DIVIAC: Library of Input Data Aware Approximate Dividers with Partial Exact Minimization

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

***Abstract*—Approximate divider designs have been extensively explored as they are widely used in image processing applications. The design of approximate dividers is predominantly accomplished through the use of functional approximation, where the Boolean functions of the subtractor sub-blocks in the dividers are approximated by replacing them with a different Boolean function. However, the prior works have explored only a few Boolean approximations and evaluated the error metrics using uniform distributions. This does not effectively explore the design space and leads to suboptimal approximate divider designs. In this work, we alleviate the limitations of prior works as follows: Firstly, we perform an extensive and systematic design space exploration to identify the Pareto-optimal approximate divider designs, where each sub-block has been reduced through exact minimization. Secondly, we do this for three input distributions, namely uniform, normal, and exponential distributions. Lastly, we also evaluate the design on the widely used image processing applications for approximate dividers, namely background removal and change detection. We aim to make the Pareto-optimal approximate divider designs available as open-source to stimulate further research.**

***Index Terms*—Approximate Computing, Restoring Array Divider, Input Distribution, Arithmetic Circuits, Synthesis**

I. Introduction

Dividers are one of the most expensive designs in terms of area, power, and delay [1]. Hence, there have been several efforts to reduce the cost of the divider design [2]. One of the methods to optimize the divider's design for error-resilient applications is by the introduction of approximation in the divider designs [3], [4]. The approximation is used as a trade-off to obtain benefits in area, power, and delay with little loss in the error-resilient application's output quality [5], [6]. The prior works have shown that functional approximation can be used to obtain approximate divider designs resulting in significant benefits as compared to the exact dividers.

In [3], the authors were one of the first to introduce the design of approximate dividers. The authors used Pass Transistor Logic (PTL) based designs to implement the approximate divider design at the transistor level. The authors presented cell-level and array-level approximation techniques, as well as applications that can benefit from divider approximation. In [7], the authors proposed three approximate subtractors, which were used to build the approximate divider designs. The authors showed that they were able to reduce the power and delay by a third as compared to the exact divider. In [4], [8], the approximate subtractors were optimized using K-Maps and were to be used for designing the approximate subtractors. In [8], the authors achieved 60% less power and

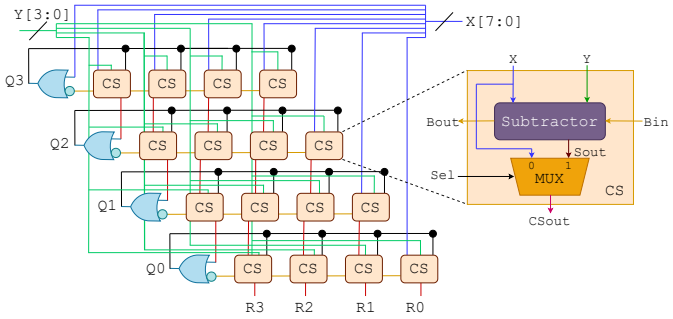


Figure 1: Restoring Array Divider

40% less delay for their best approximate divider design. In [4], the authors showed that PTL-based designs consume a large amount of energy and have a larger delay. The authors proposed CMOS-based approximate dividers that consumed 50% less energy than the exact dividers.

Recently, approximate dividers based on Logic-in-Memory (LiM) using memristors were proposed. In [9], the authors systematically explored the design space of functional approximation to generate the Pareto-optimal divider designs. However, this work focused on tailoring the divider designs for LiM, and only explored one input distribution, namely uniform distribution for error computation. In recent works [10]–[12], it has been shown that approximate circuits that are not tailored for a specific input distribution can lead to larger-than-expected deterioration in the output quality. It is also shown that better designs can be obtained if the approximate designs are tailored for the application's input distribution [13]–[15]. Input distribution-aware approximate adders and multiplier designs have been recently explored [16]–[18]. However, none of the prior works have explored input distribution-aware approximate divider designs. The following are the contributions of our work:

1) We obtain Pareto-optimal approximate divider designs using systematic functional approximation for three different input distributions.
2) The sub-block of the approximate divider designs has been reduced through exact minimization.
3) We show that, compared to prior works, we achieve better Pareto-optimal approximate divider designs and also use these designs in two image processing applications.
4) The Pareto-optimal designs are available at https://github.com/agra-uni-bremen/diviac as open source.

II. Preliminaries and Terminologies

In this section, we discuss the design of the Restoring Array Divider (RAD) and the terminologies we use to describe the Boolean function throughout this paper. The RAD design is shown in Fig. 1. It is an 8 by 4 divider, i.e., the dividend is 8 bits and the divisor is 4 bits. The result of the division, i.e., the quotient and the remainder, is 4 bits each. The basic block of the RAD is a Controlled Subtractor (CS) as shown in Fig. 1. The CS consists of a subtractor followed by a MUX that selects between the result of the subtraction and the bit of the dividend. The overflow is prevented when the n most significant bits (MSBs) of the dividend are less than the divisor. This condition ensures that the quotient requires up to 4 bits. Repeated subtraction is used to perform the division operation. The truth table values for the borrow bit are `"01110001"` (113), and for the difference bits are `"01101001"` (105) when written as a bit vector. The naming scheme used in this work follows the scheme `Borrow_Difference` using the decimal values of the truth table values to identify a specific divider circuit. Hence, the exact design is determined by `113_105`.

In this work, the mode of approximation is functional approximation, i.e., the Boolean function of the subtractor is replaced with another function. For example, one typical approximation can be to approximate both the borrow and the difference outputs for an input value of 000 to 1. Then, the approximated Boolean functions for borrow and difference will be `"11110001"` and `"11101001"`, respectively. According to our naming scheme, the approximated design will be `241_233`. For designing the approximate divider, the subtractor block in the CS is functionally approximated. Depending upon the approximation scheme, different numbers of CS blocks are selected for approximation.

III. Framework for Input-Aware Dividers

In this section, we discuss the systematic design space exploration in detail. Fig. 2 shows an overview of the flow; we refer to the circled numbers for each step (Ⓝ) from Fig. 2 while explaining the flow of DIVIAC.

First, we choose a divider circuit suitable for approximation, and identify the basic blocks which are subject to approximation (step ①). For RAD, we systematically approximate a 16-by-8 RAD with 8-bit outputs, by replacing rows, columns, and the triangular section of the divider with approximate basic blocks (step ②). These approximation methodologies have been shown to give the best benefits in the RAD design [3], [4], [9]. In a RAD, the basic block is a subtractor with three input bits and two output bits. By approximating each output bit for all possible input bit combinations, we obtain 65 536 (256×256) distinctively different approximate basic blocks. These approximated basic blocks are applied to the RAD in the previously mentioned row, column, and triangular scheme. This scheme is applied with an approximation amount N_{approx} of the values 2, 4, and 6. For example, when approximating a RAD in the row scheme for N_{approx} being 6, the six rows of the divider are replaced with all of the 65 536 approximate

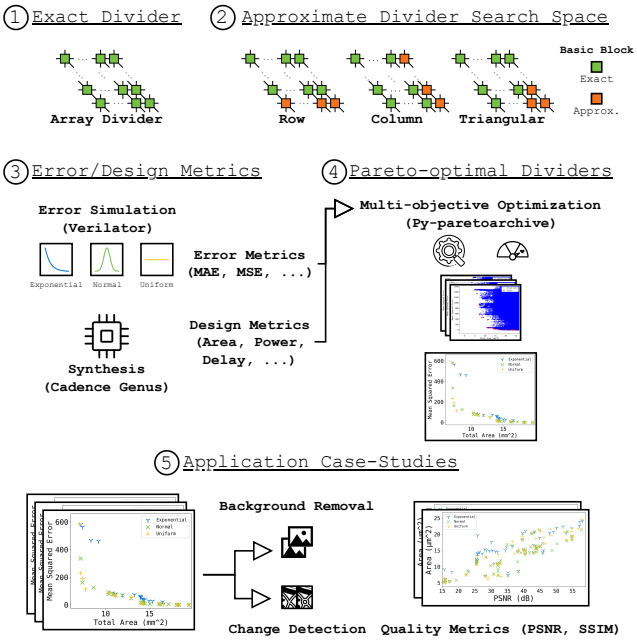
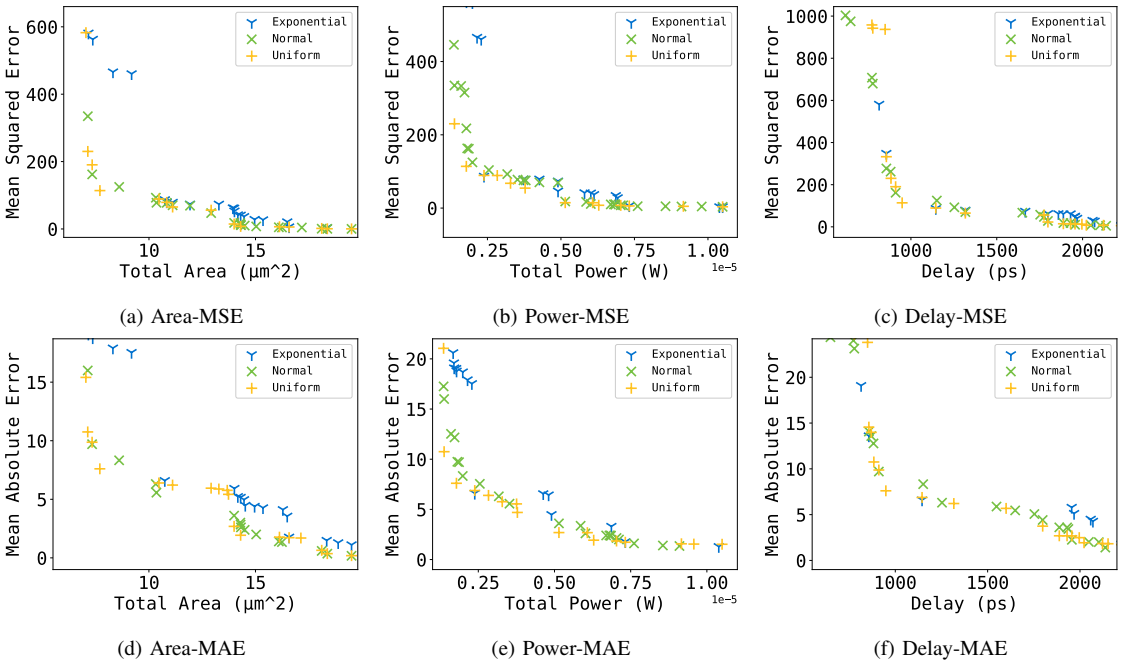


Figure 2: Overview of DIVIAC Flow

basic blocks one at a time for every approximated basic block. This is repeated for every available amount of approximation, i.e., 2, 4, and 6, leading to 196 608 approximate dividers per approximation scheme ($65\,536 \times 3$). Across all approximation schemes, i.e., row, column, and triangular, a total of 589 824 ($196\,608 \times 3$) approximate RAD are obtained for design space exploration. Next, error metrics are obtained through simulation with Verilator [19] and design metrics are obtained through synthesis with Cadence Genus tool using ASAP7 7 nm open-source PDK [20] (step ③). For the design metrics, we consider area, power, and delay of the divider circuits in the mentioned PDK. For the error metrics, we consider Mean Absolute Error (MAE) and Mean Square Error (MSE) as error metrics, although any other error metric can be utilized. Recent work showed a distinct difference when utilizing different input distributions for input samples, other than uniformly generated random input samples [16]. Hence, we perform the error simulations with inputs sampled from different input distributions. By utilizing exponential and normal distributions, next to the uniform distribution, we consider this recent parameter in the design space of approximate computing. We performed exact minimization of the approximate basic blocks before obtaining design metrics. Because the basic blocks in the approximate circuits are small, performing exact minimization leads to different netlists with different area, power, or delay.

To obtain and analyze the Pareto-optimal dividers (step ④), firstly, we determine every Pareto-optimal set for each combination of the error metric, input distribution, and design metric, leading to 18 different Pareto-optimal sets ($2 \times 3 \times 3$). Secondly, we analyze the Pareto-optimal sets that have the same error metric and design metric pair (e.g., Area and MSE)



(a) Area-MSE (b) Power-MSE (c) Delay-MSE

(d) Area-MAE (e) Power-MAE (f) Delay-MAE

Figure 3: Comparison of all Pareto optimal solutions for each input distribution for 8 and 16 bit exsy. Each row shows the designs for both error metrics (MSE and MAE) for each bit width respectively.

with different input distributions. In this way, the impact of the different input distributions for all dividers in the Pareto-optimal sets can be compared. As a last step of the framework, the dividers from the Pareto-optimal sets are utilized in application case studies, such as background removal and change detection for images (step ⑤). By obtaining output quality metrics like Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM), the Pareto-optimal sets can be compared on real applications beyond error simulations. This additional analysis enables the input distribution-aware error simulation in the context of related works.

IV. Evaluation

A. Experimental Setup

Following the framework illustrated in Section III, we performed the design space exploration of approximate RAD across 589 824 approximate dividers. For the error simulation, input samples were generated from exponential, normal, and uniform distributions, respectively. We generated 10 000 input pairs for the divider for each input distribution. Through a parallelized flow based on Verilator, every error simulation was compiled and simulated in a total run time of 4716 min (78.6 h or approx. 3 days). Hence, the error simulation was performed with a speed of 0.479 s per design. To evaluate and compare each design against one another, we synthesized the designs and evaluated them in terms of their power consumption (W), delay (ps), and area (μm^2). To synthesize the designs, we processed 16 parallel batches, each containing 256 designs, to accelerate the synthesis process. On average, each design required 9 s for synthesis. The use of 16 parallel batches was constrained by the limited Genus tool licenses; with more licenses and sufficiently powerful machines, additional batches could be run in parallel.

The Pareto-optimal sets for each triplet of design metric, error metric, and input distribution were determined in a parallel manner, and were performed utilizing the py-paretoarchive library [21], [22] in Python. Through parallelization, the determination of the Pareto-optimal designs took 99 s in total, hence processing one Pareto-optimal set every 2.75 s. The synthesis was performed on an AMD EPYC 7302P 16-Core Processor with 3 GHz and 512 GB of memory. The simulations and determination of Pareto-optimal sets were performed on an Intel(R) Xeon(R) Gold 6240 36-Core Processor with 2.6 GHz and 376 GB of memory. The presentation and discussion of the results are structured into three parts: 1) showing the impact of input distributions on the Pareto-optimal dividers, then 2) showing the results of the ablation with and without exact synthesis pre-optimization, and 3) showing the results of the quality metrics obtained for the application case study with background removal and change detection.

B. Results and Discussion

a) Impact of input distributions on the Pareto-optimal dividers: Fig. 3 shows the results of each pair of design metric and error metric evaluated, each containing the Pareto-optimal set for the respective pair for inputs sampled from an exponential distribution (blue, $\curlyvee$), a normal distribution (green, $\times$), and a uniform distribution (yellow, $+$). The subplots are arranged in two rows and three columns. The top row contains the MSE error metric pairs (Figs. 3a to 3c), the bottom row contains the MAE error metric pairs (Figs. 3d to 3f). The first column contains the area design metric pairs

Table I: Comparison of dividers from prior works. Each divider of the compared works is encoded with their approximate borrow (B) and difference (D) functions. If a divider from a prior work is part of any Pareto-optimal set, its marked with ✓, else ✗. Furthermore, the parameter column shows the design space parameter, for which the divider is in a Pareto-optimal set. N/A = Not Applicable.

Prior Work	Basic Block (B_D)	Pareto?	Design Space Parameter (Synthesis, Approx. Scheme, N_{approx}, Design Metric, Error Metric, Distribution)
[4]	113_113	✓	(ESY,Row,2,Area,MAE,E) (ESY,Row,2,Area,MSE,E) (HSY,Triangular,2,Area,MAE,E) (HSY,Triangular,2,Area,MAE,N) (HSY,Triangular,4,Area,MAE,U) (HSY,Triangular,2,Area,MSE,E) (HSY,Triangular,2,Area,MSE,N) (HSY,Triangular,4,Area,MSE,U)
[4]	241_241	✗	N/A
[4]	115_115	✗	N/A
[4]	117_117	✗	N/A
[8]	113_121	✓	(HSY,Row,2,Delay,MAE,E) (HSY,Row,2,Delay,MSE,E) (HSY,Row,2,Area,MAE,E) (HSY,Triangular,4,Area,MAE,N) (HSY,Triangular,4,Area,MAE,U) (HSY,Row,2,Area,MSE,E) (HSY,Triangular,4,Area,MSE,N) (HSY,Triangular,4,Area,MSE,U) (HSY,Triangular,4,Power,MAE,E) (HSY,Triangular,4,Power,MAE,N) (HSY,Triangular,4,Power,MAE,U) (HSY,Triangular,4,Power,MSE,E) (HSY,Triangular,4,Power,MSE,N) (HSY,Triangular,4,Power,MSE,U)
[8]	113_109	✗	N/A
[8]	113_97	✗	N/A
[8]	113_73	✗	N/A
[7]	51_43	✗	N/A
[7]	85_77	✗	N/A
[7]	240_232	✗	N/A

(Figs. 3a and 3d), the second column contains the power design metric pairs (Figs. 3b and 3e,), and the third column contains the delay design metric pairs (Figs. 3c and 3f). Each plot in particular shows the design metric on the x-axis (area, power, delay), and the error metric on the y-axis (MSE, MAE). Across the different utilized input distributions, it can be observed that dividers sampled through exponential and normal distributions cover the dividers found through uniform distributions, while additionally revealing further approximate dividers that are not in the uniform Pareto-optimal set. These additional dividers are important, as they enable engineers to match specific requirements for error metrics and design metrics better than only errors by utilizing a uniform input distribution. Furthermore, the increased number of dividers in the Pareto-optimal sets allows for further optimization to tailor dividers for specific applications. To briefly highlight this, we discuss some of the differences for particular metric pairs.

For the area design metric (Figs. 3a and 3d), the Pareto-optimal set of dividers reveals additional better dividers than dividers in the Pareto-optimal sets of uniform or exponential distribution. Particularly, for approximate dividers with around $10\,\mu m$ and $5\,\mu m$ the normal distribution shows better dividers. For a moderate amount of approximation that leads to dividers with an area greater than $15\,\mu m$, some dividers from the

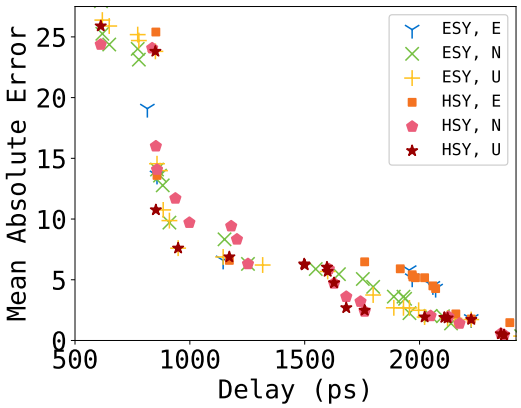
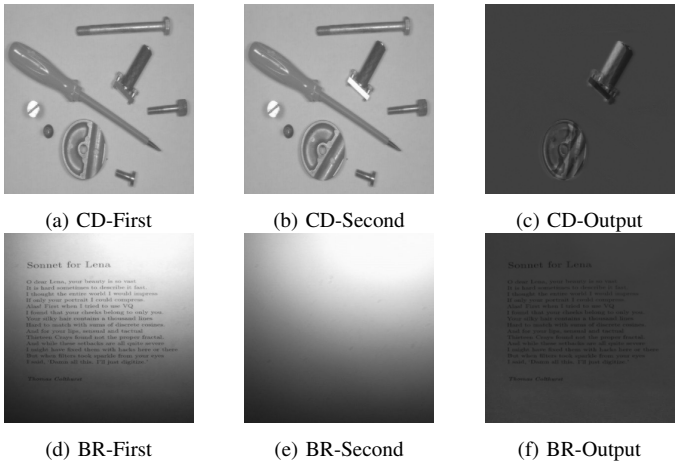


Figure 4: Ablative comparison of ESY and HSY for exponential, normal, and uniform input distributions.

Pareto-optimal set of the normal distribution are better than the uniform and exponential distributions. Furthermore, the gaps between two different points are also smaller when considering normal and exponential distributions. We also compared how dividers from prior works perform compared to our obtained Pareto-optimal sets. Tab. I shows dividers from prior works. The first column references the prior work a divider is from, the second column specifies the borrow (B) and difference (D) function used to approximate the dividers basic blocks, the third column shows if the divider is found in any Pareto-optimal set, and the last column shows the particular design space parameters for which the respective divider is in a Pareto-optimal set. While the first and third dividers in Tab. I from [4] and [8], respectively, are part of our Pareto-optimal sets, all other dividers do not perform good enough to be part of any Pareto-optimal set for any pair of design metrics and error metrics. Particularly, introducing higher errors in the divider without sufficient design space exploration of other dividers leads to non-optimal approximate dividers. As a reference, the basic block 113_105 specifies the correct borrow and difference functions. For example, the divider with the basic block 241_241 from [4] is not part in any Pareto-optimal set, but in our work, we identified a Pareto-optimal divider with the basic block 241_15 (the same borrow function, although the difference function is another). The divider 241_15 in our Pareto-optimal set for delay and MAE has an area of $2122\,\mu m^2$ with a MAE of 0.185, while any divider with the functions 241_241 has a higher area if the MAE is comparable.

b) Ablation with and without exact synthesis pre-optimization: As an additional design space parameter, we considered a pre-optimization step before the actual design synthesis. By applying Exact Synthesis (ESY) methods to the basic blocks of the RAD, additional approximate dividers can be revealed. As ESY methods are not scalable, they cannot be applied to the whole approximate circuit without significantly impacting the time to explore the design space efficiently. Hence, we explored the divider designs with the additional pre-



(a) CD-First (b) CD-Second (c) CD-Output

(d) BR-First (e) BR-Second (f) BR-Output

Figure 5: Application Images for change detection (CD) and background removal (BG).

optimization step, referred to as ESY, and without, referred to as Heuristic Synthesis (HSY), for an ablative study. For brevity, we illustrate the results for a particular pair of design metric and error metric (delay and MAE), but the trends are similar across the investigated metrics.

Fig. 4 shows the Pareto-optimal sets for delay and MAE across all three input distributions for divider designs synthesized with our pre-optimization step (ESY) and without the pre-optimization step (HSY). While in general, the dividers' designs across the Pareto-optimal sets follow the same trend, it can be seen that ESY and HSY can reveal different approximate divider designs. For both ESY and HSY, the different input distributions behave mostly consistently with the observations for Fig. 3. The utilization of this design space parameter revealed further divider designs, hidden without the utilization of pre-optimization or different input distributions.

V. Application case-study

In the last step, we illustrate the effectiveness of the obtained approximate dividers in a case study, utilizing all Pareto-optimal designs in two image processing applications. We use Change Detection (CD) and Background Removal (BG) applications from digital image processing, error-resilient applications in which approximate dividers can be used to obtain a feasible output image quality with better circuit performance. The input images for the CD applications are shown in Fig. 5a and Fig. 5b; the output image is shown in Fig. 5c. The input images for the BG applications are shown in Fig. 5d and Fig. 5e; the output image is shown in Fig. 5f.

For both applications, we perform the image processing task for every available Pareto-optimal approximate divider. The output image for a particular approximate divider is compared against the results obtained with an exact divider. This comparison is measured through the PSNR and SSIM quality metrics. For brevity, we discuss the quality metrics for the area design metric, but the trends are transferable to other design metrics as well.

Fig. 6 shows four plots, with the first row (Figs. 6a and 6b) showing the results of the CD application and the second row (Figs. 6c and 6d) showing the results of the BG application. The first column contains the plots with the PSNR metric (Figs. 6a and 6c, while the second row shows the plots with the SSIM metric (Figs. 6b and 6d). For both applications, the PSNR highlights the previously identified differences between the utilized input distributions further. Additional approximate dividers are revealed and available compared to the utilization of uniform distribution alone, providing engineers with more design choices to establish better performances for a target PSNR. Particularly, for the BG application at PSNRs between $30\,\mathrm{dB}$ to $40\,\mathrm{dB}$ shows a high density of Pareto-optimal dividers, which are not established with a uniform distribution. In the CD application and the same PSNR range, approximate dividers with better PSNR metrics are seen for exponential and normal distributions. Both of these trends can be observed as well for the SSIM quality metric on both applications. Here, a SSIM of 0.97 (for CD) and 0.95 (for BG) show approximate dividers from exponential and normal distributions, respectively.

The results of the evaluation can be summarized in three points. 1) We showed how the input distributions impact the design space exploration and the resulting Pareto-optimal sets, in favor of the consideration of the input distribution as a design space parameter. Utilization of exponential and normal distributions reveals additional approximate divider designs that are otherwise unidentified with the use of a uniform input distribution. 2) Through ablation of our pre-optimization step, additional designs were identified while confirming the trends across input distributions. 3) The utilization of case-study applications confirms the obtained Pareto-optimal approximate divider sets for real-world examples like CD and BG.

VI. Conclusion

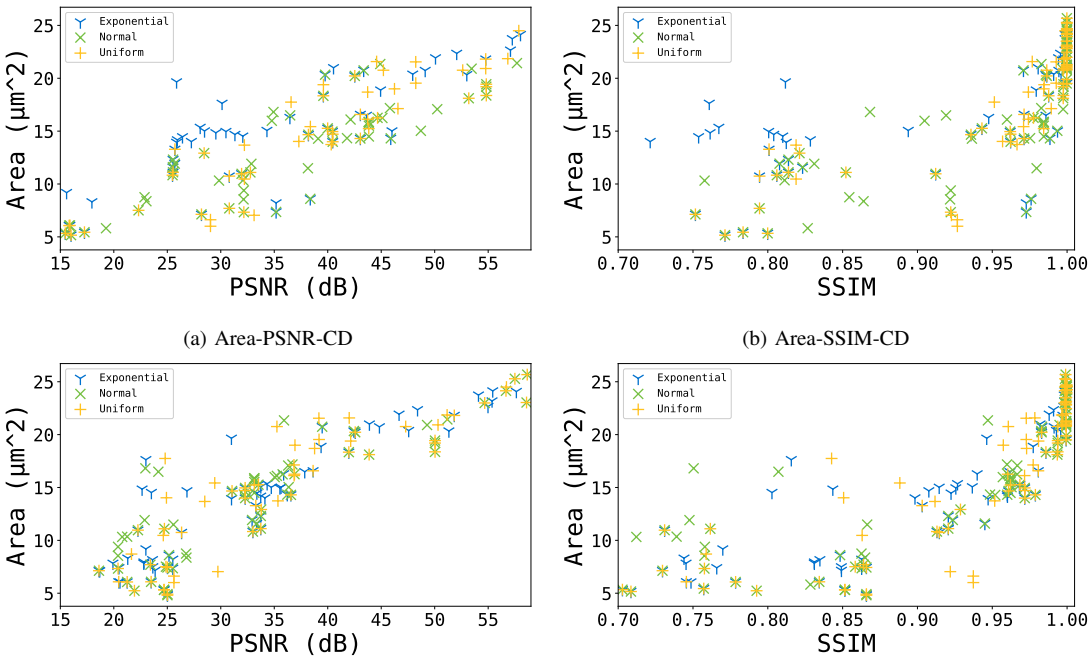
In this paper, we propose DIVIAC, an input distribution-aware library of approximate RAD. We evaluated over half a million designs to identify the Pareto-optimal designs for various combinations of design and error metrics across three input distributions. We performed exact minimization on the basic subtractor blocks in the divider, and then performed synthesis and compared it against heuristic-based synthesis. We also then used the Pareto-optimal designs in two image processing applications.

VII. Acknowledgements

This work was supported by the German Research Foundation (DFG) within the project PLiM (DR 287/35-2, project number 406079023), the German Ministry for Research, Technology and Space (BMFTR) within project ExaVerse (grant number 01IW25003), and the Data Science Center of the University of Bremen (DSC@UB) funded by the State of Bremen.

References

[1] P. Behrooz, "Computer arithmetic: Algorithms and hardware designs," *Oxford University Press*, vol. 19, pp. 512 583–512 585, 2000.



(a) Area-PSNR-CD

(b) Area-SSIM-CD

(c) Area-PSNR-BG

(d) Area-SSIM-BG

Figure 6: Application case-study for change detection (CD) and background removal (BG). Plots show PSNR and SSIM for both applications across exponential, normal, and uniform distributions.

[2] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 4, Aug. 2017.

[3] L. Chen, J. Han, W. Liu, and F. Lombardi, "On the design of approximate restoring dividers for error-tolerant applications," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2522–2533, 2015.

[4] C. Jha and J. Mekie, "Design of novel CMOS based inexact subtractors and dividers for approximate computing: An in-depth comparison with PTL based designs," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2019, pp. 174–181.

[5] 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.

[6] Y. Wu, C. Chen, W. Xiao, X. Wang, C. Wen, J. Han, X. Yin, W. Qian, and C. Zhuo, "A survey on approximate multiplier designs for energy efficiency: From algorithms to circuits," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 29, no. 1, Jan. 2024.

[7] K. M. Reddy, M. Vasantha, Y. N. Kumar, and D. Dwivedi, "Design of approximate dividers for error tolerant applications," in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2018, pp. 496–499.

[8] A. Gorantla and P. Deepa, "Design of approximate subtractors and dividers for error tolerant image processing applications," *Journal of Electronic Testing*, vol. 35, no. 6, pp. 901–907, 2019.

[9] C. K. Jha, S. Ahmadi-Pour, and R. Drechsler, "MARADIV: Library of magic-based approximate restoring array divider benchmark circuits for in-memory computing using memristors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 7, pp. 2635–2639, 2023.

[10] M. Barbareschi, S. Barone, A. Bosio, B. Deveautour, A. Piri, and M. Traiola, "Automatic generation of input-aware approximate arithmetic circuits," in *2025 IEEE 28th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*. IEEE, 2025, pp. 139–144.

[11] 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)*. IEEE, 2023, pp. 179–182.

[12] 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*, vol. 71, no. 7, pp. 3282–3293, 2024.

[13] M. Rezaalipour, M. Rezaalipour, M. Dehyadegari, and M. N. Bojnordi, "AxMAP: Making approximate adders aware of input patterns," *IEEE Transactions on Computers*, vol. 69, no. 6, pp. 868–882, 2020.

[14] M. Masadeh, O. Hasan, and S. Tahar, "Input-conscious approximate multiply-accumulate (MAC) unit for energy-efficiency," *IEEE access*, vol. 7, pp. 147 129–147 142, 2019.

[15] C. K. Jha, S. Ahmadi-Pour, and R. Drechsler, "Input distribution aware library of approximate adders based on memristor-aided logic," in *2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)*. IEEE, 2024, pp. 577–582.

[16] 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.

[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] 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.

[19] W. Snyder, "Verilator," https://verilator.org, 2025.

[20] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectronics Journal*, vol. 53, p. 105–115, Jul. 2016.

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

[22] 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.