Journal of
Low Power Electronics and Applications

MDPI

Article

Designing Energy-Efficient Approximate Multipliers

Stefania Perri [1,*], Fanny Spagnolo [2], Fabio Frustaci [2] and Pasquale Corsonello [2]

[1] Department of Mechanical, Energy and Management Engineering, University of Calabria, 87036 Rende, Italy
[2] Department of Informatics, Modeling, Electronics and System Engineering, University of Calabria, 87036 Rende, Italy
* Correspondence: s.perri@unical.it; Tel.: +39-0984-494-765

Abstract: This paper proposes a novel approach suitable to design energy-efficient approximate multipliers using both ASIC and FPGAs. The new strategy harnesses specific encoding logics based on bit significance and computes the approximate product performing accurate sub-multiplications by applying an unconventional approach instead of using approximate computational modules implementing traditional static or dynamic bit-truncation approaches. The proposed platform-independent architecture exhibits an energy saving of up to 80% over the accurate counterparts and significantly better behavior in terms of accuracy loss with respect to competitor approximate architectures. When employed in 2D digital filters and edge detectors, the novel approximate multipliers lead to an energy consumption up to ~82% lower than the accurate counterparts, which is up to ~2 times higher than that obtained by state-of-the-art competitors.

Keywords: approximate computing; binary multipliers; energy efficiency; digital circuits

Citation: Perri, S.; Spagnolo, F.; Frustaci, F.; Corsonello, P. Designing Energy-Efficient Approximate Multipliers. *J. Low Power Electron. Appl.* **2022**, *12*, 49. https://doi.org/10.3390/jlpea12040049

Academic Editors: Luis Parrilla Roure, Antonio García and Encarnación Castillo

Received: 31 August 2022
Accepted: 23 September 2022
Published: 27 September 2022

1. Introduction

Inspired by the observation that exact (or precise) computations are not always necessary in most modern applications, approximate computing is nowadays a widely used paradigm for designing error-resilient circuits that can trade accuracy for energy [1,2].

The topic of several papers [3–26] is the design of energy-efficient approximate arithmetic circuits realized either by using Application Specific Integrated Circuits (ASICs) or Field Programmable Gate Arrays (FPGAs). Among them, in particular, approximate integer multipliers received a great deal of attention [3–19]. Despite the generality of the adopted approximation logic, implementing such circuits into either ASIC or FPGA technologies may lead to quite different energy, timing, and area behavior due to the different utilization of available resources. For example, the simplest approximation strategy often adopted in ASIC designs is bit-truncation [3–6]. When applied statically [3,6], it allows the pruning of the hardware resources used to compute a pre-established number of least significant bits (LSBs) of the product. Conversely, dynamic truncation techniques [4,5] allow the energy saving to be tuned on a time-varying quality target. As an efficient alternative, the static approach proposed very recently in [7] exploits a small inner multiplier to process m-bit segments of the operands and adopts a correction technique to improve the error performance. On the contrary, the approximation approaches presented in [8,9] accumulate the partial products (PPs) with approximate circuits that save energy, introducing a reasonable accuracy loss.

Unfortunately, the approaches conceived for ASIC designs are often not effective when adopted for FPGA implementations. Indeed, in most cases, they lead to energy consumptions higher than the accurate multipliers. For this reason, alternative strategies specific to FPGA designs are proposed in [10–19]. Most of them exploit Booth's algorithm, which is simplified by either truncating specific bits of the PPs [14] or approximating the encoding logic [15]. Others are based on modular approaches [12,16–19] that allow high-order multipliers to be implemented involving approximate low-order sub-multipliers. However, these

methods are based on platform-specific optimizations that allow approximate operations to be efficiently mapped within Look-Up-Tables (LUTs), and, as a consequence, they do not perform as well when implemented in ASIC.

The above overview of the state-of-the-art counterparts discloses that none of the above papers' proposed design methods have either been demonstrated on both ASICs and FPGAs or shown the potentiality of being competitive on both platforms. Indeed, although they are described using the Very High-Speed Integrated Circuits Hardware Description Language (VHDL), the above designs can be synthesized and implemented onto both FPGAs and ASIC, and they can achieve energy-delay trade-offs quite far from that reached by counterparts natively optimized for a specific platform. Therefore, they do not appear to be good candidates for the platform-independent design approach that we want to propose.

With the aim of introducing a platform-independent approximate multiplication strategy, this paper presents a new technique that allows modular energy-efficient approximate multipliers to be designed. In contrast to existing approaches [8–19], the proposed multiplier provides inexact results by using unconventional accurate sub-multipliers on approximated input operands, instead of performing approximate computations. Thanks to this strategy, the novel multiplier can be coded at the VHDL level, avoiding any specific platform-dependent primitives. Therefore, it is suitable to be processed by any synthesis tool and for any hardware platform without requiring specific modifications concerning the target technology. It is worth underlining that the approaches discussed in previously published papers [3–19] significantly differ from the method presented here.

To demonstrate the effectiveness of the proposed strategy, experiments were performed in both the ASIC and FPGA domains. In the former case, we achieve an energy saving over the accurate baseline of up to more than 80%, which, at a comparable number of effective bits (NoEB), is quite better than that obtained by the approximate multiplier recently presented in [7]. A significant advantage in terms of energy reduction with higher NoEB is achieved with respect to the architectures described in [8,9]. The results obtained in comparison with the competitors [15–19] also show that, among FPGA-based implementations, the proposed strategy reaches the best energy-accuracy trade-off.

Similar to previous works, novel multipliers were included in the design of approximate 2D image filters and edge detectors. In the former application, the proposed design consumes ~82% less energy than the accurate baseline, without introducing any Structural Similarity index (SSIM) [26] degradation, thus overcoming achievements in [8]. Conversely, the edge detection tests demonstrate that the proposed multiplier allows a higher edge-detected percentage to be reached with an energy saving only 0.88% lower than [14].

2. Background and Related Works

In this section, the behavior of conventional $n \times m$ integer multipliers and some representative static approximation strategies are briefly described. In order to do this, let us assume, without loss of generality, that the n-bit multiplicand $A_{[n-1:0]} = a_{n-1}, \ldots, a_0$ and the m-bit multiplier $B_{[n-1:0]} = b_{n-1}, \ldots, b_0$ are 2′s complement numbers represented as given in (1). As it is well known, the basic multiplication algorithm first computes the bitwise ANDs between the operand A and the bits of B. Then, in order to obtain the generic partial product PPj, with $j = 0, \ldots, m-1$, the j-th result produced by the AND operation related to the bit b_j, is left shifted by j bit positions and sign extended to $(n + m)$ bits. Finally, as shown by (2), the exact product $Pe_{[n+m-1:0]}$ is calculated by accumulating the m computed PPs. It is important to highlight that the simpler behavior of a multiplier processing unsigned operands can be easily derived from (1) and (2) by just removing the initial minus sign.

$$\begin{aligned} A_{[n-1:0]} &= -a_{n-1} \cdot 2^{n-1} + \textstyle\sum_{i=0}^{n-2} a_i \cdot 2^i \\ B_{[m-1:0]} &= -b_{m-1} \cdot 2^{m-1} + \textstyle\sum_{j=0}^{m-2} b_j \cdot 2^j \end{aligned} \quad (1)$$

$$Pe_{[n+m-1:0]} = -b_{m-1} \cdot 2^{m-1} \cdot A_{[n-1:0]} + \sum_{j=0}^{m-2} b_j \cdot 2^j \cdot A_{[n-1:0]} \quad (2)$$

When the radix-2r Booth's algorithm is adopted, the m bits of the signed multiplier B are split into $\lceil \frac{m}{r} \rceil$ $(r+1)$-bit groups, with 1-bit overlaps. An encoded digit is extracted from each group and used to generate the partial product PPi (with $i = 0, \ldots, \lceil \frac{m}{r} \rceil - 1$) as a multiple of A. As an example, with $r = 2$ the generic encoded digit can assume the values 0, ± 1 and ± 2, whereas with $r = 3$, it can be equal to 0, ± 1, ± 2, ± 3, and ± 4. Each PPi is sign extended and left shifted by $r \times i$ bit positions to be aligned to the other partial products for the accumulation that furnishes the exact product $Pe_{[n+m-1:0]}$ as given in (3). In this case, in order to treat unsigned inputs correctly, A and B must be zero extended to $(n+1)$- and $(m+1)$-bit, respectively.

$$Pe_{[n+m-1:0]} = \sum_{j=0}^{\lceil \frac{m}{r} \rceil - 1} 2^{r \cdot j} \cdot PP_j \quad (3)$$

Both the above multiplication algorithms are suitable for the modular approach that can be applied, as an example, by splitting the operands A and B into two sub-words, namely, $A_M = a_{n-1} \ldots a_{ka}$, $A_L = a_{ka-1} \ldots a_0$, $B_M = b_{m-1} \ldots b_{kb}$, and $B_L = b_{kb-1} \ldots b_0$. In this case, the product Pe is calculated as shown in (4), where $P_{MM} = A_M \times B_M$, $P_{ML} = A_M \times B_L$, $P_{LM} = B_M \times A_L$, and $P_{LL} = A_L \times B_L$.

$$Pe_{[n+m-1:0]} = 2^{ka+kb} \cdot P_{MM} + 2^{ka} \cdot P_{ML} + 2^{kb} \cdot P_{LM} + P_{LL} \quad (4)$$

It is worth noting that, in the case of signed operands, while the sub-words A_M and B_M still represent 2's complement numbers, the sub-words A_L and B_L are unsigned numbers. This makes the management of signs information necessary to compute the sub-products P_{ML}, P_{LM}, and P_{LL} much simpler than what is required for calculating P_{MM}. Obviously, the overall computation is even easier when unsigned operands are processed. Furthermore, it is easy to understand that, independent of the adopted algorithm, the modular approach could be applied recursively to compute the sub-products, as shown, for example, in [27].

The above formulations suggest that several strategies are viable to design efficient approximate multipliers. For example, the ASIC implementations presented in [8,9] compute the PPs conventionally as the bitwise AND between A and B, and then accumulate them by means of approximate compressors. Depending on the approximation level adopted and the chosen truncation, four approximate multipliers (named 1StepFull, 1StepTrunc, 2StepFull, and 2StepTrunc) are presented in [8], each providing a different trade-off between speed, power, and accuracy. Conversely, the two approximate multipliers (called C-N and C-FULL) described in [9] differ from each other in the way they use approximate 4-2 compressors. While the architecture C-N exploits the approximate 4-2 compressors only to process the LSBs of the PPs, thus limiting the errors introduced with respect to the exact product, the design C-FULL uses the approximate 4-2 compressors on the entire PPs, thus saving more energy, but sacrificing the accuracy.

The approaches known for FPGA-based designs make use of quite different approximation logics. The architectures (called AxBM1 and AxBM2) proposed in [14] approximate the radix-8 Booth's multiplier by exploiting new encoders on purpose designed to compute inexact PPs and to be efficiently mapped within LUT primitives. A different method is applied in [15] to design a radix-4 Booth's approximate multiplier (called BA) by taking advantage also of a LUT-level optimization strategy. In such a case, the logic operations performed by the LUTs responsible for computing the two LSBs of the PPs are removed or approximated, thus saving energy consumption and hardware resources. Alternative ways to exploit efficiently LUT-optimized implementations are presented in [16–19]. In [16], a 4×2 approximate multiplier is used as the basic block to design higher-order multipliers. In such a basic block, the PPs are computed by performing the bitwise AND between the multiplicand A and the bits of B, then they are grouped two by two and added by means of the fast carry chains available in modern FPGAs [28,29]. The modular $w \times w$ multiplier, designed as described in [16], performs the generic computation by summing four $w/2 \times w/2$ approximate sub-products using either an accurate or an approximate

ternary adder, thus leading to two architectures, named CA and CC, respectively. In a similar way, the modular designs proposed in [17,18] exploit 4×4 and 2×2 sub-multipliers to implement higher order multipliers. Finally, the open-source library presented in [19] collects several 8-bit approximate circuits, including 471 8×8 multipliers designed using conventional multiplication structures.

3. The Novel Approximation Strategy

The architecture of the proposed multiplier is illustrated in Figure 1. It relies on a double-stage encoding logic to simplify the multiplication by minimizing the number of non-zero bits involved in the accumulation of partial products. During the first stage, the inputs $A = a_{n-1} \dots a_0$ and $B = b_{m-1} \dots b_0$ are split into the sub-words $A_M = a_{n-1} \dots a_{ka}$, $A_L = a_{ka-1} \dots a_0$, $B_M = b_{m-1} \dots b_{kb}$, and $B_L = b_{kb-1} \dots b_0$, with ka and kb being chosen at design time. Then, the least significant sub-words A_L and B_L are partitioned into non-overlapping 3-bit groups and encoded through an on-purpose conceived method based on a backward propagation action. The encoded digits CDx are properly aligned and OR-ed in overlapped positions, thus obtaining the approximate versions of the least significant sub-words A_{La} and B_{La}, corresponding to the closest power-of-two. In the second stage, four sub-products are calculated by multiplying the sub-words A_M, B_M, A_{La} and B_{La}. While the most significant term $P_{MM} = A_M \times B_M$ is computed through a conventional multiplier, the others are obtained by using the new radix-4 encoding logic (NR4EL) here indicated with the operator Θ, which performs an accurate multiplication on approximated input operands, having at most one non-zero partial product. That is, $P_{MLa} = A_M \, \Theta B_{La}$, $P_{LMa} = B_M \Theta \, A_{La}$, and $P_{LLa} = A_{La} \, \Theta B_{La}$. Both encoding logics above mentioned will be detailed later. Finally, the sub-products P_{MM}, P_{MLa}, P_{LMa}, and P_{LLa} are aligned, sign-extended, and accumulated to compute the final approximate product Pa, as given in (5).

$$Pa_{[n+m-1:0]} = 2^{ka+kb} \cdot P_{MM} + 2^{ka} \cdot P_{MLa} + 2^{kb} \cdot P_{LMa} + P_{LLa} \tag{5}$$

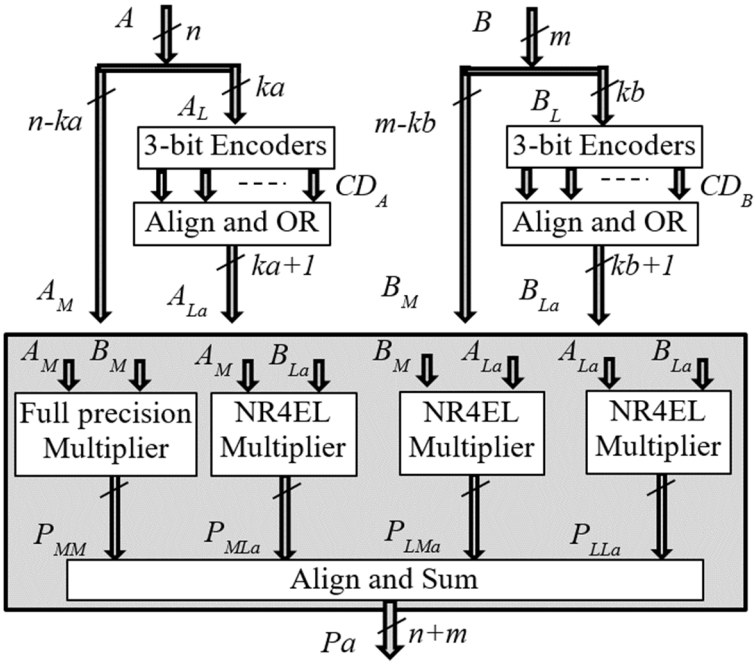


Figure 1. The top-level architecture of the proposed multiplier.

3.1. The New 3-Bit Encoding Logic for Least Significant Sub-Words

As illustrated in Figure 2, before being encoded with the proposed method, the unsigned sub-words A_L and B_L are zero extended to be treated as positive numbers.

Furthermore, additional zeros are put beside the least significant positions (as schematized with the red dots in Figure 2) if needed to obtain an integer number of non-overlapping 3-bit groups. The most significant group is encoded by using the novel logic E3bMG, whereas for the less significant bit positions, the encoding rules E3bG are applied. As visible in Figure 2, such an encoding logic is based on a back-propagation action sustained by the signals P_{in} and P_{out}. As shown in the following, coded digits CDx are then aligned and OR-ed to finally furnish A_{La} and B_{La}.

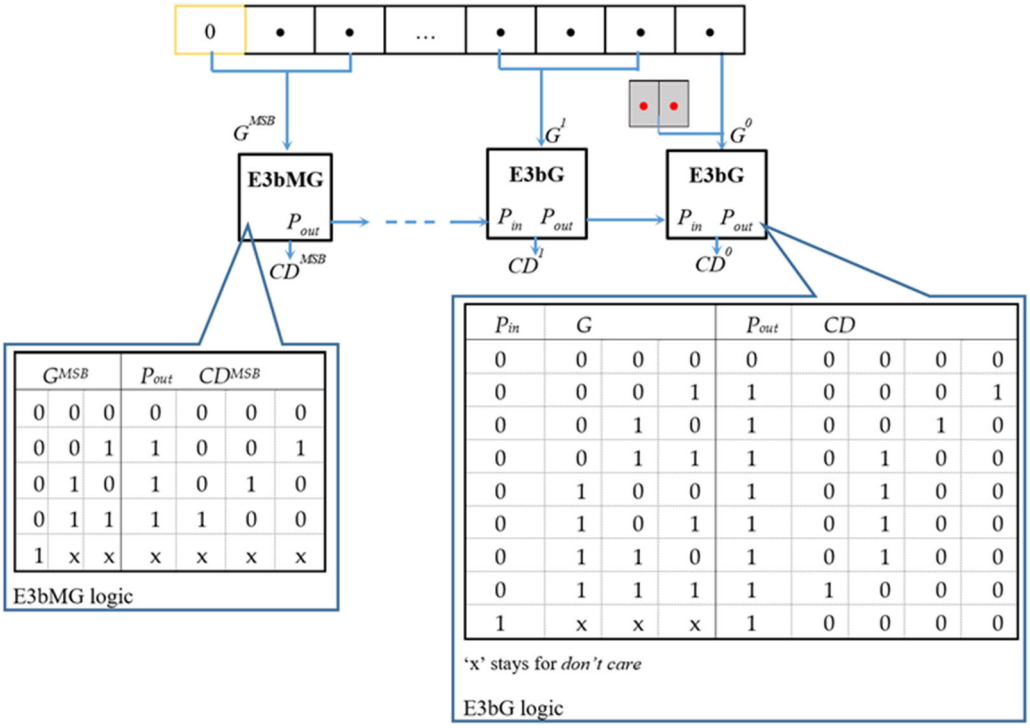


Figure 2. Partitioning and encoding logic applied to the sub-words A_L and B_L.

It is worth noting that the above encoding strategy introduces an approximation to the closest power of two. As detailed in the following, this property allows simplifying the logic required to compute the accurate sub-products P_{MLa}, P_{LMa}, and P_{LLa}. In addition, it must be pointed out that, in this context, the novel E3bMG and E3bG logic perform much better than the conventional leading one detection. To understand this, as an example, let us consider the 8-bit numbers 127 and 65. While the proposed encoding provides the approximate values 128 and 64, thus leading to an absolute error equal to 1 in both the cases, the conventional technique approximates both the values to 64, thus causing an absolute error equal to 63 and 1, respectively.

3.2. The NR4EL Multiplication

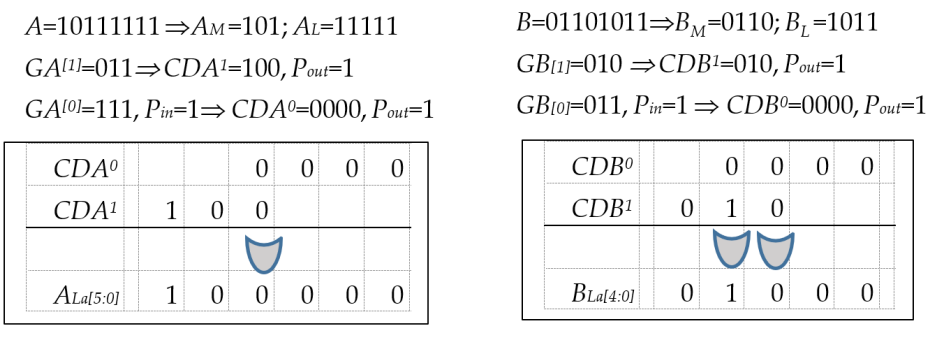
Starting from the observation that the approximate sub-words A_{La} and B_{La} are represented as power-of-two, containing at most one non-zero bit, a further original encoding step is here proposed to exploit this property in computing the sub-products P_{MLa}, P_{LMa}, and P_{LLa}. That is: A_{La} and B_{La} are split into 3-bit groups, with 1-bit overlaps, and zero extended if needed to complete the most significant group. The NR4EL summarized in Figure 3 is applied to each 3-bit group GL to obtain the corresponding partial product PP. Since the approximate sub-words contain at most one non-zero bit, the NR4EL can output just three possible values: 0, MD, and $2 \times MD$, with MD being the multiplicand (i.e., A_M or B_M or A_{La}). Moreover, the computations of the sub-products P_{MLa} and P_{LMa} involve at most one non-zero partial product, whereas at most just one bit is asserted among all the partial products computed to calculate P_{LLa}. Due to this, partial products are then

accumulated by simple logic ORs rather than addition circuits, thus ensuring that a quite significant energy reduction is expected with respect to conventional approaches.

GL			*PP*
0	0	0	0
0	0	1	0
0	1	0	*MD*
0	1	1	-
1	0	0	*2×MD*
1	0	1	-
1	1	0	-
1	1	1	-

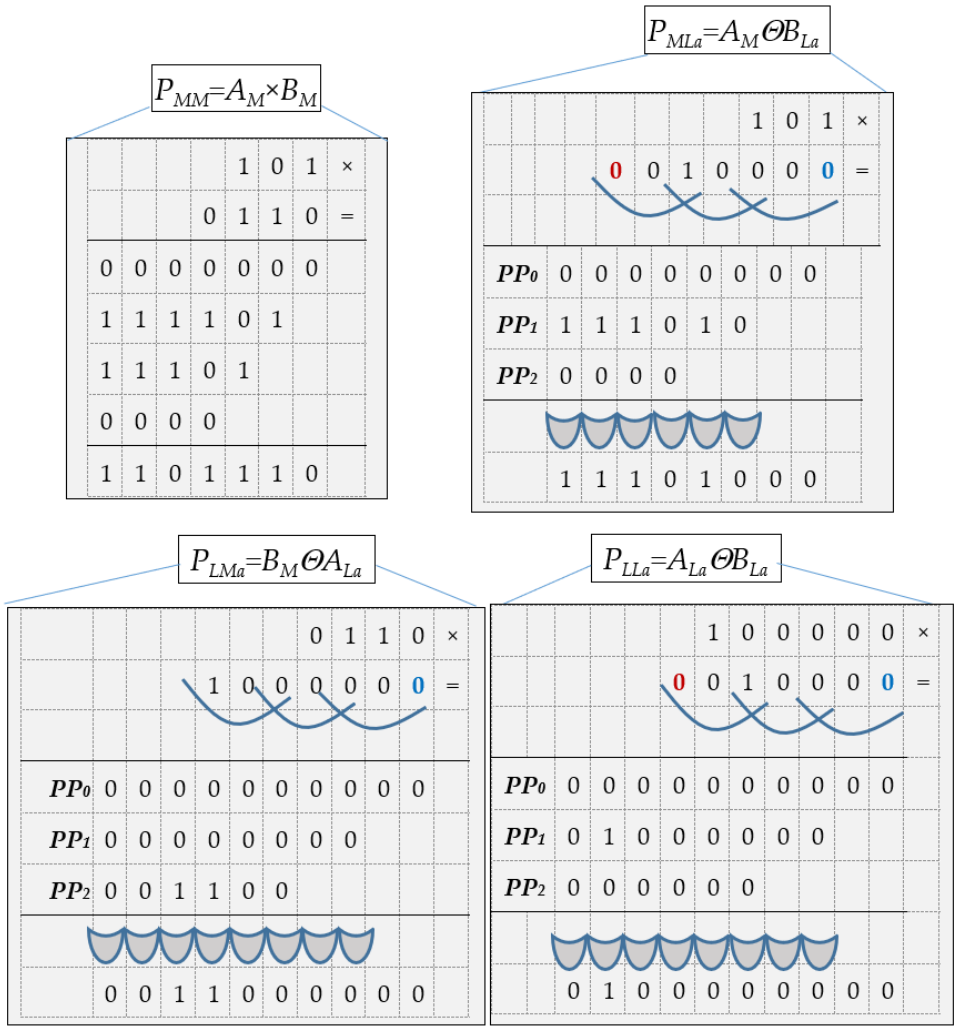
Figure 3. The novel encoding logic NR4EL.

It is worth noting that, due to the approximation made on the least significant bits of the input operands, the proposed NR4EL logic leads to hardware requirements quite different from that of a conventional radix-4 Booth multiplier. Just as a comparison, let us refer to the example illustrated in Figure 4, where $n = m = 8$ and the configuration $ka = 5$ and $kb = 4$ is used to perform the multiplication by the novel approximate strategy. The input operands A and B are firstly partitioned into the most significant (A_M, B_M) and least significant (A_L,B_L) parts. The latter are zero-extended and encoded through the 3-bit logic shown in Section 3.1. Coded digits are aligned taking into account that their significance is dictated by the bit positions involved in the 3-bit groups from which they are calculated. The approximate values A_{La} and B_{La} are then obtained by simply ORing their overlapped bits. As discussed above (see Figure 1), P_{MM} is computed by a full precision conventional multiplier, whereas P_{MLa}, P_{LMa}, and P_{LLa} exploit the NR4EL multiplication logic. In contrast to the Booth multiplier, the proposed one, thanks to its coding strategy, allows any additional circuit for the computation of P_{MLa}, P_{LMa}, and P_{LLa} to be avoided, as illustrated in Figure 4b. The sub-products obtained in this way are aligned, sign extended, and summed to finally furnish the approximate product Pa, as shown in the last step of Figure 4, which also reports the exact product Pe.



(**a**)

Figure 4. *Cont.*



(**b**)

P_{MM} (signed)	**1**	**1**	**0**	**1**	**1**	**1**	**0**	0	0	0	0	0	0	0	0	0
P_{MLa} (signed)	1	1	1	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	0	0	0	0	0
P_{LMa} (signed)	0	0	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	0	0	0	0
P_{LLa} (unsigned)	0	0	0	0	0	0	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Pa=-6656	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0
Pe=-6955	1	1	1	0	0	1	0	0	1	1	0	1	0	1	0	1

(**c**)

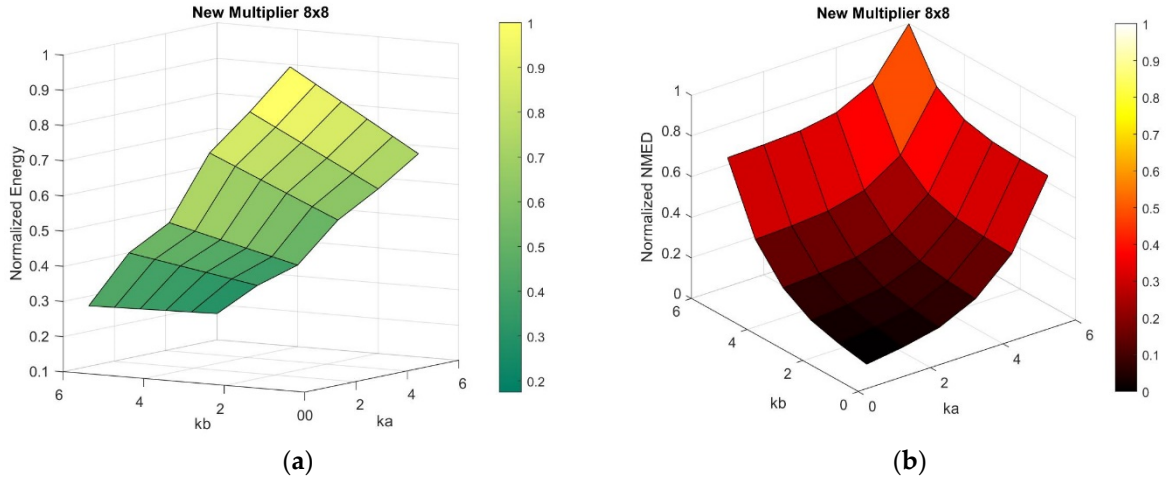
Figure 4. An example of multiplication through the proposed approximate approach: (**a**) approximate A and B; (**b**) compute P_{MM}, P_{MLa}, P_{LMa}, P_{LLa}; (**c**) compute the approximate product Pa.

4. Accuracy and Implementation Results

To prove the effectiveness and the high flexibility of the proposed method, several signed $n \times m$ approximate multipliers were implemented using both ASIC and FPGA realization platforms. In the following, New_{ka_kb} indicates a multiplier designed as described here that approximates *ka* LSBs of *A* and *kb* LSBs of *B*. This section presents results obtained for both symmetric and asymmetric designs. Performances achieved by our proposal are discussed and compared with competitors. All quality measures, in terms of average error (AE), error rate (ER), normalized mean error distance (NMED), mean relative error distance (MRED), defined as reported [30], and number of effective bits (NoEB), introduced in [8], have been obtained through exhaustive C++ simulations. It is worth noting that accuracy tests for multipliers with operands word lengths greater than 16-bit are excessively time consuming. Therefore, as in all the previous works [3–19] for such cases, only the hardware characteristics are provided.

4.1. Design Space Exploration

It is important to note that the possibility of differently setting *ka* and *kb* represents a further degree of freedom that can be exploited to finely tune the energy and accuracy of the multiplier to the requirements of a given specific application. This property leads to a design space wider than that bounded by using *ka* = *kb* and it cannot always be obtained by other techniques, such as those based on approximate compressors. In Figure 5, the normalized energy-NMED design space for the 8 × 8 multiplier is illustrated for *ka* and *kb* varying in the range 1–6.



(**a**) (**b**)

Figure 5. The design spaces for the 8 × 8 multiplier: (**a**) normalized energy; (**b**) normalized NMED.

Just as an example, with respect to the symmetric *ka* = *kb* = 4 scenario, approximating one more bit on one operand (e.g., *ka* = 4 and *kb* = 5) leads to a 7% higher energy gain with an NMED increased by only ~0.005. On the contrary, the *ka* = 3 and *kb* = 4 configuration reduces the NMED by ~0.002 and the energy gain with respect to the precise architecture by ~3.5%. Such an analysis can be useful to optimize the parameters *ka* and *kb* for a given scenario. As an example, for the image processing applications referred to in Section 5, the configuration with *ka* = 2 and *kb* = 6 is particularly efficient, given the significantly different nature of the operands to be multiplied. However, the optimizations of *ka* and *kb* and the realization of a design framework are beyond the scope of this paper.

4.2. ASIC Implementations

For purposes of a fair comparison with state-of-the-art counterparts, 8 × 8 and 16 × 16 signed multipliers were implemented using the TSMC40 nm CMOS 1.1 V and the ST28 nm UTBB-FDSOI 1 V technologies. They were synthesized with the Cadence Genus™ tool version 19.11 at the minimum delay constraint inserting registers as the driving and the loading logic, with the output flip-flops having 0.1 pF load capacitances. The energy consumption was analyzed using the Value Change Dump (VCD) files extracted for 100,000 random vectors.

Tables 1 and 2 collect results obtained in terms of delay (D), silicon area (A), energy (E), average error (AE), error rate (ER), and number of effective bits (NoEB). The behavior of each approximate multiplier is clearly appreciable in comparison with the precise baseline versions realized with the same technology process.

Table 1. Hardware Characteristics of the 8 × 8 ASIC Designs.

Circuit	Process	D (ps)	A (um^2)	E (pJ)	AE	ER%	NoEB
Baseline [8]	40 nm	564	986	1.29		PRECISE	
1StepFull [8]	40 nm	500	524	0.81	42.3	30	9.47
1StepTrunc [8]	40 nm	500	310	0.47	2.3×10^2	96	7.89
2StepFull [8]	40 nm	419	428	0.72	8.7×10^2	84	5.59
2StepTrunc [8]	40 nm	375	171	0.3	1.0×10^3	99	5.46
Our Baseline	40 nm	506	780.4	0.24		PRECISE	
New_{2_6}	40 nm	519	529.6	0.04	0.324	91.4	6.79
Baseline [9]	28 nm	260	196	0.046		PRECISE	
C-N [9]	28 nm	248.6	175	0.041	n.a.	9	10.8
C-Full [9]	28 nm	216	155	0.031	n.a.	40	5.44
Our Baseline	28 nm	280	370	0.16		PRECISE	
New_{2_6}	28 nm	284	360	0.031	0.324	91.4	6.79

Table 2. Hardware Characteristics of the 16 × 16 ASIC Designs.

Circuit	Process	D (ps)	A (um^2)	E (pJ)	AE	ER%	NoEB
Baseline [8]	40 nm	800	2595	3.58		PRECISE	
1StepFull [8]	40 nm	746	1859	2.94	3.57×10^4	61	16.04
1StepTrunc [8]	40 nm	730	1002	1.56	1.45×10^5	100	14.66
2StepFull [8]	40 nm	667	1147	2.01	3.77×10^6	97	9.36
2StepTrunc [8]	40 nm	650	700	1.29	3.86×10^6	100	9.35
Our Baseline	40 nm	720	2362	7.2		PRECISE	
New_{8_8}	40 nm	737	1814	1.62	8867.18	99.87	10.12
Baseline [9]	28 nm	375	920	3.52		PRECISE	
C-N [9]	28 nm	363	821	2.94	n.a.	47	17.53
C-Full [9]	28 nm	318	727	2.11	n.a.	88	5.44
Our Baseline	28 nm	445	1016	4.68		PRECISE	
New_{8_8}	28 nm	446	849	1.2	8867.18	99.87	10.12

The New_{2_6} signed design achieves an energy saving higher than 80%, with a negligible impact on the speed performances. The 2StepTrunc signed architecture [8] shows an energy saving with respect to its baseline of ~76%, and, even though it reaches an interesting delay reduction, the achieved quality level is quite lower than the New_{2_6}. On the other side, while the C-Full circuit [9] dissipates the same energy as the proposed one, it shows a much lower gain with respect to the baseline and achieves a NoEB lower than the New_{2_6}. Furthermore, it must be considered that the architectures in [9] operate only on unsigned operands. The above analysis confirms the effectiveness of the proposed

approach in reducing the number of non-zero bits within the tree of partial products in favor of energy efficiency. Indeed, the strategies proposed in [8,9], being, respectively, based on LSB truncation and approximate compressors, just partially simplify the adder circuits responsible for the accumulation of the partial products.

The energy gain obtained over the baseline generally deteriorates with the operands word length. From Table 2, it can be observed that the New_{8_8} 16 × 16 signed multiplier saves ~75% of the energy, whereas [8] saves at most ~63%. Surprisingly, [9] shows a ~8% improvement in this figure. However, the quality level of the 16 × 16 New_{8_8} multiplier still overcomes the competitors. On the other hand, [8,9] achieve area and delay reductions remarkably higher than the new designs.

In order to evaluate how the ASIC designs trade-off energy saving (*EnSv*), accuracy, area, and delay, the figure of merit defined in (6) and the comprehensive cost function given in (7) are introduced.

$$FM_{ASIC} = EnSv\% \times NoEB \tag{6}$$

$$CF_{ASIC} = \frac{D \times A \times E}{NoEB} \tag{7}$$

Figure 6 plots the normalized values of FM_{ASIC} (NFM) and CF_{ASIC} (NCF) and shows that the FM_{ASIC} achieved by the New_{2_6} circuit is 12% and 34% higher than 1StepTrunc [8] and CSSM [7], respectively. Indeed, at a comparable NoEB, the signed 8 × 8 architectures demonstrated in [7] reach a power saving ~20% lower. The graceful behavior of the proposed multiplier is confirmed by the CF_{ASIC}, which is up to 13 times lower than that of the competitors.

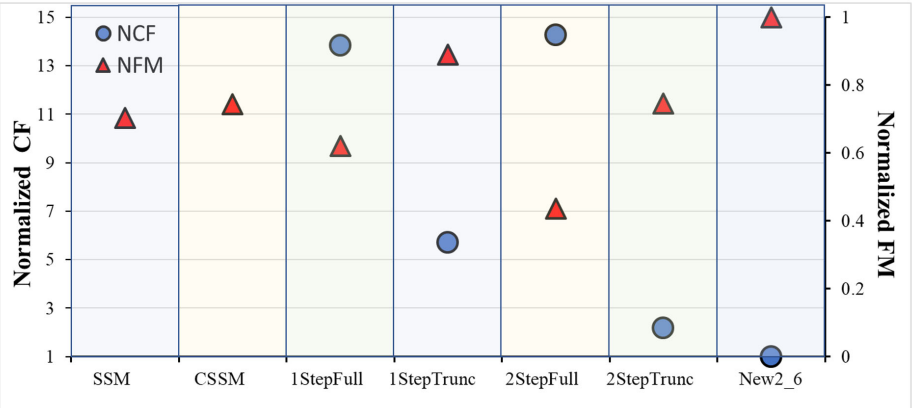


Figure 6. Normalized FMASIC and CFASIC of 8 × 8 signed designs (SSM [7], CSSM [7], 1StepFull [8], 1StepTrunc [8], 2StepFull [8], 2StepTrunc [8].

The FM_{ASIC} also reveals that, among the 16 × 16 designs, 1StepTrunc [8] reaches the best trade-off. However, the FM_{ASIC} of the proposed New_{8_8} signed multiplier is only 5% lower and up to 2.6 times higher than other competitors referenced in Table 2.

4.3. FPGA Implementations

Tables 3 and 4 collect hardware characteristics of 8 × 8 and 16 × 16 approximate multipliers implemented on a Xilinx XC7VX330T FPGA device. Data related to competitors are extracted from the original papers.

Table 3. Hardware Characteristics and accuracy of the 8 × 8 FPGA Designs.

Configuration	#LUTs	D(ns)	E(pJ)	AE	ER (%)	MRED
BA [15]	37	3.41	4.22	85.01	90.56	0.091
Trunc [1] [15]	43	2.15	3.06	149.78	93	0.121
S2 [15]	86	4.89	7.42	118.875	34.19	0.0223
CA [16]	57	3.13	4.73	54.19	8.36	0.0029
CC [16]	56	1.98	3.55	1592.26	80.46	0.13
S1 [17]	92	4.99	7.1	1842.44	86.46	0.362
S3 [18]	81	5.19	7.41	101.94	8.42	0.0121
S5 [19]	110	4.43	9.75	127.11	84.43	0.049
New_{4_4}	82	2.4	7.2	0.0664	89.69	1.5×10^{-4}
New_{2_6}	68	2.13	3.8	0.324	91.35	2.1×10^{-3}

[1] The two LSBs of each input operand are truncated [15].

Table 4. Hardware Characteristics and accuracy of the 16 × 16 FPGA Designs.

Configuration	#LUTs	D (ns)	E (pJ)	MRED	MED	NMED
AxBM1 [14]	194	3.68	18.03	5.0×10^{-4}	9233.62	8.6×10^{-6}
AxBM2 [14]	161	3.45	14.21	3.0×10^{-4}	7623.1	7.1×10^{-6}
New_{11_11}	183	3.03	15.15	4.0×10^{-3}	13,194.9	1.2×10^{-5}

Table 3 shows that the circuits BA and Trunc [15] achieve the lowest resource requirements and energy dissipation, respectively. Conversely, CC [16] reaches the highest speed performance. However, the above architectures are characterized by MRED values quite higher than those achieved by the multipliers designed using the strategy here proposed. Indeed, the circuit New_{4_4} achieves the lowest MRED. Results in Table 3 show that New_{4_4} and New_{2_6} architectures achieve the best energy-quality-delay trade-off, significantly overcoming their counterparts.

Table 4 compares a 16 × 16 architecture based on the proposed approach to the competitors AxBM1 and AxBM2 [14], and it reports the MRED, the MED, and the NMED because those metrics are used in [14]. It can be noted that the multipliers AxBM1 and AxBM2 achieve better energy-quality-delay trade-offs. However, such a result is obtained by adopting specific and strictly platform-dependent LUT-level optimizations, which prevent exploiting the AxBM1 and AxBM2 within ASIC designs as efficiently. Even without exploiting any specific optimization, the New_{11_11} architecture is ~12% faster than [14] and reaches a more than acceptable energy-quality behavior. As a final remark, it is worth noting that none of the competitors evaluated in Tables 1–4 have the ability to perform well by using both ASIC and FPGA platforms.

In order to show how the operands word length and the adopted configuration affect the behavior of the novel multipliers, further implementations have been characterized. All the obtained results are summarized in Tables 5 and 6, where the results presented in Tables 3 and 4 are also reported, to provide a clearer picture. The former collects the achieved accuracy, whereas the latter reports the hardware characteristics in comparison with the competitors [15–18] and the accurate IP cores.

Table 5. Accuracy of the novel multipliers at various operands word lengths.

Multiplier	Configuration	AE	MRED	NMED	NoEB
8 × 8	New_{4_4}	0.0664	1.50×10^{-4}	0.009	8.343
	New_{5_3}	0.476	5.70×10^{-4}	0.0127	7.78
	New_{2_6}	0.324	2.10×10^{-3}	0.024	6.79
12 × 12	New_{8_8}	82.556	2.50×10^{-3}	0.00943	8.306
16 × 16	New_{8_8}	8867.18	8.38×10^{-3}	1.53×10^{-5}	10.118
	New_{11_11}	82,031.17	3.97×10^{-3}	1.22×10^{-5}	7.1

Table 6. Hardware characteristics obtained with the VIRTEX 7 Device.

n × m	Configuration	#LUTs	D (ns)	E (pJ)
8 × 8	New_{4_4}	82	2.4	7.2
	New_{5_3}	78	2.21	4.42
12 × 12	New_{8_8}	177	2.8	11.2
	BA [15]	79	5.3	10.17
	Trunc [15]	102	3.52	8.97
	S1 [17]	228	6.98	20.8
	S2 [15]	189	6.37	20.77
	S3 [18]	185	7.11	20.39
	Accurate IP core	162	4.2	19.79
16 × 16	New_{8_8}	270	3.4	20.4
	New_{11_11}	183	3.03	15.15
	BA [15]	144	7.64	21.15
	Trunc [15]	214	4.1	14.76
	S1 [17]	228	6.98	20.8
	S2 [15]	330	6.59	20.39
	S3 [18]	296	7.33	18.58
	CA [16]	245	4.98	26.5
	CC [16]	240	2.38	16.16
	Accurate IP core	286	4.27	34.35
24 × 24	New_{16_16}	565	3.6	28.8
	BA [15]	301	10.99	48.26
	Trunc [15]	514	6.07	53.97
	S1 [17]	895	9.43	101.63
	S2 [15]	777	9.45	97.48
	S3 [18]	697	9.69	92.35
	Accurate IP core	627	5.98	77.25
32 × 32	New_{16_16}	937	4.95	64.35
	CA [16]	1013	6.98	58.84
	CC [16]	992	3.02	33.04
	Accurate IP core	1037	7.23	151.83

From Table 6, it is pretty evident that the LUT-optimized approximate design BA [15] is the cheapest one and often dissipates less energy than competitors. Conversely, at least one of the configurations examined for the new multiplier performs better than S1 [17], S2 [15], and S3 [18]. In fact, the amounts of LUTs required by the newly proposed 8 × 8, 12 × 12, 16 × 16, and 24 × 24 implementations are up to ~38%, ~22%, ~54%, and ~37% lower, respectively. It is also worth noting that the designs S1, S2, and S3 always utilize more LUTs than the accurate design. Furthermore, it can be seen that the amount of LUTs required by the designs CA and CC [16] rapidly grows with the operands bit-width, thus becoming higher than the novel multipliers starting from 16 × 16 implementations.

Table 6 also shows that CC implementation always leads to the lowest energy consumption. However, it must be taken into account that both the architectures CA and CC operate on unsigned inputs [16]. The energy improvement achieved by the proposed

approximation strategy over the accurate counterpart increases with the operands bit-width: the ~34% energy saving reached in the case of 8 × 8 multipliers, grows to ~43%, ~56%, ~63%, and ~70% for the 12 × 12, 16 × 16, 24 × 24, and 32 × 32 implementations, respectively. The novel designs also exhibit an appreciable energy savings ranging from ~20% to 72%, with respect to the competitors S1, S2, S3, BA, and CA. Conversely, their energy consumption is comparable to AxBM1 and AxBM2 [14] (see also Table 4).

5. Case Study: Image Processing Applications

As an example of applications, the proposed approximate multipliers have been exploited in the realization of two image processing sub-systems, commonly adopted as benchmarks in similar works [7–15,17]: the 2D filtering and the edge detector. While the former convolves the input image with a single kernel, the latter performs convolutions with two kernels that compute horizontal and vertical gradients of the input image. Both the sub-systems are based on the 8 × 8 New_{2_6} multiplier and receive the kerne values as external inputs. Therefore, they can support different edge detectors and filters. However, for purposes of comparison with previous works, the Sobel operator and the 2D Gaussian smoothing filters have been referenced. The energy consumption of complete systems was analyzed with 100,000 random vectors at the maximum toggle rates. Whereas, the accuracy was examined using images from the USC-SIPI dataset [31] as test benches. Accuracy results discussed in the following are calculated by averaging those obtained for all the 256 × 256 and 512 × 512 images available in [31]. Sample images reported in Figure 7 show that the new approximate multipliers work well in both the referred image processing applications.

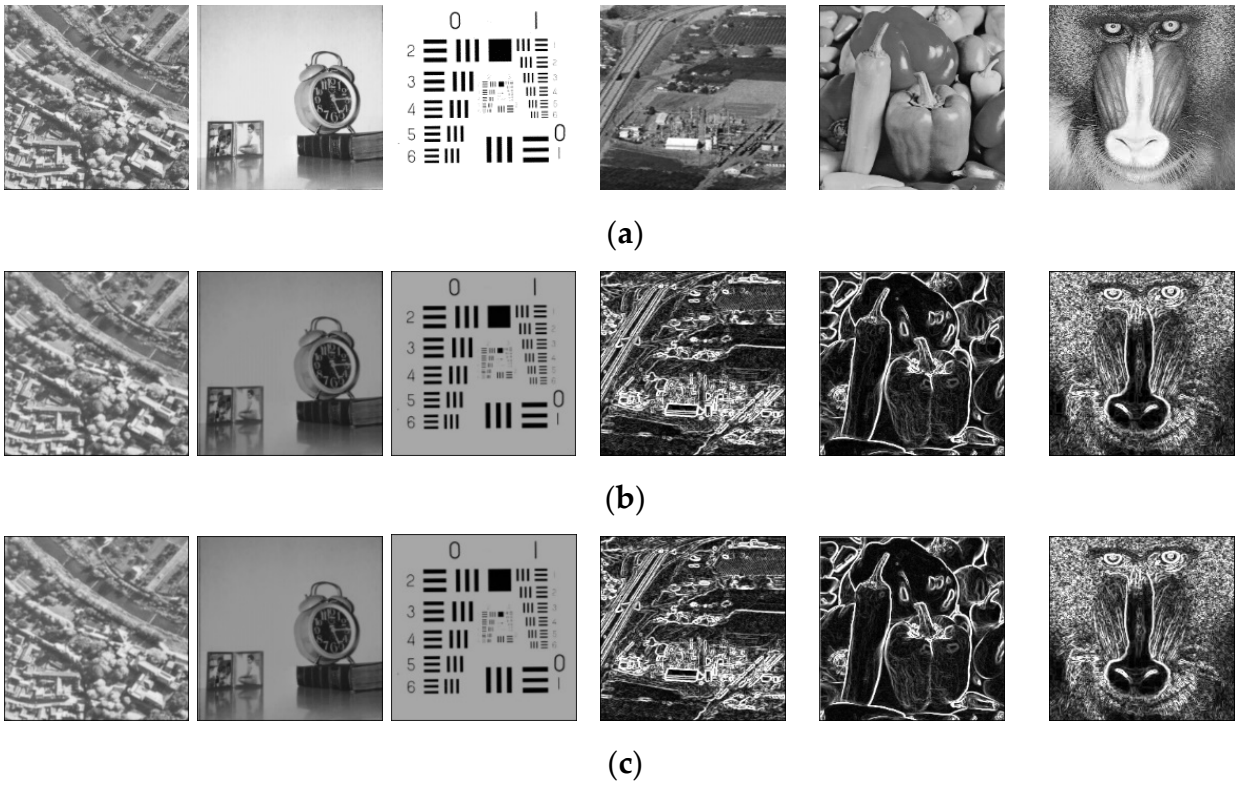


Figure 7. Sample images: (**a**) original; (**b**) precise filtering; (**c**) approximate filtering.

It is worth pointing out that, in order to analyze the behavior of the designed sub-systems on different FPGA devices, they have been implemented within Xilinx VIRTEX 7 XC7VX485 and Altera CYCLONE 006YE144A7G chips. Table 7 summarizes the hardware characteristics of the implemented sub-systems at different filter sizes. Moreover, it reports the accuracy achieved when the 2D Gaussian Smoothing Filtering is performed and averaged over the processed testbench images. The Peak Signal Noise to Ratio (PSNR) and the Structural Similarity (SSIM) [26] quality metrics have been selected for purposes of comparison with the approximate filters presented in [15].

Table 7. Comparison of 2D Filters on FPGA Devices.

Multiplier Used	Device	Filter Size	Hardware Characteristics				Quality Metrics	
			#LUT/LE	#FFs	D (ns)	E (pJ)	PSNR	SSIM
8×8 New_{2_6}	VIRTEX 7 XC7VX485	3×3	664	164	4.9	98	52.9	1
		5×5	1935	420	5.87	299.3	54.68	1
		7×7	3781	804	6.8	632.4	60.75	1
	CYCLONE10LP 006YE144A7G	3×3	1118	164	14.4	89.57	52.9	1
8×8 BA [15]	VIRTEX 7 XC7VX485	3×3	398	163	6.3	100.8	50.5	0.98
		5×5	1221	419	6.8	326.5	51.85	0.99
		7×7	2411	803	7.5	682.5	52.36	0.99
8×8Accurate IP	VIRTEX 7 XC7VX485	3×3	722	164	5.5	143	∞	1
		5×5	2025	420	6.9	414	∞	1
		7×7	3976	804	8.6	842.8	∞	1
	CYCLONE10LP 006YE144A7G	3×3	1010	164	14	204.7	∞	1

To provide a complete overview, the behavior of LUT-optimized accurate filters, referenced as the baselines and employing the 8×8 accurate IP core multiplier, is also shown. It is worth pointing out that, in terms of SSIM, the approximate filters based on the novel multipliers achieve the same behavior as the accurate implementations. Moreover, when compared to the filter based on the BA multiplier presented in [15], in terms of PSNR, the novel design exhibits an improvement ranging from ~4.8% to ~16%, achieved for the 3×3 and the 7×7 filter size, respectively. Xilinx VIRTEX 7 implementations exhibit an energy reduction with respect to the baseline that, depending on the filter size, varies between ~25% and ~32%, with an energy improvement up to ~8.5% achieved in comparison with [15]. The Altera CYCLONE implementation achieves a ~56% energy reduction over the baseline. Table 7 also shows that the approximate filters designed as proposed here are up to ~21% and ~22% faster than the baselines and the counterparts characterized in [15], respectively. Finally, it must be noted that, since the architectures proposed in [15] exploit FPGA-specific optimizations, they achieve appreciable reductions in terms of utilized logic resources, with respect to the accurate IP-based implementations.

Table 8 compares several 3×3 Sobel edge detectors based on 8×8 approximate multipliers. The energy gains and the edge detection accuracies achieved with respect to the precise baselines are reported. While AxBM2 [14] achieves the highest energy gain and the architecture in [10] obtains the best accuracy level, the proposed strategy leads to an appreciable trade-off, even though it does not exploit any specific and strictly platform-dependent LUT-level optimization. On the other hand, this is the reason for which, in contrast to the competitors, the approximation approach proposed here can be efficiently employed also in ASIC designs, as clearly visible in Table 9. The latter reports percentage gains in terms of area, delay, and energy, achieved over the accurate baselines, and SSIM degradations attainable by several approximation techniques in Gaussian smoothing filtering. It can be observed that the proposed method significantly outperforms the competitors. It is worth highlighting that the approximation strategy presented here maintains the same

accuracy achieved by the accurate baseline. Conversely, the approach exploited in [8] reduces the SSIM by up to 8%.

Table 8. Comparison of Approximate 3 × 3 Edge Detectors.

	Energy Gain	Edge Detected
New	25.53%	99.21%
AxBM1 [14]	21.25%	97.45%
AxBM2 [14]	26.41%	98.45%
BA [15]	22.47%	98.96%
[10]	18.6%	99.23%
[11]	16.55%	98.70%

Table 9. Comparison of 3 × 3 2D Filters on ASIC.

Multiplier Used	Technology	Area Gain	Delay Gain	Energy Gain	SSIM Loss
1StepFull [8]	TSMC 40 nm	-	-	36%	1%
1StepTrunc [8]		-	-	63%	1.3%
2StepFull [8]		-	-	44%	7.5%
2StepTrunc [8]		-	-	76%	8%
New_{2_6}	TSMC 40 nm 1.1 V	8.85%	0%	82.6%	0%
	ST UTBB- FDSOI 28 nm 1 V	5.1%	−4%	50.6%	0%

Additional tests demonstrated that the approximate multipliers designed as proposed here work well also when employed in 5 × 5 and 7 × 7 Gaussian smoothing filters. In fact, energy gains close to 25% and 80% are still achieved by FPGA and ASIC designs, respectively, with PSNR higher than 50 dB and without causing SSIM degradation.

6. Conclusions

This paper presented a novel approach to designing energy-efficient platform-independent approximate multipliers. In fact, even without exploiting specific low-level optimizations, the proposed approximate approach leads to efficient designs in both ASIC and FPGAs. This is a quite remarkable advantage over existing counterparts, given that, even though any design described using VHDL can be synthesized and implemented onto any realization platform, the energy-delay trade-off achieved is typically quite far from that reached by counterparts natively optimized for a specific platform.

The novel strategy directly approximates the operands received as inputs. In order to do this in a smart way, thus limiting the overall accuracy loss, an innovative encoding logic has been introduced. The approximation method here proposed has been applied to several signed multipliers with different operands word lengths. A thorough analysis performed in terms of accuracy metrics and hardware characteristics demonstrated that the novel approximation strategy achieves remarkable energy savings in both FPGA-based and ASIC implementations. The ASIC designs have shown that the novel approximation technique achieves the best energy improvement over the accurate baseline and overcomes several competitors in terms of NoEB.

The proposed technique has been applied to design approximate 2D filters and edge detectors. When implemented onto FPGA devices, the novel approximate filters exhibit an energy consumption up to ~32% lower than the optimized baselines. Moreover, the achieved energy-delay product is more than 24% lower than its state-of-the-art counterparts [15]. Even better behaviors have been observed for the ASIC designs that consume more than 80% less energy than the baselines without affecting the accuracy achieved in terms of SSIM.

Author Contributions: Conceptualization, S.P., P.C. and F.S.; methodology, S.P., P.C. and F.F.; software, S.P.; validation, S.P., P.C. and F.S.; formal analysis, S.P. and P.C.; writing—original draft preparation, S.P. and P.C.; writing—review and editing, S.P., P.C., F.S. and F.F.; funding acquisition, S.P. and P.C. All authors have read and agreed to the published version of the manuscript.

Funding: The activity of F.S. was funded by Ministero dell'Università e della Ricerca (PON Ricerca & Innovazione—Grant 1062_R24_INNOVAZIONE).

Data Availability Statement: Not applicable.

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

References

1. Alioto, M. Ultra-Low Power VLSI Circuit Design Demystified and Explained: A Tutorial. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2012**, *59*, 3–29. [CrossRef]
2. Jiang, H.; Santiago, F.J.H.; Mo, H.; Liu, L.; Han, J. Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications. *Proc. IEEE* **2020**, *108*, 2108–2135. [CrossRef]
3. Chang, C.-H.; Satzoda, R.K. A low error and high performance multiplexer-based truncated multiplier. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2010**, *18*, 1767–1771. [CrossRef]
4. Frustaci, F.; Perri, S.; Corsonello, P.; Alioto, M. Approximate Multipliers with Dynamic Truncation for Energy Reduction via Graceful Quality Degradation. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**, *67*, 3427–3431. [CrossRef]
5. Hashemi, S.; Bahar, R.I.; Reda, S. DRUM: A Dynamic Range Unbiased Multiplier for approximate applications. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, USA, 2–6 November 2015.
6. 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]
7. 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]
8. Esposito, D.; Strollo, A.G.M.; Napoli, E.; De Caro, D. Approximate Multipliers Based on New Approximate Compressors. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2018**, *65*, 4169–4182. [CrossRef]
9. Strollo, A.G.M.; Napoli, E.; De Caro, D.; Petra, N.; Di Meo, G. 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]
10. Venkatachalam, S.; Adams, E.; Lee, H.J.; Ko, S.-B. Design and analysis of area and power efficient approximate booth multipliers. *IEEE Trans. Comput.* **2019**, *68*, 1697–1703. [CrossRef]
11. Waris, H.; Wang, C.; Liu, W. Hybrid low radix encoding-based approximate booth multipliers. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**, *67*, 3367–3371. [CrossRef]
12. Kulkarni, P.; Gupta, P.; Ercegovac, M. Trading accuracy for power with an underdesigned multiplier architecture. In Proceedings of the 24th Internatioal Conference on VLSI Design, Chennai, India, 2–7 January 2011.
13. 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.
14. Waris, H.; Wang, C.; Liu, W.; Lombardi, F. AxBMs: Approximate Radix-8 Booth Multipliers for High-Performance FPGA-Based Accelerators. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 1566–1570. [CrossRef]
15. Ullah, S.; Schmidl, H.; Sahoo, S.S.; Rehman, S.; Kumar, A. Area-Optimized Accurate and Approximate Softcore Signed Multiplier Architecture. *IEEE Trans. Comput.* **2021**, *70*, 384–392. [CrossRef]
16. Ullah, S.; Rehman, S.; Shafique, M.; Kumar, A. High-Performance Accurate and Approximate Multipliers for FPGA-based Hardware Accelerators. *IEEE Trans. Comput. -Aided Des. Integr. Circuits Syst.* **2022**, *41*, 211–224. [CrossRef]
17. Rehman, S.; El-Harouni, W.; Shafique, M.; Kumar, A.; Henkel, J. Architectural-space exploration of approximate multipliers. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, USA, 7–10 November 2016.
18. Ullah, S.; Rehman, S.; Prabakaran, B.S.; Kriebel, F.; Hanif, M.A.; Shafique, M.; Kumar, A. Area-optimized low-latency approximate multipliers for FPGA-based hardware accelerators. In Proceedings of the 55th Annual Design Automation Conference, San Francisco, CA, USA, 24–28 June 2018.
19. Mrazek, V.; Hrbacek, R.; Vasicek, Z.; Sekanina, L. EvoApprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, Switzerland, 27–31 March 2017.
20. Imed, B.D. Implementation of a Fuel Estimation Algorithm Using Approximated Computing. *J. Low Power Electron. Appl.* **2022**, *12*, 17.
21. Preatto, S.; Giannini, A.; Valente, L.; Masera, G.; Martina, M. Optimized VLSI Architecture of HEVC Fractional Pixel Interpolators with Approximate Computing. *J. Low Power Electron. Appl.* **2020**, *10*, 24. [CrossRef]

22. Coelho, D.F.G.; Cintra, R.J.; Bayer, F.M.; Kulasekera, S.; Madanayake, A.; Martinez, P.; Silveira, T.L.T.; Oliveira, R.S.; Dimitrov, V.S. Low-Complexity Loeffler DCT Approximations for Image and Video Coding. *J. Low Power Electron. Appl.* **2018**, *8*, 46. [CrossRef]
23. Balasubramanian, P.; Maskell, D.L. Hardware Optimized and Error Reduced Approximate Adder. *Electronics* **2019**, *8*, 1212. [CrossRef]
24. Tastan, I.; Karaca, M.; Yurdakul, A. Approximate CPU Design for IoT End-Devices with Learning Capabilities. *Electronics* **2020**, *9*, 125. [CrossRef]
25. Perri, S.; Spagnolo, F.; Frustaci, F.; Corsonello, P. Efficient Approximate Adders for FPGA-Based Data-Paths. *Electronics* **2020**, *9*, 1529. [CrossRef]
26. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef] [PubMed]
27. Perri, S.; Corsonello, P.; Cocorullo, G. Efficient recursive multiply architecture for FPGAs. *Electron. Lett.* **2005**, *41*, 1314–1316. [CrossRef]
28. 7 Series FPGAs Configurable Logic Block User Guide, UG474 (v1.8) September 27. 2016. Available online: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf (accessed on 22 July 2022).
29. Intel® Stratix® 10 Logic Array Blocksand Adaptive Logic Modules User Guide, UG-S10LA, April 4. 2020. Available online: https://www.intel.com/content/dam/www/programmable/us/en/pdf/literature/hb/stratix-10/ug-s10-lab.pdf (accessed on 22 July 2022).
30. Liang, J.; Han, J.; Lombardi, F. New Metrics for the Reliability of Approximate and Probabilistic Adders. *IEEE Trans. Comput.* **2013**, *62*, 1760–1771. [CrossRef]
31. SIPI Image Database. 2019. Available online: http://sipi.usc.edu/database/database.php?volume=misc (accessed on 22 July 2022).