

Approximate Reciprocal-based Divider

Ali Ghaderi¹, Nima Amirafshar² , Hadi Shahriar Shahhoseini¹ , and Nima TaheriNejad^{2,3} 

¹School of Electrical Engineering, Iran University of Science and Technology, Iran,

²Institute of Computer Engineering (ZITI), Heidelberg University, Germany,

³Institute of Computer Technology (ICT), Technische Universität Wien (TU Wien), Austria

Email: ali_ghaderi99@elec.iust.ac.ir, nima.amirafshar@ziti.uni-heidelberg.de, shahhoseini@iust.ac.ir,
nima.taherinejad@ziti.uni-heidelberg.de

Abstract—Recent years have seen a growing demand for smart, portable, and high-speed processing systems, posing significant challenges for digital circuit design in terms of power consumption, silicon area, and latency. Approximate Computing (AC) has emerged as an effective approach to address these challenges by allowing controlled computational errors in error-tolerant applications. In this paper, a novel approximate divider architecture based on a reciprocal algorithm is proposed. The proposed design achieves improvements of 86% in area, 92% in delay, and 79% in power consumption compared with an exact divider. The accuracy of the proposed divider is evaluated in image and signal processing applications. For change detection, it achieves a Structural Similarity Index Measure (SSIM) of 0.99 and a Peak Signal-to-Noise Ratio (PSNR) of 55 dB, compared with average values of 0.97 and 47.5 dB in state-of-the-art designs. In foreground extraction, the average SSIM and PSNR are 0.96 and 37.54 dB, respectively. For heartbeat QRS detection using the Pan–Tompkins algorithm, the proposed divider achieves a PSNR of 60 dB, demonstrating its suitability for error-resilient applications.

Index Terms—Approximate computing, reciprocal divider, image processing, signal processing, energy efficiency.

I. INTRODUCTION

Approximate Computing (AC) is an emerging computing paradigm that intentionally sacrifices computational accuracy to improve hardware efficiency in terms of execution time, memory usage, chip area, and energy consumption [1]–[4]. In fact, by exploiting the inherent error tolerance of many applications, it enables high-performance computing under strict power and resource constraints. This paradigm is therefore well suited for applications such as machine learning, scientific computing, and image and signal processing, where small inaccuracies can be tolerated in exchange for significant performance and energy gains [5], [6].

While approximate adders and multipliers have received considerable attention [4]–[6], approximate dividers remain comparatively underexplored despite their importance in numerically intensive workloads. The iterative nature of division and its sensitivity to operand scaling make it a strong candidate for algorithmic simplification, for example through truncated iterations of Newton–Raphson methods, coarse initial estimates, or quotient digit prediction, leading to tangible gains in throughput and energy efficiency. Such approximate dividers are particularly attractive for edge and embedded applications (e.g., IoT sensors, real-time control systems, and mobile devices), where tight energy budgets and latency constraints

dominate design objectives, as well as for graphics and simulation workloads where small, controlled errors are acceptable.

The main contribution of this paper is a novel approximate reciprocal-based algorithm that converts the division operation into multiplication using a simple and compact circuit, achieving higher power efficiency than conventional reciprocal-based approaches. By approximating the reciprocal of the divisor, the proposed design eliminates costly divider circuits and replaces them with simpler multiplier units that operate faster and consume less power, thereby improving energy efficiency and reducing latency. The remainder of this paper is organized as follows. Section II reviews related work, followed by a detailed description of the proposed method in Section III. Section IV presents the experimental results, and Section V presents and discusses application scenarios. Finally, Section VI concludes the paper with final remarks and directions for future research.

II. RELATED WORK

Historically, hardware division has been implemented using iterative subtraction-based array dividers, such as Restoring Array Dividers (RADs) [7] and Non-Restoring Array Dividers (NRADs), or by employing iterative algorithms such as the Newton–Raphson and Goldschmidt methods. Although exact designs guarantee high accuracy, they are often impractical for energy-constrained systems. Approximate divider designs aim to deliberately relax accuracy requirements while maximizing efficiency [8]. The following subsections summarize recent research trends in this area.

A. Approximate Array-based Dividers

Array dividers are natural candidates for approximation due to their regular structure, which consists of cascaded subtractor cells arranged sequentially [9]. Approximation can be introduced at the cell level, within specific regions of the array, or through dynamic adaptation of the array structure.

At the cell level, the most common strategy is to replace exact Full Subtractor (FS) cells with simplified Approximate Subtractor Cells (APSCs) [7]. APSC designs, such as Simplified Approximate Subtractor Cells (SAPSCs), achieve hardware savings by reducing the logic complexity of the difference output while keeping the borrow-out signal exact. This approach preserves correct borrow propagation and confines approximation errors to local effects. Approximation can also be selectively applied to specific regions of the array.

For example, the least significant bits (LSBs) of the quotient contribute less to overall accuracy, making them suitable candidates for approximation. Techniques such as Triangle Replacement (TR) and Horizontal Replacement (HR) define spatial patterns in which exact subtractors are replaced with APSCs. Architectures such as Approximate Restoring Divider (AXDrs) and Approximate Non-Restoring Divider (AXDnrs) adopt this principle, offering controllable error levels through selective approximation [7].

More advanced techniques allow the divider structure to dynamically adapt to input data or operating conditions. For example, the Dynamic Approximate Divider (DAXD) employs reduced-width restoring arrays when input characteristics permit, thereby improving energy efficiency at the cost of occasional overflow errors. To address such limitations, the Adaptively Approximate Divider (AAXD) uses pruning schemes guided by the position of the leading “1” in the divisor, together with error-correction logic to preserve accuracy. Similarly, the Reconfigurable Energy-Efficient Approximate Divider (READ) introduces reconfigurable subtractor cells that enable switching between exact and approximate computation modes, providing flexibility for systems that must balance energy constraints with accuracy requirements.

B. Multiplicative and Logarithmic Approximate Dividers

In some studies, division is reformulated as multiplication with the reciprocal of the divisor, thereby leveraging more efficient multiplier designs. Approximation in this domain is achieved either by simplifying the reciprocal estimation process or by adopting logarithmic number system (LNS) transformations. SEERAD (Rounding-based Approximate Divider) converts division into multiplication by approximating the reciprocal of the divisor using lookup tables (LUTs) and rounding operations [10]. TruncApp (Truncation-Based Approximate Divider) approximates division by multiplying a truncated dividend with an approximate reciprocal of the divisor. This architecture can also integrate approximate multipliers to further improve efficiency and often outperforms exact SRT radix-4 dividers in both speed and power consumption [11]. CADE (Configurable Approximate Divider for Energy Efficiency) targets floating-point division by approximating mantissa division. A small LUT is used for error compensation, enabling fine-grained control over the accuracy–efficiency trade-off [12].

Logarithmic-based approaches exploit the property that division can be transformed into subtraction in the logarithmic domain, as expressed by

$$\log_2\left(\frac{A}{B}\right) = \log_2(A) - \log_2(B) \quad (1)$$

Mitchell’s approximation, $\log_2(1+x) \approx x$, forms the basis of many logarithmic approximate divider designs. LEAD (Logarithmic-Based Approximate Divider) rounds operands to the nearest power of two and replaces division with shift operations and simple arithmetic. Its signed extension,

S-LEAD, supports signed division using the same principle [13]. LPCAD (Logarithmic Conversion with Piecewise Constant Approximation) improves upon Mitchell’s method by employing piecewise constant approximation (PCA), with constants heuristically selected to minimize the Mean Relative Error Distance (MRED) [14]. FPDME (Floating-Point Divider with Minimum MRED) approximates mantissa division $(1+M_x)/(1+M_y)$ using a linear function whose coefficients are derived via linear programming, ensuring minimal MRED [15]. INZeD and FaNZeD introduce bias-correction techniques to achieve near-zero error accumulation, making them suitable for iterative algorithms [15]. AXHD (Approximate Hybrid Divider) combines array-based and logarithmic approaches by computing the most significant bits (MSBs) exactly while approximating the least significant bits (LSBs) using logarithmic techniques [16]. RAPID (Pipelined Approximate Multiplier and Divider) is an FPGA-oriented design that employs LUT-based reciprocal approximation and pipelined datapaths to achieve high throughput while maintaining energy efficiency [17].

III. PROPOSED APPROXIMATE DIVIDER

In this section, we propose an approximate, hardware-efficient divider design based on a reciprocal algorithm. The key idea is to avoid the direct implementation of division, which is generally resource intensive, and instead reformulate the operation as a multiplication with a precomputed approximate reciprocal. Let us consider the exact division of two numbers:

$$Q = \frac{A}{B} \quad (2)$$

where A and B denoted the 8-bit integer dividend and the 4-bit divisor, respectively, and Q represents the final 8-bit integer quotient. To approximate this operation, we exploit the fact that the reciprocal of the divisor ($1/B$) can be expressed using a rational approximation as

$$\frac{1}{B} \approx \frac{K}{2^n} \quad (3)$$

where n is a fixed constant and K is an integer approximation of the scaled reciprocal. In this design, n is chosen to be 8, since the dividend A is 8-bit and the divider output is required to be 8-bit (i.e., Q). From Equation (3), the scaling factor K is computed as shown in Equation (4).

$$K = \frac{2^n}{B} \quad (4)$$

Thus, the division operation can be reformulated based on Equation (5). This transformation enables the division to be implemented using two simpler operations: (i) mapping the divisor B to the corresponding integer value K , and (ii) performing the integer multiplication of A and K , followed by a truncation operation, since n is chosen as a fixed constant.

$$\frac{A}{B} \approx \frac{A \times K}{2^8} \quad (5)$$

As illustrated in Figure 1, once K is obtained, the multiplication $A \times K$ is performed. For this operation, an exact

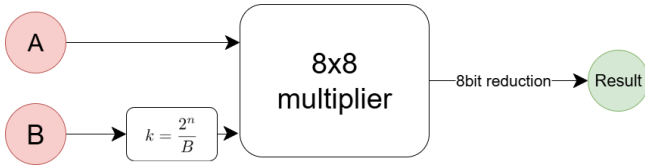


Fig. 1: Architecture of the proposed divider.

8-bit unsigned multiplier described in [5] is used. Considering 8-bit operands for both A and K , this multiplication produces a 16-bit result. However, because the division by 2^n (with $n = 8$) corresponds to discarding the lower 8 bits of the product, only the upper 8 bits of the multiplication result are retained. This truncation step reduces hardware overhead by eliminating the need for additional shifting units or fractional-handling circuitry.

To implement the mapping from B to K , eight complete truth tables are constructed, one for each bit of K , based on the 4-bit divisor $B = (b_3b_2b_1b_0)_2$ over all possible input combinations. As shown in Equations (6) to (13), the minimized Boolean functions for each bit of $K = (k_7k_6k_5 \dots k_0)_2$ are then derived using the Quine–McCluskey method, which systematically reduces logic redundancy. As a result, the mapping can be implemented with minimal combinational logic, significantly reducing hardware complexity.

$$k_0 = b_3b_2\bar{b}_1\bar{b}_0 + \bar{b}_3b_2b_1\bar{b}_0 + b_1b_0 + \bar{b}_3\bar{b}_1b_0 \quad (6)$$

$$k_1 = b_3\bar{b}_2b_1b_0 + \bar{b}_3b_2b_1\bar{b}_0 + b_3b_1\bar{b}_0 + \bar{b}_3\bar{b}_1b_0 \quad (7)$$

$$k_2 = \bar{b}_3b_2b_1b_0 + b_3\bar{b}_2b_1b_0 + b_3b_2\bar{b}_1\bar{b}_0 + \bar{b}_3\bar{b}_2b_0 + b_3\bar{b}_1b_0 \quad (8)$$

$$k_3 = b_3\bar{b}_2b_1\bar{b}_0 + \bar{b}_2\bar{b}_1b_0 + \bar{b}_3b_2b_1\bar{b}_0 \quad (9)$$

$$k_4 = b_3b_2\bar{b}_1\bar{b}_0 + b_3b_1\bar{b}_0 + b_3\bar{b}_1b_0 + b_1b_0 + \bar{b}_3\bar{b}_1 \quad (10)$$

$$k_5 = b_3\bar{b}_2\bar{b}_1\bar{b}_0 + \bar{b}_3b_2b_1b_0 + \bar{b}_3b_2b_1\bar{b}_0 + \bar{b}_3\bar{b}_1b_0 \quad (11)$$

$$k_6 = \bar{b}_3b_2\bar{b}_1\bar{b}_0 + \bar{b}_3\bar{b}_2b_0 \quad (12)$$

$$k_7 = \bar{b}_3\bar{b}_2b_1\bar{b}_0 + \bar{b}_3\bar{b}_2\bar{b}_1b_0 \quad (13)$$

Through this approach, the proposed divider achieves a substantial reduction in hardware complexity compared with conventional division circuits. The use of reciprocal mapping combined with logic minimization enables a low-area, high-speed implementation, making the proposed method well suited for digital signal processing and embedded hardware applications.

IV. EXPERIMENTS AND RESULTS

This section presents a detailed evaluation of the hardware cost and accuracy analysis of the proposed approximate divider, highlighting both its strengths and limitations as revealed through experimental results. The proposed approximate divider was designed in Verilog and synthesized using 45-nm NanGate technology with Cadence Genus to evaluate critical path delay, power consumption, and area, as reported in Table I. In addition, a behavioral model of the divider was implemented in Python to evaluate accuracy. To quantify the error of the proposed divider, the MRED [20] is used as the

error metric. The MRED is computed over all possible input combinations in the Python implementation.

As shown in Table I, the proposed divider achieves an MRED of 1.8%, indicating that the approximation error remains at an acceptable level. Furthermore, compared with an exact non-restoring divider [18], the proposed design achieves improvements of 86% in area and 92% in critical path delay, along with a 79% reduction in power consumption.

Compared with other approximate dividers implemented using the same technology, as summarized in Table I, the proposed divider achieves average improvements of 71% in area, 56% in power consumption, and 60% in critical path delay. Compared with TruncApp [11], although TruncApp achieves a smaller area, the proposed divider provides 56% lower power consumption, 23% lower delay, and an 8.2% improvement in MRED. Compared with HEADiv(R) [18], the proposed design achieves improvements of 65% in area, 88% in delay, and 70% in power consumption; however, HEADiv(R) exhibits a lower MRED. Compared with SEERAD [10], the proposed divider achieves improvements of 61% in area, 33% in delay, and 71% in power consumption, with an MRED of 1.6%.

V. APPLICATIONS IN IMAGE AND SIGNAL PROCESSING

The performance of the proposed approximate divider is evaluated in image and signal processing applications to demonstrate its effectiveness in practical, real-world scenarios.

1) **Change Detection:** Image change detection is the process of identifying and analyzing differences between two or more images of the same scene acquired at different times. It has wide applications in areas such as remote sensing, surveillance, and medical imaging [8].

To evaluate the effectiveness of the proposed design, experiments were conducted using the CDnet dataset [27]. The performance of our approximate divider was compared with other divider implementations, with detailed results reported in Table II. This comparison highlights the accuracy and efficiency trade-offs introduced by approximation while demonstrating the suitability of the proposed design for change detection applications. Compared with an exact divider, the proposed design exhibits a PSNR degradation of 5 dB, which remains acceptable for image processing applications. Moreover, compared with the average performance of other approximate designs, the proposed divider achieves a higher PSNR.

2) **Foreground Extraction:** Foreground extraction is the process of isolating and identifying moving or salient objects of interest in an image or video sequence, typically by separating them from the background. It is a fundamental step in applications such as surveillance (object and human activity detection), human–computer interaction, medical imaging, and autonomous systems [14]. Similar to the change detection experiment, this algorithm was implemented in Python, and the exact divider was replaced with the proposed approximate divider. The results, compared with other approximate divider designs, are presented in Table III. The comparison indicates that the proposed divider achieves performance close to the

TABLE I: Hardware and accuracy comparison of the proposed divider with exact and approximate designs.

Divider	Area (μm^2)	Delay (nS)	Power (mW)	PDP (pJ)	MRED (%)
Exact Non-Restoring [18]	2335	14.45	0.39	5.64	0
HEADiv(C) [18]	1315	8.53	0.27	2.30	0.73
HEADiv(R) [18]	901	8.72	0.12	1.05	1.82
SAADI (6,5) (Combinatorial Logic) [19]	1879	11.84	0.32	3.79	1.82
SAADI (5) (Combinatorial Logic) [19]	1606	10.20	0.30	3.06	2.67
SAADI (4) (Combinatorial Logic) [19]	1334	8.55	0.27	2.31	2.70
SAADI (3) (Combinatorial Logic) [19]	1063	6.90	0.23	1.59	3.47
SAADI (6) (Resource Sharing) [19]	804	13.02	0.09	1.18	-
SAADI (5) (Resource Sharing) [19]	804	10.85	0.09	0.98	-
SAADI (4) (Resource Sharing) [19]	804	8.68	0.09	0.78	-
SAADI (3) (Resource Sharing) [19]	804	6.51	0.09	0.59	-
SEERAD (4) [10]	588	2.67	0.16	0.43	2.47
SEERAD (3) [10]	439	2.31	0.12	0.28	4.64
TruncApp (4) [11]	391	1.56	0.26	0.41	4.26
TruncApp (3) [11]	305	1.37	0.19	0.26	10.03
Proposed	313.88	1.049	0.0823	0.086	1.8

TABLE II: Comparison of the proposed design with other approximate designs for change detection.

Name	PSNR(dB)	SSIM
Exact	60	0.99
LPCAD(3,3) [14]	37	0.95
LPCAD(3,4) [14]	42	0.98
LPCAD(3,6) [14]	46	0.98
LPCAD(3,8) [14]	53	0.99
PLSAD(8) [21]	47.5	0.99
PLSAD(6) [21]	43.7	0.98
PLSAD(4) [21]	32.7	0.89
INT(8/4) [14]	36	0.92
Proposed	55	0.99

TABLE III: Comparison of the proposed divider with foreground extraction approaches.

Name	PSNR(dB)	SSIM
Exact	50	0.99
LPCAD(2,10)	43	0.98
ALD [22]	35	0.96
FaNZeD [15]	37	0.96
TrunApp4 [11]	32	0.88
FPAD33 [23]	33	0.92
FPAD43 [23]	36	0.93
AAXD [24]	40	0.95
Proposed	37.54	0.96

average of state-of-the-art approximate designs, while exhibiting a PSNR reduction of approximately 12 dB compared with the exact divider.

3) *Heartbeat QRS Detection Using the Pan-Tompkins Algorithm*: Studies indicate that the processing stage accounts for nearly 70% of the total energy consumption in wearable health-monitoring nodes. Both sensing mechanisms and signal processing algorithms in bio-signal analysis exhibit a certain degree of error resilience, making them promising candidates for approximate computing techniques. Among these algorithms, the widely adopted Pan-Tompkins method remains the de facto standard for QRS detection in electrocardiogram (ECG) signals, where cardiac arrhythmias manifest as irregular waveforms [25]. The Pan-Tompkins algorithm was implemented in Python, and its exact division operations were replaced with the proposed approximate divider. The results, reported in Table IV, show that the performance of

TABLE IV: Comparison of the proposed design with other approaches for heartbeat QRS detection.

Name	PSNR(dB)
Exact	60.0
RAPID3 [25]	34.7
RAPID10 [25]	39.4
SIMDIVE [26]	41.4
DRUM/AAXD [24]	28.5
Proposed	60.0

the proposed divider is equivalent to that of the exact divider, demonstrating its suitability for QRS detection applications.

VI. CONCLUSION

In this paper, an approximate computing technique is applied to a divider architecture to improve hardware efficiency and reduce resource utilization. The proposed approach achieves a favorable trade-off among computational accuracy, power consumption, and circuit complexity, while maintaining acceptable accuracy. The proposed approximate divider is based on a reciprocal approximation algorithm and demonstrates improvements of 86% in area, 92% in critical path delay, and 79% in power consumption compared with an exact non-restoring divider. The divider was evaluated across multiple image processing applications, including change detection and foreground extraction, as well as a signal processing application for heartbeat QRS detection. Experimental results show that the proposed design achieves acceptable performance compared with other state-of-the-art approximate dividers, confirming its suitability for practical applications.

Future research can be directed toward several promising extensions of this study. One direction is the design of higher bit-width dividers based on the proposed 8-bit divider architecture, enabling scalability to applications that require increased accuracy. Another possible direction is the design of approximate floating-point dividers that preserve accuracy in critical bit regions while achieving high hardware efficiency.

ACKNOWLEDGMENT

The authors gratefully acknowledge the funding support from the Hector Stiftung (2304191).

REFERENCES

- [1] Honglan Jiang, Francisco Javier Hernandez Santiago, Hai Mo, Leibo Liu, and Jie Han. Approximate arithmetic circuits: A survey, characterization, and recent applications. *Proceedings of the IEEE*, 108(12):2108–2135, 2020.
- [2] K. Manikanta Reddy, Y. B. Nithin Kumar, Dheeraj Sharma, and M. H. Vasantha. Low power, high speed error tolerant multiplier using approximate adders. In *2015 19th International Symposium on VLSI Design and Test*, pages 1–6, 2015.
- [3] Suganthi Venkatachalam and Seok-Bum Ko. Design of power and area efficient approximate multipliers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(5):1782–1786, 2017.
- [4] Ayad M. Dalloo, Amjad Jaleel Humaidi, Ammar K. Al Mhdawi, and Hamed Al-Raweshidy. Approximate computing: Concepts, architectures, challenges, applications, and future directions. *IEEE Access*, 12:146022–146088, 2024.
- [5] Nima Amirafshar, Gulafshan Gulafshan, Hadi Shahriar Shahhoseini, and Nima TaheriNejad. Prim: Hybrid array-compressor multipliers with carry disregard and or-based approximation. In *2025 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2025.
- [6] Salar Shakibhamedan, Nima Amirafshar, Ahmad Sedigh Baroughi, Hadi Shahriar Shahhoseini, and Nima TaheriNejad. Ace-cnn: Approximate carry disregard multipliers for energy-efficient cnn-based image classification. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(5):2280–2293, 2024.
- [7] Abhay Shriram, Ayush Tiwari, U. Anil Kumar, Babu Ravi Teja Karri, Sreehari Veeramachaneni, and Syed Ershad Ahmed. Power efficient approximate divider architecture for error resilient applications. pages 1–6, 2022.
- [8] Komathy Vanitha Krishnan, Anila Satish, and Pradeev raj Krishnan. Design of energy efficient approximate subtractors and restoring dividers for error tolerant applications. *Microelectronics Journal*, 131:105668, 2023.
- [9] Chandan Kumar Jha, Khushboo Qayyum, Muhammad Hassan, and Rolf Drechsler. Farad: Automated formal verification of approximate restoring array dividers. In *2025 38th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)*, pages 43–48, 2025.
- [10] Reza Zendegani, Mehdi Kamal, Arash Fayyazi, Ali Afzali-Kusha, Saeed Safari, and Massoud Pedram. Seerad: A high speed yet energy-efficient rounding-based approximate divider. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1481–1484, 2016.
- [11] Shaghayegh Vahdat, Mehdi Kamal, Ali Afzali-Kusha, Massoud Pedram, and Zainalabedin Navabi. Truncapp: A truncation-based approximate divider for energy efficient dsp applications. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017, pages 1635–1638, March 2017.
- [12] Mohsen Imani, Ricardo Garcia, Andrew Huang, and Tajana Rosing. Cade: Configurable approximate divider for energy efficiency. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 586–589, 2019.
- [13] Omkar G. Ratnaparkhi and Madhav Rao. Lead: Logarithmic exponent approximate divider for image quantization application. In *Proceedings of the Great Lakes Symposium on VLSI 2022*, GLSVLSI '22, page 437–442, New York, NY, USA, 2022. Association for Computing Machinery.
- [14] Yong Wu, Honglan Jiang, Zining Ma, Pengfei Gou, Yong Lu, Jie Han, Shouyi Yin, Shaojun Wei, and Leibo Liu. An energy-efficient approximate divider based on logarithmic conversion and piecewise constant approximation. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(7):2655–2668, 2022.
- [15] Gennaro Di Meo, Antonio Giuseppe Maria Strollo, and Davide De Caro. Novel low-power floating-point divider with linear approximation and minimum mean relative error. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 70(12):5275–5288, 2023.
- [16] Weiqiang Liu, Tao Xu, Jing Li, Chenghua Wang, Paolo Montuschi, and Fabrizio Lombardi. Design of unsigned approximate hybrid dividers based on restoring array and logarithmic dividers. *IEEE Transactions on Emerging Topics in Computing*, 10(1):339–350, 2020.
- [17] Zahra Ebrahimi, Muhammad Zaid, Mark Wijtvliet, and Akash Kumar. Rapid: Approximate pipelined soft multipliers and dividers for high throughput and energy efficiency. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(3):712–725, 2022.
- [18] Hanghang Wang, Ke Chen, Bi Wu, Chenghua Wang, Weiqiang Liu, and Fabrizio Lombardi. Headiv: A high-accuracy energy-efficient approximate divider with error compensation. In *Proceedings of the 17th ACM International Symposium on Nanoscale Architectures*, NANOARCH '22, New York, NY, USA, 2023. Association for Computing Machinery.
- [19] Jackson Melchert, Setareh Behrooz, Jingjie Li, and Younghyun Kim. Saadi-ec: A quality-configurable approximate divider for energy efficiency. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(11):2680–2692, 2019.
- [20] Zdenek Vasicek. Formal methods for exact analysis of approximate circuits. *IEEE Access*, PP:1–1, 12 2019.
- [21] Chaoyuan Wu, Weiwei Shi, Yida Yuan, Zhuoliang Zou, Zhihong Mo, and Jiangwei He. Area-delay-energy-efficient approximate dividers based on piecewise linear fitting of surface. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(11):5017–5029, 2024.
- [22] John N. Mitchell. Computer multiplication and division using binary logarithms. *IRE Transactions on Electronic Computers*, EC-11(4):512–517, 1962.
- [23] Yong Wu, Honglan Jiang, Zining Ma, Pengfei Gou, Yong Lu, Jie Han, Shouyi Yin, Shaojun Wei, and Leibo Liu. An energy-efficient approximate divider based on logarithmic conversion and piecewise constant approximation. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(7):2655–2668, 2022.
- [24] Jonghyun Jeong and Youngmin Kim. Asad-rd: Accuracy scalable approximate divider based on restoring division for energy efficiency. *Electronics*, 10(1), 2021.
- [25] Zahra Ebrahimi, Muhammad Zaid, Mark Wijtvliet, and Akash Kumar. Rapid: Approximate pipelined soft multipliers and dividers for high throughput and energy efficiency. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(3):712–725, 2023.
- [26] Zahra Ebrahimi, Salim Ullah, and Akash Kumar. Simdive: Approximate simd soft multiplier-divider for fpgas with tunable accuracy. In *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, GLSVLSI '20, page 151–156, New York, NY, USA, 2020. Association for Computing Machinery.
- [27] Yi Wang, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, Yannick Benezeth, and Prakash Ishwar. Cdnets 2014: An expanded change detection benchmark dataset. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 393–400, 2014.