Automatic generation of input-aware approximate arithmetic circuits

Mario Barbareschi, Salvatore Barone, Alberto Bosio, Bastien Deveautour, Ali Piri, Marcello Traiola

Automatic generation of input-aware approximate arithmetic circuits

Mario Barbareschi[1], Salvatore Barone[2], Alberto Bosio[3], Bastien Deveautour[4], Ali Piri[3], Marcello Traiola[5]

[1]*Department of Electrical Engineering and Information Technologies, University of Naples Federico II, Naples, Italy*
[2]*Department of Computer Science and Technologies, Pegaso Digital University, Naples, Italy*
[3]*Univ Lyon, ECL, INSA Lyon, CNRS, UCBL, CPE Lyon, INL, UMR5270, 69130 Ecully, France*
[4]*Univ Nantes, IETR, Nantes, France*
[5]*Univ Rennes, CNRS, Inria, IRISA - UMR 6074, F-35000 Rennes, France*
[1]{name.surname}@unina.it [2]salvatore.barone@unipegaso.it, [3]{name.surname}@ec-lyon.fr,
[4]bastien.deveautour@univ-nantes.fr, [5]marcello.traiola@inria.fr

***Abstract*—Approximate Computing (AxC) is systematically applied across various abstraction levels to reduce overheads and enhance the performance of applications such as image processing and machine learning. However, AxC does not typically consider the specific workload (i.e., data input) of a given application. For instance, in signal processing applications like filters, some inputs are constants (filter coefficients), which allows for an additional level of approximation by considering the specific input distribution. This method is known as "Input-Aware Approximation" (IAA) and has shown potential advantages in previous studies. Unfortunately, existing input-aware design methodologies lack scalability as they mostly depend on ad-hoc, non-automatic design approaches, limiting their applicability. In this paper, we investigate how the input-aware approximate design approach can be integrated into a systematic, generic, and automatic design flow. We employ state-of-the-art approximation and multi-objective optimization techniques to achieve input-awareness. Our experimental results, focusing on classical signal processing applications like FIR filters, demonstrate that the input-aware approach can provide significant savings in both area and power consumption.**

***Index Terms*—Approximate Computing, Multi-objective Optimization, Approximate Arithmetic Circuits, Input-Aware Approximation, Energy Efficiency.**

I. Introduction

A promising strategy for boosting computing efficiency is Approximate Computing (AxC). This concept encompasses a broad range of techniques that exploit the inherent resilience of applications, leading to enhanced efficiency across all layers of the computing stack. These layers range from basic transistor-level design to software implementations. These techniques can have diverse impacts on both hardware performance and output quality.

The efficiency and performance of numerous time-consuming and resource-intensive applications can be significantly enhanced by exploiting the AxC design paradigm, as demonstrated in scientific literature [1]. AxC leverages various sources of error resiliency in applications by systematically trading off output quality for performance gains or resource savings [2]. It has been successfully adopted in several fields, including signal and image processing, big data analytics, and machine learning, for both software and hardware applications.

Unlocking the full potential of AxC requires addressing several challenges, all stemming from the necessity to consider the target application. These challenges include: (I) identifying portions of the application suitable for approximation, (II) determining appropriate approximation techniques, (III) choosing a relevant error metric for error assessment, and (IV) selecting the approximate configurations that provide optimal trade-offs between the introduced error and the resulting benefits.

Regarding the design of arithmetic operators, a wide range of scientific works employ either manual or automatic methodologies to create approximate components, primarily aiming at hardware resource savings. Although these operators have been successfully utilized in numerous applications, including image processing [3], [4], machine learning, and artificial intelligence [5], [6], customizing components to their specific applications can unlock further approximation opportunities, as recent studies have shown.

In fact, additional savings may be hindered when approximation flows prioritize generalization and multi-use. On the other hand, exploiting the typical workload of a specific application can lead to better results in terms of both error and performance, through a finer-grained approximation of its components.

Unfortunately, most of the input- and application-aware design methodologies proposed in the literature lack scalability, as they primarily rely on manual techniques. This limitation restricts their applicability to small-to-medium-sized applications and results may heavily depend on the designer's expertise. Furthermore, these methodologies often do not incorporate error assessment at the component level, relying instead on manual speculations about the probability of single

input bits being either logic-1 or logic-0. Consequently, they may lead to sub-optimal outcomes or, worse, components that cause unacceptable quality degradation at the application level [7].

In this paper, we introduce an input-aware design strategy within a systematic, generic, and automatic approximate design flow. Instead of focusing on individual input bits, we consider workloads at the component level as a set of n-bit long input assignments, where n represents the number of Primary Inputs (PIs) of the component in question. We define component-level workloads by considering the frequency of occurrence for component-level input values, allowing for precise error assessment at the component level.

We utilize state-of-the-art approximation techniques and Multi-objective Optimization Problem (MOP) with input and application awareness. To evaluate our method, we target a common application in signal processing: a Finite-Impulse Response (FIR) filter. We compare our approach against both manual input-aware and automatic non-input-aware approaches. The results empirically demonstrate that the input-aware approach significantly outperforms the non-input-aware approach, offering up to 20% additional savings in both area and power consumption. Furthermore, the findings highlight that manual methodologies may not achieve adequate results.

The remainder of this paper is structured as follows: Section II discusses related works and relevant contributions from the scientific literature, Section III discusses our automatic input-aware methodology while Section IV presents the experimental setup for the evaluation and comparison purposes, and also discusses results. Finally, Section V draws conclusions.

II. Related Works

Contributions from the scientific literature on designing approximate arithmetic circuits can be divided into two main categories. A significant number of papers focus on manual approaches and methodologies that explore the RTL scheme of circuits or their architecture to introduce approximation [8]–[12]. At the same time, considerable effort has been invested in developing systematic and generic methodologies [13]–[15]. The most recent contributions within this latter category aim to optimize both the error introduced by approximation and the hardware resource requirements simultaneously [3], [16], [17], yielding impressive results across various application fields, including image processing and artificial intelligence applications [4], [18], [19].

Nevertheless, although the aforementioned methodologies have proven to be quite effective, they have been developed without fully considering the final application. Specifically, they do not take advantage of the potential for further approximation that could be possible by relaxing error constraints, which might only need to be met for a subset of the entire input set.

In fact, the authors of [7] demonstrated that the workload of a given application can be leveraged to achieve finer granularity of approximation, resulting in better outcomes in terms of silicon area requirements and energy efficiency, with minimal impact on application accuracy. However, approaches like the one in [7] face scalability issues because they rely on largely manual characterization techniques for workloads and manual manipulations of circuits based on the probability of individual input bits being either logic-1 or logic-0 to introduce approximation.

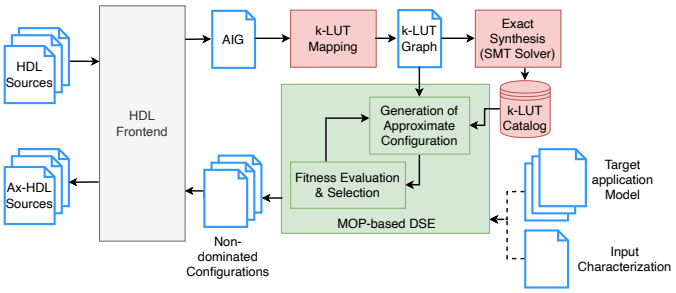


Fig. 1: Overview of the automatic workflow.

From this perspective, we believe our proposal takes a significant step forward. We demonstrate how the application-aware design philosophy can be integrated with a generic and automatic approximate design methodology, combining state-of-the-art approximation and multi-objective optimization techniques with input-awareness.

III. Automatic Input-Aware Approximation

In this section, we discuss how state-of-the-art automatic approximate design methodologies can be leveraged to design approximate arithmetic components tailored to their final applications. Although we reference the approach from [3], the method we discuss is not limited to a specific approximation technique. In our honest opinion, the effort required to integrate application-awareness with other techniques is negligible.

In Figure 1, we outline the overall process. Starting with HDL source code describing the design under study, we initiate with a phase of catalog-based AIG rewriting to generate a k-LUT operator catalog. Next, to manage the exponential growth of potential approximate configurations while maintaining acceptable quality of results and reducing hardware requirements, we use the generated catalog in conjunction with input characterization and the target application to conduct a multi-objective optimization (MOP) based design space exploration (DSE). This process yields non-dominated configurations (i.e., HDL source code of the approximate solutions), which can then be synthesized.

Since the methodology is based on non-trivial local rewriting of And-Inverter Graphs (AIGs) and MOP, in the following we provide the reader with essentials concerning these building blocks. Then we discuss how workload-awareness can improve the approximation flow.

A. Catalog-based AIG-Rewriting

An AIG serves as a structural representation for Boolean networks, based on directed acyclic graphs. In an AIG, nodes fall into two categories: PI nodes, lacking incoming edges and

hence no driving node, and logic-AND nodes, possessing exactly two incoming edges. Additionally, nodes lacking outgoing edges, meaning they don't drive any other node, are termed Primary Outputs (POs). Edges in an AIG denote physical connections between nodes and can be either complemented or not. Conventionally, the polarity of complemented edges is represented by "0". The AIG is considered non-canonical, implying that multiple different AIGs can represent the same Boolean function.

The method described in [3] utilizes the AIG representation of digital circuits to introduce approximation, specifically leveraging k-feasible cuts, or k-cuts.

A k-cut of a node n, referred to as the "root," is a set of at most K nodes within the AIG such that every path from a PI to n traverses at least one node from the set.

In summary, given the AIG of a combinational circuit, the approach outlined in [3] begins by enumerating k-feasible cuts and then introduces approximation by replacing k-cuts with approximate ones. These approximate k-cuts are generated by utilizing Exact Synthesiss (ESs). Given that f represents the Boolean function implemented by a k-cut, ES seeks those k-cuts that implement a function $f' \neq f$ requiring fewer resources compared to f, while still satisfying error constraints expressed in terms of the Hamming distance between f and f'.

The primary challenge lies in finding replacements that result in Pareto-optimal trade-offs between the quality of results and hardware requirements. This involves addressing two major concerns: (I) the exponential growth in the number of approximate configurations with the size of the involved Boolean functions, and (II) the conflicting design goals of preserving result quality while aiming for reduced hardware requirements.

Addressing these issues typically involves utilizing MOP-based Design Space Exploration (DSE), as commonly seen in literature [4], [17]–[19]. Therefore, we approach the aforementioned challenges by optimizing both the error, evaluated using an appropriate metric, and the hardware requirements of the resulting circuits, estimated based on the number of AIG nodes.

B. Bringing input-awareness into the approximation flow

The application and its unique characteristics are crucial in determining component-level workloads, and this can be approached in two different ways.

The first method involves thoroughly profiling the target application, documenting the values of every input at the component level. For example, in a FIR filter, comprehensive profiling would consider both operands of each arithmetic operation, such as the coefficients and signal samples. This approach would generate a reference workload for error measurement at the component level, closely tied to the specific application-level workload used as a reference. However, this method could result in significant quality degradation if the actual data slightly deviates from the reference data during runtime. This issue is particularly pronounced in machine learning applications. For instance, with a Deep Neural Network (DNN), using a test dataset, or a portion of it, as a reference workload during profiling might optimize components like multipliers for that dataset. However, the deployed approximate DNN might perform poorly due to reduced generalization.

Instead of fully profiling the target application as described above, the second approach involves a deeper consideration of the application's peculiarities and characteristics during the component-level workload definition. For example, only a subset of the inputs at the component level may be profiled, while the remaining inputs are randomly selected to enhance overall robustness against slight changes in operating conditions. In the case of a multiplier, we could profile just one of its inputs, using random values from an appropriate set for the other input during error assessment. Unfortunately, this simple approach might lead to sub-optimal results and poor performance, as it does not account for the probability distribution of input values, which is unlikely to be uniform [7].

To address this challenge, we consider the occurrence frequency of values for one of the inputs during profiling and define the component-level workload based on this frequency, as described in Equation (1). This workload is formed by the union of (v, x_v) pairs, which are generated as a Cartesian product between the set $\{v\}$—containing a single profiled input value from V, the set of profiled values for one of the inputs—and vectors x_v from X_v, a set of randomly selected values that satisfy constraints (2).

$$W = \bigcup_{v \in V} \{v\} \times X_v \tag{1}$$

$$\begin{aligned} x \neq y \ \forall x, y \in X_v \\ |X_v| \propto f(v) \ \forall v \in V \\ m_v \leq |X_v| \leq M_v, \ \forall v \in V \end{aligned} \tag{2}$$

In equation (2) elements from X_v are required to be unique, therefore that $(v, x_v) \in W$ are unique. Also $|X_v|$, i.e., the size of X_v, is proportional to the occurrence frequency $f(v)$ for each of the values $v \in V$, and $|X_v|$ is bounded by m_v and M_v.

It is worth also noting that a different X_v set is conveniently defined for each $v \in V$.

The goal of (1) and (2) is first, to obtain high coverage for frequent input assignments while simultaneously guaranteeing minimum coverage for less-frequent assignments. second, to restrict the size of the component-level workload $|W| = \sum_{v \in V} |X_v| \leq \sum_{v \in V} M_v$, and hence to exploit the potential further degree of approximation resulting from the reduced number of input assignments on which application-level error constraints apply.

In addition, we define the *coverage for one single value* ρ_v (3) and the *coverage of the dataset* ρ_W (4) as metrics to help the designer to select values appropriately for $|X_v|$, m_v and M_v.

$$\rho_v = \frac{|X_v|}{|\mathbb{B}^n|} = \frac{1}{2^n}|X_v| \tag{3}$$

$$\rho_w = \frac{|W|}{|\mathbb{B}^{2n}|} = \frac{1}{2^{2n}} \sum_{v \in V} |X_v| \tag{4}$$

However, exploring the optimal degree of coverage w.r.t. (3) and (4) is beyond the scope of this paper, since determining an optimal value for these parameters require minimizing (4) while maximizing (3), which are conflicting goals.

IV. Experimental Results

In our experiments, we leverage on the FIR filter as case study. The filter is defined in Equation (5) where x_j is the j^{th} sample of the input signal, b_i is the i^{th} coefficient of the signal and y_n is the n^{th} sample of the output signal. FIR filters are one of the most common signal processing applications, and furthermore, this case-study allows us to compare easily our automatic input-aware method against that discussed in [20].

$$y_n = \sum_{i=0}^{N} b_i \cdot x_{n-i} \tag{5}$$

The aim of this case-study is the design of a multiplier to perform the $b_i \cdot x_{n-i}$ part of Equation (5), simultaneously minimizing both the error due to approximation and hardware requirements. The FIR filter we considered here have 500 Hz cut-off frequency, 60 dB attenuation in the stop band, and use fixed-point arithmetic. It is a low-pass filter that adopts the Q1.7 fixed-point arithmetic.

1) Designing multipliers using the automatic input-aware approach: According to the proposed automatic input-aware approximation flow, suitable Q1.7 multipliers are designed by locally assessing the error due to approximation. In this case, for each of the mentioned multipliers, we applied the methodology discussed in Section III on Q1.7 multipliers, while resorting to the Mean relative Error Distance (MRED) [21] as error metric. Hardware requirements are always estimated based on the number of AIG nodes, as in [3].

The reference dataset for both the filters have been generated according to (1) and (2). In particular, we profiled the occurrence frequency of filter coefficients, randomly picking the remaining input in the [-0.5, 0.5] range, according to (2).

2) Evaluation and comparison: After the DSE, To evaluate the generated input-aware multipliers, we compared them to those from [20]. To demonstrate the practicality of input awareness, we conducted simulations once using all possible inputs for an 8-bit signed multiplier and once with the coefficients of the FIR filter as described previously.

The error metrics considered were Error Probability, MRED, and Absolute Worst-Case Error (AWCE) defined in [21]. As can be seen in Figure 2, the general-purpose approximate multipliers from [20] exhibit similar behavior across both workloads (i.e., all possible inputs and the filter coefficients), with differences arising mainly due to the lower number of inputs. However, the input-aware multipliers, being specifically generated for this application, may perform worse with inputs other than those they are designed for. Nonetheless, with the given coefficients, the average error probability of the generated input-aware multipliers is 21%, the average MRED is 0.027, and the average AWCE is 0.001, which are significantly lower than those of the other multipliers. It is also worth mentioning that multipliers 9 and 13 are completely accurate with the given workload.

in order to evaluate our approach in terms of circuit characteristics and application level accuracy, we assess the impact of approximation by measuring the Peak Signal-to-Noise Ratio (PSNR) between the output signals produced by the FIRs filter with multipliers from [20] and our input-aware approximate multipliers. Furthermore, we have simulated the multipliers targeting the 45 nm standard cell library [22] to calculate delay, area, and power consumption.

Figure 3 report results for the 8-bit FIR Filters where red triangles denote results from our automatic input-aware approach while the blue stars denotes the multipliers from [20]

Among the multipliers from EvoApprox, such as the accurate mul8s_1KV6, and our input-aware multipliers, which operate accurately for specific inputs (multiplier numbers 9 and 13), the input-aware multipliers demonstrate advantages in terms of reduced area, lower power consumption, and increased speed. Even when comparing approximate multipliers with the same PSNR in the FIR filter, the input-aware multipliers exhibit greater efficiency in power consumption and delay.

V. Conclusion

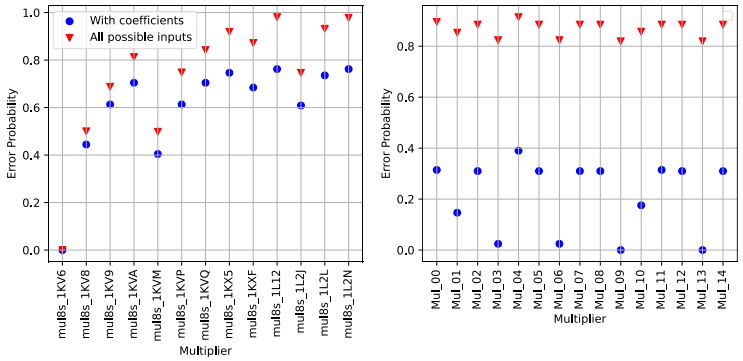
In this paper, we explored the integration of the input-aware design philosophy into a systematic, generic, and automated approximate design flow. Specifically, we demonstrated how component-level workloads can be automatically derived from application-level workloads, enabling the design of input-aware approximate components that minimize both application-level error and hardware resource usage. This process leverages state-of-the-art approximation techniques and multi-objective optimization (MOP) while performing component-level error assessments. To evaluate our method and benchmark it against current methodologies, we focused on a common signal processing application: FIR filters. Our empirical results show that, given the specific workload the circuit will handle, input-aware circuits generated by our automatic method outperform those created by non-input-aware approaches.

References

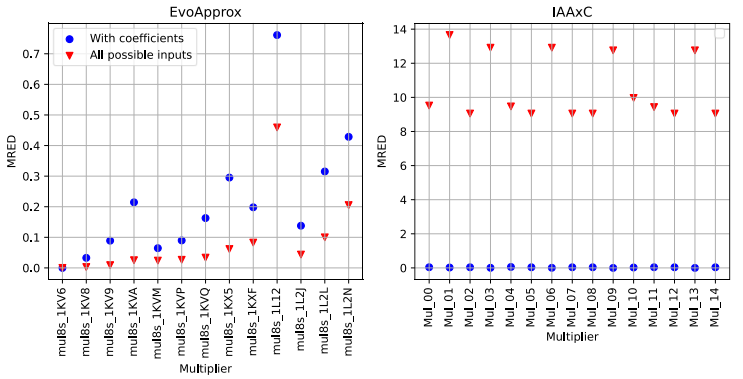
[1] A. Bosio, D. Ménard, and O. Sentieys, Eds., *Approximate Computing Techniques: From Component- to Application-Level*. Cham: Springer International Publishing, 2022. [Online]. Available: https://link.springer.com/10.1007/978-3-030-94705-7

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

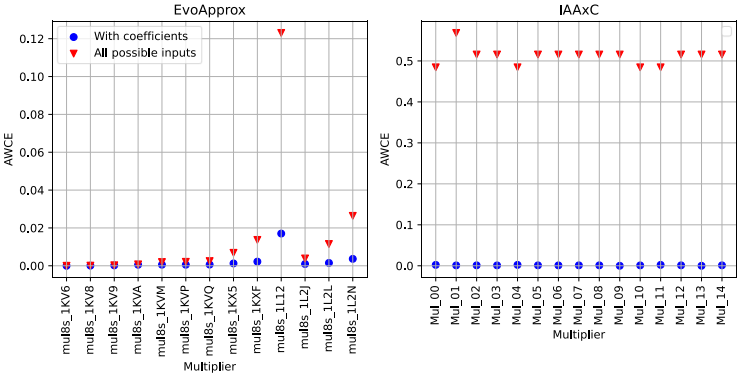
[3] M. Barbareschi, S. Barone, N. Mazzocca, and A. Moriconi, "A Catalog-based AIG-Rewriting Approach to the Design of Approximate Components," *IEEE Transactions on Emerging Topics in Computing*, 2022.



(a) Error probability



(b) Mean Relative Error Distance



(c) Absolute Worst Case Error

Fig. 2: Accuracy characterization of the multipliers with FIR filter and all possible inputs. coefficients

[4] M. Barbareschi, S. Barone, A. Bosio, J. Han, and M. Traiola, "A Genetic-algorithm-based Approach to the Design of DCT Hardware Accelerators," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 18, no. 3, pp. 1–25, Jul. 2022.

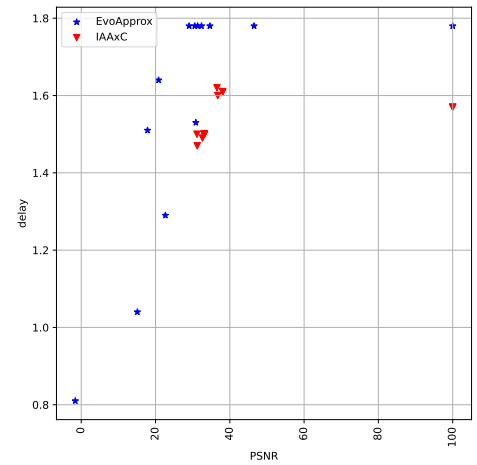
[5] V. Mrazek, M. A. Hanif, Z. Vasicek, L. Sekanina, and M. Shafique, "autoAx: An Automatic Design Space Exploration and Circuit Building Methodology utilizing Libraries of Approximate Components," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, Jun. 2019, pp. 1–6, iSSN: 0738-100X.

[6] V. Mrazek, L. Sekanina, and Z. Vasicek, "Libraries of Approximate Circuits: Automated Design and Application in CNN Accelerators," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 4, pp. 406–418, Dec. 2020.

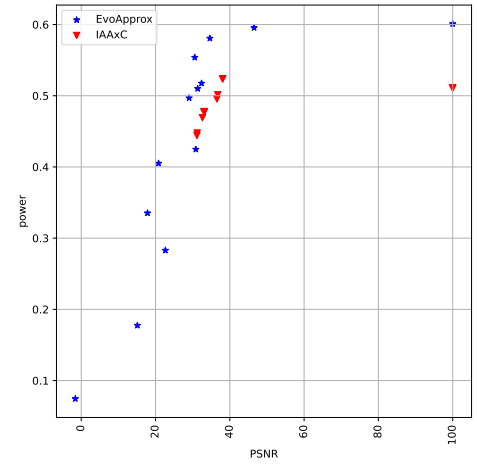
[7] A. Piri, S. Saeedi, M. Barbareschi, B. Deveautour, S. D. Carlo, I. O'Connor, A. Savino, M. Traiola, and A. Bosio, "Input-aware approximate computing," in *2022 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, 2022, pp. 1–6.

[8] H. Jiang, C. Liu, F. Lombardi, and J. Han, "Low-power approximate unsigned multipliers with configurable error recovery," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 1, pp. 189–202, 2018.

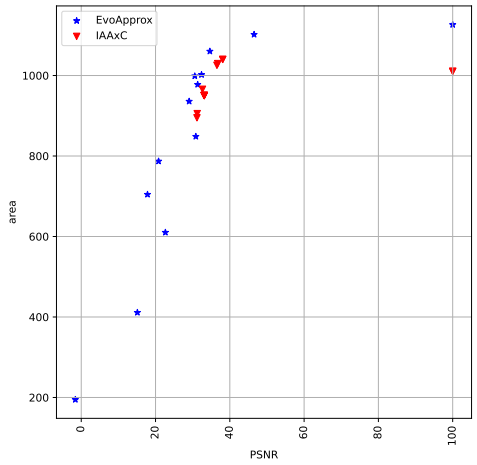
[9] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi,



(a) Delay-PSNR



(b) Power-PSNR



(c) Area-PSNR

Fig. 3: PSNR and hardware resources for 8-bits FIR while using input-aware approximate multipliers and multipliers from [20].

"Design-efficient approximate multiplication circuits through partial product perforation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 10, pp. 3105–3117, 2016.

[10] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra, G. Saggese, and G. Di Meo, "Approximate Multipliers Using Static Segmentation: Error Analysis and Improvements," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 6, pp. 2449–2462, Jun. 2022.

[11] E. Zacharelos, I. Nunziata, G. Saggese, A. G. Strollo, and E. Napoli, "Approximate Recursive Multipliers Using Low Power Building Blocks," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 3, pp. 1315–1330, Jul. 2022.

[12] H. Waris, C. Wang, W. Liu, J. Han, and F. Lombardi, "Hybrid Partial Product-Based High-Performance Approximate Recursive Multipliers," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 1, pp. 507–513, Jan. 2022.

[13] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: Systematic logic synthesis of approximate circuits," in *DAC Design Automation Conference 2012*, Jun. 2012, pp. 796–801, iSSN: 0738-100X.

[14] K. Nepal, S. Hashemi, H. Tann, R. I. Bahar, and S. Reda, "Automated High-Level Generation of Low-Power Approximate Computing Circuits," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 1, pp. 18–30, Jan. 2019.

[15] L. Witschen, M. Awais, H. Ghasemzadeh Mohammadi, T. Wiersema, and M. Platzner, "CIRCA: Towards a modular and extensible framework for approximate circuit generation," *Microelectronics Reliability*, vol. 99, pp. 277–290, Aug. 2019.

[16] L. Sekanina, Z. Vasicek, and V. Mrazek, "Automated Search-Based Functional Approximation for Digital Circuits," in *Approximate Circuits*, S. Reda and M. Shafique, Eds. Cham: Springer International Publishing, 2019, pp. 175–203.

[17] S. Barone, M. Traiola, M. Barbareschi, and A. Bosio, "Multi-Objective Application-Driven Approximate Design Method," *IEEE Access*, vol. 9, pp. 86 975–86 993, 2021.

[18] M. Barbareschi, S. Barone, and N. Mazzocca, "Advancing synthesis of decision tree-based multiple classifier systems: an approximate computing case study," *Knowledge and Information Systems*, pp. 1–20, Apr. 2021.

[19] A. Piri, S. Pappalardo, S. Barone, M. Barbareschi, B. Deveautour, M. Traiola, I. O'Connor, and A. Bosio, "Input-aware accuracy characterization for approximate circuits," in *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2023, pp. 179–182.

[20] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "Evoapprox8b: library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 258–261.

[21] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.

[22] "FreePDK45 | NC State EDA." [Online]. Available: https://eda.ncsu.edu/freepdk/freepdk45/