electronics

MDPI

Article

Design of Generalized Enhanced Static Segment Multiplier with Minimum Mean Square Error for Uniform and Nonuniform Input Distributions

Gennaro Di Meo *, Gerardo Saggese, Antonio G. M. Strollo and Davide De Caro

Department of Electrical Engineering and Information Technology, University of Naples Federico II, 80125 Naples, Italy

* Correspondence: gennaro.dimeo@unina.it

Abstract: In this paper, we analyze the performances of an Enhanced Static Segment Multiplier (ESSM) when the inputs have both uniform and non-uniform distribution. The enhanced segmentation divides the multiplicands into a lower, a middle, and an upper segment. While the middle segment is placed at the center of the inputs in other implementations, we seek the optimal position able to minimize the approximation error. To this aim, two design parameters are exploited: m, defining the size and the accuracy of the multiplier, and q, defining the position of the middle segment for further accuracy tuning. A hardware implementation is proposed for our generalized ESSM (gESSM), and an analytical model is described, able to find m and q which minimize the mean square approximation error. With uniform inputs, the error slightly improves by increasing q, whereas a large error decrease is observed by properly choosing q when the inputs are half-normal (with a *NoEB* up to 18.5 bits for a 16-bit multiplier). Implementation results in 28 nm CMOS technology are also satisfactory, with area and power reductions up to 71% and 83%. We report image and audio processing applications, showing that gESSM is a suitable candidate in applications with non-uniform inputs.

Keywords: approximate multiplier; static segmentation; low-power; approximate computing

Citation: Di Meo, G.; Saggese, G.; Strollo, A.G.M.; De Caro, D. Design of Generalized Enhanced Static Segment Multiplier with Minimum Mean Square Error for Uniform and Nonuniform Input Distributions. *Electronics* **2023**, *12*, 446. https://doi.org/10.3390/electronics12020446

Academic Editor: Spyridon Nikolaidis

Received: 20 December 2022
Revised: 10 January 2023
Accepted: 12 January 2023
Published: 15 January 2023

1. Introduction

The reduction of power consumption in DSP algorithms is a primary concern for the feasible realization of electronic systems and calls for the adoption of suitable design strategies. Convolution, dot product, and correlation are well diffused operations in applications ranging from telecommunication to image and audio processing, and make the design critical due to the extensive employment of adders and multipliers. As an example, IoT and mobile devices, which implement deep learning and machine learning algorithms, demand quantization techniques, down-sampling, and arithmetic approximations to reduce the hardware complexity [1–3]. In telecommunication, the suppression of noise in transceivers, necessary for improving the receiver sensitivity, requires cancellation methods based on adaptive filtering [4–6]. The huge number of multipliers, used for adaptation, increases the power consumption and demands specific techniques aimed to reduce area and power while preserving the quality of results [7–11]. Low-power designs are also required for audio applications [12] in which banks of filters realize operations as equalization and denoising.

Since multipliers are responsible for large power consumption in DSP algorithms, hardware-efficient designs are required for achieving acceptable performances. As the nature of many DSP algorithms is error tolerant (as adaptive filtering or image and audio processing), the Approximate Computing paradigm constitutes a valuable means of improving the hardware performances of multipliers, providing a way to approximate the design at the cost of a tolerable accuracy loss. Approximations can be introduced in the partial product generation stage, in the partial product matrix (PPM) compression step,

or in the final carry propagate adder of the multiplier. Since the PPM compression stage is rich in half-adders and full-adders, the approximation of the compression circuit can lead to a significant hardware improvement. In [13], the authors involve AND and OR gates to merge the partial product generation stage with the compression step, while [14] deletes some rows from the PPM at design time. In [15] a recursive approach is proposed, in which the multiplier is decomposed into small approximate units. The paper [16] shows a compression scheme in which OR gates substitute half-adders and full-adders, whereas [17] improves this technique by compensating the mean approximation error. In [18], fast counters encode the partial products by following a stacking approach, whereas the works [19–24] analyze multipliers with approximate 4–2 compressors. In these papers, the full-adders required for the realization of the exact compressor are substituted by simple logic at the cost of an error in the computation, and the carry chain between compressors is broken in order to optimize the critical path and to moderate the glitch propagation. In [20], the authors propose three compressors with different levels of accuracy, while [21] designs an error recovery module to improve the quality of results. The paper [22] shows a statistical approach for ordering the partial products in approximate 4–2 compressors, and analyzes the performances when different compressors are employed in the same multiplier. In [23], compressors with positive and negative mean error are interleaved in order to minimize the approximation effects, whereas [24] prefers NAND and NOR gates to AND and OR gates for achieving high speed performances.

The fixed-width technique is a further approach able to reduce the power, providing a way to discard some columns of the PPM [25,26]. In this case, properly weighing the partial products in the truncated PPM reduces the approximation error of the multiplier [26].

Different from the previous works, the segmentation method reduces the bit-width of the multiplicands with the aim to downsize the multiplier. The papers [27,28] describe a dynamic segment method (DSM) in which the segment is selected starting from the leading one bit of the multiplicand. While [27] adds a '1' bit at the least significant position of the segment for accuracy recovery, ref. [28] revises the multiplication as a multiply-and-add operation and applies operand truncation for further simplification. On the contrary, the paper [29] proposes a static segment method (SSM), which reduces the complexity of the selection mechanism by choosing between two fixed m-bit segments, with $n/2 \leq m < n$ and n that is the number of bits of the inputs. At the same time, an Enhanced SSM multiplier (ESSM) is also proposed in [29], which allows for selecting between three fixed portions of the inputs: the m most significant bits (MSBs), the m least significant bits (LSBs), and the m central bits of the inputs. The paper [30] improves the accuracy of the SSM multipliers by reducing the maximum approximation error, whereas in [31] the authors propose a hybrid approach in which a static stage is cascaded to a dynamic stage. In these cases, error metric results reveal satisfactory accuracy when the inputs have uniform distribution, along with acceptable power improvements with respect to the exact and the DSM multipliers. At the same time, these works do not offer an analysis with non-uniform distributed input signals; in addition, the work [29] does not show a detailed analysis of the hardware implementation of the ESSM multiplier.

In this paper, we analyze the performances of the ESSM multiplier as a function of the input stochastic distribution and propose a novel implementation able to minimize the mean square approximation error. Indeed, the statistical properties of a signal affect the probability of assuming values in a range, giving high probability ranges and low probability ranges. Starting from this observation, our idea is to properly place the central segment (named middle segment in the following) in order to minimize the segmentation error in the high probability ranges. To this aim, two design parameters are exploited: m, which defines the size of the multiplier, and q, which defines the position of the middle segment. For the error analysis, we consider inputs with uniform and non-uniform distribution, taking into consideration half-normal signals for demonstration in this last case, and also describe an analytical model able to find the optimal position q_{opt} that minimizes the multiplier error in a mean square sense.

Simulation results match with the theoretical analysis, exhibiting accuracy performances dependent on the input stochastic distribution and on the choice of m and q. Best error metrics are achieved with the middle segment placed toward the MSBs if the inputs are uniform, and with middle segment placed at the center of the inputs if the distribution is half-normal. Electrical analyses also show remarkable hardware improvements if compared with the exact multiplier, whereas only an acceptable degradation is registered with respect to the SSM multipliers. Assessments of image and audio processing applications confirm these trends, showing performances that depend on the position of the middle segment.

The paper is organized as follows: Section 2 shows the static segment method, also describing the correction technique of [30] and the enhanced segmentation presented in [29]. Then, Section 3 describes the hardware structure of the proposed gESSM, along with the analytical model used to minimize the mean square value of the approximation error. Section 4 shows the results in terms of error metrics, electrical performances, and applications in image and audio processing. A comparison with the state-of-the-art is also proposed. Section 5 further compares the multipliers finding the pareto-optimal implementations, and Section 6 concludes the paper.

2. Static Segment Method

2.1. Static Segment Multiplier and Correction Technique

The SSM technique shown in [29] provides for selecting m-bit segments from the multiplicands, with $n/2 \leq m < n$, in order to employ a smaller $m \times m$ multiplier instead of a nxn multiplier. As shown in Figure 1 for the unsigned signal A, if the $n - m$ MSBs (i.e., a_{15}, a_{14}, ..., a_{10}) are low the least significant m bits of the input are chosen, forming the segment A_{L}. On the contrary, if any bit of the n-m MSBs is high the most significant m bits are selected, forming the segment A_{H}. It is worth noting that the segmentation introduces an error when A_{H} is chosen since the bits belonging to e_A are truncated (i.e., a_5, a_4, . . . , a_0 in Figure 1). In addition, m is the only parameter able to define the accuracy and the size of the multiplier.

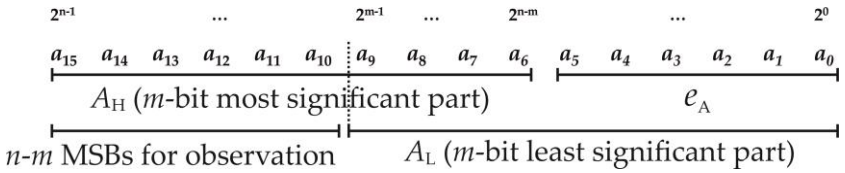


Figure 1. Segmentation of the signal A with n = 16 bits and m = 10 bits.

Then, defining α_{A} as the OR between the $n - m$ MSBs of A, the segmented input A_{ssm} is

$$A_{ssm} = \begin{cases} A_L \; if \; \alpha_A = 0 \\ A_H \; if \; \alpha_A = 1 \end{cases} \tag{1}$$

A similar expression holds also for the input B and the corresponding segment B_{ssm}. Then, the segmented multiplication is

$$\gamma_{ssm} = \left(A_{ssm} \cdot 2^{SHa,ssm}\right) \cdot \left(B_{ssm} \cdot 2^{SHb,ssm}\right) = (A_{ssm} \cdot B_{ssm}) \cdot 2^{SHssm} \tag{2}$$

with SHa,ssm, SHb,ssm that are

$$\begin{aligned} SHa,ssm &= \begin{cases} 0 & if \; \alpha_A = 0 \\ n-m & if \; \alpha_A = 1 \end{cases} \\ SHb,ssm &= \begin{cases} 0 & if \; \alpha_B = 0 \\ n-m & if \; \alpha_B = 1 \end{cases} \end{aligned} \tag{3}$$

and SH = SHa,ssm + SHb,ssm, defining the left-shift used to express the result on $2{\cdot}n$ bits:

$$SHssm = \begin{cases} 0 & if\ \alpha_A = 0\ ,\ \alpha_B = 0 \\ n-m & if\ \alpha_A = 0\ ,\ \alpha_B = 1\ or\ if\ \alpha_A = 1\ ,\ \alpha_B = 0 \\ 2{\cdot}(n-m) & if\ \alpha_A = 1\ ,\ \alpha_B = 1 \end{cases} \quad (4)$$

Figure 2a depicts the hardware implementation of the SSM multiplier. The multiplexers on A and B apply the segmentation choosing between the most significant and least significant portions of the inputs, whereas two OR gates compute the selection flags α_A and α_B. After the $m \times m$ multiplier, a further multiplexer realizes the left-shift described in (4).

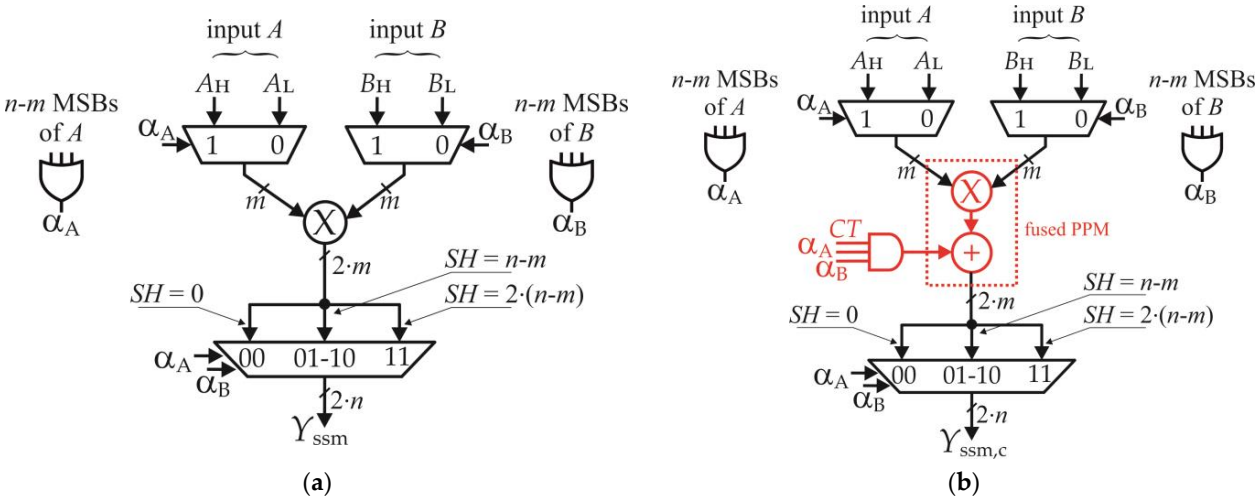


Figure 2. Approximate multiplier with (**a**) static segment method and (**b**) segmented multiplier with the correction technique of [30].

The accuracy of the SSM multiplier is improved in [30] by minimizing the approximation error in the case $\alpha_A = 1$, $\alpha_B = 1$ (i.e., when both inputs are truncated). Here, the authors estimate the committed error as

$$CT = 2^{2n-2m} \cdot \sum_{k=0}^{m-1} ct_k 2^k \quad (5)$$

with

$$Ct_k = (a_{k+n-m} b_{n-m-1}) OR (b_{k+n-m} a_{n-m-1}) \quad (6)$$

and add CT to the approximate product for compensation:

$$\gamma_{ssm,c} = (A_{ssm} \cdot B_{ssm} + CT) \cdot 2^{SH} \quad (7)$$

As detailed in [30], using two or three terms of the summation (5) sufficiently improves the accuracy.

Figure 2b shows the implementation of the corrected SSM multiplier (named cSSM in the following). The correction term CT is combined with the product $A_{ssm} \cdot B_{ssm}$ if $\alpha_A = 1$ and $\alpha_B = 1$ (see the AND gate highlighted in red). It is also worth noting that the correction technique has a minimum impact on the hardware performances since a fused PPM is employed for realizing the (7).

2.2. Enhanced SSM Multiplier

The ESSM multiplier described in [29] allows for selecting between three segments of the input, each one having m bits (see Figure 3a). In this implementation, the middle segment A_M is placed at the center of the signal (i.e., $(n-m)/2$ bits on the left with respect

to the LSB, see the figure). As the position of A_M is fixed, m is again the only design parameter which defines the accuracy and the size of the multiplier.

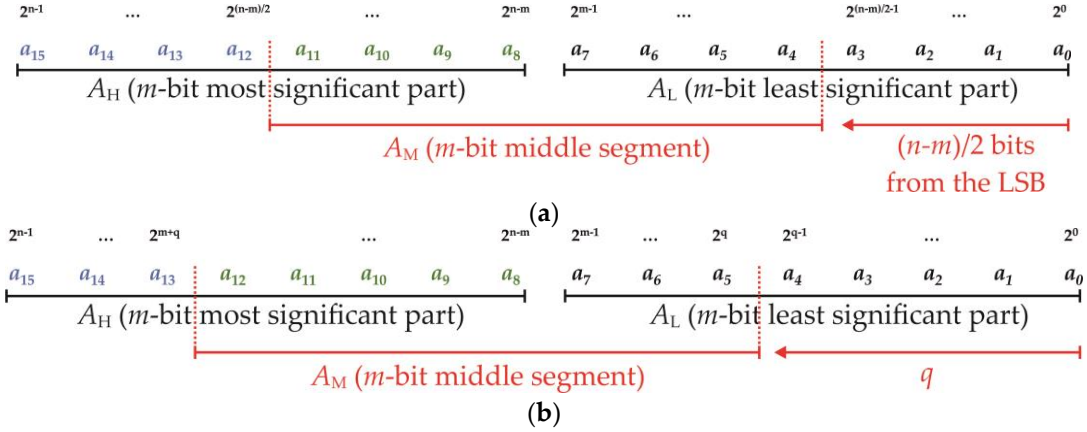


Figure 3. Segmentation of the input A in the case n = 16 and m = 8 with (**a**) the ESSM method of [29] and (**b**) the proposed generalized ESSM method in the case q = 5.

In this case, two control flags are required for the selection, named α_{AH} and α_{AM} in the following. Therefore, defining α_{AH} as the OR of the first $(n-m)/2$ MSBs of A (i.e., $a_{15}, a_{14}, \ldots, a_{12}$, highlighted in blue in Figure 3a), and α_{AM} as the OR of the remaining $(n-m)/2$ MSBs (i.e., $a_{11}, a_{10}, \ldots, a_8$, highlighted in green in Figure 3a), the segment A_{essm} is computed as

$$A_{essm} = \begin{cases} A_L & if\ (\alpha_{AH}, \alpha_{AM}) = (0,0) \\ A_M & if\ (\alpha_{AH}, \alpha_{AM}) = (0,1) \\ A_H & if\ (\alpha_{AH}, \alpha_{AM}) = (1,0)\ or\ (1,1) \end{cases} \quad (8)$$

A similar expression holds also for the segment B_{essm}, with the flags α_{BH}, α_{BM} that handle the segmentation.

Therefore, the approximate product is

$$\gamma_{essm} = \left(A_{essm} \cdot 2^{SHa,essm}\right) \cdot \left(B_{essm} \cdot 2^{SHb,essm}\right) = (A_{essm} \cdot B_{essm}) \cdot 2^{SHessm} \quad (9)$$

with SHa,essm, SHb,essm that are

$$SHa,essm = \begin{cases} 0 & if(\alpha_{AH}, \alpha_{AM}) = (0,0) \\ (n-m)/2 & if(\alpha_{AH}, \alpha_{AM}) = (0,1) \\ n-m & if(\alpha_{AH}, \alpha_{AM}) = (1,0)\ or\ (1,1) \end{cases}$$
$$SHb,essm = \begin{cases} 0 & if(\alpha_{BH}, \alpha_{BM}) = (0,0) \\ (n-m)/2 & if(\alpha_{BH}, \alpha_{BM}) = (0,1) \\ n-m & if(\alpha_{BH}, \alpha_{BM}) = (1,0)\ or\ (1,1) \end{cases} \quad (10)$$

and SHessm defined in Table 1.

Table 1. Left-shift for the ESSM multiplier.

α_{AH}, α_{AM}, α_{BH}, α_{BM}	*SHessm*
(0000)	0
(0001), (0100)	$(n-m)/2$
(0010), (0011), (0101), (1000), (1100)	$n-m$
(0110), (0111), (1001), (1101)	$(3/2)\cdot(n-m)$
(1010), (1011), (1110), (1111)	$2\cdot(n-m)$

As shown in the table, the left-shift SHessm ranges between five possible values, thus requiring a 5×1 multiplexer to extend the result on $2{\cdot}n$ bits.

3. Proposed Generalized ESSM Multiplier

3.1. Hardware Implementation

With the aim to improve the accuracy of the multiplier presented in the previous section, we generalize the ESSM method by placing A_{M} in any possible position between the LSB and the MSB. With reference to Figure 3a, let us suppose A to be in the range $[2^{12}, 2^{13})$ with high probability, which means that the bit a_{12} is high and the bits a_{15}, a_{14}, a_{13} are low with high probability. The segmentation scheme of Figure 3a mostly chooses the segment A_{H}, approximating the input with resolution 2^8, whereas A_{M}, able to offer a finer accuracy, is less used. In order to improve the performances, we can choose a segmentation scheme as in Figure 3b, allocating the middle segment in order to collect the bits a_{12}, a_{11}, $\ldots$, a_5. In this way, the selection mechanism mostly chooses A_{M} allowing a finer resolution (that is 2^5 instead of 2^8) with beneficial effects on the overall accuracy. As a consequence, choosing the position of A_{M} in dependance on the input statistical properties allows us to optimize the accuracy of the multiplier.

As shown in Figure 3b, the parameter q defines the position of A_{M} with respect the LSB of the input (in this example $q = 5$). Therefore, two parameters are used for the design: m, which defines the accuracy and the size of the multiplier, and q, which improves the accuracy of the segmentation. Please note also that q defines the resolution of A_{M}, which is 2^{q} (see Figure 3b).

By noting that A_{M} and A_{L} overlap if $q = 0$, and that A_{M} and A_{H} overlap if $q = n - m$, we choose q in the range $[1, n - m - 1]$ to select three distinct segments. In addition, if $q = (n - m)/2$ we get the ESSM multiplier presented in [29].

The selection flag α_{AH} is computed by OR-ing the first $n - (m + q)$ MSBs of A (i.e., a_{15}, a_{14}, a_{13} in Figure 3b, depicted in blue), whereas α_{AM} is computed by OR-ing the remaining q MSBs (i.e., $a_{12}, a_{11}, \ldots, a_8$ in Figure 3b, depicted in green). Then, the segmented inputs A_{essm}, B_{essm} are computed as in (8), with the following expressions for SHa,essm, SHb,ssm:

$$\begin{aligned} SHa, essm &= \begin{cases} 0 & if\ (\alpha_{AH}, \alpha_{AM}) = (0,0) \\ q & if\ (\alpha_{AH}, \alpha_{AM}) = (0,1) \\ n-m & if\ (\alpha_{AH}, \alpha_{AM}) = (1,0)\ or\ (1,1) \end{cases} \\ SHb, essm &= \begin{cases} 0 & if\ (\alpha_{BH}, \alpha_{BM}) = (0,0) \\ q & if\ (\alpha_{BH}, \alpha_{BM}) = (0,1) \\ n-m & if\ (\alpha_{BH}, \alpha_{BM}) = (1,0)\ or\ (1,1) \end{cases} \end{aligned} \tag{11}$$

Likewise, the approximate product is computed as in (9) with the final left-shift SHessm defined in Table 2. Now, SHessm ranges between six possible values, thus calling for a 6×1 multiplexing scheme.

Table 2. Left-shift for the generalized ESSM multiplier.

α_{AH}, α_{AM}, α_{BH}, α_{BM}	*SHessm*
(0000)	0
(0001), (0100)	q
(0101)	$2{\cdot}q$
(0010), (0011), (1000), (1100)	$n - m$
(0110), (0111), (1001), (1101)	$n - m + q$
(1010), (1011), (1110), (1111)	$2{\cdot}(n - m)$

Figure 4 depicts the hardware implementation of the generalized ESSM multiplier (named gESSM in the following). The 3×1 multiplexers allow for selecting between the most significant, the middle, and the least significant part of the inputs, whereas a small $m \times m$ multiplier computes the approximate product. The left-shift is realized by cascading

two 3 × 1 multiplexers, where the first multiplexer applies the shift SHa,essm due to the flags α_{AH}, α_{AM}, and the second one applies the shift SHb,essm due to α_{BH}, α_{BM}. It is worth noting that this approach prevents the usage of large multiplexers with beneficial effects on the hardware performances of the multiplier.

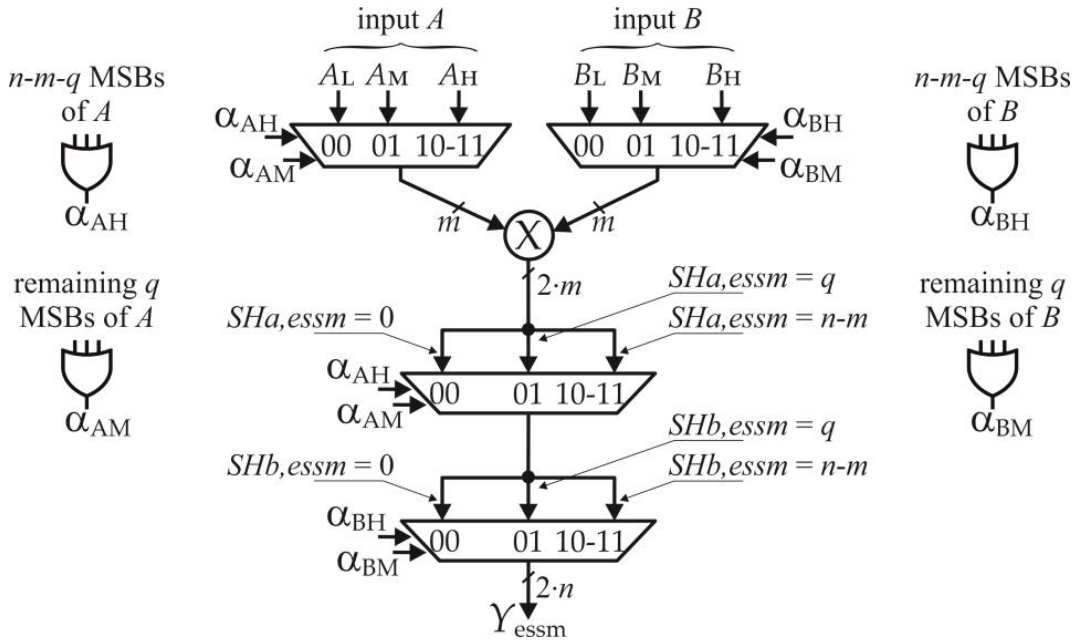


Figure 4. Block diagram of the proposed generalized ESSM multiplier.

3.2. Minimization of the Mean Square Approximation Error

In this paragraph, we find m and q in order to minimize the mean square approximation error at the output of the multiplier under the hypothesis of both uniform and non-uniform distributed input signals. In the following, we consider inputs with half-normal distribution in the non-uniform case, whose probability density function is as follows:

$$f(A) = \frac{\sqrt{2}}{\sigma\sqrt{\pi}} \cdot e^{-\frac{A^2}{2\sigma^2}} \quad for\ A \geq 0 \tag{12}$$

where σ, being the standard deviation of the underlying normal variable, is also related to the standard deviation of A.

Before proceeding, let us assume A and B to be independent, and let us re-write equation (8) as follows with the help of Figure 3b:

$$A_{essm} = \begin{cases} A_L & if\ A < 2^m \\ A_M & if\ 2^m \leq A < 2^{m+q} \\ A_H & if\ 2^{m+q} \leq A < 2^n - 1 \end{cases} \tag{13}$$

where the conditions $A < 2^m$, $2^m \leq A < 2^{m+q}$, and $2^{m+q} \leq A < 2^n - 1$ recall the conditions (α_{AH}, α_{AM}) = 00, (α_{AH}, α_{AM}) = 01 and (α_{AH}, α_{AM}) = 10 or 11, respectively. Defining $A'_{essm} = A_{essm} \cdot 2^{SHa,essm}$ and $B'_{essm} = B_{essm} \cdot 2^{SHb,essm}$, we can write the exact inputs as follows:

$$\begin{aligned} A &= A'_{essm} + e_A \\ B &= B'_{essm} + e_B \end{aligned} \tag{14}$$

with e_A, e_B that are the truncation errors due to the segmentation:

$$e_A = \begin{cases} 0 if\ A < 2^m \\ \sum_{k=0}^{q-1} a_k 2^k if\ 2^m \le A < 2^{m+q} \\ \sum_{k=0}^{n-m-1} a_k 2^k if\ 2^{m+q} \ll A \ll 2^n - 1 \end{cases} \quad e_B \begin{cases} 0 if\ B < 2^m \\ \sum_{k=0}^{q-1} b_k 2^k if\ 2^m \le B < 2^{m+q} \\ \sum_{k=0}^{n-m-1} b_k 2^k if\ 2^{m+q} \le B \le 2^n - 1 \end{cases} \tag{15}$$

In (15) we assume bits a_k, b_k to be independent from A and B, respectively, and to be uniform random variables with probability $\frac{1}{2}$ of being '1'.

Using (14), the exact product is:

$$\begin{aligned} \gamma = A \cdot B &= \left(A'_{essm} + e_A\right) \cdot \left(B'_{essm} + e_B\right) \\ &= A'_{essm} \cdot B'_{essm} + A'_{essm} \cdot e_B + B'_{essm} \cdot e_A + e_A \cdot e_B \end{aligned} \tag{16}$$

Since the gESSM computes only the term $A'_{essm} \cdot B'_{essm}$, the segmentation error is:

$$e_{essm} = A'_{essm} \cdot e_B + B'_{essm} \cdot e_A + e_A \cdot e_B \tag{17}$$

Re-writing (14) as $A'_{essm} = A - e_A$, $B'_{essm} = B - e_B$ and substituting in (17), we find:

$$e_{essm} = A \cdot e_A + B \cdot e_B - e_A \cdot e_B \tag{18}$$

Neglecting the small term $e_A \cdot e_B$ for the sake of simplicity, we compute the mean square approximation error by squaring (18) and by using the expectation operator:

$$E\left[e^2_{essm}\right] = E\left[A^2\right] \cdot E\left[e^2_B\right] + E\left[B^2\right] \cdot E\left[e^2_A\right] + 2 \cdot E[A \cdot e_A] \cdot E[A \cdot e_B] \tag{19}$$

Since A and B have the same distribution, we have $E[A^2] = E[B^2]$, as well as $E[e^2{}_A] = E[e^2{}_B]$ and $E[A \cdot e^2{}_A] = E[B \cdot e^2{}_B]$ for the previous hypothesis. Therefore, Equation (19) becomes

$$E\left[e^2_{essm}\right] = 2 \cdot E\left[A^2\right] \cdot E\left[e^2_A\right] + 2 \cdot E[A \cdot e_A]^2 \tag{20}$$

As the computation of $E[A \cdot e_A]^2$ is not straightforward, we can exploit the Cauchy–Schwarz inequality $E[A \cdot e_A]^2 \le E[A_2] \cdot E[e^2{}_A]$ to find the upper limit of $E[e^2{}_{essm}]$:

$$E\left[e^2_{essm}\right] \le 4 \cdot E\left[A^2\right] \cdot E\left[e^2_A\right] \tag{21}$$

Here, $E[A^2]$ depends on the statistic of the input signal, whereas $E[e^2{}_A]$, which is the mean square value of the approximation error committed on A, depends on m and q. Then, as suggested by the above inequalities, minimizing the upper limit (i.e., minimizing $E[e^2{}_A]$) minimizes the overall mean square approximation error of the multiplier.

Starting from (15), we can write $E[e^2{}_A]$ as follows:

$$E\left[e^2_A\right] = E\left[\left(\sum_{k=0}^{q-1} a_k 2^k\right)^2\right] \cdot P(A_M) + E\left[\left(\sum_{k=0}^{n-m-1} a_k 2^k\right)^2\right] \cdot P(A_H) \tag{22}$$

with $P(A_\mathrm{M})$ and $P(A_\mathrm{H})$ that are the probability of having A in the ranges $[2^\mathrm{m}, 2^{\mathrm{m+q}})$ and $[2^{\mathrm{m+q}}, 2^\mathrm{n} - 1]$, respectively. Table 3 collets the expressions of $P(A_\mathrm{M})$ and $P(A_\mathrm{H})$ for the uniform and the half-normal cases, where $erf(\cdot)$ represents the so-called error function (details on the computation are reported in Appendix A for the half-normal case).

Table 3. Probability of selecting A_M and A_H as a function of the input distribution.

Input Stochastic Distribution	$P(A_M)$	$P(A_H)$
Uniform	$\frac{1}{2^n-1}\cdot\left(2^{m+q}-2^m\right)$	$\frac{1}{2^n-1}\cdot\left(2^n-1-2^{m+q}\right)$
Half-normal	$erf\left(\frac{2^{m+q}}{\sigma\sqrt{2}}\right)-erf\left(\frac{2^m}{\sigma\sqrt{2}}\right)$	$erf\left(\frac{2^n-1}{\sigma\sqrt{2}}\right)-erf\left(\frac{2^{m+q}}{\sigma\sqrt{2}}\right)$

We underline that the presence of $P(A_M)$ and $P(A_H)$ in (22) highlights the relation between the approximation error and the stochastic distribution of the inputs.

Solving the expectations in (22), we find the following expression for $E[e^2{}_A]$ (refer to Appendix B for details):

$$E\left[e_A^2\right] = \left[\frac{1}{6}\left(4^q-1\right)+\frac{1}{4}2^1\left(2^{q-1}-1\right)-\frac{1}{6}\left(4^{q-1}-1\right)\right]\cdot P(A_M) + \left[\frac{1}{6}\left(4^{n-m}-1\right)+\frac{1}{4}2^{n-m}\left(2^{n-m-1}-1\right)-\frac{1}{6}\left(4^{n-m-1}-1\right)\right]\cdot P(A_H) \quad (23)$$

with $P(A_M)$ and $P(A_H)$ that also depend on m and q (see Table 3).

The behavior of $E[e^2{}_A]$ with respect to m and q is shown in Figure 5, compared to the simulation results. In this study, the input A is an n = 16 bits integer signal with uniform distribution in Figure 5a, and half-normal distribution with σ = 1024, 2048, and 16,384 in Figure 5b–d. We achieve the simulation results by segmenting 10^6 input samples of A and by computing the mean square value of the approximation error.

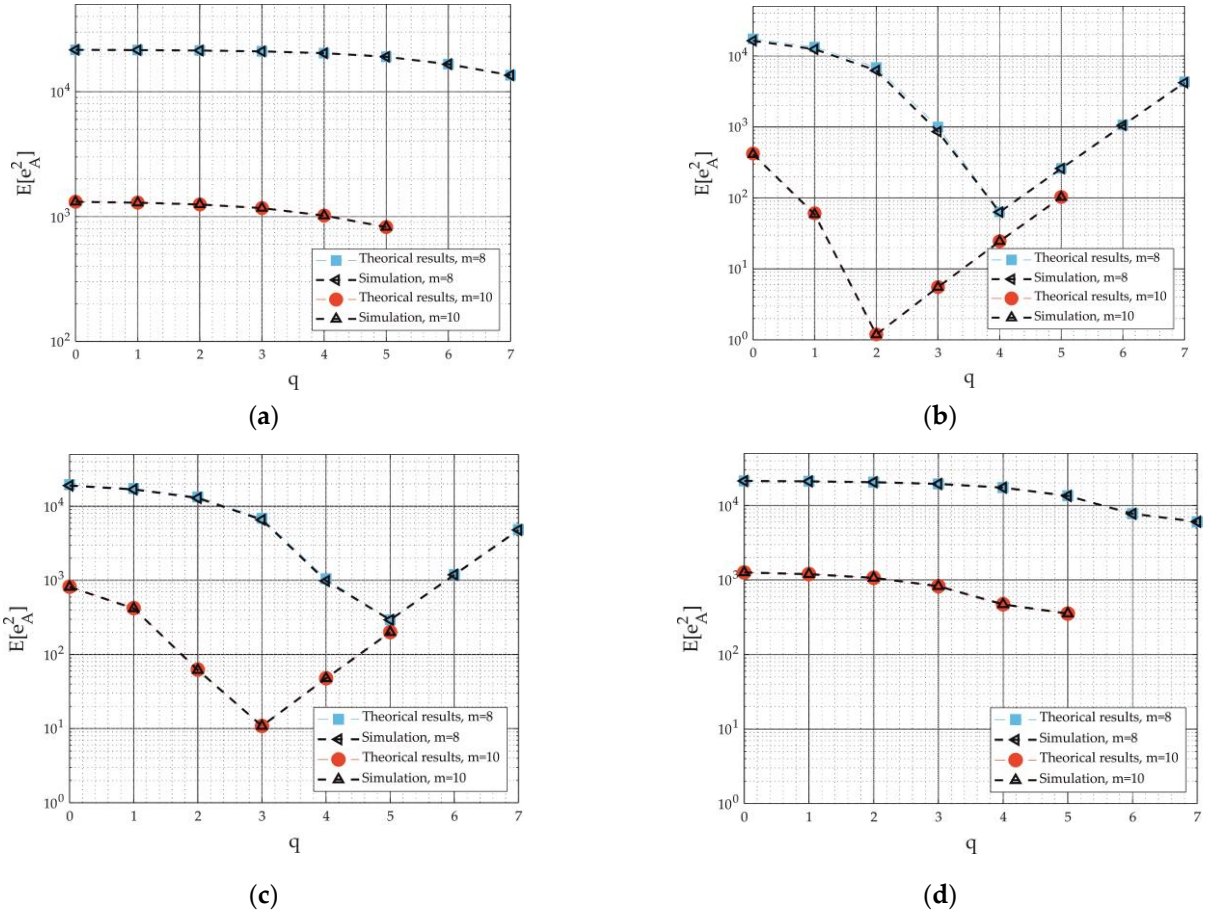


Figure 5. Mean square error on the input signal A as a function of m and q with (**a**) uniform distribution and half-normal distribution in the cases of (**b**) σ = 1024, (**c**) σ = 2048, and (**d**) σ = 16,384. In this example, A is an integer signal expressed on n = 16 bits.

As shown, the theoretical results perfectly match with the simulations. For fixed m, increasing q decreases $E[e^2{}_A]$ in the uniform case. Therefore, the optimal point q_{opt}, able to minimize $E[e^2{}_A]$, is the maximum value of q (that is $q_{opt} = n - m - 1$). On the other hand, $E[e^2{}_A]$ shows minima in Figure 5b,c, with optimal points in $q_{opt} = 4$, $m = 8$ and $q_{opt} = 2$, $m = 10$ for σ=1024, and $q_{opt} = 5$, $m = 8$ and $q_{opt} = 3$, $m = 10$ for σ = 2048. When σ becomes large, $E[e^2{}_A]$ again decreases with q, making $q = n - m - 1$ the best choice.

In conclusion, a proper selection of q is of paramount importance for optimizing the accuracy of the multiplier, leading to placing the middle segment in any position between the LSB and the MSB of A (in contrast with [29] which always fixes the middle segment at the center of the input). In addition, the statistical properties of the input signals strongly affect the optimal value of q, as demonstrated by the results of Figure 5.

4. Results

4.1. Assessment of Accuracy

We study the accuracy of the gESSM by exploiting the error metrics commonly used in the literature. To this end, let us define the approximation error $E = Y - Y_{apprx}$ and the Error Distance $ED = |E|$, with Y and Y_{apprx} that are the exact and the approximate product. Naming $avg(\cdot)$ and Y_{max} the average operator and the maximum value of Y, respectively, with $Y_{max} = (2^n - 1)^2$, we define the Normalized Mean Error Distance as $NMED = avg(ED)/Y_{max}$, the Mean Relative Error Distance as $avg(ED/Y)$, and the Number of Effective Bits as $NoEB = 2{\cdot}n - \log_2(1 - E_{rms})$, with E_{rms} being the root mean square value of E.

Figure 6 depicts the *NoEB* as a function of q for $m = 8, 10$. Please note that the cases $q = 4$, $m = 8$ and $q = 3$, $m = 10$ give the ESSM described in [29], and that for $q = 0$ we obtain the performances of the SSM multiplier. In this analysis, the error performances are computed by multiplying 10^6 input samples, expressed on $n = 16$ bits, considering both uniform and half-normal distribution with σ = 2048 for the sake of demonstration. As shown in Figure 6a, the *NoEB* slowly improves with q, achieving the best result for $q_{opt} = n - m - 1$. On the other hand, the *NoEB* reaches the peak value with $q_{opt} = 5$, $m = 8$ and $q_{opt} = 3$, $m = 10$, respectively, when the inputs are half-normal. These results are in agreement with the analysis of the previous section, since the *NoEB* reflects the behavior of the overall mean square approximation error. In addition, this study confirms that the input statistical properties affect the quality of results, and that positioning A_M in the middle (as in [29]) generally does not achieve the best accuracy.

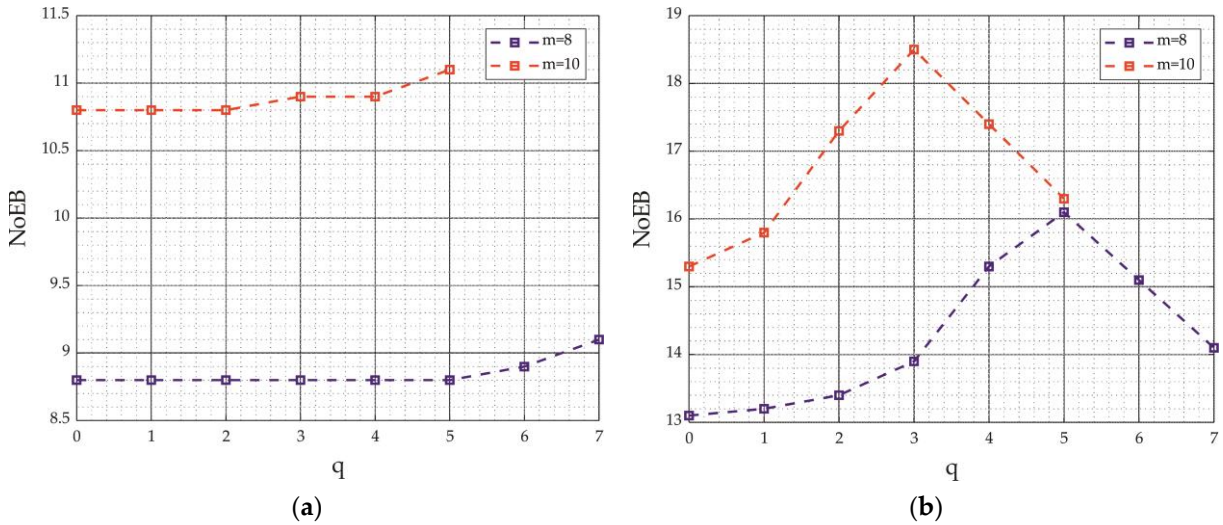


Figure 6. *NoEB* with respect to q for $m = 8$ and $m = 10$ for (**a**) uniform distributed inputs and for (**b**) half-normal distributed inputs (with σ = 2048). The number of bits of the inputs is $n = 16$.

For the sake of comparison, we also analyze the performances of SSM [29], of cSSM [30] (with three corrective terms), and of segmented multipliers described in [27,28,31]. The multipliers [13,16,19] are also investigated, which exploit approximate compression. The works [27,28] employ a dynamic segmentation, whereas [31] employs a hybrid technique by cascading a static stage and a dynamic stage. In [27] (named DRUM in the following), the parameter k defines the bit-width of the selected segment, whereas [28] (named TOSAM in the following) exploits a multiply-and-add operation for realizing the product. Here, h bits of the multiplicands are truncated, and $t = h + 4$ bits of the addends are discarded for hardware simplification. In [31] (named HSM) the static stage selects p-bit segments, whereas the dynamic one chooses $(p/2)$-bit segments. In [16] (referred as Qiqieh in the following), the parameter L defines the number of rows compressed by an OR gate, whereas [19] (referred to as AHMA in the following) compresses the PPM with approximate 4–2 compressors. We highlight that the HDL code of [27,30] is available on [32,33], respectively.

Table 4 collects the error metrics of the investigated multipliers when the inputs are uniform and half-normal (with $\sigma = 2048$), respectively. For the gESSM, we consider the points $q = 5$ and $q = 7$ for $m = 8$, and $q = 3$ and $q = 5$ for $m = 10$, which achieved best performances in the previous analysis. Please note that only the case $q = 3$, $m = 10$ places A_M at the center of the inputs as in [29].

Table 4. Error metrics of the investigated multipliers for $n = 16$ bits.

Multiplier		**Uniform Distribution**			**Half-Normal Distribution ($\sigma = 2048$)**		
		NMED	***MRED***	***NoEB***	***NMED***	***MRED***	***NoEB***
SSM [29]	$m = 8$	$1.93 \times \cdot 10^{-3}$	2.08×10^{-2}	8.8	8.29×10^{-5}	1.87×10^{-1}	13.1
	$m = 10$	4.73×10^{-4}	3.99×10^{-3}	10.8	1.46×10^{-5}	1.96×10^{-2}	15.3
cSSM [30]	$m = 8$	6.70×10^{-4}	9.49×10^{-3}	10.2	8.28×10^{-5}	1.87×10^{-1}	13.1
	$m = 10$	1.63×10^{-4}	1.73×10^{-3}	12.2	1.46×10^{-5}	1.96×10^{-2}	15.3
TOSAM [28]	$h = 3$	2.69×10^{-3}	1.05×10^{-2}	7.8	6.24×10^{-6}	9.98×10^{-3}	16.4
	$h = 4$	1.34×10^{-3}	5.27×10^{-3}	8.8	3.13×10^{-6}	5.02×10^{-3}	17.3
DRUM [27]	$k = 4$	1.41×10^{-2}	5.89×10^{-2}	5.5	3.85×10^{-5}	6.20×10^{-2}	13.7
	$k = 6$	3.51×10^{-3}	1.46×10^{-2}	7.5	9.53×10^{-6}	1.52×10^{-2}	15.7
	$k = 8$	8.82×10^{-4}	3.66×10^{-3}	9.5	2.39×10^{-6}	3.68×10^{-3}	17.7
HSM [31]	$p = 8$	1.47×10^{-2}	1.03×10^{-1}	5.5	3.53×10^{-4}	7.05×10^{-1}	11.0
	$p = 10$	7.15×10^{-3}	3.72×10^{-2}	6.5	1.02×10^{-4}	1.70×10^{-1}	12.3
	$p = 12$	3.51×10^{-3}	1.56×10^{-2}	7.5	9.76×10^{-6}	3.92×10^{-2}	15.7
Qiqieh [16]	$L = 2$	2.43×10^{-4}	2.88×10^{-3}	11.0	1.90×10^{-5}	3.27×10^{-2}	14.0
	$L = 4$	1.12×10^{-2}	5.90×10^{-2}	5.7	5.78×10^{-5}	8.35×10^{-2}	12.8
Kulkarni [13]		1.39×10^{-2}	3.32×10^{-2}	4.7	1.28×10^{-5}	1.74×10^{-2}	14.1
AHMA [19]		2.14×10^{-2}	1.18×10^{-1}	4.9	1.65×10^{-4}	2.42×10^{-1}	11.3
gESSM	$m = 8, q = 5$	1.73×10^{-3}	9.68×10^{-3}	8.8	1.06×10^{-5}	2.55×10^{-2}	16.1
	$m = 8, q = 7$	1.45×10^{-3}	1.19×10^{-2}	9.1	4.24×10^{-5}	9.93×10^{-2}	14.1
	$m = 10, q = 3$	4.26×10^{-4}	2.22×10^{-3}	10.9	1.64×10^{-6}	2.21×10^{-3}	18.5
	$m = 10, q = 5$	3.55×10^{-4}	2.30×10^{-3}	11.1	7.24×10^{-6}	9.73×10^{-3}	16.3

In the uniform case, the performances of the gESSM are very close to the SSM multiplier, with *NoEB* of about 9 and 11 bits with $m = 8$ and $m = 10$, and *NMED*, *MRED* in the ranges $[3 \times 10^{-4}, 2 \times 10^{-3}]$, $[2 \times 10^{-3}, 1.2 \times 10^{-2}]$, respectively. A modest improvement is registered only in the cases $q = 7$, $m = 8$ and $q = 5$, $m = 10$, as expected from the previous considerations, with a *NoEB* increase of 0.3 bits. Among the segmented multipliers, cSSM offers best accuracy in the uniform case, with *NoEB* improvement of 1.4 bits with respect to SSM, and *NMED*, *MRED* in the order of 10^{-4} and 2×10^{-3} (see the case $m = 10$). The other implementations exhibit lower performances in general, with *NoEB* limited between

5.5 and 9.5 bits. Only Qiqieh $L = 2$, using approximate compression technique, is able to approach a *NoEB* of 11 bits and *NMED*, *MRED* comparable to SSM, cSSM, and gESSM.

On the other hand, the accuracy of the gESSM strongly improves with respect to the SSM when the inputs are half-normal, exhibiting a *NoEB* increase up to 3 bits with $q = 5$, $m = 8$, and up to 3.2 bits with $q = 3$, $m = 10$. The *NMED* also improves, achieving values in the order of 10^{-5} with $q = 5$, $m = 8$, and 10^{-6} with $q = 3$, $m = 10$. Conversely, the cSSM multiplier does not show improvements, with performances very close to the SSM. Among the other implementations, DRUM is the only one able to offer an accuracy close to the gESSM multiplier, with *NoEB* up to 17.7 bits in the case $k = 8$.

4.2. Hardware Implementation Results

We synthesize the investigated multipliers in TSMC 28 nm CMOS technology using 0.9V standard voltage library, in Cadence Genus, exploiting a physical flow in order to improve the accuracy of the estimation of power consumption. For all the circuits, a clock period of 500ps is considered, whereas the power consumption is computed by simulating the post-synthesis netlist with 10^5 input samples, with both uniform and half-normal distribution ($\sigma = 2048$) at a toggle rate of 1GHz. In the simulation, Standard Delay Format and Toggle Count Format files are used for the annotation of the path delays and of the switching activity. At the same time, we also assess the minimum delay by synthesizing each multiplier at the maximum frequency able to allow a positive slack.

Results are collected in Table 5. As shown, the gESSM multipliers allow a reduction of area up to 71.4% with $q = 7$, $m = 8$, and in the range 47%/50% with $m = 10$. The SSM and cSSM multipliers offer superior reductions (up to 76% and 75%, respectively), whereas best results are achieved with DRUM $k = 6$ and HSM $p = 4$ (reduction of 84% and 86%, respectively). We also express the complexity of the circuits in terms of equivalent NAND count, considering as reference a two-input NAND gate with drive strength 2x and area of $0.63\mu m^2$. Also in this case, the gESSM exhibits remarkable improvements with respect to the exact multiplier and an acceptable worsening with respect to SSM and cSSM.

The gESSM reduces the minimum delay up to 12.8% with respect to the exact implementation. On the other hand, the SSM and cSSM produce faster results due to the simpler segmentation algorithm. The minimum delay of DRUM and HSM increases with k and p, respectively, up to +8.6%, whereas best performances are achieved with Qiqieh, Kulkarni, and AHMA (with reductions up to 38%).

In the case of uniform distributed inputs, the gESSM multipliers show remarkable power savings, ranging between 53.7% and 78.1%. On the other hand, the implementations SSM and cSSM are able to obtain more than 83% of power reduction with $m = 8$. DRUM $k = 4$ and HSM $p = 8$ achieve best performances, with improvements in the order of -90%.

When the input is half-normal, the power saving of gESSM is of 71% and 27% in the optimal points $q = 5$, $m = 8$, and $q = 3$, $m = 10$, and is larger than 83% in the case $q = 7$, $m = 8$. SSM and cSSM continue to exhibit high power reductions (around 88%/89% with $m = 8$), whereas power saving of DRUM $k = 4$ and HSM $p = 8$ reaches up 76.2% and 84%.

We underline that, despite the reduced power saving with half-normal distribution in the optimal points, the gESSM multipliers offer the best accuracy, showing superior error metrics if compared to the other implementations. Therefore, the loss of electrical performances is more than compensated by the reduced approximation error.

Table 5. Hardware implementation results of the investigated multipliers for $n = 16$ bits.

Multiplier		Minimum Delay [ps]	Area [μm^2]	Equivalent NAND Count	Power @1GHz (Uniform Input)	Power @1GHz (Half-Normal Input)
Exact		336	791.3	1256	1300.3	721.8
SSM [29]	$m = 8$	272 (−19.0%)	190.1 (−76.0%)	302	201.4 (−84.5%)	77.6 (−89.2%)
	$m = 10$	313 (−6.8%)	308.6 (−61.0%)	490	346.8 (−73.3%)	281.8 (−61.0%)

Table 5. *Cont.*

Multiplier		Minimum Delay [ps]	Area [μm^2]	Equivalent NAND Count	Power @1GHz (Uniform Input)	Power @1GHz (Half-Normal Input)
cSSM [30]	$m = 8$	272 (−19.0%)	197.4 (−75.0%)	313	214.7 (−83.5%)	85.6 (−88.1%)
	$m = 10$	313 (−6.8%)	352.3 (−55.5)	559	395.0 (−69.6%)	358.6 (−50.3%)
TOSAM [28]	$h = 3$	311 (−7.4%)	341.2 (−56.9%)	542	367.1 (−71.8%)	394.5 (−45.3%)
	$h = 4$	335 (−0.3%)	494.9 (−37.4%)	786	582.4 (−55.2%)	613.5 (−15.0%)
DRUM [27]	$k = 4$	257 (−23.5%)	126.5 (−84.0%)	201	155.9 (−88.0%)	171.8 (−76.2%)
	$k = 6$	357 (+6.3%)	241.9 (−69.4%)	384	389.1 (−70.1%)	377.4 (−47.7%)
	$k = 8$	365 (+8.6%)	414.3 (−47.6%)	658	691.1 (−46.9%)	656.2 (−9.1%)
HSM [31]	$p = 8$	251 (−25.3%)	112.9 (−85.7%)	179	137.3 (−89.4%)	115.7 (−84.0%)
	$p = 10$	354 (+5.4%)	204.8 (−74.1%)	325	306.3 (−76.4%)	339.6 (−53.0%)
	$p = 12$	364 (+8.3%)	347.1 (−56.1%)	551	538.2 (−58.6%)	582.8 (−19.3%)
Qiqieh [16]	$L = 2$	262 (−22.0%)	440.8 (−44.3%)	700	578.6 (−55.5%)	330.9 (−54.2%)
	$L = 4$	218 (−35.1%)	271.9 (−65.6%)	432	385.8 (−70.3%)	241.6 (−66.5%)
Kulkarni [13]		289 (−14.0%)	508.9 (−35.7%)	808	620.4 (−52.3%)	364.7 (−49.5%)
AHMA [19]		208 (−38.1%)	282.4 (−64.3%)	448	327.5 (−74.8%)	252.1 (−65.1%)
gESSM	$m = 8, q = 5$	312 (−7.1%)	235.6 (−70.2%)	347	289.9 (−77.7%)	210.63 (−70.8%)
	$m = 8, q = 7$	293 (−12.8%)	226.0 (−71.4%)	359	284.4 (−78.1%)	122.20 (−83.1%)
	$m = 10, q = 3$	327 (−2.7%)	393.4 (−50.3%)	624	510.9 (−60.7%)	524.8 (−27.3%)
	$m = 10, q = 5$	329 (−2.1%)	420.2 (−46.9%)	667	602.0 (−53.7%)	494.6 (−31.5%)

4.3. Image Processing Application

We study the performances of the investigated multipliers in image filtering applications. Named $I(x,y)$ the pixel of the input image with coordinates x, y, the filtering operation realizes the relation

$$I_f(x,y) = \sum_{i=-d}^{d} \sum_{j=-d}^{d} I(x+i, y+j) \cdot h(i+d+1, j+d+1) \tag{24}$$

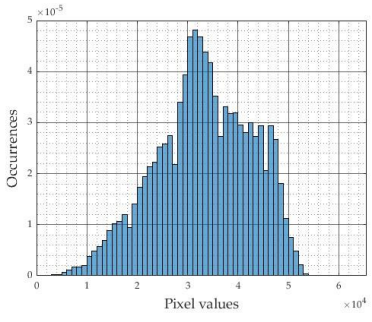
with $I_f(x,y)$ which is the pixel of the output image, and with h which is the kernel matrix. In our case, we consider a 5×5 gaussian kernel, h_{GAUSSIAN}, used for smoothing operations, and a 5×11 motion kernel, h_{MOTION}, able to approximate the linear motion of a camera. Figure 7a,b report the coefficients of h_{GAUSSIAN} and h_{MOTION}, expressed as integer numbers on $n = 16$ bits.

$$h_{GAUSSIAN} = \begin{pmatrix} 21329 & 31034 & 35166 & 31034 & 21329 \\ 31034 & 45154 & 51166 & 45154 & 31034 \\ 35166 & 51166 & 57978 & 51166 & 35166 \\ 31034 & 45154 & 51166 & 45154 & 31034 \\ 21329 & 31034 & 35166 & 31034 & 21329 \end{pmatrix}$$

(**a**)

$$h_{MOTION} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2949 & 41157 & 79365 & 6615 \\ 0 & 0 & 0 & 0 & 0 & 8471 & 46679 & 84887 & 57721 & 19513 & 0 \\ 0 & 0 & 0 & 13992 & 52200 & 90408 & 52200 & 13992 & 13992 & 0 & 0 \\ 0 & 19513 & 57721 & 84887 & 46679 & 8471 & 0 & 0 & 0 & 0 & 0 \\ 6615 & 79365 & 41157 & 2949 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(**b**)



(**c**)

Figure 7. Kernel matrix for (**a**) the gaussian and (**b**) the motion filter. (**c**) Histogram of occurrences for the Mandrill image.

For our analysis, we process three test images, Lena, Cameraman, and Mandrill, whose pixel values are represented on $n = 16$ bits. For the sake of demonstration, Figure 7c depicts the histogram of occurrences for Mandrill, showing that the probability of assuming values in $[0, 2^n - 1]$ is almost spread across the whole range. We assess the performances by exploiting the Mean Structural Similarity Index (SSIM), able to measure the similarity between images, and the Peak Signal-to-Noise ratio (PSNR), expressed in dB, taking as reference the exact filtered image.

Table 6 collects the results, showing the average SSIM and PSNR obtained with the smoothing and the motion application. In addition, the overall average SSIM and PSNR are presented for facilitating the comparisons. All the multipliers allow for achieving SSIM very close to 1, with the static segmented implementations that exhibit best results. The PSNR of cSSM strongly increases if compared with SSM (up to about +14 dB on average in the case $m = 10$), whereas the improvement is more modest with the gESSM (up to +4.1dB with $m = 8$ and +6dB with $m = 10$ on average). Again, the performances of the gESSM depend on the statistical properties of the input image and on the choice of q.

Table 6. Accuracy performances of the investigated multipliers in image processing applications.

Multiplier		**Gaussian Filter**		**Motion Filter**		**Average**	
		SSIM	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**
SSM [29]	$m = 8$	1.000	42.6	1.000	42.7	1.000	42.6
	$m = 10$	1.000	53.4	1.000	55.2	1.000	54.3
cSSM [30]	$m = 8$	1.000	58.4	1.000	54.7	1.000	56.5
	$m = 10$	1.000	68.5	1.000	67.6	1.000	68.0
TOSAM [28]	$h = 3$	1.000	48.8	0.999	54.8	0.999	51.8
	$h = 4$	1.000	63.9	1.000	63.4	1.000	63.7
DRUM [27]	$k = 4$	0.984	35.7	0.980	37.9	0.982	36.8
	$k = 6$	0.999	51.7	0.999	48.9	0.999	50.3
	$k = 8$	1.000	64.8	1.000	62.8	1.000	63.8
HSM [31]	$p = 8$	0.982	35.7	0.978	36.0	0.980	35.8
	$p = 10$	0.996	45.2	0.994	45.7	0.995	45.5
	$p = 12$	0.999	51.7	0.999	48.9	0.999	50.3
Qiqieh [16]	$L = 2$	1.000	63.9	1.000	65.0	1.000	64.5
	$L = 4$	0.982	32.0	0.981	31.2	0.981	31.6
Kulkarni [13]		0.993	39.2	0.997	42.6	0.995	40.9
AHMA [19]		0.950	25.3	0.948	25.4	0.949	25.4
gESSM	$m = 8, q = 5$	1.000	44.4	1.000	45.5	1.000	45.0
	$m = 8, q = 7$	1.000	47.1	1.000	46.3	1.000	46.7
	$m = 10, q = 3$	1.000	53.7	1.000	57.4	1.000	55.5
	$m = 10, q = 5$	1.000	60.0	1.000	60.4	1.000	60.2

The dynamic segmented multipliers exhibit large PSNR with TOSAM and DRUM (more than 60 dB), whereas performances are limited with HSM. Among multipliers with approximate compressors, only Qiqieh $L = 2$ is able to overcome 60dB of PSNR, whereas Kulkarni and AHMA show lower performances. Figure 8 offers the results obtained with the segmented multipliers for the Lena image. As shown, the results of gESSM are very close to the exact case (as demonstrated by the high values of SSIM and PSNR), whereas some degradations are registered with DRUM $k = 4$ and HSM $p = 8$.

Figure 8. Lena image filtered by means of segmented multipliers.

4.4. Audio Application

As a further example, we investigate the use of the proposed gESSM and the other multipliers for implementing an audio filter. Filtering is a well diffused operation in audio processing, able to realize frequency equalization and noise reduction. In this example, we elaborate the signal by considering a linear phase, low-pass, generalized Equiripple, 187-th order, finite impulse response (FIR) filter, with pass-band up to 0.1667 π rad/sample and stop-band from 0.1958 π rad/sample with attenuation of 85dB. The module of the impulse response is shown in Figure 9a with the taps represented as integer numbers expressed on $n = 16$ bits.

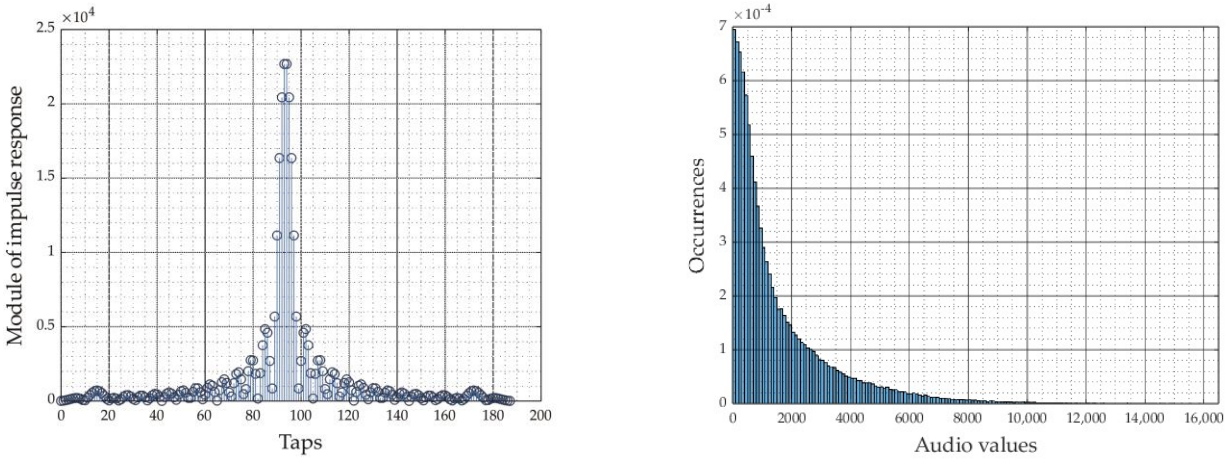


Figure 9. (**a**) Module of the impulse response of the low-pass FIR filter and (**b**) histogram of occurrences of the audio signal.

The audio signal used for this trial is p232_016.wav, from the library [34]. We also superimpose an external gaussian noise with variance of −30dB and quantize the resulting

signal on $n = 16$ bits. The histogram of occurrences of the input signal, depicted in Figure 9b, highlights a close to half-normal distribution.

For the sake of comparison, we show the mean square error (MSE) between the approximate and the exact output for each multiplier. Therefore, the lower the MSE, the better the multiplier accuracy.

Figure 10 shows the performances, with multiplications revisited as sign-magnitude operations. The results for the gESSM multipliers are highlighted in violet ($m = 8$) and in red ($m = 10$). As shown, the accuracy of the gESSM again varies in dependence on q, with best performance achieved with $q = 3$, $m = 10$. In this application, gESSM overcomes cSSM both with $m = 8$ and $m = 10$, which offer a worse MSE. In general, the gESSM performs better than the other implementations, with the exception of DRUM $k = 8$, featuring the best accuracy in this case.

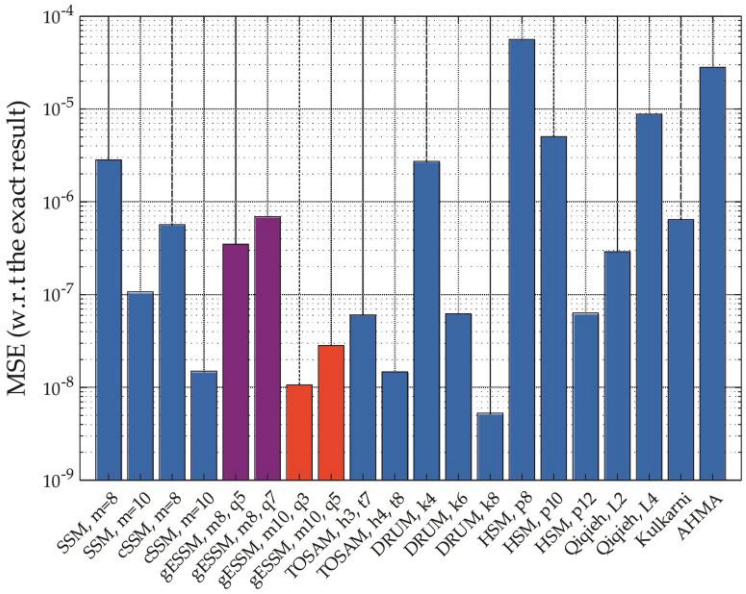


Figure 10. MSE between the approximate and the exact results. The MSE for gESSM $m = 8$ and $m = 10$ are highlighted in violet and red, respectively.

5. Discussion

As shown in the previous sections, the position of the middle segment A_M affects the accuracy of the multiplier, achieving different results dependent on the statistical properties of the inputs. Indeed, the accuracy mainly depends (i) on the probability of A of assuming values in the ranges $[2^m, 2^{m+q})$ and $[2^{m+q}, 2^n–1]$, and (ii) on the resolution of A_M.

Figure 11 shows the behavior of $P(A_M)$ and $P(A_H)$ with respect to q for $m = 8$, in the uniform case and in the half-normal distribution with $\sigma = 2048$ and $\sigma = 16{,}384$. We remember that the analytical expressions of $P(A_M)$ and $P(A_H)$ are shown in Table 3.

In the uniform case (Figure 11a), $P(A_H)$ is very close to 1 for small values of q. Therefore, A is mainly approximated with A_H, with negative effects on the multiplier accuracy. Increasing q, $P(A_M)$ increases, whereas $P(A_H)$ reduces. This improves the accuracy, since the probability of approximating A with A_M grows up. When $q = n - m - 1$, $P(A_M)$ equals $P(A_H)$. As a consequence, the segmentation fairly chooses between A_M and A_H, allowing the approximation error to minimize. Therefore, increasing q allows the error performances to improve with respect to the SSM multiplier. Nevertheless, cSSM exhibits better error results also considering the optimal gESSM, since the correction technique allows the approximation error to reduce when A_H is chosen. These trends are almost confirmed in the image processing applications, where the kernels and the input images prefer the selection of the most significant segments.

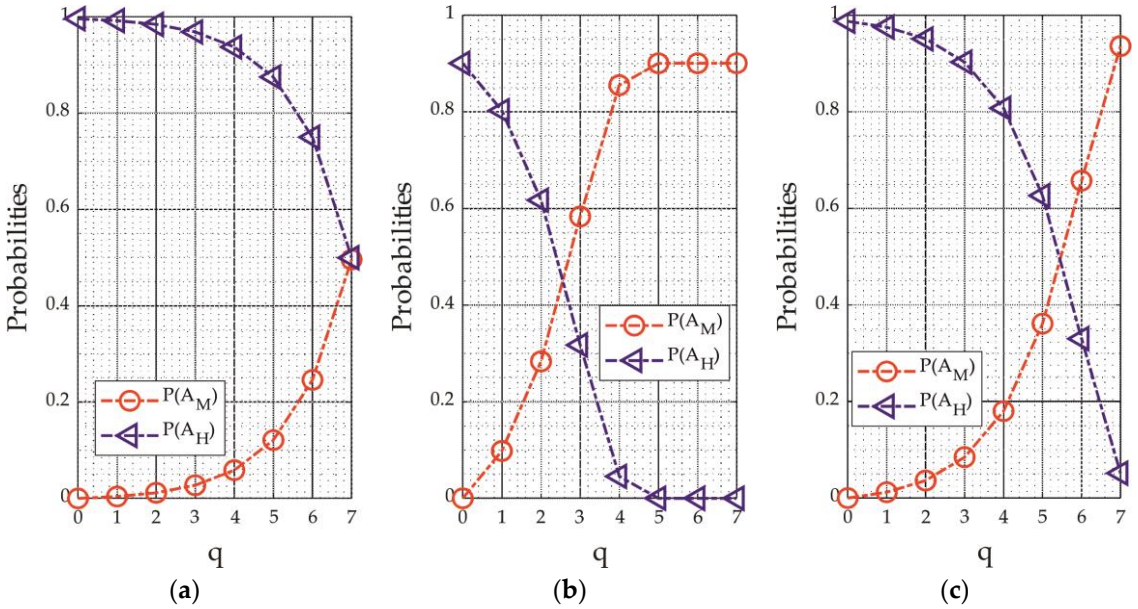


Figure 11. $P(A_M)$ and $P(A_H)$ for (**a**) the uniform distribution and the half-normal distribution in the cases (**b**) σ = 2048 and (**c**) σ = 16,384.

At the same time, the power consumption strongly reduces both with gESSM and with cSSM and SSM, whereas lower improvements are registered with the other DSM multipliers. This is mostly due to the employment of leading one detector and encoders, used to perform the dynamic segmentation. On the other hand, the power saving of gESSM is slightly weaker than SSM and cSSM due to the different selection mechanism.

The hardware performances of Qiqieh, Kulkarni, and AHMA also show interesting results, due to the reduced complexity of the PPM compression stage, but at the cost of an important loss of the quality of results.

When the distribution is half-normal, the overall mean square error presents a minimum for small values of σ. Indeed, with reference to the case σ = 2048 in Figure 11b, $P(A_M)$ increases up to $q = 5$ and is constant for $q > 5$. On the other hand, for large values of q, the resolution of A_M worsens. This leads to $q = 5$ as the optimum point since $P(A_M)$ is maximized with A_M that offers the best possible accuracy. Furthermore, Figure 5 of Section 3.2 shows also that the position of the optimal point depends on the standard deviation of the inputs: the higher σ, the higher q_{opt}. This is explained in Figure 11c for the case σ = 16,384, where $P(A_M)$ reaches the peak value only for $q = n - m - 1$, thus moving ahead the optimal point.

With reference to the case σ = 2048, the accuracy of cSSM is very close to SSM since the probability of choosing A_H is low and the correction term is practically unused. Conversely, the gESSM is able to improve the performances with a *NoEB* of 18.5 bits, also overcoming the other implementations. This scenario is confirmed by the audio processing analysis. In this application, the gESSM performs better than the cSSM, achieving a MSE of about 10^{-8}. From an electrical point of view, the power reduction offered by gESSM is remarkable when $m = 8$, and decreases if $m = 10$. Conversely, SSM and cSSM again exhibit reductions up to 89.2% and 88.1%, but at the cost of limited accuracy performances.

In order to assess the multipliers considering both the error features and the electrical performances, we show the plot of the power saving with respect to the *NMED* and the *MRED* for uniform and half-normal inputs (with σ = 2048) in Figure 12. As shown, the cSSM multipliers are on the pareto front when the inputs are uniform, offering large power saving with a high quality of results. On the contrary, when the input is half-normal, the proposed gESSM with $q = 5$, $m = 8$ and $q = 3$, $m = 10$ define the pareto front for *NMED* in the range $[9 \times 10^{-6}, 5 \times 10^{-5}]$, and *MRED* in the range $[2 \times 10^{-2}, 10^{-1}]$, offering the best

trade-off between accuracy and power consumption. Therefore, the gESSM results in the best choice when the inputs have a non-uniform distribution.

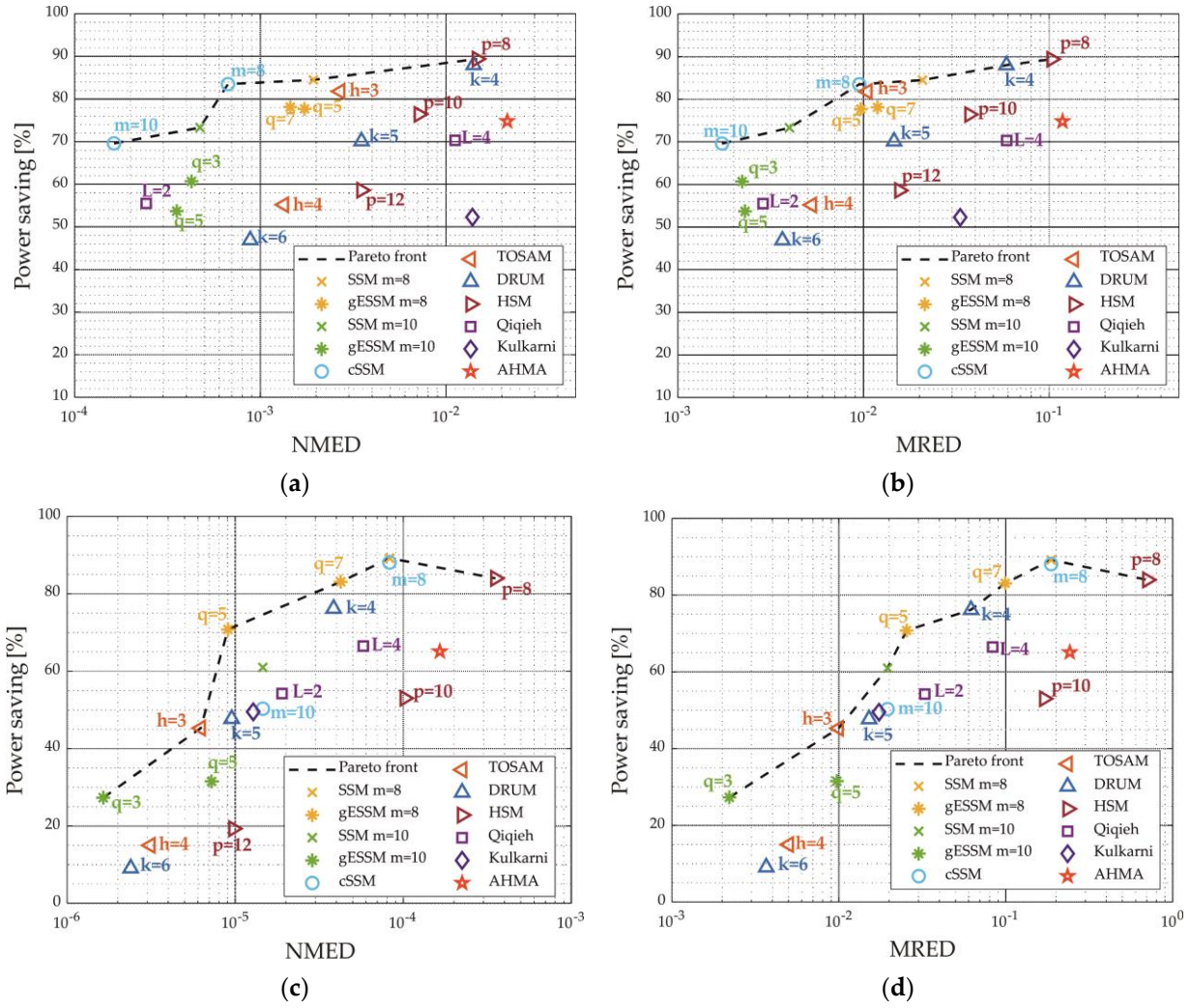


Figure 12. In the case of uniform distribution: (**a**) *NMED* vs. Power saving and (**b**) *MRED* vs. Power saving. In the case of half-normal distribution (σ = 2048): (**c**) *NMED* vs. Power saving and (**d**) *MRED* vs. Power saving.

6. Conclusions

In this paper, we have analyzed the performances of the ESSM multiplier as a function of the position of the middle segment and of the statistical properties of the input signals. While the standard implementation of the ESSM places the middle segment at the center of the input, we have moved the middle segment from the LSB to the MSB in order to find the configuration best able to minimize the mean square approximation error. To this aim, two design parameters were exploited: m, defining the accuracy and the size of the multiplier, and q, defining the position of the middle segments for further error tuning. We have described the hardware implementation of the proposed gESSM, and we have analytically demonstrated the possibility of choosing q for minimizing the overall approximation error in a mean square sense.

The error metrics reveal a strong dependence on q and on the statistical properties of the input signals. When the inputs are uniform, the best accuracy is achieved when q reaches the maximum value, whereas minimum points arise in the half-normal case (with σ = 2048). The gESSM is not able to overcome cSSM with uniform distribution, but exhibits best results with half-normal inputs (achieving *NoEB* of 18.5 bits). These trends are also

confirmed in image and audio applications, giving best results in audio filtering. The electrical performances also exhibit satisfactory results, with power reductions up to 78% and 83% in the uniform and half-normal cases, respectively.

From the comparison of the error metrics and the power saving of Figure 12, the gESSM results in the best choice when the input signal is non-uniform, offering the best trade-off between power and accuracy.

Author Contributions: Conceptualization, G.D.M., G.S. and A.G.M.S.; methodology, G.D.M. and G.S.; software, G.D.M. and G.S.; validation, G.D.M., G.S. and A.G.M.S.; formal analysis, G.D.M., G.S. and A.G.M.S.; investigation, G.D.M., G.S. and A.G.M.S.; data curation, G.D.M. and G.S.; writing—original draft preparation, G.D.M., G.S., A.G.M.S. and D.D.C.; writing—review and editing, A.G.M.S. and D.D.C.; visualization, G.D.M. and G.S.; supervision, A.G.M.S. and D.D.C.; project administration, A.G.M.S. and D.D.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The Verilog code is available on GitHub at https://github.com/GenDiMeo/gESSM, accessed on 16 February 2020.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Let consider the normal random variable A' with zero mean and standard deviation σ. The half-normal random variable A is obtained by computing the absolute value of A', i.e., $A = |A'|$.

In order to compute $P(A_M)$ and $P(A_H)$, let us consider the probability of having A in the range $[0, a]$:

$$P(0 \leq A \leq a) = \int_0^a f(A)da = erf\left(\frac{a}{\sigma\sqrt{2}}\right) \tag{A1}$$

where $f(A)$ is the pdf of A (see (12)), and $erf(\cdot)$ is the error function.

Therefore, observing that $P(A_M) = P(0 \leq A \leq 2^{m+q}) - P(0 \leq A \leq 2^m)$ and $P(A_H) = P(0 \leq A \leq 2^n - 1) - P(0 \leq A \leq 2^{m+q})$, we obtain the results shown in Table 3.

Appendix B

In order to compute (23), let us concentrate on the first summation in (22), writing the following equality:

$$E\left[\left(\sum_{k=0}^{q-1} a_k 2^k\right)^2\right] = E\left[\sum_{k=0}^{q-1}\left(a_k 2^k\right)^2\right] + 2\sum_{k=0}^{q-2} a_k 2^k \sum_{j=k+1}^{q-1} a_j 2^j \tag{A2}$$

Exploiting the linearity of the expectation operator and the independence between the bits, we obtain

$$\begin{aligned} E\left[\left(\sum_{k=0}^{q-1} a_k 2^k\right)^2\right] &= \tfrac{1}{2}\cdot\sum_{k=0}^{q-1} 2^{2k} \\ E\left[2\sum_{k=0}^{q-2} a_k 2^k \sum_{j=k+1}^{q-1} a_j 2^j\right] &= \tfrac{1}{2}\cdot\sum_{k=0}^{q-2} 2^k \sum_{j=k+1}^{q-1} 2^j \end{aligned} \tag{A3}$$

under the hypothesis $E[a_k] = 1/2$.

Therefore, observing that

$$\begin{aligned} \sum_{k=0}^{q-1} r^k &= \tfrac{1-r^q}{1-r} \\ \sum_{j=k+1}^{q-1} r^k &= \sum_{j=0}^{q-1} r^j - \sum_{j=0}^{k} r^j \end{aligned} \tag{A4}$$

with r that is a natural number, we have the following expressions after simple algebra:

$$\begin{aligned} E\left[\sum_{k=0}^{q-1}\left(a_k 2^k\right)^2\right] &= \tfrac{1}{6}\left(4^q-1\right) \\ E\left[2\sum_{k=0}^{q-2} a_k 2^k \sum_{j=k+1}^{q-1} a_j 2^j\right] &= \tfrac{1}{4}2^q\left(2^{q-1}-1\right) - \tfrac{1}{6}\left(4^{q-1}-1\right) \end{aligned} \tag{A5}$$

Applying the same reasoning for the second summation, we obtain the (23).

References

1. Spagnolo, F.; Perri, S.; Corsonello, P. Approximate Down-Sampling Strategy for Power-Constrained Intelligent Systems. *IEEE Access* **2022**, *10*, 7073–7081. [CrossRef]
2. Vaverka, F.; Mrazek, V.; Vasicek, Z.; Sekanina, L. TFApprox: Towards a Fast Emulation of DNN Approximate Hardware Accelerators on GPU. In Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 294–297. [CrossRef]
3. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2704–2713. [CrossRef]
4. Montanari, D.; Castellano, G.; Kargaran, E.; Pini, G.; Tijani, S.; De Caro, D.; Strollo, A.G.M.; Manstretta, D.; Castello, R. An FDD Wireless Diversity Receiver With Transmitter Leakage Cancellation in Transmit and Receive Bands. *IEEE J. Solid State Circuits* **2018**, *53*, 1945–1959. [CrossRef]
5. Kiayani, A.; Waheed, M.Z.; Antilla, L.; Abdelaziz, M.; Korpi, D.; Syrjala, V.; Kosunen, M.; Stadius, K.; Ryynamen, J.; Valkama, M. Adaptive Nonlinear RF Cancellation for Improved Isolation in Simultaneous Transmit–Receive Systems. *IEEE Trans. Microw. Theory Tech.* **2018**, *66*, 2299–2312. [CrossRef]
6. Zhang, T.; Su, C.; Najafi, A.; Rudell, J.C. Wideband Dual-Injection Path Self-Interference Cancellation Architecture for Full-Duplex Transceivers. *IEEE J. Solid State Circuits* **2018**, *53*, 1563–1576. [CrossRef]
7. Di Meo, G.; De Caro, D.; Saggese, G.; Napoli, E.; Petra, N.; Strollo, A.G.M. A Novel Module-Sign Low-Power Implementation for the DLMS Adaptive Filter With Low Steady-State Error. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2022**, *69*, 297–308. [CrossRef]
8. Meher, P.K.; Park, S.Y. Critical-Path Analysis and Low-Complexity Implementation of the LMS Adaptive Algorithm. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2014**, *61*, 778–788. [CrossRef]
9. Jiang, H.; Liu, L.; Jonker, P.P.; Elliott, D.G.; Lombardi, F.; Han, J. A High-Performance and Energy-Efficient FIR Adaptive Filter Using Approximate Distributed Arithmetic Circuits. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2019**, *66*, 313–326. [CrossRef]
10. Esposito, D.; Di Meo, G.; De Caro, D.; Strollo, A.G.M.; Napoli, E. Quality-Scalable Approximate LMS Filter. In Proceedings of the 2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Bordeaux, France, 9–12 December 2018; pp. 849–852. [CrossRef]
11. Di Meo, G.; De Caro, D.; Petra, N.; Strollo, A.G.M. A Novel Low-Power High-Precision Implementation for Sign–Magnitude DLMS Adaptive Filters. *Electronics* **2022**, *11*, 1007. [CrossRef]
12. Bruschi, V.; Nobili, S.; Terenzi, A.; Cecchi, S. A Low-Complexity Linear-Phase Graphic Audio Equalizer Based on IFIR Filters. *IEEE Signal Process. Lett.* **2021**, *28*, 429–433. [CrossRef]
13. Kulkarni, P.; Gupta, P.; Ercegovac, M. Trading Accuracy for Power with an Underdesigned Multiplier Architecture. In Proceedings of the 2011 24th Internatioal Conference on VLSI Design, Chennai, India, 2–7 January 2011; pp. 346–351. [CrossRef]
14. Zervakis, G.; Tsoumanis, K.; Xydis, S.; Soudris, D.; Pekmestzi, K. Design-Efficient Approximate Multiplication Circuits Through Partial Product Perforation. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2016**, *24*, 3105–3117. [CrossRef]
15. Zacharelos, E.; Nunziata, I.; Saggese, G.; Strollo, A.G.M.; Napoli, E. Approximate Recursive Multipliers Using Low Power Building Blocks. *IEEE Trans. Emerg. Top. Comput.* **2022**, *10*, 1315–1330. [CrossRef]
16. Qiqieh, I.; Shafik, R.; Tarawneh, G.; Sokolov, D.; Yakovlev, A. Energy-efficient approximate multiplier design using bit significance-driven logic compression. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, Switzerland, 27–31 March 2017; pp. 7–12. [CrossRef]
17. Esposito, D.; Strollo, A.G.M.; Alioto, M. Low-power approximate MAC unit. In Proceedings of the 2017 13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), Giardini Naxos-Taormina, Italy, 12–15 June 2017; pp. 81–84. [CrossRef]
18. Fritz, C.; Fam, A.T. Fast Binary Counters Based on Symmetric Stacking. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *25*, 2971–2975. [CrossRef]
19. Ahmadinejad, M.; Moaiyeri, M.H.; Sabetzadeh, F. Energy and area efficient imprecise compressors for approximate multiplication at nanoscale. *Int. J. Electron. Commun.* **2019**, *110*, 152859. [CrossRef]
20. Yang, Z.; Han, J.; Lombardi, F. Approximate compressors for error-resilient multiplier design. In Proceedings of the 2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), Amherst, MA, USA, 12–14 October 2015; pp. 183–186. [CrossRef]

21. Ha, M.; Lee, S. Multipliers With Approximate 4–2 Compressors and Error Recovery Modules. *IEEE Embed. Syst. Lett.* **2018**, *10*, 6–9. [CrossRef]
22. Strollo, A.G.M.; Napoli, E.; De Caro, D.; Petra, N.; Meo, G.D. Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**, *67*, 3021–3034. [CrossRef]
23. Park, G.; Kung, J.; Lee, Y. Design and Analysis of Approximate Compressors for Balanced Error Accumulation in MAC Operator. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2021**, *68*, 2950–2961. [CrossRef]
24. Kong, T.; Li, S. Design and Analysis of Approximate 4–2 Compressors for High-Accuracy Multipliers. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2021**, *29*, 1771–1781. [CrossRef]
25. Jou, J.M.; Kuang, S.R.; Chen, R.D. Design of low-error fixed-width multipliers for DSP applications. *IEEE Trans. Circuits Syst. II Analog. Digit. Signal Process.* **1999**, *46*, 836–842. [CrossRef]
26. Petra, N.; De Caro, D.; Garofalo, V.; Napoli, E.; Strollo, A.G.M. Design of Fixed-Width Multipliers With Linear Compensation Function. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2011**, *58*, 947–960. [CrossRef]
27. Hashemi, S.; Bahar, R.I.; Reda, S. DRUM: A Dynamic Range Unbiased Multiplier for approximate applications. In Proceedings of the 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, USA, 2–6 November 2015; pp. 418–425. [CrossRef]
28. Vahdat, S.; Kamal, M.; Afzali-Kusha, A.; Pedram, M. TOSAM: An Energy-Efficient Truncation- and Rounding-Based Scalable Approximate Multiplier. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *27*, 1161–1173. [CrossRef]
29. Narayanamoorthy, S.; Moghaddam, H.A.; Liu, Z.; Park, T.; Kim, N.S. Energy-Efficient Approximate Multiplication for Digital Signal Processing and Classification Applications. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2015**, *23*, 1180–1184. [CrossRef]
30. Strollo, A.G.M.; Napoli, E.; De Caro, D.; Petra, N.; Saggese, G.; Di Meo, G. Approximate Multipliers Using Static Segmentation: Error Analysis and Improvements. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2022**, *69*, 2449–2462. [CrossRef]
31. Li, L.; Hammad, I.; El-Sankary, K. Dual segmentation approximate multiplier. *Electron. Lett.* **2021**, *57*, 718–720. [CrossRef]
32. GitHub. Available online: https://github.com/scale-lab/DRUM (accessed on 18 April 2020).
33. GitHub. Available online: https://github.com/astrollo/SSM (accessed on 16 February 2020).
34. DataShare. Available online: https://datashare.ed.ac.uk/handle/10283/2791 (accessed on 21 August 2017).