Truncated Binary Multipliers With Variable Correction and Minimum Mean Square Error

Nicola Petra, *Member, IEEE*, Davide De Caro, *Senior Member, IEEE*, Valeria Garofalo, Ettore Napoli, and Antonio G. M. Strollo, *Senior Member, IEEE*

***Abstract*—Truncated multipliers compute the n most-significant bits of the $n \times n$ bits product. This paper focuses on variable-correction truncated multipliers, where some partial-products are discarded, to reduce complexity, and a suitable compensation function is added to partly compensate the introduced error. The optimal compensation function, that minimizes the mean square error, is obtained in this paper in closed-form for the first time. A sub optimal compensation function, best suited for hardware implementation, is introduced. Efficient multipliers implementation based on sub-optimal function is discussed.**

Proposed truncated multipliers are extensively compared with previously proposed circuits. Experimental results, for a 0.18 μm technology, are also presented.

***Index Terms*—Multiplication, truncated multipliers, digital arithmetic, error compensation, error analysis, least mean square method.**

I. Introduction

MULTIPLICATION is the main operation in many signal processing algorithms (filtering, convolution, fast Fourier transform (FFT), DCT etc.). As a consequence efficient parallel multipliers are desirable both in general purpose DSP processors and in application specific architectures for digital signal processing [1]–[4].

A full-width digital $n \times n$ multiplier computes the 2 n output as a weighted sum of partial-products [1], [5]. Many applications require an n bits output. A typical example is a digital, computationally intensive, ASIC in which the bit-width at the output of the arithmetic blocks is chosen on the basis of system-related accuracy issues. In these applications, 2 n bits of precision at the multiplier output are very often more than required. Having a multiplier with an n bits output can also be useful in digital signal processors where the inputs and the outputs of the datapath can be stored in registers with the same bit-width.

A truncated multiplier is an $n \times n$ multiplier that computes, with a certain approximation, the n most-significant bits of the product. The most obvious way to design a truncated multiplier uses a full-width multiplier, whose output is rounded to n bits. As an example, Fig. 1 shows the partial-products matrix of a 8×8 unsigned multiplier. In this figure the inputs x and y are assumed to be two fractional numbers ($x = \sum_{i=1}^{n} x_i 2^{-i}$ and $y = \sum_{i=1}^{n} y_i 2^{-i}$). The reader should note that, even if the example of Fig. 1 shows an unsigned multiplier, the following discussion holds equally well for signed multipliers [5].

Manuscript received July 03, 2008; revised April 07, 2009, July 24, 2009; accepted September 10, 2009. First published December 18, 2009; current version published June 09, 2010. This paper was recommended by S. Cotofana.

The authors are with the Department of Biomedical Electronic and Telecommunication Engineering, University of Napoli Federico II, 80125 Naples, Italy (e-mail: dadecaro@unina.it).

Digital Object Identifier 10.1109/TCSI.2009.2033536

As shown in Fig. 1, the partial-products are the result of AND operations between the bits of the two operands x and y. The partial-products can be divided into two subsets. The least significant part (LSP) includes the n less significant columns of the partial-products array, while the most significant part (MSP) includes the remaining columns.

The full-width multiplier output, P_{exact} is given by

$$P_{\text{exact}} = S_{\text{MSP}} + S_{\text{LSP}}. \tag{1}$$

where S_{MSP} and S_{LSP} represent the weighted sum of the elements of MSP and LSP, respectively.

The output of the full-rounded multiplier is:

$$P_{\text{full\_round}} = \text{trunc}_n(S_{\text{MSP}} + S_{\text{LSP}} + \text{LSB}/2) \tag{2}$$

where trunc_j indicates the truncation to j bits of the result, and LSB/2 is the rounding constant. LSB is the weight of the least significant bit of the truncated multiplier output p (see Fig. 1).

While giving the smallest possible error (bounded by LSB/2), a full-rounded multiplier requires the highest amount of hardware, since all the partial-products are computed and summed.

Since in a truncated multiplier the n less significant bits of the full-width product are discarded (see $\text{p}_9 \ldots \text{p}_{16}$ in Fig. 1), one is tempted to remove some of the partial-products in the LSP, to trade-off accuracy with hardware cost.

Several techniques that follow this idea have been proposed in the literature [6]–[22]. These techniques introduce suitable compensation circuits that partly compensates for the dropped terms, thereby reducing the approximation error.

In this paper we focus on truncated multipliers with so-called variable-correction approach that show best performances among published architectures [10]–[22].

As shown in Fig. 1, let us indicate as LSPmajor the h most significant columns of LSP (h is a design parameter, that can range from $h = 0$ to $h = n$), while the remaining $n_{\text{eq}} = n - h$ columns of LSP are named LSPminor. Moreover, let us name Input Correction (IC) the leftmost column of LSPminor (highlighted in gray in Fig. 1).

Fig. 2 shows an example of variable-correction truncated multiplier. The partial-products in LSPminor are dropped, and this is compensated with the introduction of a suitable function of the IC partial-products ($f(\text{IC})$). It is worth noticing that computing $f(\text{IC})$ is generally simpler than computing the sum of the elements of the LSPminor, since $f(\text{IC})$ depends on a subset of the LSPminor terms. Nevertheless, for the same reason, $f(\text{IC})$ cannot be exactly equal to the sum of the LSPminor terms. The main trade-off effort for fixed-width multiplier design is finding a compensation function that is efficiently implemented in hardware, and provides, at the same

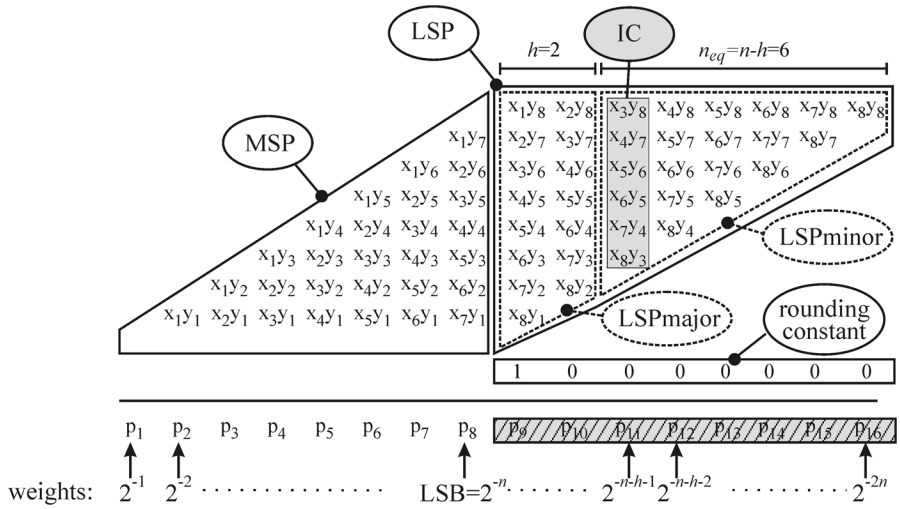

Fig. 1. Full Rounded truncated unsigned multiplier with $n = 8$.

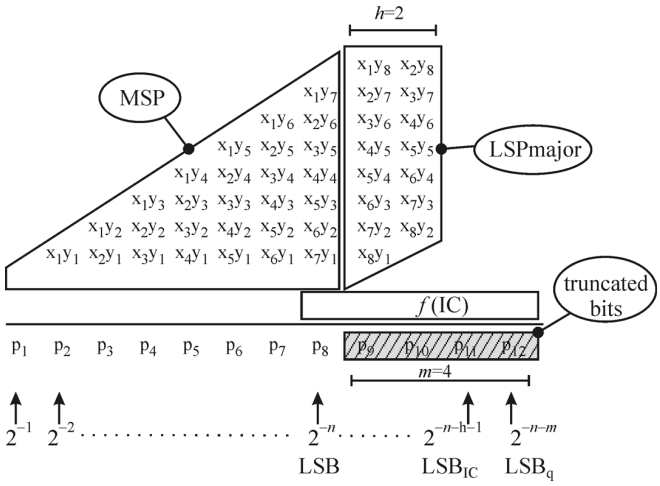

Fig. 2. Partial products matrix for the multiplier of Fig. 1, truncated with $h = 2$ and using a variable-correction scheme to compute the result.

time, a correction close to the sum of LSPminor partial-products. As will be shown in the Literature survey of Section II, the techniques proposed to date show substantial limits. In particular, they are not derived from an analytical theory but rather heuristically or with the help of exhaustive searches.

In this paper, the optimal compensation function, that minimizes the mean square error of the truncated multiplier, $f_{\mathrm{opt}}(\mathrm{IC})$, is analytically calculated. It is shown that $f_{\mathrm{opt}}(\mathrm{IC})$ is a quadratic form of the elements in IC. We also describe the derivation of a sub-optimal linear compensation function that is characterized, at the same time, by reduced approximation error and easy hardware implementation. The characteristics of the resulting truncated multipliers are extensively compared with previously proposed circuits. Many truncated multipliers, implementing eight different architectures with wordlength from 8 to 16 bits have been synthesized. Area, power and accuracy of the multipliers are investigated and compared. Experimental performances on a 0.18-μm test chip are also presented. A preliminary version of this work was earlier presented in [21].

The paper is organized as follows. The state of the art is discussed in Section II, while the error sources in the truncated multiplier scheme of Fig. 2 are analyzed in Section III. The derivation of the optimal compensation function is reported in Section IV. The linear compensation function is introduced in Section V, while Section VI analyzes the problem of coefficient quantization. After extending the treatment to signed multipliers in Section VII, the hardware implementation of truncated multipliers with linear compensation function is presented in Section VIII. Experimental results obtained from the test chip are reported in Section IX.

II. Previous Art

A first general overview of truncated multipliers is given in the work of Lim [6]. Lim identifies two main approaches for compensating the error introduced by dropping the LSPminor. The simplest approach uses a constant to estimate the weighted sum of the LSPminor elements (S_{LSPminor}). The second method, called conditional-correction in [6], uses the sum of the elements of the IC to estimate S_{LSPminor}. As explained in [6], this second approach results in a lower error since the LSPminor elements are correlated to the IC elements. The actual implementation of the conditional-correction method is not described in [6].

The constant correction method introduced by Lim is refined by Shulte and Swartzlander in [7], [8] where the rounding error is also taken into account in the computation of the constant. The multiplier output in these constant-correction methods can be written as

$$P_{\mathrm{Constant\_Correction}}l = \mathrm{trunc}_n(S_{\mathrm{MSP}} + S_{\mathrm{LSPmajor}} + \mathrm{constant}) \tag{3}$$

where S_{LSPmajor} is the weighted sum of the elements of the LSPmajor.

Kidambi *et al.* in [9] propose another constant correction method in which the LSP is eliminated altogether. This approach reduces up to 50% the area of the full-width multiplier, but introduces a rather large error, which rapidly increases with n, resulting impractical in most applications.

The accuracy of truncated multiplier is significantly improved in the variable-correction truncated multipliers [10]–[22], where a general function of the IC elements $f(\mathrm{IC})$ is used to estimate S_{LSPminor}. These techniques can be seen as a generalization of the conditional-correction method proposed by Lim [6]. The

Fig. 2 shows an example of the partial-products matrix for the multiplier of Fig. 1, truncated with $h = 2$. As shown in this example, the partial-products in LSPminor are dropped. The terms in the IC are employed to estimate the weighted sum of the elements of the LSPminor, and the multiplier output is computed as

$$P_t = \mathrm{trunc}_n[S_{\mathrm{MSP}} + S_{\mathrm{LSPmajor}} + f(\mathrm{IC})] \tag{4}$$

where $f(\mathrm{IC})$ is a suitable compensation function.

For a given h value, the hardware complexity and approximation error introduced by the scheme in Fig. 2 depends on the choice of the compensation function $f(\mathrm{IC})$.

In [10]–[12], Swartzlander *et al.* propose to add the partial-products of the IC in the rightmost column of the LSPmajor.

In [13] Stine and Duverne propose a so-called hybrid correction method, where only a subset of the IC elements are summed in the rightmost column of the LSPmajor (like [10]–[12]) and a constant is also added (like [6]–[9]), to compensate for the portion of IC that is not used.

In [14] Park *et al.* improve the approach of [10]–[12] by summing in the rightmost column of the LSPmajor all the IC terms and also another correcting bit.

In [15] a correction function for truncated multipliers with $h = 0$ is proposed by Jou *et al.*, by manipulating the partial-products in IC with a circuit composed by AND–OR gates, to obtain $f(\mathrm{IC})$. The technique in [15] is affected by a significant mean and mean square error. In addition, the circuit that calculates $f(\mathrm{IC})$ is characterized by a slow ripple architecture.

Curticapean *et al.*, [16], use a modified version of the correction circuit of [15], suitable for unsigned multipliers. The technique of [16] provides good error performances but the compensation function is still based on a slow ripple architecture. Furthermore no explanation is given to justify the improved error performances.

In [17] Van *et al.* propose a fixed-width signed multiplier architecture in which the correction function proposed in [15] is generalized by considering either the IC terms or their complements. The optimal pattern of inversions applied to the partial-products in IC is obtained through exhaustive simulation. The technique is extended to the case $h > 0$ in [18].

An attempt to analytically explain the results obtained in [18] is presented by Liao *et al.*. in [19]. The circuits proposed in [19] are however too complex to be implemented in a practical multiplier and do not improve the performances of [18].

The truncated multiplier proposed by Strollo *et al.* in [20] uses an error-compensation function that can be optimized in order to minimize either the maximum absolute error or the mean-square error. The error compensation function is obtained heuristically, observing that $f(\mathrm{IC})$ can be approximated as a weighted sum of IC terms.

The technique proposed in [20] is slightly improved by Kuang and Wang in [22], where the authors propose a modification that reduces hardware complexity with respect to [20]. The truncated multipliers proposed in [22] reduce the mean error while increasing the mean square error with respect to [20], or vice versa.

The approach proposed by Michard *et al.* in [23] is a noteworthy exception with respect to the scheme of Fig. 2. The prediction-selection method of [23] is based on a logical computation of the values to be added to S_{MSP} and to S_{LSPmajor} (prediction), followed by a simplification process (selection). Results are presented for truncated multipliers with $n = 8, 12$, and 16. The approach of [23] is not applicable to higher n values, due to the fast growing computational cost of the prediction process.

III. Error in Truncated Multipliers

A. Definitions and Assumptions

In the following, for the sake of simplicity, we will consider unsigned multipliers. Extension to signed and signed × unsigned multipliers is discussed in Section VII. Without loss of generality, we will moreover assume that the inputs of the multiplier represent fractional values in $[0, 1)$:

$$x = \sum_{i=1}^{n} x_i 2^{-i}; \quad y = \sum_{i=1}^{n} y_i 2^{-i} \tag{5}$$

The weight of the less-significant bit of x and y is given by

$$\mathrm{LSB} = 2^{-n} \tag{6}$$

The x_i and y_i bits are assumed to be independent and uniformly distributed, with a probability 1/2 of being one.

The elements of the IC (see Fig. 1) will be indicated as follows:

$$\mathrm{IC} = \left[\gamma_1, \gamma_2, \ldots, \gamma_{n_{\mathrm{eq}}}\right] \tag{7}$$

where

$$n_{\mathrm{eq}} = n - h \tag{8}$$

$$\gamma_i = x_{h+i} y_{n+1-i}. \tag{9}$$

The weighted sum of the elements in LSPminor is given by

$$S_{\mathrm{LSPminor}} = \left(2^{-n-h-1} \sum_{i=1}^{n_{\mathrm{eq}}} \gamma_i\right) + \left(\sum_{i=h+2}^{n} \sum_{j=n+h+2-i}^{n} x_i y_j 2^{-i-j}\right) \tag{10}$$

The first term on the right-hand side of (10) gives the contribution of the elements in the IC, while the second term is the contribution of the other LSPminor elements.

We define Θ the set of all the possible values of the vector IC ($2^{n_{\mathrm{eq}}}$ elements) and $\Omega(A)$ the subset of all couples (x, y) giving $\mathrm{IC} = A$. For instance, in the multiplier of Fig. 1, the subset $\Omega([1, 1, 0, 1, 1, 1])$ is composed by 48 (x, y) values, that can be written as follows:

$$(x_{8\ldots1}; y_{8\ldots1}) = (1, 1, 1, 0, 1, 1, -, -; 1, 1, 0, 1, 1, 1, -, -)$$
$$(x_{8\ldots1}; y_{8\ldots1}) = (1, 1, 1, 1, 1, 1, -, -; 1, 1, 0, 1, 1, 1, -, -)$$
$$(x_{8\ldots1}; y_{8\ldots1}) = (1, 1, 1, 0, 1, 1, -, -; 1, 1, 1, 1, 1, 1, -, -)$$

where the symbol "–" indicates a don't care value (the variable can be either 1 or 0). In general the number $n_{xy}(A)$ of (x, y) values in a set $\Omega(A)$ is equal to: $n_{xy}(A) = 3^{nz} \cdot 2^{2h}$, where nz is the number of zeros in the IC.

We define $P(A)$ the probability of IC being equal to A. For instance, in the multiplier of Fig. 1 we have: $P([1, 1, 0, 1, 1, 1]) = 48/2^{16} = 3 \times 2^{-12}$.

Finally please note that, in order to simplify the notation, in the following we will indicate the mean conditioned to all couples $(x,y) \in \Omega(A)$ in the following way:

$$\underset{\mathrm{IC}=A}{E}\{\cdots\} \triangleq \underset{(x,y)\in\Omega(A)}{E}\{\cdots\}. \tag{11}$$

B. Error Sources

The error e_{total} introduced by the truncated multiplier of Fig. 2 is computed with respect to the output of the full-width multiplier:

$$e_{\mathrm{total}} = P_t - P_{\mathrm{exact}} = f(\mathrm{IC}) - S_{\mathrm{LSPminor}} + e_{\mathrm{trunc}} \tag{12}$$

where e_{trunc} is the effect of the truncation operation.

The last equation can be rearranged as follows:

$$e_{\mathrm{total}} = e_{\mathrm{erasing}} + e_{\mathrm{trunc}} \tag{13}$$

where

$$e_{\mathrm{erasing}} = f(\mathrm{IC}) - S_{\mathrm{LSPminor}}. \tag{14}$$

Thus, two error sources affect the truncated multiplier. The first error source, e_{erasing}, is due to the use of the compensation function $f(\mathrm{IC})$ in place of the sum of the elements in LSPminor. The second error source is due to the truncation of the result.

Our aim is to minimize the mean square error

$$\varepsilon^2_{\mathrm{total}} = E\{e^2_{\mathrm{total}}\}. \tag{15}$$

Indicating with $\sigma^2_{\mathrm{total}}$ and μ_{total} the variance and the mean value of e_{total}, we have

$$\varepsilon^2_{\mathrm{total}} = (\mu_{\mathrm{total}})^2 + \sigma^2_{\mathrm{total}} \tag{16}$$

where, from (13)

$$\mu_{\mathrm{total}} = \mu_{\mathrm{erasing}} + \mu_{\mathrm{trunc}} \tag{17}$$

$$\begin{aligned}\sigma^2_{\mathrm{total}} &= \sigma^2_{\mathrm{erasing}} + \sigma^2_{\mathrm{trunc}} + 2\mathrm{COV}(e_{\mathrm{erasing}}, e_{\mathrm{trunc}}) \\ &= \sigma^2_{\mathrm{erasing}} + \sigma^2_{\mathrm{trunc}}.\end{aligned} \tag{18}$$

The simplified expression in (18) is obtained assuming that e_{trunc} is independent from e_{erasing}. From (17)–(18) one obtains

$$\varepsilon^2_{\mathrm{total}} = (\mu_{\mathrm{erasing}} + \mu_{\mathrm{trunc}})^2 + \sigma^2_{\mathrm{erasing}} + \sigma^2_{\mathrm{trunc}}. \tag{19}$$

C. Statistical Properties of the Truncation Error (e_{trunc})

The statistical properties of the truncation error can be obtained by considering the summation matrix needed to implement the truncated multiplier. As it can be seen in the example of Fig. 2, $f(\mathrm{IC})$ is represented with an least significant bit of 2^{-n-m}. The truncation error is given by the weighted sum of the truncated bits

$$e_{\mathrm{trunc}} = -\sum_{i=n+1}^{n+m} p_i \cdot 2^{-i}. \tag{20}$$

The mean value e_{trunc} can be easily computed by assuming that each discarded bit p_i has a probability 1/2 of being one

$$\mu_{\mathrm{trunc}} = -\sum_{i=n+1}^{n+m} \frac{1}{2}2^{-i} = -\frac{\mathrm{LSB}}{2}(1-2^{-m}). \tag{21}$$

Similarly, assuming the bits p_i independent and equally likely, we compute the variance of the truncation error as

$$\sigma^2_{\mathrm{trunc}} = \sum_{i=n+1}^{n+m} \frac{1}{4}2^{-2i} = \frac{\mathrm{LSB}^2}{12}(1-2^{-2m}). \tag{22}$$

D. Statistical Properties of the Erasing Error (e_{erasing})

The mean value of the erasing error is given by

$$\mu_{\mathrm{erasing}} = E\{e_{\mathrm{erasing}}\} = \sum_{A\in\Theta}\left(\underset{\mathrm{IC}=A}{E}\{e_{\mathrm{erasing}}\}\right)P(A). \tag{23}$$

From (14) the term within the summation in (23) can be written as

$$\underset{\mathrm{IC}=A}{E}\{e_{\mathrm{erasing}}\} = f(A) - \underset{\mathrm{IC}=A}{E}\{S_{\mathrm{LSPminor}}(x,y)\}. \tag{24}$$

Let us define as $\mu_{\mathrm{LSP}}(A)$ the function whose value is equal to the mean value of S_{LSPminor} in $\Omega(A)$:

$$\mu_{\mathrm{LSP}}(A) \triangleq \underset{(x,y)\in\Omega(A)}{E}\{S_{\mathrm{LSPminor}}(x,y)\}. \tag{25}$$

By substituting (24)–(25) in (23) we have

$$\mu_{\mathrm{erasing}} = \sum_{A\in\Theta}[f(A) - \mu_{\mathrm{LSP}}(A)]\cdot P(A). \tag{26}$$

Before computing the variance of the erasing error, let us calculate first its mean square error, given by

$$\varepsilon^2_{\mathrm{erasing}} = E\{e^2_{\mathrm{erasing}}\}. \tag{27}$$

Again, we can compute $\varepsilon^2_{\mathrm{erasing}}$ by performing the averaging on $\Omega(A)$, as follows:

$$\varepsilon^2_{\mathrm{erasing}} = \sum_{A\in\Theta}\left(\underset{\mathrm{IC}=A}{E}\{e^2_{\mathrm{erasing}}\}\right)\cdot P(A). \tag{28}$$

We can write the conditional mean in (28) as

$$\underset{\mathrm{IC}=A}{E}\{e^2_{\mathrm{erasing}}\} = \underset{\mathrm{IC}=A}{\mathrm{VAR}}\{e_{\mathrm{erasing}}\} + \left(\underset{\mathrm{IC}=A}{E}\{e_{\mathrm{erasing}}\}\right)^2. \tag{29}$$

From (14), the variance in (29) is given by the variance of $\mathrm{S}_{\mathrm{LSPminor}}$, defined as

$$\sigma^2_{\mathrm{LSP}}(A) \triangleq \underset{(x,y)\in\Omega(A)}{E}\{(S_{\mathrm{LSPminor}}(x,y) - \mu_{\mathrm{LSP}}(A))^2\}. \tag{30}$$

By using (24) one has

$$\underset{\mathrm{IC}=A}{E}\{e^2_{\mathrm{erasing}}\} = \sigma^2_{\mathrm{LSP}}(A) + (f(A) - \mu_{\mathrm{LSP}}(A))^2. \tag{31}$$

By substituting (30)–(31) in (28) one finally obtains

$$\varepsilon^2_{\mathrm{erasing}} = \varepsilon^2_{\mathrm{intrinsic}} + \varepsilon^2_{\mathrm{comp}} \tag{32}$$

where

$$\varepsilon^2_{\text{intrinsic}} = \sum_{A\in\Theta} \sigma^2_{\text{LSP}}(A)\cdot P(A) \tag{33}$$

$$\varepsilon^2_{\text{comp}} = \sum_{A\in\Theta} (f(A) - \mu_{\text{LSP}}(A))^2 \cdot P(A). \tag{34}$$

The variance of the erasing error ($\sigma^2_{\text{erasing}}$) can be computed by proceeding in a way similar to the one employed to obtain (32). The final result is

$$\sigma^2_{\text{erasing}} = \varepsilon^2_{\text{intrinsic}} + \sigma^2_{\text{comp}} \tag{35}$$

where

$$\sigma^2_{\text{comp}} = \sum_{A\in\Theta} (f(A) - \mu_{\text{LSP}}(A) - \mu_{\text{erasing}})^2 \cdot P(A). \tag{36}$$

E. Optimal Compensation Function

By substituting (35) in (19) one has

$$\varepsilon^2_{\text{total}} \simeq \sigma^2_{\text{trunc}} + \varepsilon^2_{\text{intrinsic}} + (\mu_{\text{erasing}} + \mu_{\text{trunc}})^2 + \sigma^2_{\text{comp}}. \tag{37}$$

This equation shows that the mean square error of the truncated multiplier can be reduced to the sum of four positive terms.

The first term in (37), σ^2_{trunc}, is the truncation error. This error component arises from using only n output bits from the multiplier. The truncation errors is (almost) independent from the technique used to realize the truncated multiplier and is also present in the full-rounded multiplier (see Fig. 1).

The second term in (37), $\varepsilon^2_{\text{intrinsic}}$, is independent from the compensation function. This error component arises since we use only the elements in IC to estimate $\text{S}_{\text{LSPminor}}$.

The third term in (37) depends on the average error (17). This error component can be canceled by selecting a compensation function that fulfills the following condition

$$\mu_{\text{erasing}} = -\mu_{\text{trunc}}. \tag{38}$$

The last term in (37), σ^2_{comp}, also depends from the compensation function and can be minimized with a suitable selection of $f(\text{IC})$.

In conclusion we can observe that the choice of the compensation function influences only the third and the fourth terms in (37). The optimal compensation function, $f_{\text{opt}}(\text{IC})$, nullifies these two contributions. Thus, $f_{\text{opt}}(\text{IC})$ fulfills (38) and, in addition, yields $\sigma^2_{\text{comp}} = 0$. Both conditions are satisfied by the following function:

$$f_{\text{opt}}(\text{IC}) = \mu_{\text{LSP}}(\text{IC}) - \mu_{\text{trunc}}. \tag{39}$$

In fact, by substituting (39) in (26), we can easily find that (38) is verified. Moreover, with the help of (38), it can be seen that all the terms summed in (36) are zero when (39) is verified. Thus the compensation function (39) also yields $\sigma^2_{\text{comp}} = 0$. By using the optimal compensation function (39) the total mean square error is reduced to

$$\varepsilon^2_{\text{total,opt}} = \sigma^2_{\text{trunc}} + \varepsilon^2_{\text{intrinsic}}. \tag{40}$$

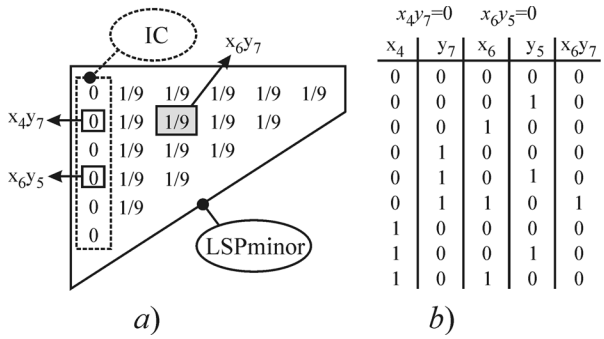

Fig. 3. Computation of μ_{LSP} for a $n = 8$ signed multiplier with $\text{IC} = 0$. (a) IC and LSPminor of the multiplier. (b) computation of the mean value of x_6y_7.

IV. Computing The Optimal Compensation Function

From (39) the computation of $f_{\text{opt}}(\text{IC})$ reduces to the calculation of $\mu_{\text{LSP}}(\text{IC})$, since μ_{trunc} is given by (21). In order to express μ_{LSP} in explicit form, let us start by computing the mean values of S_{LSPminor} when all the elements of the IC are zero. Fig. 3(a) shows what happens in this case for the 8×8 multiplier with $h = 2$ considered in Fig. 1. The mean value of each partial-product can be computed by considering all the possible values of the input bits, constrained by the condition $\text{IC} = [0,0,0,0,0]$. As an example, Fig. 3(b) shows the computation of the mean value of the partial-product x_6y_7. The value of x_6y_7 is correlated with two elements of the IC (x_4y_7 and x_6y_5) and hence with the four bits x_4, y_7, x_6, and y_5. All the possible values for these bits can be easily enumerated, taking into account the constraint that $\text{IC} = [0,0,0,0,0,0]$. The result is reported in Fig. 3(b), showing that the mean value of x_6y_7 is 1/9.

In general, it can easily be seen that the mean value of each partial-product of LSPminor is 1/9, when all the elements of the IC are zero. From Figs. 1 and 3(a) one obtains

$$\mu_{\text{LSP}}(\text{IC}=0) = \text{LSB}\, 2^{-h-1}\frac{1}{9}\sum_{i=1}^{n_{\text{eq}}-1} 2^{-i}(n_{\text{eq}} - i) \tag{41}$$

where the LSB is defined in (6). By resolving the summation in (41) we have

$$\mu_{\text{LSP}}(\text{IC}=0) = \text{LSB}\, 2^{-h-1}K \tag{42}$$

with

$$K = \frac{2}{9}\left(\frac{n_{\text{eq}}}{2} + 2^{-n_{\text{eq}}} - 1\right). \tag{43}$$

The Fig. 4(a) shows the situation when only one element of the IC is equal to 1. In this example x_6y_5 is the only nonzero element of the IC. Since $x_6y_5 = 1$ we have $x_6 = 1$ and $y_5 = 1$. This increases the mean of all the partial-products dependent on y_5 (the elements on the grey horizontal line) and on x_6 (the elements on the grey diagonal line). By enumerating the possible cases, it can easily be seen that the mean value of each grey-ed element in Fig. 4(a) is 1/3. Thus the value of μ_{LSP} can be obtained by adding to (41) a correction factor equal to

$$2^{-n-h-1}\left(1 + \frac{2}{9}\sum_{i=1}^{3} 2^{-i} + \frac{2}{9}\sum_{i=1}^{2} 2^{-i}\right).$$

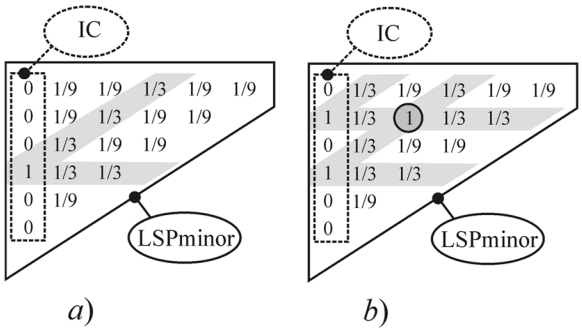

Fig. 4. Mean values of the elements of the LSP when the IC is non-zero for a 8×8 multiplier $(h = 2)$. (a) only one element of the IC is equal to 1. (b) two or more elements of the IC are equal to 1.

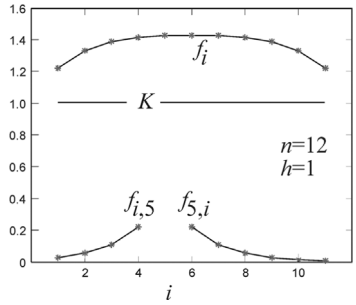

Fig. 5. Coefficients of the optimal error compensation function $f_{\rm opt}$ for $n = 12, h = 1$.

This result can be easily extended to the case in which a single element $\gamma_i (i = 1, \ldots, n_{\rm eq})$ of the IC is equal to 1. Considering all the possible cases in which only one element of the IC is equal to 1 yields

$$\mu_{\rm LSP}|_{\text{only one}\gamma_i=1} = \mu_{\rm LSP}({\rm IC} = 0) + {\rm LSB}\, 2^{-h-1} \sum_{i=1}^{n_{\rm eq}} f_i \gamma_i \tag{44}$$

where

$$f_i = \frac{13}{9} - \frac{2}{9}(2^{-i+1} + 2^{i-n_{\rm eq}}). \tag{45}$$

The above reasoning can be iterated to compute the value of $\mu_{\rm LSP}$ when two or more elements of the IC are equal to 1. In Fig. 4(b), two elements x_4y_7 and x_6y_5 of the IC are equal to 1. In this case the value of $\mu_{\rm LSP}$ differs from (44) due to the mean value of the partial-product x_6y_7 [circled in Fig. 4(b)]. In fact, since $x_4 = y_7 = x_6 = y_5 = 1$, the mean value of x_6y_7 is 1, whereas in (44) it is computed as $((1/9) + (2/9) + (2/9))$. Hence the mean value of x_6y_7 should be increased by a factor $(4/9)$ with respect to (44). In the general case in which a couple of partial-products γ_i, γ_j are equal to 1, the weight of the partial-product in the LSPminor with probability equal to 1 is $2^{-{\rm abs}(j-i)}$. Therefore the contribution to be added to (45) is$(4/9)2^{-{\rm abs}(j-i)}$. Thus, considering the contribution of all couples of elements of the IC, $\mu_{\rm LSP}$ can be computed as

$$\mu_{\rm LSP}({\rm IC}) = {\rm LSB}\, 2^{-h-1} \left[K + \sum_{i=1}^{n_{\rm eq}} f_i \gamma_i + \sum_{i=1}^{n_{\rm eq}} \sum_{j=i+1}^{n_{\rm eq}} f_{i,j} \gamma_i \gamma_j \right] \tag{46}$$

where K and f_i are defined in (43), (45) and

$$f_{i,j} = \frac{4}{9} 2^{-{\rm abs}(j-i)}. \tag{47}$$

A. Discussion

Since the elements γ_i of the IC are binary values, we have $\gamma_i = \gamma_i^2$. As a consequence, the optimal compensation function from (39), (46) can be seen as a quadratic form in the variables γ_i, augmented with a constant.

The nonlinearity of the optimal compensation function can be foreseen from Fig. 4. In fact, when a single term of the IC is high, then there is one row and one diagonal of LSPminor elements having a probability 1/3 of being high. When two terms of the IC are high, however, we have not only two rows and two diagonals with a probability 1/3 but also a single element with probability 1 in LSPminor. Thus, we have a non-linear behavior that is captured by the quadratic terms of the compensation function. Moreover, the closer the two high terms of the IC, the larger the weight of the element with probability 1 in LSPminor. Therefore, the coefficients of quadratic terms depends exponentially from $j - i$, as reported in (47).

The value of (46) decreases exponentially with h, as expected, since the larger h, the lower the weight of the terms in LSPminor. Fig. 5 shows the behavior of coefficients K, f_i and f_{ij} in (46) for a truncated multiplier with $n = 12$ and $h = 1$. The linear terms f_i exhibit a maximum for $i = (1/2)(1 + n_{\rm eq})$ and the maximum f_i value is$f_{i,\max} = (13/9) - (4/9)2^{(1/2)(1-n_{\rm eq})}$. The value of $f_{i,\max}$ tends to 13/9 as $n_{\rm eq}$ increases. As shown in Fig. 5, the maximum in the distribution of f_i is not very pronounced. The minimum of the f_i is reached for both $i = 1$ and $i = n_{\rm eq}$, is given by$f_{i,\min} = (11/9) - (4/9)2^{-n_{\rm eq}}$ and tends to 11/9 as $n_{\rm eq}$ increases.

The quadratic terms f_{ij} decrease exponentially with ${\rm abs}(j - i)$. The maximum of the quadratic coefficients is attained for ${\rm abs}(j - i) = 1$ and is 2/9. Thus, the quadratic coefficients are significantly smaller than the linear terms.

The constant term K increases almost linearly with $n_{\rm eq}$ (see (43)). In practice, K is in the same order of magnitude as f_i (in the example of Fig. 5 K is actually smaller than $f_{i,\min}$). Therefore, the optimal compensation function (46) is far from being constant. This explains the rather large mean square error of the constant-correction techniques [6]–[9].

V. Linear Compensation Function

The optimal compensation function (39), (46) can hardly be implemented in hardware, requiring the sum of a large number of terms, in the order of $n_{\rm eq}^2$. As observed before, the quadratic coefficients f_{ij} are significantly smaller than the linear terms. This suggests that a good first order approximation of the best possible error compensation function can be obtained as a simple linear combination of the elements of the IC

$$f_{\rm lin}({\rm IC}) = {\rm LSB}\, 2^{-h-1} \left[K_l + \sum_{i=1}^{n_{\rm eq}} l_i \gamma_i \right] - \mu_{\rm trunc}. \tag{48}$$

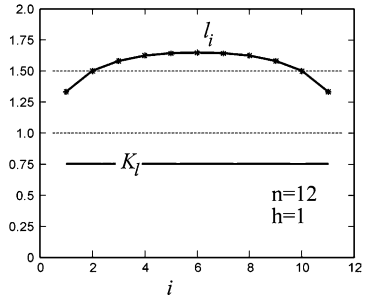

Fig. 6. Coefficients of linear compensation function $f_{\rm lin}$ for $n = 12, h = 1$.

From the discussion in Section III, the optimal values of the coefficients K_l and l_i in (48) should be obtained by zeroing and by minimizing $\sigma^2_{\rm comp}$. From (36) one has

$$\sigma^2_{\rm comp} = \sum_{A\in\Theta} (f_{\rm lin}(A) - \mu_{\rm LSP}(A) - \mu_{\rm erasing})^2 P(A) \quad (49)$$

Note that $\sigma^2_{\rm comp}$ does not depend on the constant K_l, since K_l appears both in $f_{\rm lin}(A)$ and $\mu_{\rm erasing}$ (see (26)). Therefore the optimal l_i coefficients are obtained by solving the following system:

$$\frac{\partial \sigma^2_{\rm comp}}{\partial l_r} = 0 \quad r = 1, \ldots, n_{\rm eq} \quad (50)$$

This is a linear system of $n_{\rm eq}$ equations in the $n_{\rm eq}$ unknowns l_i. The solution of this equation system is obtained in the Appendix as

$$l_i = \frac{5}{3} - \frac{1}{3}(2^{-i+1} + 2^{i-n_{\rm eq}}). \quad (51)$$

In order to obtain K_l, we impose the condition (38). This yields

$$\sum_{A\in\Theta} (f_{\rm lin}(A) - \mu_{\rm LSP}(A)) \cdot P(A) = -\mu_{\rm trunc}. \quad (52)$$

The solution of (52) is also reported in the Appendix. The result is

$$K_l = \frac{1}{6}\left(\frac{n_{\rm eq}}{2} + 2^{-n_{\rm eq}} - 1\right). \quad (53)$$

Fig. 6 shows the behavior of coefficients K_l and l_i for a truncated multiplier with $n = 12$ and $h = 1$.

Also in this case, as for Fig. 5, the distribution of linear terms l_i is symmetric and exhibits a maximum for $i = (1 + n_{\rm eq})/2$. The maximum l_i value is $l_{i,\max} = (5/3) - (1/3)2^{1/2(1-n_{eq})}$ and tends to 5/3 as $n_{\rm eq}$ increases. The minimum of the l_i is reached for both $i = 1$ and $i = n_{\rm eq}$, is given by $l_{i,\min} = (4/3) - (2/3)2^{-n_{\rm eq}}$ and tends to 4/3 as $n_{\rm eq}$ increases.

VI. Linear Coefficients Quantization

In order to implement the linear compensation function in hardware, the coefficients given by (51), (53) should be quantized on a reduced number of bits. As shown in Fig. 2, $f(\rm IC)$ is represented with an least significant bit of 2^{-n-m}. Clearly, the larger m the lower will be the error due to coefficient quantization. On the other hand, the larger m the higher the hardware complexity.

In the following we will assume $m = h + 1$. In this way the LSB of $f(\rm IC)$ is equal to the weight of the IC partial-products:

$$\mathrm{LSB}_{\rm IC} = 2^{-n-h-1} = \mathrm{LSB}\ 2^{-h-1}. \quad (54)$$

The linear compensation function with quantized coefficients is written as

$$f^*_{\rm lin}(\rm IC) = \mathrm{LSB}_{\rm IC}\left[K^* + \sum_{i=1}^{n_{\rm eq}} l^*_i \gamma_i\right] \quad (55)$$

where K^* and l^*_i are integer coefficients.

The values l^*_i can be obtained by rounding the coefficients l_i in (51) to the nearest integer. From (51) one easily obtains[1]

$$\begin{aligned} l^*_1 = l^*_2 = l^*_{n_{\rm eq}-1} = l^*_{n_{\rm eq}} = 1 \\ l^*_3 = l^*_4 = \cdots = l^*_{n_{\rm eq}-2} = 2. \end{aligned} \quad (56)$$

The value of is K^* obtained by minimizing the average error $\mu_{\rm total}$, imposing (38). After simple algebra, reported in Appendix, one obtains

$$K^* = \mathrm{round}\left(\frac{1}{2}2^{-n_{\rm eq}} + 2^h\right) = 2^h. \quad (57)$$

Therefore, the linear compensation function with quantized coefficients is given by

$$f^*_{\rm lin}(\rm IC) = \frac{1}{2}\mathrm{LSB} + \mathrm{LSB}_{\rm IC}\sum_{i=1}^{n_{\rm eq}} l^*_i \gamma_i. \quad (58)$$

VII. Signed and Signed×Unsigned Multipliers

Fig. 7 shows the partial-products matrix of a signed multiplier [1]. As it can be seen, when $h > 0$ the LSPminor is equal to the LSPminor part of an unsigned multipliers. As a consequence, the above presented expressions for the optimal compensation function are still valid.

Let us consider the case $h = 0$. It can easily be shown that the results obtained for unsigned multipliers can be also used for signed multipliers, if, instead of (9), the following definition for the terms of the IC is used

$$\gamma_i = \begin{cases} \overline{x_1 y_n}, & \text{for } i = 1 \\ x_i y_{n+1-i}, & \text{for } i = 2, \ldots, n-1 \\ \overline{x_n y_1}, & \text{for } i = n. \end{cases} \quad (59)$$

The above equation assumes that the sign-extension prevention constant, shown in Fig. 7, is added to the partial-products matrix.

The same reasoning can be applied to signed×unsigned multipliers. Also in this case, for $h > 0$ no modifications are needed with respect the unsigned case. For $h = 0$ the results of the previous sections are still valid if the following definition for IC terms is to used

$$\gamma_i = \begin{cases} x_i y_{n+1-i}, & \text{for } i = 1, \ldots, n-1 \\ \overline{x_1 y_n}, & \text{for } i = n. \end{cases} \quad (60)$$

[1]Please note that, from (51), $l_2 = q_{n_{\rm eq}-1} = (3/2) - (4/3)2^{-n_{\rm eq}} < 1.5$

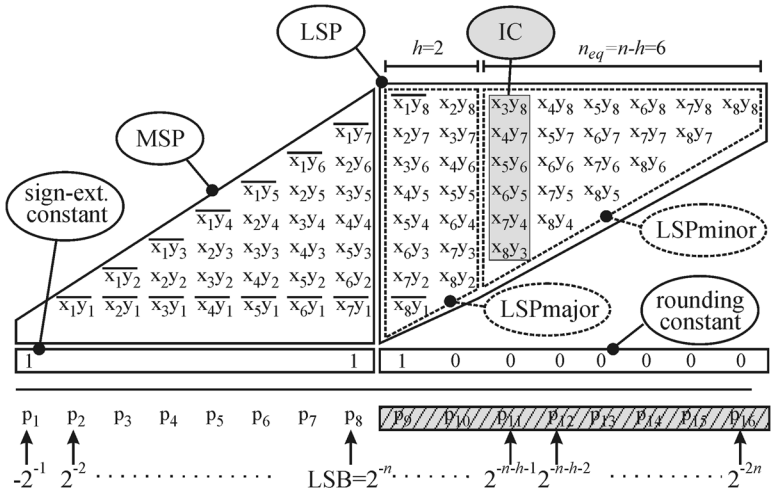

Fig. 7. Partial products matrix for a signed multiplier with $n = 8, h = 2$.

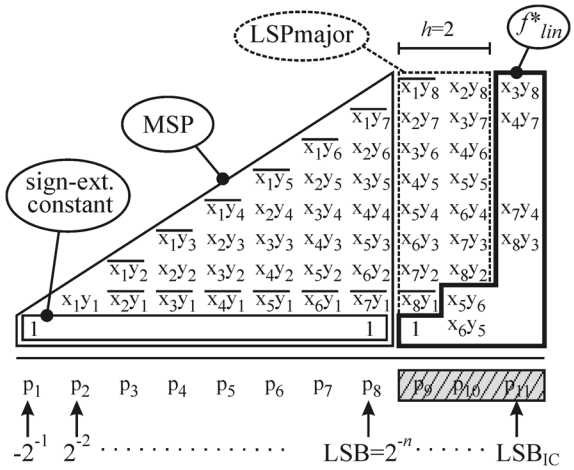

Fig. 8. Implementation of the proposed signed truncated multiplier, for: $n = 8, h = 2$.

VIII. Truncated Multipliers Implementations

The truncated multipliers based on the proposed linear quantized compensation function are efficiently implemented by summing a modified partial-products matrix (PPM). As an example, let us consider a signed multiplier with $n = 8$ and $h = 2$, (as in Fig. 1). From (56), (58) the compensation function in this case can be written as

$$f^*_{\text{lin}}(\text{IC}) = \frac{1}{2}\text{LSB} + \text{LSB}_{\text{IC}}[\gamma_1 + \gamma_2 + 2\gamma_3 + 2\gamma_4 + \gamma_5 + \gamma_6] \tag{61}$$

Thus, we simply put the terms $\gamma_1, \gamma_2, \gamma_5$ and γ_6 on the matrix column with weight LSB_{IC}, while the terms γ_3 and γ_4 are aligned on the matrix column immediately on the left, having a weight 2LSB_{IC}. The corresponding PPM is shown in Fig. 8.

In general, the PPM of the proposed truncated multiplier leaves the two extreme couples of IC partial-products (γ_1, γ_2, and $\gamma_{n_{\text{eq}}-1}, \gamma_{n_{\text{eq}}}$) on the IC column, while the remaining partial-products of the IC, having $l^*_i = 2$, are placed in the column at the left of the IC.

The modified PPM is efficiently summed by using a carry-save reduction tree, followed by a fast carry-propagate adder (CPA) [1]. The Three-Dimensional Minimization (TDM) method [24], [25], is exploited for the carry-save reduction tree. TDM minimizes the overall delay by compensating the delay asymmetries of full and half adders. In the following we will employ the approach of [24], that gives performances very close to the optimal TDM of [25] while requiring a much simpler implementation. For the CPA implementation, we will consider the Kogge-Stone parallel prefix topology [1].

TABLE I
Characteristics of Proposed Truncated Multipliers

n, h	Multiplier	Reduction Tree		Performances		
		#FA	#HA	Area ($10^3 \cdot \mu m^2$)	Power (μW/MHz)	Delay (ns)
8, -	full rounded	37	5	4.51	8.24	2.10
8, 2	proposed	31	1	3.38	6.67	2.10
16, -	full rounded	197	13	19.6	37.9	2.75
16, 1	proposed	119	1	10.7	22.2	2.75
24, -	full rounded	485	21	45.2	87.1	3.13
24, 1	proposed	275	1	23.8	49.1	3.09

A. Multiplier Performance

The Table I presents the implementation results of the truncated multiplier, for a 0.18 μm technology. The circuits have been synthesized at the gate level and simulated with SDF back-annotation. The power dissipation is obtained at gate level by annotating the switching activities obtained from the SDF simulation. Please note that we imposed the same time constraint for multipliers with the same value of n. This explains why most designs in Table I have the same delay.

The truncated multipliers provide a significant area, power and delay improvement with respect to the full-rounded multipliers. As an example for the case $n = 24, h = 1$ the developed multiplier provides a power and area reduction of 47% and 42%, respectively, while being also slightly faster than the full-rounded multiplier.

The good performance are due to the considered compensation function that is well suited to TDM implementation without the need of additional logic. As we will discuss later, this is not true for other truncated multipliers proposed in the Literature.

TABLE II
PREVIOUSLY PROPOSED TRUNCATED MULTIPLIERS CONSIDERED IN THE COMPARISONS

	Signed h=0	Unsigned h=0	Signed/Unsigned h>0
Jou *et al.* [15]	proposed in the original paper	proposed in the original paper	can be extended to this case
Van *et al.* [17],[18]	proposed in the original paper	can be extended to this case	proposed in the original paper
Curticapean *et al.* [16]	-	proposed in the original paper	can be extended to this case
Kuang *et al.* [22]	-	proposed in the original paper	can be extended to this case
Liao *et al.* [19]	proposed in the original paper	-	-
Swartzlander *et al.* [10],[11]	-	proposed in the original paper	proposed in the original paper

B. Comparison With Previous Art

In the comparison with literature results, we restrict our attention to variable-correction truncated multipliers [10]–[22] which provide much lower errors with respect to constant-correction circuits.

It is worth highlighting that not every literature contribution covers both signed and unsigned topologies and can be applied to the general case $h > 0$. A review of the approaches found in the Literature is reported in Table II. In [15] Jou *et al.* consider the case of both signed and unsigned multipliers. In [15] additional w columns in the matrix are considered, and the number of output bits is $n + w$. In our paper (see Fig. 1), similarly to the rest of the Literature, the number of output bits is equal to n and is fixed[2], while h is a design parameter that can be used to increase the accuracy without changing the weight of the output LSB. Therefore the architecture of [15] can be compared with our approach only when $h = 0$. However, as shown in Section VII, the results obtained for unsigned multiplier with $h = 0$ can be extended rather straightforwardly to cover both signed and unsigned multipliers with $h > 0$. Thus, as highlighted in Table II, we have extended the approach of [15] to the case $h > 0$, introducing a suitable rounding constant that minimizes the total mean error (as discussed in Section III, this also minimizes the total mean square error).

The same considerations apply to the multipliers proposed by Kuang *et al.* [22]. In this case, however, the original paper considers only unsigned multipliers. Therefore the technique of [22] cannot be extended to signed multipliers with $h = 0$. Please note that [22] proposes two approaches. We will consider only the first one (Type I) that provides a lower mean square error.

The truncated multipliers proposed by Van *et al.* [17], [18] include signed multipliers for $h \geq 0$. The error compensation function proposed for $h > 0$ can however be extended to the unsigned multiplier with $h = 0$.

[2]The approach proposed in this paper could be easily extended to consider $w > n$ output bits, still keeping h as an accuracy parameters.

Curticapean *et al.* [16] consider only the unsigned multiplier with $h = 0$. The architecture proposed in [16] can hence be extended (still adding a suitable rounding constant) to signed and unsigned multipliers with $h > 0$.

The variable-correction multipliers proposed by Swartzlander *et al.* [10], [11] consider the unsigned case for $h \geq 0$. This technique, therefore, cannot be extended to the signed multipliers with $h = 0$.

Finally the approach of Liao *et al.* [19] is originally developed for the signed $h = 0$ case and cannot be extended to any other case.

1) Mean and Mean Square Errors: The data in Table III show that, for the investigated multipliers, both the mean and the mean square errors decrease by increasing h. The mean square error is always larger than the truncation error variance (σ^2_{trunc}), as expected. Note that σ^2_{trunc} can be approximated as $\text{LSB}^2/12 \cong 0.083 \cdot \text{LSB}^2$, when m is sufficiently large (see (22)).

As it can be seen from Table III, the proposed truncated multipliers exhibit good error performance. When $h = 0$ only the multiplier of Kuang *et al.* [22] is able to obtain a slightly lower mean square error than our architecture. On the other hand, the multiplier of Kuang *et al.* [22] yields larger mean square error than ours for $h > 0$. The multipliers of Van *et al.* [17], [18] and Jou *et al.* [15] exhibit the lower mean square error when $h > 0$. In this conditions, however, the performances of the proposed multipliers are very close to that of Van and Jou multipliers, especially for $n \geq 12$.

It is also interesting to observe that the proposed multipliers provide a remarkable lower mean error (μ_{total}) with respect to all other multipliers. Although in most applications the mean square error represents the error metric which better describe the multiplier performances, in same application also having a low mean error can be important. As an example if one considers the errors accumulated in DCT algorithm for image processing, the mean error yields lighter or darker horizontal or vertical lines in the image, while the effects of mean square error may be less noticeable.

2) Electrical Performances (Area, Power, Delay): As we have pointed-out previously, our truncated multipliers are well suited for effective tree based implementations. This happens for most of the previously proposed truncated multipliers. The architectures of Jou *et al.* [15], Van *et al.* [17], [18] and Curticapean *et al.* [16], on the other hand, use array multiplier with a ripple architecture, that are very slow and power hungry. These architectures can also be implemented with a much more effective carry-save TDM reduction method. However, it is worth highlighting that the terms added to the partial-products matrix related to the compensation function are still obtained with a ripple AND/OR network, which increases power dissipation and propagation delay.

In order to have a fair comparison among every approach, we have implemented every multiplier by using a carry-save TDM reduction tree followed by a fast carry-propagate adder, by targeting a 0.18 μm technology.

Fig. 9 shows the results of the implementations obtained by varying the delay constraint during synthesis, in the case $n = 16, h = 1$. The figure highlights the area/delay trade-off common to all digital circuits. The reported data show that the

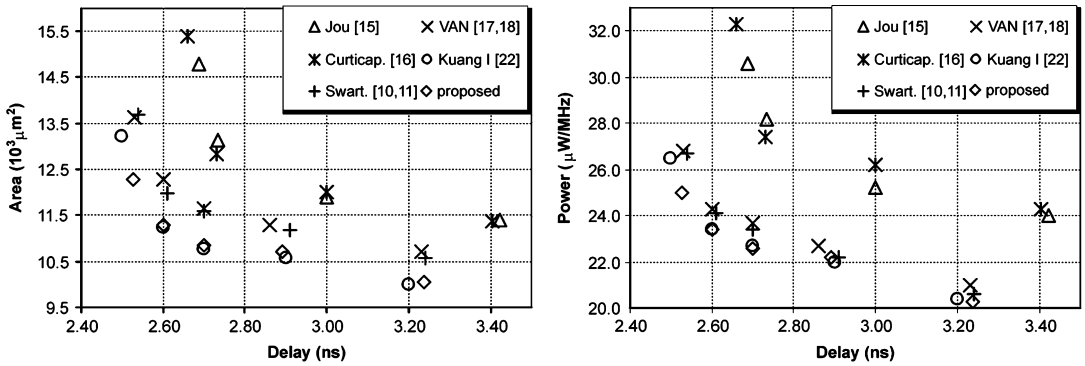

Fig. 9. Signed multipliers performances for $n = 16, h = 1$ by varying the delay constrain imposed during circuit synthesis (0.18 μm technology).

TABLE III
MEAN AND MEAN SQUARE ERRORS OF TRUNCATED MULTIPLIERS ($^*S$ = DATA APPLY TO SIGNED MULT. ONLY; $^*U$ = DATA APPLY TO UNSIGNED MULT ONLY; REMAINING DATA ARE VALID FOR BOTH SIGNED AND UNSIGNED MULT.)

n	h	ε^2_{total}							μ_{total}						
		This Paper	Jou [15]	Van [17,18]	Curtic. [16]	Kuang I [22]	Liao [19]	Swart. [10,11]	This Paper	Jou [15]	Van [17,18]	Curtic. [16]	Kuang I [22]	Liao [19]	Swart. [10,11]
8	0	0.216	0.598	0.263^{*S}	0.234^{*U}	0.213^{*U}	0.201^{*S}	0.263^{*U}	-0.017	-0.651	0.249^{*S}	0.032^{*U}	0.054^{*U}	0.000^{*S}	0.249^{*U}
	1	0.118	0.104	0.104	0.187	0.159	-	0.123	0.006	-0.060	-0.060	0.270	-0.223	-	-0.125
	2	0.090	0.087	0.087	0.108	0.101	-	0.092	0.003	-0.019	-0.019	0.139	-0.112	-	-0.063
	3	0.085	0.084	0.084	0.089	0.085	-	0.085	0.002	-0.002	-0.002	0.072	-0.058	-	-0.033
10	0	0.258	0.700	0.305^{*S}	0.285^{*U}	0.255^{*U}	0.243^{*S}	0.305^{*U}	-0.016	-0.694	0.250^{*S}	0.019^{*U}	0.054^{*U}	0.000^{*S}	0.250^{*U}
	1	0.129	0.119	0.119	0.196	0.170	-	0.134	0.007	-0.088	-0.088	0.262	-0.223	-	-0.125
	2	0.094	0.090	0.090	0.110	0.104	-	0.095	0.004	-0.038	-0.038	0.133	-0.112	-	-0.063
	3	0.086	0.085	0.085	0.090	0.088	-	0.086	0.002	-0.015	-0.015	0.068	-0.056	-	-0.032
12	0	0.300	0.780	0.347^{*S}	0.333^{*U}	0.296^{*U}	0.285^{*S}	0.347^{*U}	-0.016	-0.718	0.250^{*S}	0.011^{*U}	0.055^{*U}	0.000^{*S}	0.250^{*U}
	1	0.140	0.134	0.134	0.206	0.180	-	0.144	0.008	-0.104	-0.104	0.257	-0.223	-	-0.125
	2	0.096	0.094	0.094	0.113	0.106	-	0.097	0.004	-0.049	-0.049	0.130	-0.111	-	-0.063
	3	0.086	0.086	0.086	0.090	0.089	-	0.086	0.002	-0.022	-0.022	0.066	-0.056	-	-0.031
14	0	0.342	0.847	0.389^{*S}	0.380^{*U}	0.338^{*U}	0.326^{*S}	0.389^{*U}	-0.016	-0.732	0.250^{*S}	0.006^{*U}	0.055^{*U}	0.000^{*S}	0.250^{*U}
	1	0.151	0.147	0.147	0.216	0.191	-	0.155	0.008	-0.113	-0.113	0.254	-0.223	-	-0.125
	2	0.099	0.098	0.098	0.115	0.109	-	0.100	0.004	-0.055	-0.055	0.128	-0.111	-	-0.063
	3	0.087	0.087	0.087	0.091	0.089	-	0.087	0.002	-0.026	-0.026	0.064	0.069	-	-0.031
16	0	0.384	0.905	0.431^{*S}	0.425^{*U}	0.380^{*U}	0.368^{*S}	0.431^{*U}	-0.016	-0.740	0.250^{*S}	0.003^{*U}	0.055^{*U}	0.000^{*S}	0.250^{*U}
	1	0.160	0.160	0.160	0.227	0.201	-	0.165	0.008	-0.118	-0.118	0.252	-0.223	-	-0.125
	2	0.102	0.101	0.101	0.118	0.112	-	0.102	0.004	-0.058	-0.058	0.127	-0.111	-	-0.063
	3	0.088	0.087	0.087	0.092	0.090	-	0.088	0.002	-0.028	-0.028	0.064	-0.056	-	-0.031

proposed multiplier and the one of Kuang *et al.* exhibit very similar performances.

The multipliers of Van *et al.* and the architectures of Swartzlander *et al.* are also very similar in terms of electrical performances. Both architectures, however, are less effective than Kuang multipliers.

The multipliers proposed by Jou *et al.* and Curticapean *et al.* are not very efficient. As already pointed out, this is due to the ripple structure used in the network implementing the compensation function. This increases the delay and results in signal glitching, that increases the power dissipation. Further Jou *et al.* and Curticapean *et al.* multipliers do not exhibit good error performances (see Table III). In the following we will not consider these two architectures anymore.

It is interesting to observe that also the Van *et al.* multiplier includes a ripple error compensation network. In this case, however, this network reduces to a single multiple-input gate[3] which, in turn, can be implemented with a tree structure. This justifies why Van *et al.* multiplier shows better performances than Jou *et al.* and Curticapean *et al.* approaches.

As shown in Fig. 9, the multiplier of Kuang *et al.* has electrical performance similar to our multiplier. However, the accuracy of Kuang *et al.* multiplier is comparable to our multiplier only for $h = 0$, while for $h > 0$ the Kuang *et al.* multiplier

[3]Please note that the error compensation network proposed by *Van et al.* reduces to a single multiple-input gate only for $h > 0$. For $h = 0$ the network of *Van et al.* multiplier is similar to the networks of *Jou et al.* and *Curticapean et al.* multipliers.

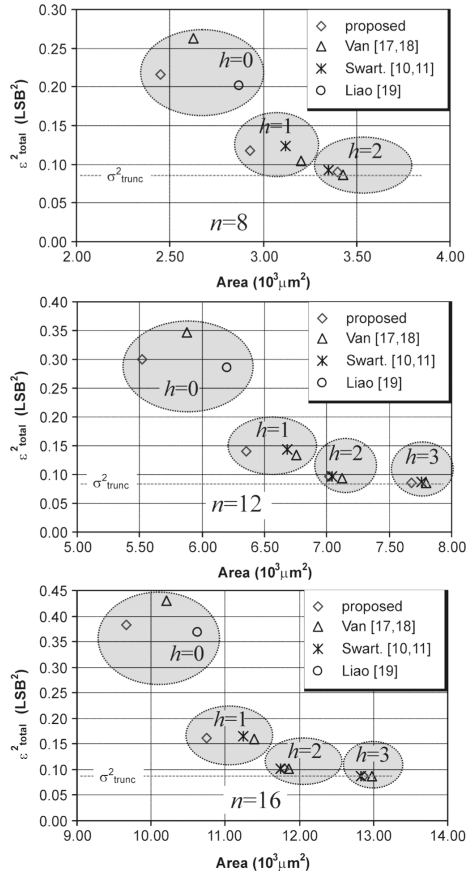

Fig. 10. Trade off between area and mean square error for different signed truncated multipliers.

yields a larger error. As a consequence, the multiplier of Kuang *et al.* has also been excluded from subsequent analyses.

C. Area Versus Accuracy Trade-Off

In practical applications the accuracy is generally constrained by system-level consideration. The designer goal is hence to achieve the lowest complexity for a specified accuracy. As we have highlighted in the previous sections, h is a trade-off parameter between accuracy and complexity that can be used to obtain an optimal design. Therefore, a comparison of the different truncated multipliers considering this trade-off between accuracy and complexity can be very useful.

To that purpose, we have simulated and implemented our truncated multipliers and the multipliers proposed by Van *et al.*, Liao *et al.* and Swartzlander *et al.*. Considering the area/delay trade-off highlighted before, for fairness, the circuits have been implemented by using the same time constraint for multipliers with the same value of n. Fig. 10 summarizes the obtained results on a diagram area vs mean square error for $n = 8, 12$ and 16 bits and $h = 0, 1, 2, 3$. Each diagram shows also the truncation error variance $(\sigma^2_{\text{trunc}})$, that represents a lower bound for the mean square error.

The data of Fig. 10 show that for $h = 0$ the proposed multiplier exhibits the best trade-off between area and accuracy. The architecture of Van *et al.* [17], [18] appears ineffective, since both area and error are larger than our multiplier. The architecture of Liao *et al.* [19], while being slightly more accurate than our technique, requires an appreciably larger area.

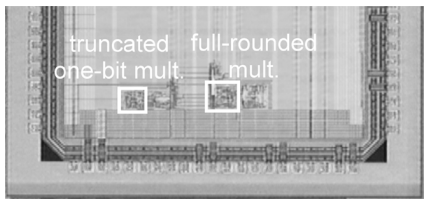

Fig. 11. Test chip micrograph realized in UMC 0.18 μm technology.

TABLE IV
EXPERIMENTAL PERFORMANCES OF THE SIGNED 16 BIT MULTIPLIERS REALIZED IN THE TEST CHIP SHOWN IN FIG. 11

Multiplier	f_{CLK} (MHz)	Area (10^3 μm^2)	P_D (μW/MHz)
truncated one-bit ($h=0$)	385	19.1	36.6
full-rounded	340	33.7	73.2

A similar picture is shown for $h = 1$, with a difference for the multiplier of Van *et al.* that shows a lower approximation error (paid, once again, with a noticeably larger area).

For $h \geq 2$ the error of the truncated multipliers is very close to the limit imposed by truncation error. In these cases, the best solution is selecting the multiplier with the lowest area, that is generally the one proposed in this paper.

IX. EXPERIMENTAL VERIFICATION

The performance of the truncated multiplier proposed in this paper have been experimentally verified on a test chip [21]. In this chip we have implemented a truncated multiplier for $n = 16, h = 0$ and also, as a comparison, a full-rounded multiplier. The technology used is 0.18 μm with six levels of metal. The photograph of the test chip is shown in Fig. 11.

The experimental performance are summarized in Table IV. The developed multiplier is able to halve the power dissipation and to reduce of about 44% the area occupation with respect to the full rounded multiplier. In addition the multiplier designed with the proposed technique is 13% faster than the full-rounded multiplier. These improvements are consistent with data in Table I. On the other hand, the actual experimental values reported in Table IV are significantly worse with respect to the data in Table I. This is due to two reasons. Firstly, an un-optimized standard-cell library (using only Manhattan geometry in the layout) was employed to fabricate the chip, while a more effective vendor-supplied library was considered for the simulations reported in Table I. Secondly, data in Table I are synthesis results that do not take into account the overhead due to place and route.

In order to verify the advantages coming from the use of the proposed truncated multiplier in a typical DSP application, we have implemented a FIR filter by using an Multiply-and-Accumulate (MAC) unit, as shown in Fig. 12. The input sequence and the taps of the filter are represented as 16 bits fractional 2's complement signed binary values. A truncated multiplier with $n = 16$ input and output bits is used to compute the multiplications (see Fig. 12), while 4 guard bits are added to the accumulator in order to avoid overflow. The final output is rounded

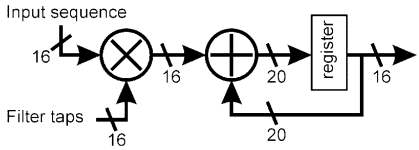

Fig. 12. Multiply-and-Accumulate (MAC) unit by using a truncated multiplier with $n = 16$ input and output bits.

TABLE V
PERFORMANCE OF THE MAC UNITS IMPLEMENTED BY USING DIFFERENT MULTIPLIERS. ERRORS ARE NORMALIZED TO THE LSB AT THE OUTPUT OF MAC UNIT

Mult. type	μ_{FIR} (LSB)	ε^2_{FIR} (LSB^2)	Area ($10^3\mu m^2$)	Relative Area	Power (μW/MHz)	Relative Power
full width	0.000	0.083	28.97	100%	71.3	100%
full round	0.031	0.116	24.22	84%	55.9	78%
trunc. (h=0)	-0.272	0.296	14.08	49%	35.5	49%
trunc. (h=1)	-0.024	0.140	15.30	53%	38.6	54%
trunc. (h=2)	0.049	0.122	16.42	57%	41.3	58%

back to 16 bits. The rounding is realized by initializing the accumulator with the rounding constant before each accumulation.

In order to estimate the error introduced by the truncated multiplier, we considered a 100 taps low-pass FIR filter. The sample frequency is assumed to be 48 MHz. The filter coefficients have been calculated with the equiripple algorithm by imposing a 1-MHz pass-band, a 1-MHz transition band and a 60-dB attenuation in the stopband.

Table V shows the performance of the MAC units, both in term of accuracy and electrical performance, by considering different types of multipliers: full-width, full-rounded, truncated (with $h = 0, 1, 2$). The circuits are designed for a maximum clock frequency of 250 MHz. The parameters $\mu_{\rm FIR}$ and $\varepsilon^2_{\rm FIR}$ are the mean and the mean square errors obtained by applying a white uniformly-distributed noise signal at the input of the filter.

From Table V it can be observed that the MAC using the full-rounded multiplier (full-rounded MAC) results in a slight reduction in area occupation (16%) and power dissipation (21%) with respect to the MAC unit using the full-width multiplier (full-width MAC). This reduction can be explained by observing that the full-rounded MAC uses a 20 bit accumulator (see Fig. 9) while the full-width MAC has a 36 bit accumulator.

The mean square error at the output of the filter $(\varepsilon^2_{\rm FIR})$ of the full-rounded MAC is comparable with the error of full-width MAC.

The MAC implementations using truncated multipliers exhibit a sensible improvement in electrical performances. As an example the implementation with $h = 0$ yields more than 50% reduction of both area occupation and power dissipation with respect to full-width MAC. This improvement, however, is paid with a rather large error $\varepsilon^2_{\rm FIR}$ at the output of the filter. This error is due to the accumulation of multiplier error over the filter taps. As show in Table V the error $\varepsilon^2_{\rm FIR}$ can be reduced by increasing h. As an example, the truncated implementation with $h = 2$ highlights a 43% reduction of area occupation and a 42% reduction of power dissipation with respect to full-width MAC, with an error comparable to a full-rounded multiplier.

These results demonstrate the benefits that can be obtained by using the proposed multipliers in a real application.

X. CONCLUSION

In this paper a thorough analysis of truncated multipliers implementing variable-correction has been presented.

It has been shown that the optimal compensation function, that minimizes the mean square error, is a quadratic form of the partial-products of the IC. A sub-optimal compensation function, best suited for hardware implementation, has also been obtained as a linear combination of the IC terms.

The practical implementation of truncated multipliers with the linear compensation function has been investigated. It has been shown that the proposed truncated multipliers can easily be realized by using a carry-save reduction tree followed by a final carry-propagate adder.

The characteristics of the truncated multipliers developed in this paper have been extensively compared with previously proposed circuits. Our analysis has shown that the proposed multipliers represent, in most cases, the best trade-off between complexity and accuracy.

Experimental performances, obtained from a test chip in 0.18 μm technology have been presented. In addition, the use of proposed truncated multipliers in a multiply and accumulate unit has been analyzed.

APPENDIX

1) Derivation of (51), (53): In order to solve the system (50) let us firstly compute the mean error $\mu_{\rm erasing}$ obtained when the linear compensation function (48) is employed. By substituting (48) and (46) in (26) we have

$$\mu_{\rm erasing} = \sum_{A\varepsilon\Theta}\left(-\mu_{\rm trunc} - H - \sum_{i=1}^{n_{\rm eq}}\alpha_i\gamma_i - \sum_{i=1}^{n_{\rm eq}}\sum_{\substack{j=1\\ j\neq 1}}^{n_{\rm eq}}\frac{f_{i,j}}{2}\gamma_i\gamma_j\right)\cdot P(A) \tag{62}$$

where we have defined

$$H = K - K_l; \quad \alpha_i = f_i - l_i \tag{63}$$

Since the terms $\gamma_i (i = 1, \ldots, n_{\rm eq})$ are independent we can write

$$\sum_{A\varepsilon\Theta}\gamma_i P(A) = \frac{1}{4}; \quad \sum_{A\varepsilon\Theta}\gamma_i\gamma_j P(A) = \frac{1}{16}; \quad \sum_{A\varepsilon\Theta}\gamma_i\gamma_j\gamma_r P(A) = \frac{1}{64}. \tag{64}$$

By using (64), with simple algebra, the (62) can be simplified as follows:

$$\mu_{\rm erasing} = -\mu_{\rm trunc} - H - \frac{1}{4}\sum_{i=1}^{n_{\rm eq}}\alpha_i - \frac{1}{16}\sum_{i=1}^{n_{\rm eq}}\sum_{\substack{j=1\\ j\neq i}}^{n_{\rm eq}}\frac{f_{i,j}}{2}. \tag{65}$$

We can now compute σ^2_{comp}. By substituting (65), (48) and (46) in (49) we have

$$\sigma^2_{\text{comp}} = \sum_{A\varepsilon\Theta} \left(\sum_{i=1}^{n_{\text{eq}}} \alpha_i \left(\gamma_i - \frac{1}{4} \right) + \sum_{i=1}^{n_{\text{eq}}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{\text{eq}}} \frac{f_{i,j}}{2} \left(\gamma_i \gamma_j - \frac{1}{16} \right) \right)^2 P(A). \quad (66)$$

By substituting (66) in the system (50), one obtains

$$\sum_{A\varepsilon\Theta} \gamma_r \cdot \left(\sum_{i=1}^{n_{\text{eq}}} \alpha_i \left(\gamma_i - \frac{1}{4} \right) + \sum_{i=1}^{n_{\text{eq}}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{\text{eq}}} \frac{f_{i,j}}{2} \left(\gamma_i \gamma_j - \frac{1}{16} \right) \right) \cdot P(A) = 0, \qquad \text{for } r = 1, \ldots, n_{\text{eq}}. \quad (67)$$

This system can be again simplified with the help of (64). With simple algebra one obtains

$$\alpha_r = -\frac{1}{2} \sum_{\substack{j=1 \\ j\neq i}}^{n_{\text{eq}}} \frac{f_{i,r}}{2}, \qquad \text{for } r = 1, \ldots, n_{\text{eq}}. \quad (68)$$

From (68) the result (51) is easily obtained.

The constant K_l (that is H) can be fixed by imposing $\mu_{\text{erasing}} = -\mu_{\text{trunc}}$. From (65) we have

$$-H - \frac{1}{4} \sum_{i=1}^{n_{\text{eq}}} \alpha_i - \frac{1}{16} \sum_{i=1}^{n_{\text{eq}}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{\text{eq}}} \frac{f_{i,j}}{2} = 0. \quad (69)$$

By substituting (68) in (69) and solving for H we found

$$H = \frac{1}{16} \sum_{i=1}^{n_{\text{eq}}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{\text{eq}}} \frac{f_{i,j}}{2}. \quad (70)$$

From this equation the result (53) is easily verified.

2) Derivation of (57): From the definition (26) one can write

$$\mu_{\text{erasing}} = \sum_{A\varepsilon\Theta} [f^*_{\text{lin}}(A) - f_{\text{lin}}(A) + f_{\text{lin}}(A) - \mu_{\text{LSP}}(A)] \cdot P(A). \quad (71)$$

By exploiting the properties of f_{lin} (IC) and by imposing $\mu_{\text{erasing}} = -\mu_{\text{trunc}}$ one obtains

$$\sum_{A\varepsilon\Theta} [f^*\text{lin}(A) - f_{\text{lin}}(A)] \cdot P(A) = 0. \quad (72)$$

Let us neglect, for the time being, that K^* is integer. With a few algebra and with the help of (64) one has

$$K^* = K_l + \frac{1}{4} \sum_{i=1}^{n_{\text{eq}}} (l_i - l_i^*) - \frac{\mu_{\text{trunc}}}{\text{LSB}_{\text{IC}}}. \quad (73)$$

By using (51), (56) one obtains

$$K^* = \frac{1}{2} 2^{-n_{\text{eq}}} + 2^h. \quad (74)$$

By rounding the last equation, (57) is obtained.

REFERENCES

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

[2] S. R. Kuang, J. M. Jou, and Y. L. Chen, "The design of an adaptive on-line binary arithmetic coding chip," *IEEE Trans Circuits Syst. I, Fundam. Theory Appl.*, vol. 45, no. 7, pp. 693–706, Jul. 1998.

[3] D. De Caro, N. Petra, and A. G. M. Strollo, "A 380 MHz direct digital synthesizer/mixer with hybrid CORDIC architecture in 0.25 μm CMOS," *IEEE J. Solid-State Circuits*, vol. 42, no. 1, pp. 151–160, Jan. 2007.

[4] J. A. Pineiro, S. F. Oberman, J. M. Muller, and J. D. Bruguera, "High-speed function approximation using a minimax quadratic interpolator," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 304–318, Mar. 2005.

[5] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Trans. Comput.*, vol. C-22, no. 12, pp. 1045–1047, Dec. 1973.

[6] Y. C. Lim, "Single-precision multiplier with reduced circuit complexity for signal processing applications," *IEEE Trans. Comput.*, vol. 41, no. 10, pp. 1333–1336, Oct. 1992.

[7] M. J. Schulte and E. E. Swartzlander Jr., "Truncated multiplication with correction constant [for DSP]," in *Workshop on VLSI Signal Process.*, Oct. 1993, pp. 388–396.

[8] M. J. Shulte and E. E. Swartzlander Jr., "Truncated multiplication with correction constant," *VLSI Signal Process. VI*, pp. 388–396, 1993.

[9] S. S. Kidambi, F. El-Guibaly, and A. Antonious, "Area-efficient multipliers for digital signal processing applications," *IEEE Trans. Circuits Syst. II, Anal. Digit. Signal Process.*, vol. 43, no. 2, pp. 90–95, Feb. 1996.

[10] E. J. King and E. E. Swartzlander Jr., "Data dependent truncated scheme for parallel multiplication," in *Proc. 31th Asilomar Conf. on Signals, Circuits Syst.*, 1997, pp. 1178–1182.

[11] E. E. Swartzlander Jr., "Truncated multiplication with approximate rounding," in *Proc. 33th Asilomar Conf. on Signals, Circuits Syst.*, 1999, pp. 1480–1483.

[12] M. J. Shulte, J. E. Stine, and J. G. Jansen, "Reduced power dissipation through truncated multiplication," in *IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, 1999, pp. 61–69.

[13] J. E. Stine and O. M. Duverne, "Variations on truncated multiplication," in *Proc. Euromicro Symp. on Digital Syst. Design*, 2003.

[14] H. Park and E. E. Swartzlander Jr., "Truncated multiplication with symmetric correction," in *Proc. Asilomar Conf. on Signals, Syst. and Comput. (ACSSC).*, Oct. 2006, pp. 931–934.

[15] J. M. Jou, S. R. Kuang, and R. D. Chen, "Design of low-error fixed-width multipliers for DSP applications," *IEEE Trans. Circuits Syst. II, Anal. Digit. Signal Process.*, vol. 46, no. 6, pp. 836–842, Jun. 1999.

[16] F. Curticapean and J. Niittylahti, "A hardware efficient direct digital frequency synthesizer," in *Proc. IEEE Int. Conf. on Electron., Circuits Syst. (ICECS 2001)*, Sept. 2–5, 2001, vol. 1, pp. 51–54, Malta.

[17] L. Van, S. Wang, and W. Feng, "Design of the lower error fixed-width multiplier and its application," *IEEE Trans. Circuits Syst. II, Anal. Digit. Signal Process.*, vol. 47, no. 10, pp. 1112–1118, Oct. 2000.

[18] L. Van and C. Yang, "Generalized low-error area-efficient fixed-width multiplies," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 8, Aug. 2005.

[19] Y. C. Liao, H. C. Chang, and C. W. Liu, "Carry estimation for two's complement fixed-width multipliers," in *Workshop on Signal Process. Syst. (SiPS)*, Banff, Canada, Oct. 2006, pp. 345–350.

[20] A. G. M. Strollo, N. Petra, and D. De Caro, "Dual-tree error compensation for high performance fixed-width multipliers," *IEEE Trans. Circuits Syst. II, Express Briefs*, vol. 52, no. 8, pp. 501–507, Aug. 2005.

[21] N. Petra, D. De Caro, and A. G. M. Strollo, "Design of fixed-width multipliers with minimum mean square error," in *Proc. IEEE Eur. Conf. on Circuits Theory and Des. (ECCTD 2007)*, Sevilla, Spain, Aug. 2007.

[22] S. R. Kuang and J. P. Wang, "Low-error configurable truncated multipliers for multiply-accumulate applications," *Electron. Lett.*, vol. 42, no. 16, pp. 904–905, Aug. 3, 2006.

[23] R. Michard, A. Tisserand, and N. Veyrat-Charvillon, "Carry prediction and selection for truncated multiplication," in *Workshop on Signal Processing Syst. (SiPS)*, Banff, Canada, Oct. 2006, pp. 339–344.

[24] 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.

[25] P. F. Stelling, C. U. Martel, V. G. Oklobdzija, and R. Ravi, "Optimal circuits far parallel multipliers," *IEEE Trans. Comput.*, vol. 47, no. 3, pp. 273–285, Mar. 1998.

Nicola Petra (M'05) received the Laurea degree (with honors) and the Ph.D. degree from the University of Napoli "Federico II" in 2002 and in 2007, respectively.

He is now working as a researcher at the Department of Biomedical Electronics and Telecommunications Engineering of the University of Napoli "Federico II," Naples, Italy. His research interests include design of digital VLSI circuits for telecommunications and high-performance arithmetic circuits. . He authored or co-authored more than 20 papers on scientific journals and international conferences.

Dr. Petra acted as a reviewer for IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATED (VLSI) SYSTEMS.

Davide De Caro (M'05–SM'09) was born in Naples, Italy, on February 9, 1973. He received the M.S. degree in electronic engineering with honors, in July 1999, and the Ph.D. degree in electronic engineering and computer science, in 2003, both from the University of Naples "Federico II," Naples, Italy.

He has worked in the area of digital integrated VLSI circuit design for the last eight years. Since March 2003 he is a Researcher at the Department of Biomedical Electronics and Telecommunication Engineering, University of Naples, Italy, where he is working on high-performance flip-flops (including both low-power and high-speed structures), VLSI implementation of arithmetic circuits (squarers, fixed-width multipliers, Reed-Solomon decoders, Galois-field multipliers), direct digital frequency synthesizers and digital mixers.

Dr. De Caro is author or co-author of more than 40 technical papers in international journals and refereed international conferences. He acted as a reviewer for IEEE Transactions on Circuits and Systems I and II, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATED (VLSI) SYSTEMS, IEEE JOURNAL OF SOLID-STATE CIRCUITS and IEEE TRANSACTIONS ON COMPUTERS.

Valeria Garofalo was born in 1981. She received the degree in telecommunication engineering, with honours, from the University of Naples "Federico II," Naples, Italy, in October 2006, discussing a thesis whose title is "Analysis and design of code compression techniques for ARM7TDMI processor". She has been working with the VLSI research group at the Department of Biomedical Electronic and Telecommunication Engineering of the University of Naples, working toward Ph.D. degree.

Her research interests are the architecture and algorithms for real-time code decompression in embedded systems (software compression, hardware decompression), VLSI digital circuits design with special emphasis on arithmetic digital circuits (fixed-width multipliers and FIR filters) and microcontrollers that works in conjunction with innovative gas sensors.

Ettore Napoli was born in Italy in 1971. He received the Electronic engineering degree (with honors) in 1995, the Ph.D. degree in electronic engineering in 1999, and the physics degree (with honors) in 2009, all from the University of Naples "Federico II," Naples, Italy.

He has been Associate Professor, University of Napoli since 2005, and was a Research Associate at the Engineering Department of the University of Cambridge, Cambridge, U.K., in 2004. His scientific interests include modeling and design of power semiconductor devices and VLSI circuit design. In the power devices field his main interests are the PiN diode, the vertical IGBT, superjunction devices and Lateral IGBTs. In the VLSI field, his interests are high speed arithmetic subsystems and advanced flip-flops.

Prof. Napoli is author or co-author of more than 80 papers published in international journals and conferences.

Antonio Giuseppe Maria Strollo (M'05–SM'06) received the Laurea degree (*cum laude*) in electronic engineering and the Ph.D. degree in electronic engineering and computer science from the University of Napoli Federico II, Italy, in 1988 and 1992, respectively. From 1990 to 1998 he was a research assistant in the Department of Electronic Engineering at the University of Napoli, Italy.

In November 1998, he was appointed an associate professor at the University of Napoli Federico II and has been a full professor since November 2002. From 2005 to 2008 he has been the head of the Department of Biomedical Electronic and Telecommunication Engineering of the same University. His initial research activities covered the area of power electronics. In this field he has worked on switching power converter simulation, modeling and simulation of power devices (SIT, IGBT, superjunction), SPICE modeling of PiN diodes and IGBTs, characterization techniques, electro-thermal modeling of power devices, optimization techniques of power bipolar devices with local lifetime control. His current research interests include design and analysis of VLSI circuits. In particular, he is working on advanced architectures for direct-digital frequency synthesis, techniques for clock dithering in digital ASICs, high-performance arithmetic circuits, high speed flip-flops. He has published more than 110 papers on international journals and refereed conferences.

Dr. Strollo is currently serving as associate editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS.