compressors. As an example, the 12×12 multiplier of Fig. 8 requires a total of 3 compression levels to obtain the inputs of the CPA (the critical path of the partial-products reduction tree includes a maximum of three compressors), while a standard Dadda multiplier requires 5 compression

levels. ii) the proposed approximate compressors are XOR-less and are hence faster (and smaller) compared to standard full-adders. For the 12×12 exact multiplier, the synthesis results show that the delay for partial products compression is 491ps, while CPA delay is 231ps; for the 2StepsFull the partial products compression delay is reduced to 259ps, while CPA delay is 291ps. Thus, speed improvement is mainly due to partial products reduction (the small difference in CPA delay can be attributed to the optimizations/cell resizing operated by synthesizer). In the 2StepsTrunc topology an additional speed improvement is related to the smaller CPA size. This improvement is limited when using parallel-prefix CPA, due to logarithmic dependence of delay to CPA size. For the 12×12 multiplier, the synthesis results reveal that the CPA delay takes 237ps while partial product compression delay is 301ps (the slight variation of the partial product compression delay, between 2StepFull and 2StepsTrunc, is due to different optimization choices of the synthesizer).

The approximate multipliers proposed in [27] also show good electrical performance. The architecture $L = 4$ is the fastest among investigated circuits, with power behavior comparable to our 2StepsTrunc circuit. The architectures $L = 2$ and $L = 3$ show average power dissipation like 2StepsFull and 1StepsTrunc. The approximate multipliers of [29] are less effective, with the AM2 topology slower and more power hungry than exact multiplier. The approximate multiplier proposed in [10] shows electrical performance comparable to our 1StepFull topology.

B. Error Performance

The error metrics considered in the following for the approximate multipliers are [6]:

- The probability of an incorrect result, Error Rate, (ER).
- The average value of the errors produced by the multiplier, Mean Error, (ME).
- The root mean square of the errors produced by the multiplier (ERMS). Instead of reporting ERMS values, we prefer to introduce the Number of Effective bits, NoEB, which for a $n \times n$ approximate multiplier is defined as: $NoEB = 2n - \log_2(1 + ERMS)$. We prefer to use this parameter instead of ERMS since it gives a handy indication of the number of output bits that are possibly free from error.

The error performance of the investigated approximate multipliers is shown in Table XI. Reported results have been obtained through Monte Carlo simulations (relative error 1% confidence level 99%). The last three rows in this Table report, for each approximate multiplier, the error metrics averaged among the four considered sizes (8×8, 12×12, 16×16, and 20×20). Please note that we defined the error as:

$$E = Y_{EXACT} - Y_{APPROX} \tag{42}$$

Where Y_{EXACT} is the output computed by the exact multiplier, while Y_{APPROX} is the approximate multiplier result. Negative ME values in Table XI refer to approximate multipliers that over-estimate the result.

The approximate multipliers proposed in this paper give very good error results. The 1StepFull topology consistently yields the lowest error rate and mean error compared to the other circuits, with a much larger number of effective bits. The 1StepTrunc also shows good ME and NoEB, with, however, a rather large error rate. Only the approximate multiplier of [27] with $L = 2$ shows error performance comparable to 2StepsFull and 2StepsTrunc. The other investigated circuits ([27] with $L = 3$ and $L = 4$, [29] and [10]) show a much larger mean error and a reduced NoEB, which is lower than 5 bits even for 20×20 multipliers.

The results in Table X and Table XI show that the circuits proposed in this paper provide very good error-electrical performance trade-off. For instance, the 1StepFull topology gives 23% average power saving with respect to exact multiplier with the best error performance with respect to the state of art. The 2StepsTrunc multiplier obtains error performance comparable to [27] $L = 2$, showing however, better power saving (68% with respect to 44%).

To better investigate the error-electrical performance trade-off, Fig. 10 shows the power dissipation values versus the NoEB for all the investigated approximate multipliers.

The proposed multipliers show excellent power-accuracy trade-off, compared to state of art multipliers, allowing targeting applications with different error resiliency attitude. Indeed, in Fig. 10, all proposed multipliers (with exception of 2StepsFull) constitute the rightmost plots, i.e. they exhibit higher NoEB for the same power.

The 1StepFull offers the best performance in terms of NoEB at moderate power dissipation improvement. The 1StepTrunc version reduces power consumption at expense of NoEB (which is however higher than previously proposed multipliers). Aggressive power saving is exhibited by 2StepsTrunc which gives the lowest power dissipation, at a price in terms of NoEB.

Let us consider, as an example, a 20×20 multiplier. Among previously proposed multipliers, the one showing better power-accuracy trade-off is [27] $L = 2$, showing 12.97 NoEB with 3.34 μW/MHz power dissipation. The 1StepTrunc topology gives both better NoEB (18.2) and lower power dissipation (2.35 μW/MHz); the 2StepsTrunc yields a much-reduced power dissipation (only 1.90 μW/MHz, i.e. 43% lower) with 10.92 NoEB.

Table XII reports the error performance of signed multipliers. Due to the presence of complemented terms (see Section III-G), the error performance of all the analyzed multipliers are worsened compared to Table XI, with the only exception of [27], [29]-AM5 that exhibits better performance in the signed scenario.

The proposed multipliers, the ones of [27] and [29]-AM3 exhibit a small NoEB degradation (e.g. for the 20×20 multiplier, the NoEB of 2StepsTrunc goes from 10.92 in the unsigned case, to 10.67 in the signed one). The multipliers [29]-AM2, [29]-AM4 and [10], show larger NoEB degradation. In any case, we can draw conclusions analogous to those of unsigned multipliers: the 1StepFull gives the best performance in terms of error probability, mean error and NoEB. The other proposed topologies outperform the state of art multiplier, giving error performance comparable with [27] $L = 2$.

VI. Applications

We have investigated the use of approximate multipliers in two applications: image filtering, and adaptive Least-Mean-Square (LMS).

TABLE XI
ERROR PERFORMANCE OF APPROXIMATE UNSIGNED MULTIPLIERS

		This Paper				[27]			[29]				[10]
		1Step Full	1Step Trunc	2Steps Full	2Steps Trunc	(L=2)	(L=3)	(L=4)	AM2	AM3	AM4	AM5	
8×8	ER	0.22	0.97	0.49	0.97	0.49	0.66	0.78	0.81	0.47	0.47	1	0.47
	ME	22.5	2.10×10^{2}	3.50×10^{2}	5.10×10^{2}	2.30×10^{2}	6.60×10^{2}	2.10×10^{3}	1.36×10^{3}	3.62×10^{3}	1.80×10^{3}	-7.17×10^{3}	9.00×10^{2}
	NoEB	9.93	8.03	6.35	6.17	7.03	5.7	4.06	4.77	2.68	3.69	0.68	4.69
12×12	ER	0.36	1	0.76	1	0.71	0.88	0.93	0.94	0.68	0.68	1	0.68
	ME	6.50×10^{2}	5.50×10^{3}	3.60×10^{4}	4.00×10^{4}	1.60×10^{4}	1.20×10^{5}	3.70×10^{5}	3.50×10^{5}	9.34×10^{5}	4.60×10^{5}	-1.85×10^{6}	2.30×10^{5}
	NoEB	13.34	11.37	7.68	7.64	8.98	6.23	4.74	4.76	2.68	3.69	0.68	4.69
16×16	ER	0.54	1	0.91	1	0.84	0.95	0.98	0.98	0.81	0.81	1	0.81
	ME	2.60×10^{4}	1.30×10^{5}	2.60×10^{6}	2.70×10^{6}	1.00×10^{6}	7.70×10^{6}	4.80×10^{7}	8.90×10^{7}	2.39×10^{8}	1.20×10^{8}	-4.74×10^{8}	6.00×10^{7}
	NoEB	16.31	14.77	9.76	9.74	10.97	8.24	5.72	4.76	2.68	3.69	0.68	4.68
20×20	ER	0.67	1	0.97	1	0.91	0.98	1	1	0.89	0.89	1	0.89
	ME	8.70×10^{5}	3.10×10^{6}	2.80×10^{8}	2.80×10^{8}	6.70×10^{7}	7.60×10^{8}	6.20×10^{9}	2.30×10^{10}	6.12×10^{10}	3.10×10^{10}	-1.21×10^{11}	1.50×10^{10}
	NoEB	19.31	18.2	10.92	10.92	12.97	9.64	6.72	4.76	2.68	3.68	0.68	4.68
Avg.	ER	0.45	0.99	0.78	0.99	0.74	0.87	0.92	0.93	0.71	0.71	1	0.71
	ME	2.24×10^{5}	8.09×10^{5}	7.07×10^{7}	7.07×10^{7}	1.70×10^{7}	1.92×10^{8}	1.56×10^{9}	5.77×10^{9}	1.54×10^{10}	7.78×10^{9}	-3.04×10^{10}	3.77×10^{9}
	NoEB	14.72	13.09	8.68	8.62	9.99	7.45	5.31	4.76	2.68	3.69	0.68	4.69

TABLE XII
ERROR PERFORMANCE OF APPROXIMATE SIGNED MULTIPLIERS

		This Paper				[27]			[29]				[10]
		1Step Full	1Step Trunc	2Steps Full	2Steps Trunc	(L=2)	(L=3)	(L=4)	AM2	AM3	AM4	AM5	
8×8	ER	0.30	0.96	0.84	0.99	0.72	0.87	0.99	0.99	0.94	0.94	1	0.94
	ME	42.3	2.26×10^{2}	8.74×10^{2}	1.04×10^{3}	4.87×10^{2}	1.44×10^{3}	6.88×10^{3}	1.12×10^{4}	1.22×10^{4}	1.14×10^{4}	1.17×10^{4}	5.68×10^{3}
	NoEB	9.47	7.89	5.59	5.46	6.40	4.99	3.05	1.67	2.16	2.24	2.05	3.24
12×12	ER	0.43	1	0.93	1	0.85	0.97	1	1	0.98	0.98	1	0.98
	ME	9.67×10^{2}	5.83×10^{3}	6.43×10^{4}	7.01×10^{4}	3.24×10^{4}	2.39×10^{5}	9.21×10^{5}	2.86×10^{6}	3.12×10^{6}	2.92×10^{6}	3.02×10^{6}	1.46×10^{6}
	NoEB	13.05	11.29	7.18	7.15	8.36	5.64	3.94	1.67	2.16	2.24	2.05	3.24
16×16	ER	0.61	1	0.97	1	0.92	0.97	1	1	1	1	1	1
	ME	3.57×10^{4}	1.45×10^{5}	3.77×10^{6}	3.86×10^{6}	2.10×10^{6}	7.76×10^{6}	9.01×10^{7}	7.32×10^{8}	8.00×10^{8}	7.47×10^{8}	7.75×10^{8}	3.73×10^{8}
	NoEB	16.04	14.66	9.36	9.35	10.35	8.23	5.17	1.67	2.16	2.24	2.05	3.24
20×20	ER	0.72	1	0.99	1	0.96	0.99	1	1	1	1	1	1
	ME	1.03×10^{6}	3.27×10^{6}	3.53×10^{8}	3.55×10^{8}	1.34×10^{8}	1.30×10^{9}	1.21×10^{10}	1.87×10^{11}	2.05×10^{11}	1.91×10^{11}	1.99×10^{11}	9.56×10^{10}
	NoEB	19.18	18.13	10.67	10.67	12.34	9.13	6.20	1.67	2.16	2.24	2.05	3.24
Avg.	ER	0.52	0.99	0.93	1	0.86	0.95	1	1	0.98	0.98	1	0.98
	ME	2.67×10^{5}	8.55×10^{5}	8.92×10^{7}	8.97×10^{7}	3.40×10^{7}	3.27×10^{8}	3.05×10^{9}	4.69×10^{10}	5.15×10^{10}	4.79×10^{10}	4.99×10^{10}	2.40×10^{10}
	NoEB	14.44	12.99	8.20	8.16	9.36	7.00	4.59	1.67	2.16	2.24	2.05	3.24

A. Image Filtering

We have investigated the quality-power trade-off of approximate multipliers in a Gaussian smoothing image filtering, which is a typical error resilient application [3]. The Gaussian kernel has zero mean and 1.5 standard deviation and can be obtained using the fspecial('gaussian', [3 3], 1.5) Matlab command. The convolution of the 'Lena' image with the Gaussian kernel is performed through Matlab-HDL cosimulations where the multiplications are mapped by approximate multipliers. To compare the filtered images, we used the structural similarity index SSIM. The SSIM is a well-established metric to quantify the visibility of errors (differences) between a distorted image and a reference image. The SSIM was introduced in [31] to improve on traditional methods such as peak signal-to-noise ratio (PSNR) and mean squared error (MSE).

The resulting filtered images are shown in Fig. 11. With exception of [29] AM2-AM5 and [27] L = 4, the investigated approximate multipliers perform quite well in terms of SSIM. A better outlook is offered by the quality-power trade-off shown in Fig. 12, where the y-axis reports the SSIM percentage degradation, while the x-axis reports the power saved with respect to the exact implementation. From this figure we can observe that (i) the proposed 1StepFull exhibits the lowest quality degradation (only 0.68% SSIM reduction) with 37% power saving, (ii) by slightly increasing the tolerable image quality degradation the proposed 1StepTrunc achieve the best trade-off, offering 2% of SSIM degradation as [10] and [27] L = 2, but at higher power saving (64%), (iii) if power dissipation is the priority, proposed 2StepsTrunc is the best choice, achieving 77% power saving with 8% image quality degradation.

B. LMS Filtering

Adaptive Least-Mean-Square (LMS) is the most popular adaptive filtering algorithm, with applications ranging from adaptive noise cancellation to system identification and channel equalization [32]. LMS aims to minimize the mean squared error (MSE) between the output of the adaptive filter (typically a Finite Impulse Response filter, FIR) and a desired signal. The MSE is minimized in a recursive manner, updating step-by-step the coefficients of the FIR filter. The minimization happens in an approximated way, using a stochastic gradient descent algorithm. Therefore, the LMS algorithm is characterized by an inherent grade of imprecision and constitute a fertile ground to employ approximate hardware circuits.

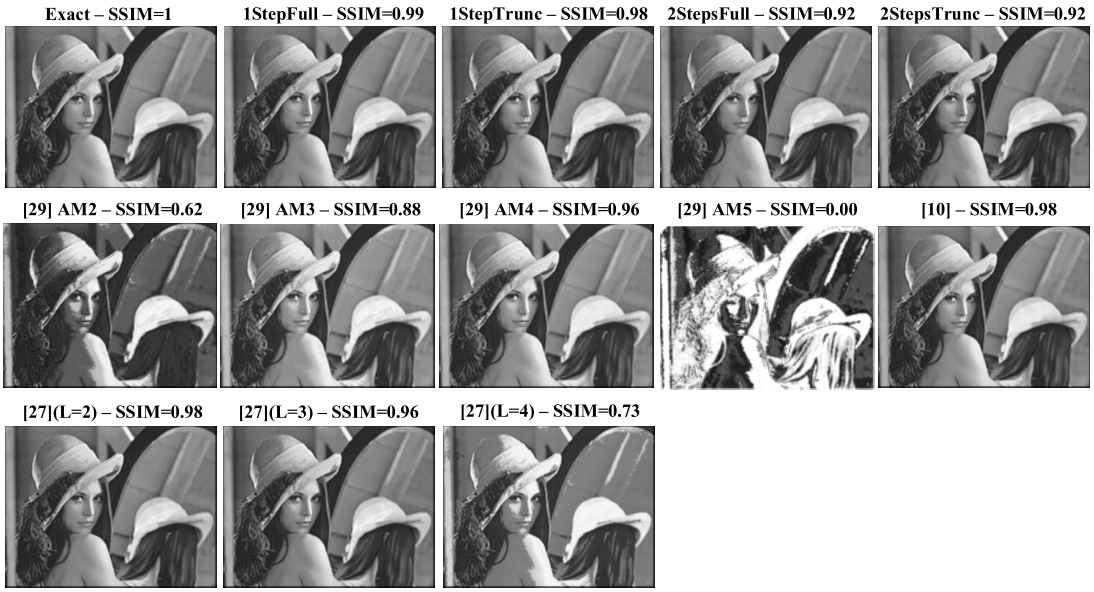

Fig. 11. Gaussian smoothed Lena images processed with exact and approximate multipliers.

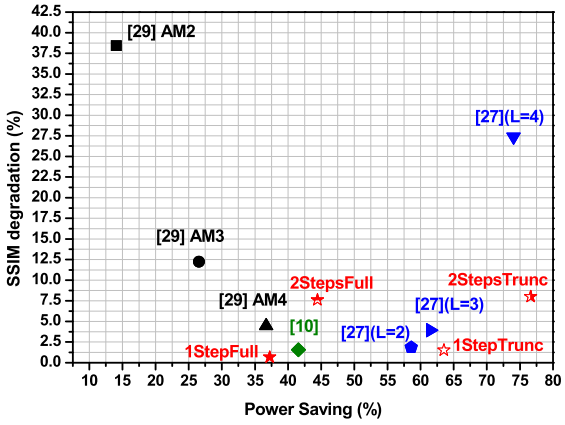

Fig. 12. Quality-power trade-off for Gaussian smoothing application (circuit [29] AM5 is out of scale).

In this paper we evaluate the convergence error degradation due to the usage of approximate multipliers in an adaptive FIR filter. We consider a system identification scenario (Fig. 13), in which the signal $d(n)$ is the output of the filter to be identified. The LMS algorithm adjusts the FIR filter coefficients $w(n)$ to mimic the response of the filter to be identified. In our test case we employed as filter to be identified an IIR Butterworth low-pass filter with bandpass gain equal to 0 dB (from zero to 0.2 π · rad/sample) and stopband attenuation of -60 dB at 0.3125π ·rad/sample. The LMS FIR filter uses 125 taps and is implemented by using 16-bit fixed point representation (only 16×16 signed multipliers are employed). The convergence of the algorithm has been evaluated through 100 simulations, each one with a duration of 5×10^4 iterations, using as input $x(n)$ (see Fig. 13) a white Gaussian signal with zero mean, 0.3 standard deviation and range in $[-1, 1)$. The adaptive filters have been synthesized by imposing a maximum clock frequency of 290 MHz. Mean error compensation has been applied at system level for all the investigated approximate circuits.

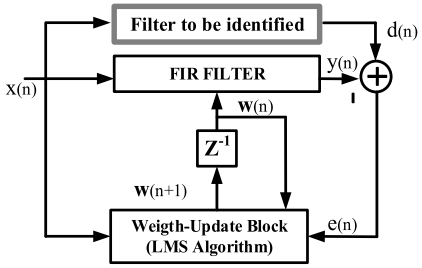

Fig. 13. LMS System identification.

Figure 14 shows the magnitude of the harmonic response of the investigated adaptive filters. The LMS filter using exact multipliers achieves a stop-band attenuation of 60.3dB, Fig. 14 (a). The adaptive filters using proposed 1StepFull and 1StepTrunc exhibit a good convergence behavior, with stop-band attenuation of 60.7dB and 61.6dB respectively (see Fig. 14 (c) and Fig. 14 (d)). The LMS filter using [27] L = 2 multiplier shows poor convergence properties, with a stop-band attenuation as low as of 35.8dB; this implementation suffers also from a large ripple in the bandpass (Fig. 14 (b)). The LMS filters using others approximate multipliers fail to converge.

Fig. 15 shows the error-power trade-off of the investigated circuits. Here, y-axis reports, as a figure of merit related to the precision, the stop-band attenuation. The x-axis reports the power saving obtained by using approximate multipliers (as opposed as exact multipliers).

The adaptive filters using proposed 1StepFull and 1StepTrunc, in addition to the excellent convergence property, allow to achieve 4% and 26% system-level power saving, respectively (please note that these values differ from data reported in Table X, since they take into account not only the power dissipated by the multipliers but also the contribution of pipeline registers and additional logic; moreover, the synthesis timing constraint used for LMS filter synthesis is different from the one assumed in Table X, being related to the critical path of the adaptive filter).

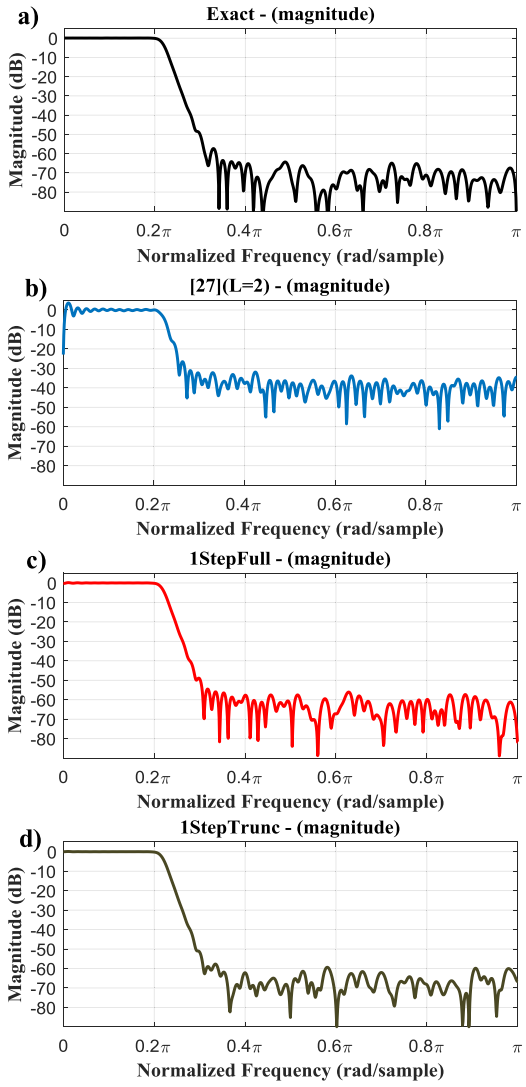

Fig. 14. Harmonic response (magnitude) of LMS adaptive filters based on exact and approximate multipliers (only results for filters that do not fail to converge are reported).

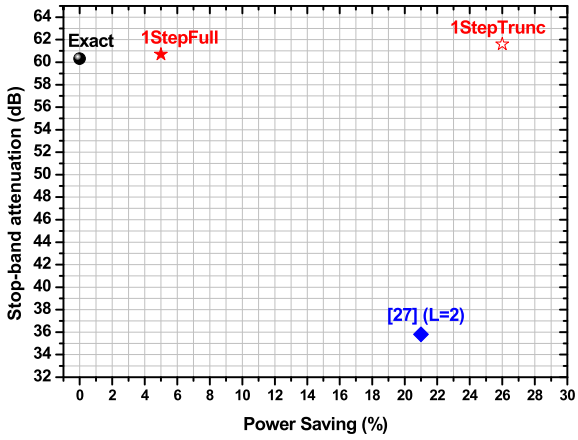

Fig. 15. Error-power trade-off for the investigated approximate adaptive LMS filters.

VII. Conclusion

The paper presents novel approximate compressors, that overcome previously proposed circuits in terms of error performance and circuit complexity. The approximate compressors are used to build approximate multipliers, using an algorithm that allocates the approximate compressors with the aim of optimize electrical performance while providing small error. For each operand size, we developed four approximate multiplier versions, with different precision vs electrical performance trade-off.

The circuits developed in this paper and previously proposed approximated multipliers, have been synthesized by using a 40nm CMOS technology. Syntheses show that our circuits provide very good error-electrical performance trade-off, compared with previously proposed approximate multipliers. We have also investigated the use of approximate multipliers in two applications: image filtering and LMS system identification. In both cases, proposed approximate multipliers show remarkable good results, compared with the state of the art.

The proposed approach could, in principle, be applied also to Booth-encoded multipliers. In this case, however, each partial product is obtained by a Booth selector, rather than a simple AND gate. Thus, for each partial product, the probability of being high is different compared to a non-Booth multiplier. Additional investigation is required to design ad-hoc approximate compressors optimized for Booth- multipliers.

As final remark, the paper shows that the tolerable approximation level strongly depends on the application, motivating the need of energy-efficient design-time (or run-time) quality-configurable systems.

References

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Proc. 18th IEEE Eur. Test Symp.*, May 2013, pp. 1–6.

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

[3] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proc. 50th ACM/EDAC/IEEE Des. Automat. Conf. (DAC)*, Austin, TX, USA, May/Jun. 2013, pp. 1–9.

[4] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. New York, NY, USA: Oxford Univ. Press, 1999.

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

[6] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, and J. Han, "A comparative evaluation of approximate multipliers," in *Proc. IEEE/ACM Int. Symp. NANOARCH*, Jul. 2016, pp. 191–196.

[7] 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.

[8] 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.

[9] S. Balamurugan and P. S. Mallick, "Error compensation techniques for fixed-width array multiplier design—A technical survey," *J. Circuits, Syst. Comput.*, vol. 26, no. 3, p. 1730003, Mar. 2017.

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

[11] C.-H. Lin and I.-C. Lin, "High accuracy approximate multiplier with error correction," in *Proc. IEEE 31st ICCD*, Oct. 2013, pp. 33–38.

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

[13] L. Dadda, "Some schemes for parallel multipliers," *Alta Freq.*, vol. 34, pp. 349–356, Mar. 1965.

[14] 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.
[15] P. F. Stelling, C. U. Martel, V. G. Oklobdzija, and R. Ravi, "Optimal circuits for parallel multipliers," *IEEE Trans. Comput.*, vol. 47, no. 3, pp. 273–285, Mar. 1998.
[16] J. Um and T. Kim, "An optimal allocation of carry-save-adders in arithmetic circuits," *IEEE Trans. Comput.*, vol. 50, no. 3, pp. 215–233, Mar. 2001.
[17] 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 Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 10, pp. 1985–1997, Oct. 2004.
[18] S. Veeramachaneni, K. M. Krishna, L. Avinash, S. R. Puppala, and M. B. Srinivas, "Novel architectures for high-speed and low-power 3-2, 4-2 and 5-2 compressors," in *Proc. 20th Int. Conf. VLSI Des. (VLSID)*, 2007, pp. 324–329.
[19] D. Baran, M. Aktan, and V. G. Oklobdzija, "Energy efficient implementation of parallel CMOS multipliers with improved compressors," in *Proc. Low-Power Electron. Des. Conf. (ISLPED)*, 2010, pp. 147–152.
[20] A. K. Verma and P. Ienne, "Automatic synthesis of compressor trees: Reevaluating large counters," in *Proc. Des., Automat. Test Eur. Conf. Exhib.*, 2007, pp. 443–448.
[21] D. Kelly, B. Phillips, and S. Al-Sarawi, "Approximate signed binary integer multipliers for arithmetic data value speculation," in *Proc. Conf. Des. Archit. Signal Image Process. (DASIP)*, Sophia Antipolis, France, Oct. 2009.
[22] 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.
[23] 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.
[24] 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.
[25] 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.
[26] R. Marimuthu, Y. E. Rezinold, and P. Mallick, "Design and analysis of multiplier using approximate 15-4 compressor," *IEEE Access*, vol. 5, pp. 1027–1036, 2017.
[27] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, and A. Yakovlev, "Energy-efficient approximate multiplier design using bit significance-driven logic compression," in *Proc. Des., Automat. Test Eur. Conf. Exhib. (DATE)*, 2017, pp. 7–12.
[28] D. Esposito, A. G. M. Strollo, and M. Alioto, "Low-power approximate MAC unit," in *Proc. IEEE PRIME*, Giardini Naxos, Italy, Jun. 2017, pp. 81–84.
[29] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, and J. Henkel, "Architectural-space exploration of approximate multipliers," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, Austin, TX, USA, Nov. 2016, pp. 1–8.
[30] J. Riordan, *An Introduction to Combinatorial Analysis*. Hoboken, NJ, USA: Wiley, 1958.
[31] 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. 600–612, Apr. 2004.
[32] B. Farhang-Boroujeny, *Adaptive Filters: Theory and Applications*. Hoboken, NJ, USA: Wiley, 2013.

Darjn Esposito (M'17) was born in Italy in 1989. He received the M.S. degree (Hons.) in electronic engineering and the Ph.D. degree in information technology and electrical engineering from the University of Napoli Federico II, Italy, in 2013 and 2017, respectively. In 2016, he was a Visiting Scholar with the National University of Singapore. He is currently a Post-Doctoral Researcher with the University of Napoli Federico II. His research interests include design of digital VLSI circuits, with emphasis on low-power approximate circuits.

Antonio Giuseppe Maria Strollo (M'05–SM'06) received the Laurea 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 was the Head of the Department of Electronic and Telecommunication Engineering from 2005 to 2008. He has published over 120 papers on international journals and conferences. His current research interests are design and analysis of VLSI circuits. From 2009 to 2012, he served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I; currently he is an Associate Editor of *Integration* (VLSI Journal).

Ettore Napoli was born in Italy in 1971. He received the Ph.D. degree in electronic engineering in 1999, the degree (Hons.) in electronic engineering in 1995, and the degree (Hons.) in physics 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 Federico II since 2005.

He has authored or co-authored over 100 papers published in international journals and conferences. His scientific interests include modeling and design of power semiconductor devices and VLSI circuit design.

Davide De Caro (M'05–SM'09) 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 involved in the area of digital integrated VLSI circuit design for the last 14 years. Since 2003, he has been a Researcher with the Department of Electrical Engineering and Information Technology. He has authored over 50 technical papers in international journals and refereed international conferences.

Nicola Petra (M'05) received the Laurea degree and the Ph.D. degree from the University of Napoli Federico II in 2002 and 2007, respectively. He is currently a Researcher with the Department of Electronics and Telecommunications Engineering, University of Napoli Federico II. He has authored or co-authored over 30 papers on scientific journals and international conferences. His research interests include design of digital VLSI circuits for telecommunications and high-performance arithmetic circuits.