Cooperative Arithmetic-Aware Approximation Techniques for Energy-Efficient Multipliers

Vasileios Leon, Konstantinos Asimakopoulos, Sotirios Xydis, Dimitrios Soudris, Kiamal Pekmestzi
School of Electrical & Computer Engineering, National Technical University of Athens, Greece
{vleon,asimakopoulos,sxydis,dsoudris,pekmes}@microlab.ntua.gr

ABSTRACT

Approximate computing appears as an emerging and promising solution for energy-efficient system designs, exploiting the inherent error-tolerant nature of various applications. In this paper, targeting multiplication circuits, i.e., the energy-hungry counterpart of hardware accelerators, an extensive exploration of the error–energy trade-off, when combining arithmetic-level approximation techniques, is performed for the first time. Arithmetic-aware approximations deliver significant energy reductions, while allowing to control the error values with discipline by setting accordingly a configuration parameter. Inspired from the promising results of prior works with one configuration parameter, we propose 5 hybrid design families for approximate and energy-friendly hardware multipliers, consisting of two independent parameters to tune the approximation levels. Interestingly, the resolution of the state-of-the-art Pareto diagram is improved, giving the flexibility to achieve better energy gains for a specific error constraint imposed by the system. Moreover, we outperform prior works in the field of approximate multipliers by up to 60% energy reduction, and thus, we define the new Pareto front.

KEYWORDS

Approximate Computing, Computer Arithmetic, ASIC, Energy Efficiency, Design Space Exploration

1 INTRODUCTION

Due to the recent failure of Dennard scaling, power consumption and the related energy dissipation have raised as first class concerns in the design of integrated circuits. In addition, the pervasive nature of modern computing systems has led to an increased need for high performance and energy efficiency. Towards this direction, approximate (or inexact) computing is considered as a promising paradigm shift for energy-efficient systems design [6], exploiting the inherent resilience of various applications. This relaxation in the requirements for exactness is evident in several emerging domains, e.g., machine learning, multimedia, etc. [2], favored due to several factors, such as the limited human perception, the probabilistic/statistical calculations, the user's intention to accept results of lower quality, etc.[1]. Thus, error is considered as a commodity that can be traded for significant gains in performance, area, power, and energy [15].

Targeting to take advantage of the error tolerance, massive research has been reported in the field of approximate computing at various layers of software and hardware [18]. Focusing on the hardware level, approximations can be applied at different design layers of abstraction, i.e., the application, algorithmic, gate and transistor layers [5]. Regarding circuit designs, the main targets are the adders [8] and the multipliers [13], i.e., the core units of hardware accelerators. Extensive research has also been conducted in approximate processors, using neural networks [4], quality programmable vectors [26] and approximate custom instructions [11].

Approximate methods have been extensively applied in the design of inexact circuits, due to delivering lower dynamic and leakage power consumption. Circuit approximations can be introduced via voltage over-scaling (VOS) [21], over clocking (OC) [10], and logic simplification [8, 9, 12, 13, 17]. In this paper, we focus on approximations applied in arithmetic circuits, and specifically the hardware multipliers, the most energy-hungry components of accelerators involving computationally intensive kernels (DSP, neural networks, etc.). The majority of existing works on inexact multipliers explores approximations either on the partial product generation [12, 13, 27] or the partial product accumulation [16, 19, 22]. These approximation targets are synergistic and can be applied in collaboration, increasing the total energy/area savings [9, 17].

Past research activities on approximate multipliers have shown that the direct application of inexact adders in the partial product accumulation is not very efficient in terms of accuracy, hardware complexity and other performance metrics [19]. On the other hand, approximations on the partial product generation deliver simpler partial product arrays, and thus, there is significant reduction in the critical paths and the accumulation complexity [13]. Although vertical cross-layer approximation techniques have recently emerged [25, 28] showing promising results, the full potential of horizontal, i.e., within the same level of design abstraction, and cooperative approximation techniques still remains an open issue for further exploration and exploitation. In this work, we explore for the first time, the efficiency of cooperative arithmetic-level approximate techniques targeting energy-efficient multiplier designs. We focus and analyze cooperative approximation on the arithmetic level, i.e., the algorithmic level of arithmetic design, due to its high impact on both the dynamic and the static power consumption of the underlying arithmetic circuits. Moreover, the implementation delivered at this level can be straightforwardly adopted in a vertical cross-level design approach and further optimized.

More specifically, we introduce 5 design families for inaccurate yet energy-friendly multipliers, that can configure their accuracy with two independent parameters. To further motivate the impact

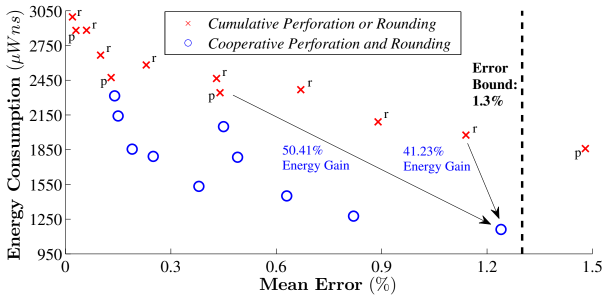


Figure 1: Energy-Error Pareto plot showing the benefits of applying cooperative approximate techniques: (i) increased Pareto resolution (ii) better energy gains w.r.t. an error constraint.

and the potential of the proposed cooperative arithmetic-aware approximation, Fig. 1 shows the Pareto points of *Perforation*, *Rounding*, and *Cooperative Perforation-Rounding*. The blue points label two different perforation configurations, with each one being combined with five rounding configurations. As observed, except for forming the Pareto front, the combination of the two techniques increases the resolution. Moreover, it provides a better circuit in terms of energy for a specific error constraint, i.e., 50.41% and 41.23% energy gains vs. the solutions delivered by *Perforation* and *Rounding*, respectively.

The main contributions of our work are summarized as follows:

- An extensive exploration of the error-energy trade-off, when combining arithmetic-aware approximation techniques, is performed for the first time.
- The resolution of the state-of-the-art error-energy Pareto front is improved, as the approximations are tuned by multiple parameters.
- A rigorous error analysis is attached for each hybrid technique in order to study the error scaling w.r.t. the aggressiveness of the applied approximations.
- A detailed experimental evaluation of industrial strength is provided, showing the constant efficiency of the proposed designs, compared to prior Pareto fronts, in two design scenarios (high performance, ISO-delay).

The rest of this paper is organized as follows. Section 2 includes a brief overview of prior works in the field of approximate multipliers. In Section 3, the proposed approximate multipliers are introduced by describing the applied approximations, and presenting their impact on the accuracy. Section 4 includes the experimental evaluation of our design through comparisons with similar approximate circuits of the literature. Finally, Section 5 concludes this work.

2 PRIOR ART

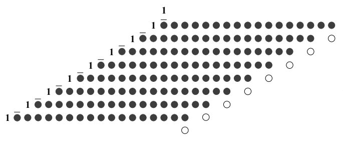
Arithmetic-aware approximations in circuits have been extensively studied in the past, as they deliver significant energy gains at the application/system level. Interestingly, we attempt to categorize the approximate techniques of the literature into the following groups: (i) *Pruning/Elimination*, (ii) *Radix Encoding*, (iii) *Rounding*, (iv) *Dynamic Scaling*. More specifically, we provide a closer look at each category, by describing their basic approximation concept and presenting representative state-of-the-art works.

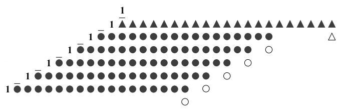
Elimination/Pruning: targeting logic minimization, approximations are introduced by discarding either bits, terms, or nodes of any arithmetic circuit. In [23], the authors presented a methodology and a CAD tool to automatically trade accuracy for area, power and delay savings, by applying gate level pruning to any combinational circuit. This methodology is quite effective especially on arithmetic circuits where there is a notion of bit significance. Moreover, probabilistic pruning has been presented in [14] using a greedy approach to implement approximate circuits. One of the main techniques of elimination/pruning is the partial product perforation, as proposed by Zervakis *et al.* [27]. In this technique, partial products are omitted, and thus, simpler partial product matrices are generated. One possible downside is that error is increased exponentially while more partial products are excluded.

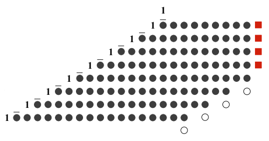
Radix Encoding: A wide range of approximate multiplication circuits is based on inexact radix encodings. All the techniques that have been proposed in the literature lead to less partial product bits or even reductions in the total number of the partial products. Liu *et al.* [17] designed approximate radix-4 encoders by transforming the Karnaugh map of the accurate encoding, and combined them with an inexact compressor. Moving to radix-8 multipliers, Jiang *et al.* [9] employed an approximate adder for producing $\pm 3A$. In [12], the authors tackled the increased circuit complexity of the very high radix-2^k encoding by rounding the high radix values to their nearest power of two, creating simple partial product generators.

Rounding: several approximate multipliers are designed by applying rounding or truncation and correction techniques on the partial products. A representative work is the truncated multiplier proposed in [24], where partial product LSBs are eliminated vertically, and a correction constant is added to reduce the total error. Nevertheless, this correction constant produces a non-zero component. Finally, Zhang *et al.* [29] designed an approximate multiplier by dividing the partial product matrix into two parts: the main part (MP) that is accurately accumulated, and the truncated part (TP), that is further partitioned into TP_{major} and TP_{minor}, with the latter being produced through a probabilistic approach.

Dynamic Scaling: in this category, the aggressiveness of the approximations is defined either by the system or according to the inputs. In [20], the authors statically capture and multiply m-bit segments, either starting from the MSB or ending at the LSB, achieving scalable accuracy. A limitation of the above technique is the difficulty in scaling to higher inputs widths, and thus, its benefits are reduced as the input size grows. Based on the varying bit significance, Hashemi *et al.* [7] proposed a more fine-grained input segmentation, using leading one detector circuits to locate the most significant '1' in each operand. However, this dynamic segmentation implies extra circuits for the signed multiplications, increasing the total hardware overhead. Recently, in [13] the authors proposed a hybrid approximate multiplier by combining two orthogonal techniques: the partial product perforation [27] and a new partial product rounding. The approximation is tuned on the runtime by multiplexing the input bits, and thus, perforation and rounding are configured according to the desired accuracy.


(a) Accurate


(b) *High Radix*


(c) *Rounding*

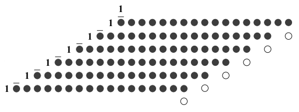

(d) *Perforation*

Figure 2: Partial product matrices of 16-bit multipliers: (a) accurate design and (b), (c), (d) inexact designs using the state-of-the-art approximate techniques.

3 COOPERATIVE ARITHMETIC-AWARE APPROXIMATION TECHNIQUES

Before presenting the proposed inexact multipliers, we make a brief introduction in the examining state-of-the-art approximation techniques. For the rest of the discussion, we consider a $n \times n$ multiplier. As a baseline, we consider the accurate radix-4 multiplier of Fig. 2a.

High Radix [12]: considering a configuration parameter $k \geq 4$, that is an even number, the multiplicand B is partitioned in two segments: the MSB part that consists of $n-k$ bits and the LSB part of k bits. The MSB part is encoded using the radix-4 encoding, whereas the LSB part is encoded with the approximate radix-2^k encoding. In Fig. 2b, the partial product matrix of *High Radix* for $k = 6$ is presented. The least significant partial product (black triangles) is approximately produced from the radix-64 encoding, substituting two accurate radix-4 partial products.

Rounding [13], [3]: this technique discards bits from the partial products and introduces correction terms in order to compensate the losing accuracy. There exist two types of rounding: the *Symmetric Rounding*, which eliminates the same number of bits from each partial product, and the *Asymmetric Rounding*, where less bits are eliminated from the most significant partial products. In this work, we focus on *Asymmetric Rounding*, but the same exploration can be performed using *Symmetric Rounding*. Fig. 2c illustrates the partial product matrix of *Asymmetric Rounding*, with rounding performed at the $t = 8$ column of the matrix. The red squares are the corrections terms

Perforation [27]: this technique eliminates p successive partial products starting from the least significant ones. In Fig. 2d, the partial product matrix of *Perforation* for $p = 2$ is presented.

High Radix and *Rounding* are considered approximations in the logic level of design abstraction, while, *Perforation* is an algorithmic-level approximation. Thereby, we designed approximate multipliers using all the possible combinations of the logic-level techniques (*High Radix & Rounding* and *High Radix & High Radix* – *Rounding & Rounding* is not viable), and we further employed *Perforation* in these designs. Moreover, we explored the viability of combining each one of the logic-level techniques with *Perforation*. Interestingly, we combined only *Rounding* and *Perforation*, as *High Radix & Perforation* is equivalent to *Perforation*, and thus, it can be safely excluded for further examination.

3.1 High Radix & High Radix

In this technique, B is encoded using radix-4 (MSB part) and approximate radix-2^k (LSB part) as in *High Radix* [12], whereas A is divided in two parts, and only the LSB is encoded with approximate radix-2^m. The following equations define the encoding of B and A:

$$B = -2^{n-1} b_{n-1} + \sum_{i=0}^{n-2} 2^i b_i = B_1 + y_0^{R2^k} \tag{1}$$

$$\text{where } B_1 = \sum_{j=k/2}^{n/2-1} 4^j y_j^{R4}, \; y_j^{R4} = -2b_{2j+1} + b_{2j} + b_{2j-1} \tag{2}$$

$$\text{and } y_0^{R2^k} = -2^{k-1} b_{k-1} + 2^{k-2} b_{k-2} + \cdots + b_0 \tag{3}$$

$$A = -2^{n-1} a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i = A_1 + x_0^{R2^m} \tag{4}$$

$$\text{where } A_1 = -2^{n-1} a_{n-1} + \sum_{j=m}^{n-2} 2^i a_i + 2^{m-1} a_{m-1} \tag{5}$$

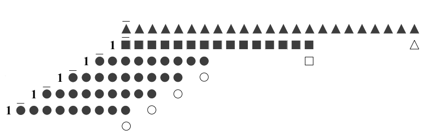
$$\text{and } x_0^{R2^m} = -2^{m-1} a_{m-1} + 2^{m-2} a_{m-2} + \cdots + a_0 \tag{6}$$

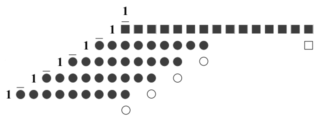
Regarding B, the MSB part is encoded with $(n-k)/2$ digits $y_j^{R4} \in \{0, \pm1, \pm2\}$, while the LSB part is encoded with $y_0^{R2^k} \in \{0, \pm1, \pm2, \pm3, \ldots, \pm(2^{k-1}-1), -2^{k-1}\}$. Similarly, the LSB part of A is encoded using $x_0^{R2^m}$. Considering (1) and (4), the multiplication $A \times B$ can now be performed as $A_1B_1 + B_1 x_0^{R2^m} + A y_0^{R2^k}$.

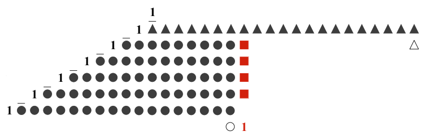
Due to the increased logic complexity of calculating the radix values of $y_0^{R2^k}$, all the values that are not power of two and the $k-4$ smallest powers of two are rounded to their nearest of the four largest powers of two or 0 [12]. Thereby, the LSB part of B is approximately encoded with $\hat{y}_0^{R2^k} \in \{0, \pm2^{k-4}, \pm2^{k-3}, \pm2^{k-2}, \pm2^{k-1}\}$. Similarly, we encode the LSB part of A, using $\hat{x}_0^{R2^m}$. Therefore, the multiplication is performed approximately as follows:

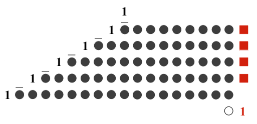
$$A \times B|_{k,m} = A_1 B_1 + B_1 \hat{x}_0^{R2^m} + A \hat{y}_0^{R2^k} \tag{7}$$

The partial product matrix of the approximate multiplier that uses double high radix encoding (DRAD) is shown in Fig. 3a. The least significant partial product (triangles) is $A\hat{y}_0^{R2^k}$, while the next one (squares) is $B_1 \hat{x}_0^{R2^m}$. The rest partial products denote the accurate part A_1B_1. Fig. 4a presents how the Mean Relative Error Distance (MRED) of DRAD is affected by the parameters k, m, when using uniform distribution as input. The derived results show that, even though the error range is small ([0.15%, 1.65%]), it increases rapidly creating blank error segments.


(a) DRAD$|_{8,8}$


(b) DRADP$|_{8,8}$


(c) RADR$|_{6,8}$


(d) RADRP$|_{3,8}$

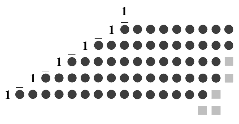
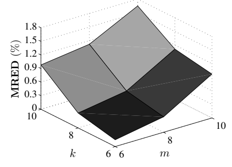
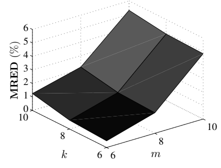

(e) ROUP$|_{3,10}$

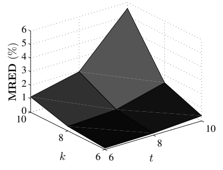
Figure 3: Proposed partial product matrices of 16-bit inexact multipliers using cooperative approximation techniques: (a) *High Radix & High Radix*, (b) *High Radix & High Radix with Perforation*, (c) *High Radix & Rounding*, (e) *High Radix & Rounding with Perforation*, (e) *Rounding with Perforation.*



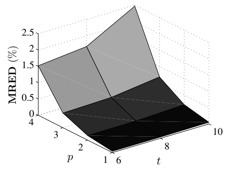
(a) DRAD$|_{k,m}$



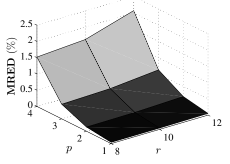
(b) DRADP$|_{k,m}$



(c) RADR$|_{k,t}$



(d) RADRP$|_{p,t}$



(e) ROUP$|_{p,r}$

Figure 4: Mean Relative Error Distance (MRED) variation w.r.t. the approximation configuration parameters for the 16-bit proposed inexact multipliers.

3.2 High Radix & High Radix with Perforation

Targeting to increase the approximations, we employed *Perforation* [27] in the DRAD multiplier, creating DRADP. More explicitly, we eliminated the least significant partial product that is labeled by the factor $A\hat{y}_0^{R2^k}$ in (7), as shown in Fig. 3b. Regarding MRED, this technique delivers higher error values (Fig. 4b), starting from 0.49%, while the rapid error scaling is again observed.

3.3 High Radix & Rounding

Using only the approximate encoding of B, i.e., $\hat{B} = B_1 + \hat{y}_0^{R2^k}$, we combine *High Radix* with *Rounding*, creating the RADR multiplier. More specifically, we truncate the t least significant columns of the partial product matrix generated by the radix-4 encoding (AB_1):

$$A \times B|_{k,t} = AB_1|_t + A\hat{y}_0^{R2^k} \quad (8)$$

Considering the encoding signals one_j and two_j for y_j^{R4}, the correction term [3] presented in (9) is inserted for each partial product. Additionally, an extra "1" is added to the correction terms' column as a form of rounding up, in order to reduce the total error, as shown in Fig. 3c.

$$cor_j = one_j \vee two_j \quad (9)$$

The benefit of this technique is that it smooths the rapid error scaling of RAD [12], by adding multiple values between two consecutive RAD errors, especially for the small k values.

3.4 High Radix & Rounding with Perforation

As in DRAD, we employed *Perforation* [27] in the RADR multiplier, and thus, we eliminated the partial product produced by the high-radix encoding, i.e., $A\hat{y}_0^{R2^k}$. We note that this technique (RADRP) is equivalent to combining *Rounding* [3] with *Perforation* [27], as the removal of the radix-2^k partial product is equal to the perforation of $k/2$ radix-4 partial products. As a result, the approximation configurations of RADRP are p for perforation and t for rounding.

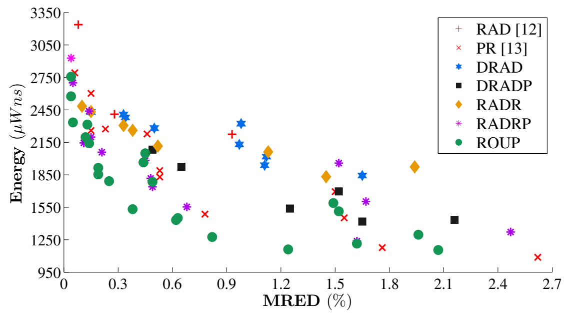
The MRED diagram of RADRP is presented in Fig. 4d. This technique is characterized by small error scaling, i.e., from 0.04% ($p = 1$) to 2.47% ($p = 4$), with several intermediate values.

3.5 Rounding with Perforation

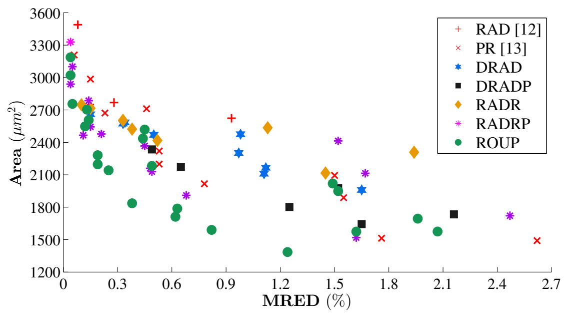
Since the RADRP multiplier practically implements *Rounding* [3] with *Perforation* [27], we chose to combine the rounding technique proposed in [13] with *Perforation*. In this hybrid technique, we perforate the p least significant partial products, and apply rounding to the r_i-bit of the i non-perforated partial product, with $i = p + 1, p + 2, \ldots, n/2$. Specifically, the $r_i - 1$ LSBs of A are truncated, and a_{r_i-1} is added with the remaining part:

$$\hat{A}_{r_i} = \langle a_{n-1}a_{n-2} \cdots a_{r_i} \rangle + a_{r_i-1} \quad (10)$$

As can be observed, A is rounded to a different bit for each partial product (*Asymmetry Rounding*), contrary to [13], where the

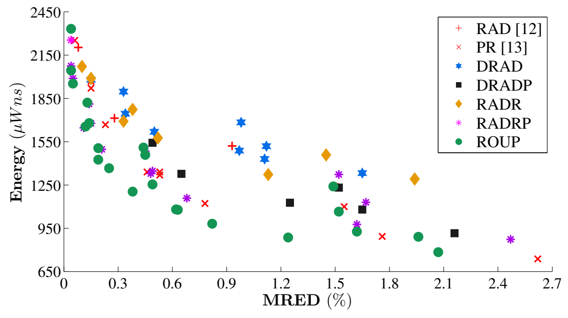


(a) Energy–Error

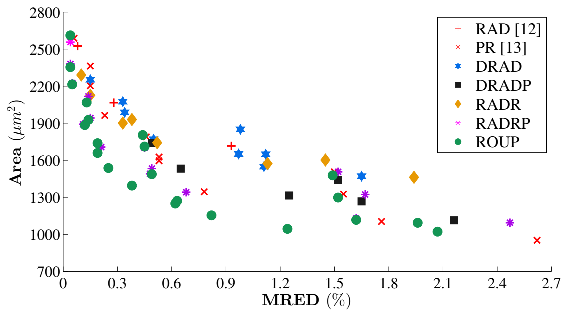


(b) Area–Error

Figure 5: Pareto analysis of the inexact multipliers when synthesized at their critical path delay. RAD [12] and PR [13]: prior state-of-the-art Pareto fronts.



(a) Energy–Error



(b) Area–Error

Figure 6: Pareto analysis of the inexact multipliers when synthesized at the same relaxed clock. RAD [12] and PR [13]: prior state-of-the-art Pareto fronts.

rounding of A is the same for all the partial products. Finally, we note that the addition of a_{r_i-1} is implemented by adding only a XOR gate in the conventional radix-4 correction terms (non-filled circles in Fig. 2a), as explained in the algorithmic optimizations of [13], and thus, the correction term is modified as shown in (11). The correction terms cor_j are labeled with the gray squares in Fig. 3e.

$$cor_j = (sign_j \oplus a_{r_i-1}) \wedge (one_j \vee two_j) \quad (11)$$

The MRED values of ROUP (Fig. 4e) are similar to those of RADRP. However, the maximum error is smaller (2.07%) compared to RADRP (2.47%).

4 EXPERIMENTAL EVALUATION

In this section, we experimentally evaluate the efficiency of the cooperative approximation techniques in terms of energy (power dissipation × delay) and occupied area, by providing comparative results w.r.t. prior state-of-the-art Pareto fronts [12, 13] and similar approximate multipliers of the literature [7, 9, 17, 24].

The following families of state-of-art approximate multipliers are considered for comparison: (i) RAD [12] that applies the high radix encoding, (ii) PR [13] that combines perforation with symmetric rounding, (iii) R8-ABM [9] that employs the radix-8 encoding by approximating the generation of 3A, (iv) R4-ABM [17] that uses inexact radix-4 encoders, (v) TMC [24] that truncates columns of the partial product matrix and inserts correction terms, (vi) DRUM [7] selects a bit segment from each operand starting from the leading '1', and sets the LSBs of the truncated values to '1'.

Several designs from each family have been synthesized to cover an extended range of MRED values, utilized as error bounds. We use industrial-strength tools for our experimentation. The synthesis of the designs is performed with *Synopsys Design Compiler* and the *TSMC* 65-nm standard cell library. *Mentor Graphics QuestaSim* has been employed for simulation purposes, while the power consumption is measured with *Synopsys PrimeTime*. Moreover, synthesis and simulation were performed at 1V, i.e., the nominal supply voltage. All the designs were synthesized and simulated under two different design scenarios: (i) at their circuit-specific critical path delay and (ii) at the same relaxed clock constraint, i.e., 0.8ns, enabling the study of both their effectiveness but also their sustainability under relaxed design margins.

In Fig. 5, we present the energy–error and area–error Pareto plots, when all the multipliers are synthesized at their critical path delay, while Fig. 6 includes the same Pareto analysis at a relaxed clock. The purpose of these schemes is to highlight the most efficient circuits in terms of energy and area under different circumstances, i.e., critical path delay and ISO delay. Regarding the high radix multipliers (DRAD, DRADP and RADR), they increase the resolution of the RAD [12] front, with DRADP constituting an even better

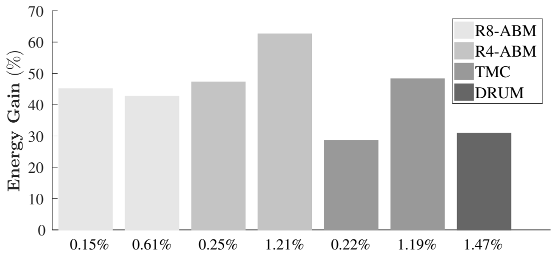


Figure 7: Energy gains of the most efficient proposed multiplier (ROUP) w.r.t. previous state-of-the-art multipliers for the same error constraint (mentioned in X axis).

alternative in most cases. We note that we present only three RAD [12] multipliers ($k = 6, 8, 10$), as for $k \geq 12$ large error values are produced, a negative feature of this technique that is overcome with the cooperative approximation techniques. Furthermore, for small error values PR [13] is not considered the most energy-efficient solution, as several configurations of the cooperative approximation techniques provide better energy results. Overall, as shown in all figures, the Pareto front is formed exclusively by the ROUP multiplier, that delivers the best energy/area–error trade-off, outperforming the previous state-of-the-art fronts [12, 13].

Fig. 7 demonstrates the energy gains of using the most efficient proposed multiplier, i.e., ROUP, by comparing it with the designs of [7, 9, 17, 24]. Specifically, we introduce the same mean error, and we compare the retried energy consumptions. To explore various scenarios, such as the energy consumption at different acceptable error bounds, for the multipliers R8-ABM [9], R4-ABM [17] and TMC [24], we performed comparisons for small errors (i.e., up to 0.25%) and for larger values (i.e., up to 1.21%), while for DRUM [7] we chose the most-efficient proposed configuration. The derived results show that ROUP provides significant gains ranging from 28.51% to 62.55%.

5 CONCLUSION

In this work, we perform an exhaustive exploration of using cooperative arithmetic-level approximation techniques for designing inexact multipliers. Targeting to exploit the full potential of the state-of-the-art approximate methods, we propose 5 hybrid multipliers that increase the flexibility of adjusting the energy–error trade-off. The efficiency of the cooperative techniques is evaluated against an extensive design space exploration of industrial strength, resulting in better energy solutions given an error bound, increased Pareto resolution and a new Pareto front outperforming state-of-the-art approximate techniques.

REFERENCES

[1] S. T. Chakradhar and A. Raghunathan. 2010. Best-Effort Computing: Re-thinking Parallel Software and Hardware. In *Design Automation Conference.* 865–870.

[2] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan. 2013. Analysis and Characterization of Inherent Application Resilience for Approximate Computing. In *Design Automation Conference.* 1–9.

[3] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi. 2004. Design of Low-Error Fixed-Width Modified Booth Multiplier. *IEEE Transactions on Very Large Scale Integration Systems* 12, 5 (May 2004), 522–531.

[4] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger. 2012. Neural Acceleration for General-Purpose Approximate Programs. In *IEEE/ACM International Symposium on Microarchitecture.* 449–460.

[5] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy. 2013. Low-Power Digital Signal Processing Using Approximate Adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 1 (Jan 2013), 124–137.

[6] J. Han and M. Orshansky. 2013. Approximate computing: An emerging paradigm for energy-efficient design. In *IEEE European Test Symposium (ETS).* 1–6.

[7] S. Hashemi, R. I. Bahar, and S. Reda. 2015. DRUM: A Dynamic Range Unbiased Multiplier for Approximate Applications. In *IEEE/ACM International Conference on Computer-Aided Design.* 418–425.

[8] H. Jiang, J. Han, and F. Lombardi. 2015. A Comparative Review and Evaluation of Approximate Adders. In *Great Lakes Symposium on VLSI.* 343–348.

[9] H. Jiang, J. Han, F. Qiao, and F. Lombardi. 2016. Approximate Radix-8 Booth Multipliers for Low-Power and High-Performance Operation. *IEEE Trans. Comput.* 65, 8 (Aug 2016), 2638–2644.

[10] X. Jiao, Y. Jiang, A. Rahimi, and R. K. Gupta. 2017. SLoT: A supervised learning model to predict dynamic timing errors of functional units. In *Design, Automation and Test in Europe.* 1183–1188.

[11] M. Kamal, A. Ghasemazar, A. Afzali-Kusha, and M. Pedram. 2014. Improving efficiency of extensible processors by using approximate custom instructions. In *Design, Automation and Test in Europe.* 1–4.

[12] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi. 2018. Approximate Hybrid High Radix Encoding for Energy-Efficient Inexact Multipliers. *IEEE Transactions on Very Large Scale Integration Systems* 26, 3 (March 2018), 421–430.

[13] V. Leon, G. Zervakis, S. Xydis, D. Soudris, and K. Pekmestzi. 2018. Walking through the Energy-Error Pareto Frontier of Approximate Multipliers. *IEEE Micro* 38, 4 (Jul-Aug 2018), 40–49.

[14] A. Lingamneni, C. Enz, K. Palem, and C. Piguet. 2013. Synthesizing Parsimonious Inexact Circuits Through Probabilistic Design Techniques. *ACM Transactions on Embedded Computing Systems* 12, 2s (May 2013), 93:1–93:26.

[15] A. Lingamneni, C. Enz, K. Palem, and C. Piguet. 2014. Highly Energy-Efficient and Quality-Tunable Inexact FFT Accelerators. In *IEEE Custom Integrated Circuits Conference.* 1–4.

[16] C. Liu, J. Han, and F. Lombardi. 2014. A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery. In *Design, Automation and Test in Europe.* 1–4.

[17] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi. 2017. Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing. *IEEE Trans. Comput.* PP (2017).

[18] S. Mittal. 2016. A Survey of Techniques for Approximate Computing. *Comput. Surveys* 48, 4 (May 2016).

[19] A. Momeni, J. Han, P. Montuschi, and F. Lombardi. 2015. Design and Analysis of Approximate Compressors for Multiplication. *IEEE Trans. Comput.* 64, 4 (Apr 2015), 984–994.

[20] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim. 2015. Energy-Efficient Approximate Multiplication for Digital Signal Processing and Classification Applications. *IEEE Transactions on Very Large Scale Integration Systems* 23, 6 (June 2015), 1180–1184.

[21] R. Ragavan, B. Barrois, C. Killian, and O. Sentieys. 2017. Pushing the limits of voltage over-scaling for error-resilient applications. In *Design, Automation and Test in Europe.* 476–481.

[22] K. M. Reddy, Y. B. N. Kumar, D. Sharma, and M. H. Vasantha. 2015. Low power, high speed error tolerant multiplier using approximate adders. In *International Symposium on VLSI Design and Test.* 1–6.

[23] J. Schlachter, V. Camus, K. V. Palem, and C. Enz. 2017. Design and Applications of Approximate Circuits by Gate-Level Pruning. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25, 5 (May 2017), 1694–1702.

[24] M. J. Schulte and E. E. Swartzlander. 1993. Truncated Multiplication with Correction Constant. In *IEEE Workshop on VLSI Signal Processing.* 388–396.

[25] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel. 2016. Invited - Cross-layer Approximate Computing: From Logic to Architectures. In *Design Automation Conference.* 99:1–99:6.

[26] S. Venkataramani, V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan. 2013. Quality programmable vector processors for approximate computing. In *IEEE/ACM International Symposium on Microarchitecture.* 1–12.

[27] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi. 2016. Design-Efficient Approximate Multiplication Circuits Through Partial Product Perforation. *IEEE Transactions on Very Large Scale Integration Systems* 24, 10 (Oct 2016), 3105–3117.

[28] G. Zervakis, S. Xydis, K. Tsoumanis, D. Soudris, and K. Pekmestzi. 2015. Hybrid Approximate Multiplier Architectures for Improved Power-Accuracy Trade-Offs. In *International Symposium on Low Power Electronics and Design.* 79–84.

[29] Z. Zhang and Y. He. 2018. A Low-Error Energy-Efficient Fixed-Width Booth Multiplier With Sign-Digit-Based Conditional Probability Estimation. *IEEE Transactions on Circuits and Systems II: Express Briefs* (Feb 2018), 236–240.