NPN-ApproxLib: DNN-Aware Approximate Multiplier Design via NPN Classes

Sallar Ahmadi-Pour[1], Chandan Kumar Jha[1], Sajjad Parvin[1], Rolf Drechsler[1,2]
[1]Institute of Computer Science, University of Bremen, Bremen, Germany
[2]Cyber-Physical Systems, DFKI, Bremen, Germany
{sallar, chajha, parvin, drechsler}@uni-bremen.de

***Abstract*—Approximate circuits have shown immense potential in error-resilient applications, with *Deep Neural Networks (DNNs)* being one of the most prominent examples. Current works explore approximate circuit errors only based on synthetic input samples and isolated from their target application. While automated approaches exist that explore a large number of approximate designs by a specified bound on the circuit error, it has recently been shown that this can lead to larger than expected errors. Verifiable approximate circuits based on functional approximation alleviate this issue. However, approaches that explore verifiable designs are limited due to the large design space and therefore cannot evaluate the circuit in the target application during the exploration. In this paper, we introduce a methodology, based on *Negation-Permutation-Negation (NPN)* classes, to explore a reduced number of approximate multipliers based on the error within DNN applications to overcome limitations of previous works. Additionally, we will provide NPN-ApproxLib as an open-source available repository[1] to stimulate further research.**

***Index Terms*—Approximate Circuit, Pareto-optimal, Artificial Intelligence, Synthesis, Error Metric**

I. Introduction

Approximate computing has emerged as a technique for error-resilient applications, allowing engineers to trade-off between circuit performance and error impact in applications [1], [2]. While the possible benefits of utilizing approximate circuits in DNNs are vast, current works explore approximate circuits by only evaluating the circuits for synthetic inputs and not the target application [3]. Without realistic input patterns, these approaches leads to the inclusion of inadequate approximate circuits as a result [3]–[6]. Works, like EvoApproxLib [7], automatically explore approximate designs based on error bounds defined for the circuits, which demonstrates the limitations previously mentioned. As the approximate designs are simulated without the applications and selected by error bounds, circuits with the same error metrics can show drastically different error behavior within applications [8]. By contrast, the state-of-the-art approximate circuit library FV-LIDAC mitigates the limitations by considering non-uniform input sample distributions for error characterization [9]. Additionally, FV-LIDAC explores approximate designs systematically and exhaustively with structural approximation, which allows verification [8] compared to works like EvoApproxLib. However, an application-based error characterization for approximate designs is impossible due to the large search space required for the systematic and structural approach. In fact, most works don't consider characterizing the approximate circuit errors within the target application when exploring the space of approximate designs.

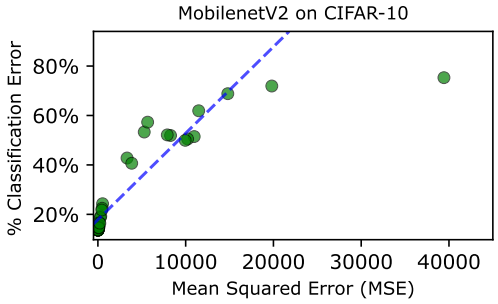


Figure 1: Non-linear correlation of FV-LIDAC circuits for MSE and error for MobileNetV2 with CIFAR-10.

Hence, to demonstrate that isolated error metrics of approximate circuits are not enough, we look at the correlation between an error metric and an error in an application. Fig. 1 shows the *Mean Square Error (MSE)* (x-axis) and classification error within MobileNetV2 with the CIFAR-10 (y-axis) for the FV-LIDAC library of Pareto-optimal 8-bit approximate multipliers. It is important to note that the non-linear correlation between MSE and classification error suggests that the application inputs and input distributions derived from applications must be taken into account. As a result, this limits the use of isolated circuit error metrics (mainly based on error distance) for the selection of approximate circuits for applications like DNNs. Instead, when exploring approximate circuit designs, applications should be considered either through input sample distributions or through direct application-aware simulations. Works like FV-LIDAC [9] have considered different input sampling distributions for closer consideration of target applications by utilizing a search space that doesn't scale for application-aware simulations. By utilizing NPN classes, we can reduce the number of explored Boolean functions for sum and carry approximations, which mitigates the scalability issues of FV-LIDAC.

In particular, our contributions are:

- Proposing a systematic approximation method, using NPN classes, significantly reducing the search space explored in state-of-the-art approximate circuit libraries to 600 instead

This research has been supported by German Research Foundation (DFG) within the project PLiM (DR 287/35-2, project number 406079023), the German Ministry for Research, Technology and Space (BMFTR) with project ExaVerse (grant number 16IW25003), and the Data Science Center of the University of Bremen (DSC@UB) funded by the State of Bremen.

[1]https://github.com/agra-uni-bremen/npn-approxlib

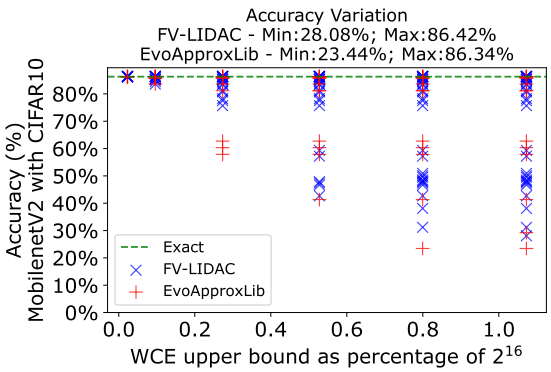


Figure 2: Variation of AI accuracy using $<1.5\,\%$ WCE error bound when utilizing state-of-the-art approximate multipliers.

of ≈392k while preserving verifiability of the circuits and identifying sufficient Pareto-optimal approximate designs.
- Considering the error impact within DNNs for the *Design Space Exploration (DSE)* of approximate circuits, alleviating limitations of error metrics obtained using synthetic inputs.

II. Motivation

When considering the use of approximate multipliers in DNN applications, the error of the multiplier with respect to synthetic inputs is not sufficient. To highlight this in more detail, we demonstrate this effect using a motivating example. We utilize the EvoApproxLib and FV-LIDAC libraries for approximate circuits to gather 8-bit multipliers and evaluate them for their *Worst Case Error (WCE)* as well as their accuracy in an image classification application. In particular, WCEs were collected with the MECALS tool [10] utilizing relaxed equivalence checking. The DNN accuracy was obtained by using MobileNetV2 on the CIFAR-10 dataset with the first Conv2D layer replaced with its approximate counterpart.

Fig. 2 shows the spread of classification accuracy with MobileNetV2 on the CIFAR-10 when utilizing a single approximate Conv2D layer for a given error of $<1.5\,\%$ WCE. The x-axis shows the WCE as a percentage of the maximum unsigned integer, and the y-axis shows the performance of MobileNetV2 utilizing the approximate multipliers. The shown approximate multipliers are chosen by setting the WCE bound to be less than $1.5\,\%$ of $2^{16} = 65536$. This is a commonly chosen error bound for approximate circuits leading to improved hardware performance (area, power delay) while retaining enough quality within the application [11]–[13]. Although a lot of approximate multipliers are available, the error bound itself is not enough to obtain a feasible choice of circuits leading to a good performance. Even with WCE of less than $1.5\,\%$, when used in applications, the spread in accuracy ($23.4\,\%$ and $86.3\,\%$ for EvoApproxLib and between $28.1\,\%$ and $86.4\,\%$ for FV-LIDAC), i.e., the same error bounds lead to multiple designs meeting the specification, causes issues as the error does not reflect the larger than expected deterioration [8]. Such a trend is not unique to this example, this can also be seen for other architectures and datasets (e.g., LeNet on Fashion-MNIST).

Table I: Comparison of relevant related works

Work	Search Space	NPN Classes	Error Metric	FV	AI Eval.
TCAS-I'21 [6]	Selected	✗	NMED, E_{MAC} *1	✗	Partial
TVLSI'22 [22]	EA	✗	MSE *2	✗	Partial
TCAS-II'22 [23]	Selected	✗	NMAE, MRE	✗	Partial
ESL'23 [24]	Selected	✗	NMAE, MAE, EP, MRE	✗	Partial
TCAS-II'25 [5]	Selected	✗	MRE, MAE, NMAE, MSE, NMSE, ER	✗	Partial
EvoApproxLib [7]	EA	✗	MAE, WCE, MRE, EP	✗	None *3
FV-LIDAC [9]	Full-scale	(✗)	MAE, MSE	✓	Partial
This work	NPN-constrained	✓	DNN Error	✓	Full

FV = Functional Verification, EA = Evolutionary Algorithm, Selected = Manually selected approximation, Full-scale = Systematic search
*1 The authors introduce a error metric based on one specific DNN application. *2 Custom error for EA-based optimization based on MSE. *3 Additional other works utilize this library when evaluating DNN applications, but the original work does not.

Hence, when having to consider the impact on the DNN accuracy, optimizing for traditional error metrics is clearly not enough. In order to appropriately consider the DNN accuracy when exploring approximate multipliers, the exploration of approximate circuit designs should yield systematically approximated circuits for verifiability and a small enough search space leading to faster DSE. This means that rather than WCE, verification is required for approximate circuits.

III. Related Work

Approximate computing has emerged as a promising technique across the software and hardware layers within computers [3], [14], [15]. Particularly for approximate circuits, a plethora of works exist regarding different arithmetic circuits [16]–[18], synthesis techniques [19]–[21], and other approaches [14], [15]. Due to the vast amount of works, we focus on the most relevant related works regarding approximate multipliers.

An overview and comparison of the relevant related works is shown in Tab. I. The first column shows the work, the second column shows the approach used to explore the search space, the third column indicates the use of NPN classes, the fourth column shows the utilized error metrics for quantifying the circuit error, the fifth column shows if functional verification is performed, and the sixth column shows if the circuits were evaluated with *Artificial Intelligence (AI)*-based applications (such as DNNs). In the second column, *selected* refers to manually crafted and chosen approximate circuits without an automated approach, while *EA* refers to the use of an evolutionary algorithm, *full-scale* refers to a systematic exploration of the search space, and *NPN-constrained* refers to the constrained search space by NPN class functions. Approximate multipliers that are categorized as part of NPN classes get a check mark in the third column. In the fifth column, *partial* refers to AI-based case studies without search space exploration based on target applications.

Works [5], [6], [23], [24] propose manually crafted multipliers by manipulating the Boolean function of the basic blocks, which are often driven by the error reduction of the basic block. By avoiding an exploration of a search space, the

works don't propose a range of circuits and potentially miss out on better circuits. The works described in [22] and [7] utilize evolutionary algorithms to explore and choose circuits, which makes the process non-deterministic, and the resulting multipliers are not retaining structures beneficial for circuit verification. This possibly leads to deterioration of the actual errors in the applications contrary to the circuits error metrics and circuit approximations that cannot be reconstructed by a specification. In [9], a large search space of approximate circuits is explored automatically, with a systematic approximation approach leading to verifiable approximate circuits. However, the large search space leads to scalability limitations. Furthermore, the works explore AI applications such as DNNs but only consider them as case studies after selecting the approximate multipliers (hence the partial evaluation).

With NPN-ApproxLib, we overcome these limitations by enabling a reduced search space for scalable automated approaches, a systematic approximation approach for verification, and a consideration of DNN accuracy/error metrics for search space exploration.

IV. NPN-Class-based Approximate Circuits

In this work, the main approach for the systematic but constrained generation of approximate *Dadda Tree Multiplier (DTM)* is based on the ten NPN classes for three-input functions described by the authors of [25]. Originally, these ten NPN classes were only studied for their use in synthesis and their optimality in representing circuit graphs with the most Boolean expressiveness. In [25], two Boolean functions are defined to be part of the same NPN class if one function can be obtained by applying input negation, input permutation, or output negation. While not considered for use in arithmetic circuits as such, the described three-input NPN functions can be utilized to describe candidate functions for the sum and carry functions inside *Full Adders (FAs)* and *Half Adders (HAs)* used in approximate multipliers. From the ten NPN classes, the functions can be utilized in all 100 (10×10) combinations for sum and carry, leading to 100 approximate DTMs. For each DTM, the number of approximated columns can also be adjusted from 2 to 12, in steps of 2, leading to 600 (100×6) approximate DTMs as the search space. This column-wise approximation is performed in the same manner as in state-of-the-art FV-LIDAC [9], which describes the column-wise approximation for DTMs in additional detail. This naturally creates a constrained set of candidate circuits, reducing the number of circuits to be searched for Pareto-optimal approximate multipliers, which can be explored much faster. Naturally, the choice of Boolean function from each of the ten NPN classes is not necessarily trivial and influences the quality of the search space. For this work, we describe two distinct sets of approximate DTMs which we study in the derivation of Pareto-optimal approximate DTMs as follows:

1. NPN_{Base} are the ten functions described in [25], which were shown to be most effective for synthesis. The choice of these functions is based on availability, without further preference. Please note, one of the 100 (10×10) function combinations of sum and carry will contain the exact FA functions (sum $= a \oplus b \oplus c_{in}$, carry $=$ Majority(a, b, c_{in})).

2. NPN_{RD} are ten functions, chosen to have the smallest possible Hamming distance (i.e., error distance) to the sum function and carry function, respectively. The ten functions for the sum and the carry are distinct and are chosen to also be different from their exact counterparts (in this case, the next possible function was chosen). The choice of these functions is based on the intuition that many studies in approximate circuits follow: reducing the Hamming distance/error distance of the approximate basic block (often an FA). Hence, using the NPN_{RD} set contains a function from each NPN class where the distance was reduced to the smallest non-exact Boolean function with respect to the exact sum and carry functions.

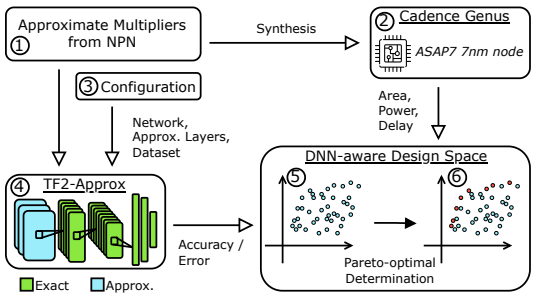
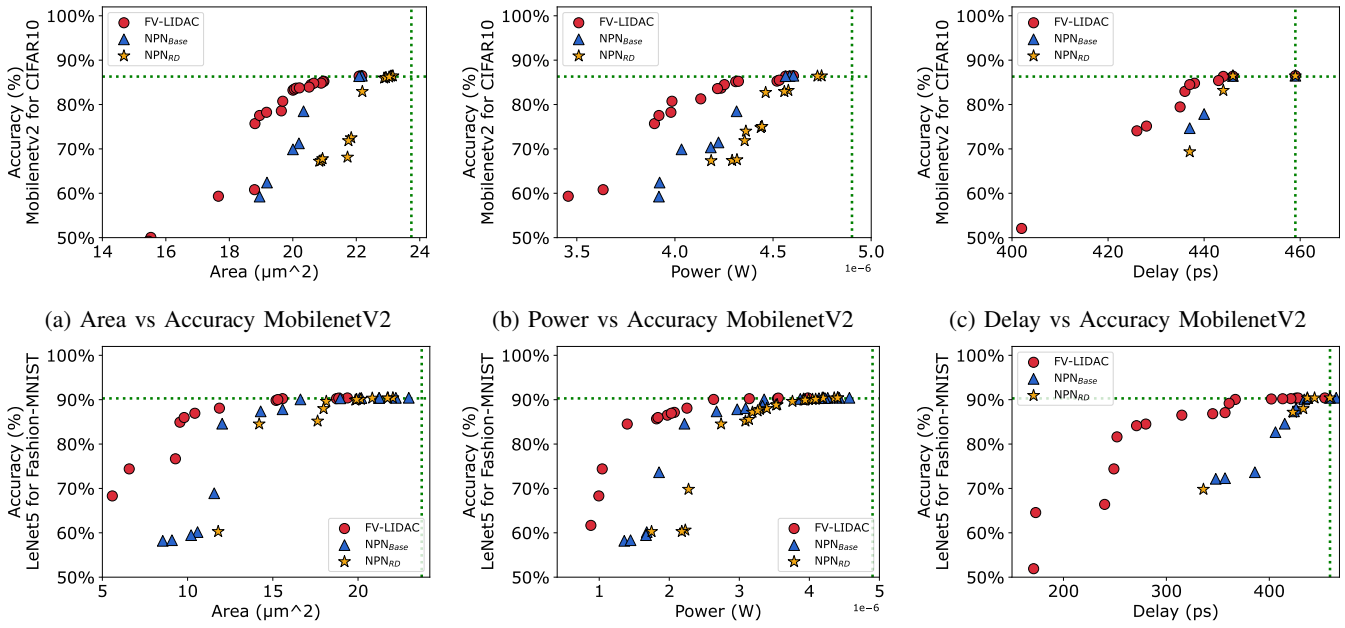


Figure 3: Overview of the NPN-ApproxLib approach.

The remainder of our flow is described in Fig. 3: ① starts with the 1200 approximate DTM (600 from NPN_{Base} and 600 from NPN_{RD}). In ②, these circuits are synthesized in order to collect the area, power, and delay metrics of each approximate DTM. For our experiments, we prepare the approximate DNN applications and *Look-Up Table (LUT)*-based representations of each DTM for the TF-Approximate [26] framework in step ③. Specifically, we configured LeNet5 with Fashion-MNIST, and all Conv2D layers were replaced with their approximate multiplication counterparts (with one approx. DTM) and MobileNetV2 with CIFAR-10, with the first Conv2D layer replaced with an approximate multiplication counterpart. In ④, the execution of each network for each of the 1200 approximate DTMs is performed with TF-Approximate. Step ⑤ describes the preparation of each search space which will consist of one design metric (area, power, delay) and one DNN accuracy metric (%-accuracy with LeNet5 on Fashion-MNIST, %-accuracy with MobileNetV2 on CIFAR-10). Finally, in ⑥, for each search space we determine the Pareto-optimal approximate DTM with py-paretoarchive [27], [28].

V. Evaluation

In order to assess the approximate DTM circuits from the NPN_{Base} and NPN_{RD} sets, we obtain Pareto-optimal circuits when considering the pairs of design metrics and DNN accuracies (e.g., area and %-Accuracy for MobileNetV2). For a reference of the quality of our NPN-based approximate DTM, we compare our Pareto-optimal circuits against the Pareto-optimal circuits of FV-LIDAC after passing through the flow described in Fig. 3. To obtain comparable Pareto frontiers, we collect all available 8-bit DTM from FV-LIDAC for all



(a) Area vs Accuracy MobilenetV2 (b) Power vs Accuracy MobilenetV2 (c) Delay vs Accuracy MobilenetV2

(d) Area vs Accuracy LeNet5 (e) Power vs Accuracy LeNet5 (f) Delay vs Accuracy LeNet5

Figure 4: Pareto-optimal designs of NPN$_{Base}$, NPN$_{RD}$ and FV-LIDAC for area, power, and delay vs. DNN accuracy.

distributions and design metrics (243 distinct approximate DTMs) and perform synthesis as well as embedding the approximate multipliers in DNNs with TF-Approximate. The primary question such a comparison should inspect is how close we can get to the Pareto-optimal circuits of FV-LIDAC while requiring a fraction of the search space explored. Next we also inspect in more detail how well the different circuits perform in DNN applications like LeNet5 and MobileNetV2 with datasets like Fashion-MNIST and CIFAR-10.

Our experiments were executed on an Intel(R) Xeon(R) Gold 6240 36-Core Processor with a frequency of 2.6 GHz and 376 GB of memory. The synthesis of the circuits was performed with Cadence Genus, using the ASAP7 7 nm open-source PDK [29]. To execute the DNN models with approximate layers, we utilized TF-Approximate [26]. The search for Pareto-optimal circuits was performed with the Pyparetoarchive Python library [27], [28].

Fig. 4 shows the different comparisons of Pareto-optimal approximate DTM of NPN$_{Base}$, NPN$_{RD}$, and FV-LIDAC. The figure consists of six plots, with the first row showing MobileNetV2 results (Figs. 4a to 4c) and the second row showing LeNet5 results (Figs. 4d to 4f). Each column shows the plots for one design metric (area, power, delay), hence each plot shows one particular metric combination (e.g., area vs. accuracy for MobileNetV2 in Fig. 4a). Each plot shows the design metric on the x-axis (lower is better) and the accuracy in the DNN application on the y-axis (higher is better). The Pareto-optimal circuits of FV-LIDAC are shown as red markers (●), NPN$_{Base}$ as blue markers (▲), and NPN$_{RD}$ as yellow markers (★). Each plot also features green dotted vertical and horizontal lines marking the area, power, delay, and accuracy of the exact DTM, respectively. For better readability of the plots, the lower axis limit is greater than 0.

In general, approximate multipliers from the NPN$_{Base}$ and NPN$_{RD}$ sets reveal good Pareto-optimal approximate multipliers for smaller improvements on the design metrics. For an area improvement of up to 2 µm^2, power improvement of up to 0.5×10^{-6} W, or delay improvement of up to 20 ps, the accuracy within Mobilenetv2 of NPN$_{Base}$ and NPN$_{RD}$ only drops by a few percent. This effect is stronger for the smaller LeNet5, where more approximate layers are configured (area improvement up to 5 µm^2, power improvement up to 2×10^{-6} W, and delay improvement up to 50 ps). For larger improvements in area, power, and delay compared to the exact DTM, we observe a deviation from FV-LIDAC, with NPN$_{Base}$ revealing better approximate DTMs than NPN$_{RD}$.

With our proposed approach of utilizing NPN classes as search space constraint, we were able to cut down the search space from 393 216 (FV-LIDAC) to 600 (our approach), which is a reduction by roughly 655×. With our experiments, we further demonstrate that NPN classes introduce an additional design space parameter for tailored approximate circuits.

VI. Conclusion

In this paper, we proposed NPN-ApproxLib, a DNN-aware approximate multiplier design search through the use of NPN classes. By utilizing NPN classes, the shortcomings of state-of-the-art are mitigated while preserving properties like verifiability. NPN-ApproxLib considers the error in the DNN application as a search space parameter of the approximate multiplier design space. As part of our evaluation, we showed how NPN classes drastically reduced the search space (655×) while revealing similar circuits to state-of-the-art. Lastly, to stimulate further research, we make NPN-ApproxLib open-source available (https://github.com/agra-uni-bremen/npn-approxlib).

REFERENCES

[1] B. C. Schaefer, "Close enough to correct: Approximate computing," *Global Communications*, vol. 2026, 2026.

[2] W. Liu, F. Lombardi, and M. Schulte, "Approximate computing: from circuits to applications," in *IEEE Proceedings*, vol. 108, no. 12, 2020, pp. 2103–2107.

[3] 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.

[4] Z. Vasicek, "Formal methods for exact analysis of approximate circuits," *IEEE Access*, vol. 7, pp. 177 309–177 331, 2019.

[5] A. Kumari and R. P. Palathinkal, "Design and analysis of energy efficient approximate multipliers for image processing and deep neural network," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 72, no. 2, pp. 854–867, 2024.

[6] G. Park, J. Kung, and Y. Lee, "Design and analysis of approximate compressors for balanced error accumulation in mac operator," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 7, pp. 2950–2961, 2021.

[7] 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.

[8] C. K. Jha, M. Hassan, and R. Drechsler, "cecapprox: Enabling automated combinational equivalence checking for approximate circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2024.

[9] S. Ahmadi-Pour, S. Parvin, C. K. Jha, and R. Drechsler, "Fv-lidac: Formally verified library of input data aware approximate arithmetic circuits," *ACM Transactions on Design Automation of Electronic Systems*, 2025.

[10] C. Meng, W. Qian, and G. D. Micheli, "Simulation-guided approximate logic synthesis under the maximum error constraint," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2026.

[11] H. Mo, Y. Wu, H. Jiang, Z. Ma, F. Lombardi, J. Han, and L. Liu, "Learning the error features of approximate multipliers for neural network applications," *IEEE Transactions on Computers*, vol. 73, no. 3, pp. 842–856, 2023.

[12] P. Choudhary, L. Bhargava, M. Fujita, and V. Singh, "Lut-based arithmetic circuit approximation with formal guarantee on worst case relative error," in *2023 IEEE 24th Latin American Test Symposium (LATS)*. IEEE, 2023, pp. 1–2.

[13] M. Češka jr, M. Češka, J. Matyáš, A. Pankuch, and T. Vojnar, "Approximating complex arithmetic circuits with guaranteed worst-case relative error," in *International Conference on Computer Aided Systems Theory*. Springer, 2019, pp. 482–490.

[14] V. Leon, M. A. Hanif, G. Armeniakos, X. Jiao, M. Shafique, K. Pekmestzi, and D. Soudris, "Approximate computing survey, part i: Terminology and software & hardware approximation techniques," *ACM Computing Surveys*, vol. 57, no. 7, pp. 1–36, 2025.

[15] ——, "Approximate computing survey, part ii: Application-specific & architectural approximation techniques and applications," *ACM Computing Surveys*, vol. 57, no. 7, pp. 1–36, 2025.

[16] A. Dalloo, A. Najafi, and A. Garcia-Ortiz, "Systematic design of an approximate adder: The optimized lower part constant-or adder," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 8, pp. 1595–1599, 2018.

[17] M. A. Hanif, A. Arous, and M. Shafique, "Dreamx: A data-driven error estimation methodology for adders composed of cascaded approximate units," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 11, pp. 3348–3357, 2024.

[18] C. K. Jha, A. Nandi, and J. Mekie, "Single exact single approximate adders and single exact dual approximate adders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 7, pp. 907–916, 2023.

[19] 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*, vol. 11, no. 1, pp. 70–81, 2022.

[20] C. Meng, A. Mishchenko, W. Qian, and G. De Micheli, "Efficient resubstitution-based approximate logic synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 6, pp. 2040–2053, 2024.

[21] C. Meng, W. Qian, and A. Mishchenko, "Alsrac: Approximate logic synthesis by resubstitution with approximate care set," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[22] Z. Li, S. Zheng, J. Zhang, Y. Lu, J. Gao, J. Tao, and L. Wang, "Adaptable approximate multiplier design based on input distribution and polarity," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 12, pp. 1813–1826, 2022.

[23] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, "An ultra-efficient approximate multiplier with error compensation for error-resilient applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 2, pp. 776–780, 2022.

[24] Z. Aizaz, K. Khare, and A. Tirmizi, "Approximate row-merging-based multipliers for neural network acceleration on fpgas," *IEEE Embedded Systems Letters*, vol. 16, no. 2, pp. 126–129, 2023.

[25] D. S. Marakkalage, E. Testa, H. Riener, A. Mishchenko, M. Soeken, and G. De Micheli, "Three-input gates for logic synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 10, pp. 2184–2188, 2020.

[26] F. Vaverka, V. Mrazek, Z. Vasicek, and L. Sekanina, "Tfapprox: Towards a fast emulation of dnn approximate hardware accelerators on gpu," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 294–297.

[27] T. Glasmachers, "A fast incremental bsp tree archive for non-dominated points," in *Evolutionary Multi-Criterion Optimization*, H. Trautmann, G. Rudolph, K. Klamroth, O. Schütze, M. Wiecek, Y. Jin, and C. Grimme, Eds. Cham: Springer International Publishing, 2017, pp. 252–266.

[28] ehwFIT, "py-paretoarchive," https://github.com/ehw-fit/py-paretoarchive, 2023.

[29] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "ASAP7 PDK and cell library," https://github.com/The-OpenROAD-Project/asap7, 2023.