# Efficient Decoder-Driven Multiplier for Power-Efficient Computing

Deepak Giri[1], Sachin Bandewar[2]
*M.Tech.Scholar[#1], Assistant Professor[#2]*
*Department of Electronics and Communication Engineering*
*Sri Satya Sai College of Engineering, RKDF University, Bhopal, (M.P.) India*
*sachin.bandewar9@gmail.com*

* Corresponding Author: Deepak Giri

**Abstract: -** This study investigates the potential benefits of approximate computing within the realm of image processing, with a specific focus on the Decoder Driven Multiplier (DDM) architecture. The research leverages the inherent error-resilient characteristics of image processing applications, aiming to reduce computational resource requirements. Through extensive experimentation and analysis, it is demonstrated that the DDM architecture effectively reduces the generation of Partial Products (PPs) during calculations, thereby simplifying the overall computational process. Results from both error and hardware evaluations indicate notable improvements in energy efficiency and area utilization compared to existing designs. Furthermore, the study showcases the architecture's ability to strike an optimal balance between computational quality and resource efficiency. As a result, this work not only holds promise for enhancing image processing but also opens avenues for exploring the wider applicability of approximate computing in diverse computational domains.

**Keywords: -** Field Programmable Gate Array, Artificial Intelligence, Machine Learning, Internet of Things

## 1. Introduction

The increasing necessity for energy-efficient computation is driven by several factors, encompassing environmental sustainability, cost savings, and improved performance across a diverse array of applications. This approach to computing centers on the development of hardware and software solutions that execute tasks with minimal energy consumption [1]. Multiple factors contribute to the mounting significance of energy efficiency. The awareness of the imperative to combat climate change and protect the environment has led to initiatives aimed at reducing energy consumption across various industries, including the computing sector. Facilities like data centers and high-performance computing installations consume substantial amounts of electricity, making enhancing their energy efficiency instrumental in reducing their environmental impact [2].
Approximation computing revolves around the utilization of imprecise methods to perform calculations, prioritizing gains in energy efficiency, speed, and resource utilization over achieving exact results. This approach is particularly advantageous in scenarios where ultra-high precision is not essential, such as in battery-powered devices, real-time applications, and resource-constrained environments like IoT systems [3]. This technique can substantially reduce energy consumption and enhance computational speed. Numerous research papers have proposed diverse types of approximate multipliers, employing techniques like error accumulation and truncation of least significant bits, to attain these efficiency gains. Various approximate compressors and multipliers have been introduced in the literature to minimize delay and power consumption in computing tasks [4][5]. These designs, such as ACM3 and ACM4, primarily apply approximation techniques to the least significant bits (LSBs) while maintaining the utmost precision in the most significant bits (MSBs). Various models, such as the broken-array, Wallace tree, and error-tolerant multipliers, employ approximation at different stages of the multiplication process. Some even employ logarithmic approximations to further streamline calculations. These techniques aim to establish more energy-efficient and expeditious computing systems [6]-[10].

## 2. Literature Review

Nimbi et al. [1] introduced a unique multiplier design grounded in decoder logic, aiming to minimize the number of partial products produced.
Sathish et al.[2] introduced a novel approach that leverages the segmentation of partial products through recursive multiplication by compressors in their approximate multipliers.
Liu et al. [3] introduced a Mitchell multiplication algorithm refined through piecewise linear logarithmic approximation.

Arya et al.[4] introduced the Reconfigurable Energy-efficient Approximate Divider (READ) built upon the fixed restoring array divider structure. When benchmarked against a precise 16⁄8 divider design, READ demonstrated a 49% boost in energy efficiency and operated 1.26 times faster, with only slight accuracy deviations.
SenthilKuma et al.[5] introduced a compressor design that simplifies system operations by cutting down on the number of XOR gates utilized. The results highlighted a notable reduction in power usage with only a slight compromise in accuracy.
Zhu et al.[6] introduced designs for approximate-truncated Booth multipliers (ATBMs) that leverage their novel approximate modified radix-4 Booth encoders (AMBEs), approximate 4-2 compressors (ACs), and a tiered truncation of partial products.
Qui et al.[7] introduced a pair of innovative approximate circuit design techniques anchored in machine learning. Instead of intricate manual evaluations, these methods prioritize input-error patterns to shape the approximate circuits.
Devi et al.[8] introduced a perspective on computational efficiency as technological scaling nears its boundaries. Their focus was on the approximate design of the RB-Normal Binary (NB) converter within the RB multiplier, taking into account the error characteristics of both the estimated Booth encoders and the RB compressors.
Mittal et al.[9] offer an overview of methods related to approximate computing (AC). The review delves into identifying sections of programs suitable for approximation, ways to ensure output quality, and the integration of AC across various processing units, including CPUs, GPUs, and FPGAs.
Narayanamoorthy et al.[10] introduced multiplier designs that allow a balance between computational precision and energy usage during the design phase.
Jiang et al.[11] offer an analysis and categorization of contemporary designs in approximate arithmetic circuits, encompassing elements like adders, multipliers, and dividers. The study conducts an exhaustive comparison of the error rates and circuitry traits across these designs, aiding in grasping the nuances of each.
Parhami et al.[12] have made significant contributions through their investigative studies and tool development, elevating computer architecture from a mere art form to a notably quantitative domain within computer science and engineering.
Liang et al.[13] introduced novel criteria for assessing both the dependability and energy efficiency of probabilistic and approximate adders.
Yi et al.[14] introduced an innovative circuit design that alters the circuit's architecture without affecting its operation. AWM3 achieves area reductions of up to 48.077% and minimizes delay by up to 46.633%. Meanwhile, AWM4 showcases reductions of up to 53.846% in size and up to 56.482% in processing time
Bhardwaj, et al.[15] introduces an approximate multiplication technique that takes into account bit-width for enhancing the design of our multiplier. Through simulations, the team found that the designs have an average accuracy ranging between 99.85% and 99.965%.
Rao et al.[16] introduced the Rounding-based AM (RAM) by adapting the Karatsuba algorithm. Through simulations, it was revealed that the newly introduced RAM designs for 8 and 16 bits showcased significant improvements. Specifically, when integrated with the ISFA, the RAM's SSIM and PSNR values ranged from 1.44% to 84.47% and 0.28% to 24.4% respectively, outperforming the ISFA-integrated existing AMs.
Yang et al.[17] introduced three innovative compressors designed for partial product reduction (PPR) within multipliers, ensuring they maintain a level of accuracy. These designs incorporate both approximation and truncation strategies.
Ha et al.[18] introduced an enhanced design that builds upon a prior approximate 4–2 compressor model by incorporating an error correction component.
Hashimi et al.[19] developed a multiplier with an emphasis on achieving an unbiased error distribution. The errors introduced have a Gaussian distribution, characterized by an almost negligible average and standard deviations ranging from 0.45% to 3.61%.
Venkatachalam et al.[20] introduced optimized approximate multipliers where the multiplier's partial products undergo modifications using generate and propagate signals. These approximations were incorporated into two distinct 16-bit multiplier models. Synthesis data indicates that the newly proposed multipliers offer power efficiencies of 72% and 38% when set against a standard multiplier.

## 3. Proposed Methodology

The proposed Dynamic Data Multiplier (DDM) architecture aims to efficiently perform multiplication by using a unique combination of decoder logic blocks and traditional AND gate logic. The multiplier A is divided into groups: (a0a1,a2a3,a4a5) are groups of 2 bits each, while a6 and a7 are the most significant bits. The decoder logic blocks generate approximate partial products (PPs) for the lower significant bits (a0a1,a2a3,a4a5). The most significant bits a6 and a7 generate exact PPs using traditional AND gate logic. The approximation is only in the least significant portion of the result, which minimizes the overall error. Since the decoder logic takes 2 bits at a time, there are only four unique bit combinations to consider for each pair. In summary, the DDM uses a hybrid approach, blending approximate and exact calculations to achieve efficient multiplication. It uses novel decoder

logic for the less significant bits and conventional logic for the most significant bits, thus minimizing error while saving on hardware resources.

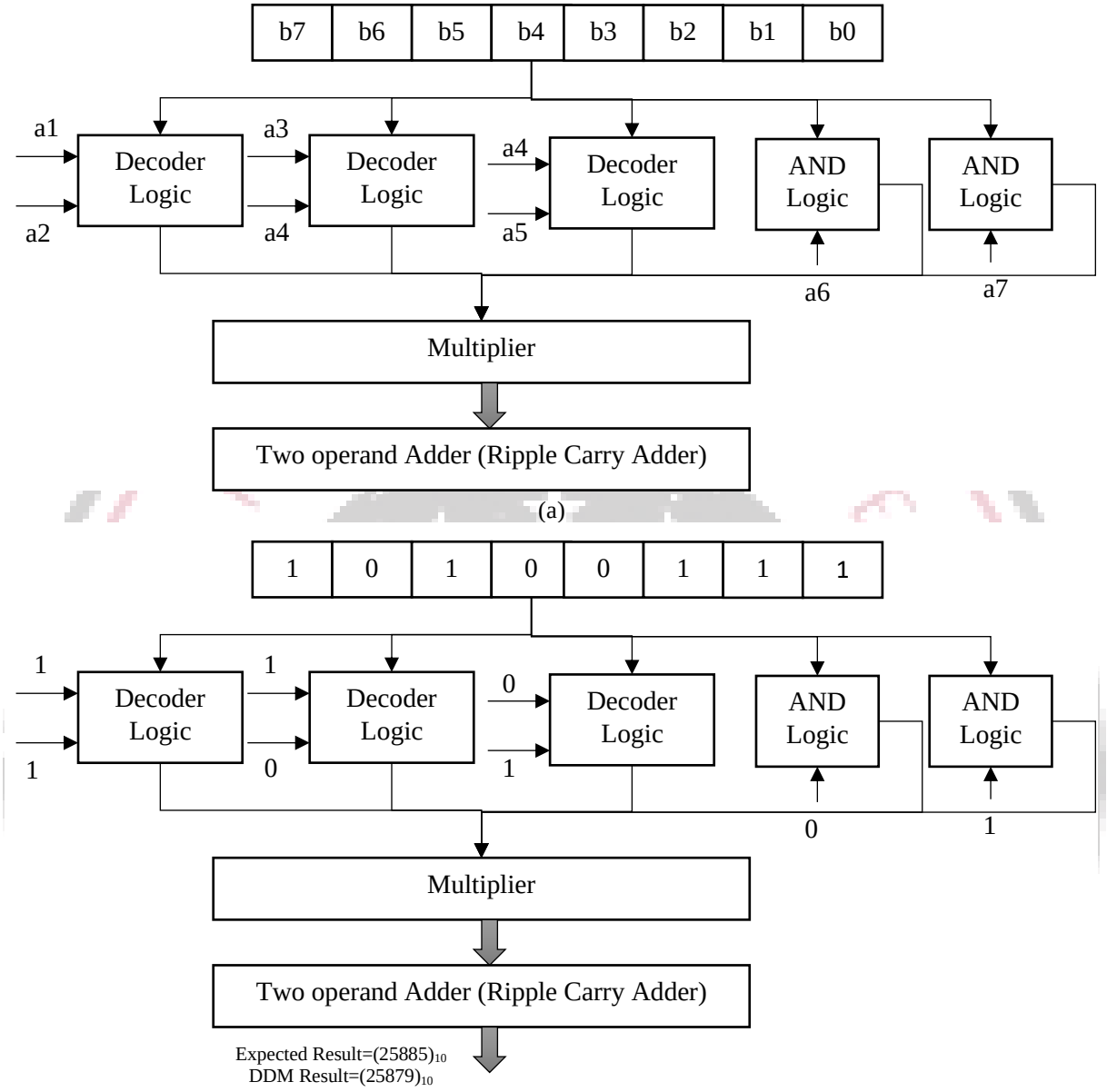


**Fig. 1: (a) Block Diagram of DDM architecture (b) DDM technique illustrated using a numerical example**

The proposed DDM architecture aims to optimize multiplier operations by using a specialized decoder logic block for partial product (PP) generation. In this architecture, the multiplier A is divided into three 2-bit groups (a0a1, a2a3, a4a5) for which approximate PPs are generated. On the other hand, the most significant bits a7 & a6 are used to generate exact PPs using conventional AND gate logic. By generating approximate PPs for the lower significant bits, computational efficiency is improved. Limiting the error to the least significant bits ensures that the overall output remains relatively accurate. Moreover, using 2-bit groups for the select logic narrows down the possible bit combinations to just four unique cases, further simplifying the architecture. The functionality of the decoder logic block is determined by these 2-bit select inputs, as detailed in Table 1.

**Table 1: Decoder Logic Truth Table**

| ax+1 | ax | Decoder output |
|---|---|---|
| 0 | 0 | Zero |
| 0 | 1 | b7-b0 |
| 1 | 0 | (b7-b0) <<1 |
| 1 | 1 | (b7-b0) OR ((b7-b0) <<1) |

The DDM architecture uses a 2-bit select input for its decoder logic block, which leads to four possible combinations, as detailed in a table. The decoder block passes the multiplicand B as output when the multiplier bits are "01". Approximate partial products (PPs) are only generated when the select inputs ax+1 are "11". In this case, the multiplicand B and its left-shifted version B<<1 are added using OR logic to obtain the PP. The output of each decoder logic block is organized based on the positional weight of the multiplier bits. The most significant bits a7 and a6 are used to generate exact PPs using traditional AND gate logic, thereby enhancing the accuracy of the overall multiplier operation.

The proposed DDM architecture consists of two main stages:

Partial Product Generation (PPG) Stage: Decoder logic blocks take 2-bit select inputs and a multiplicand to generate approximate PPs for the less significant bits, while the most significant bits a7a6 produce exact PPs. The logic for this is illustrated in Fig.1(a).

Accumulation Stage: The PPs are then processed by a 3:2 carry-save adder, reducing the initial 5 PP rows into just two. This minimizes the number of adders needed, resulting in a significant reduction in both area and power usage.

Finally, the two remaining rows from the accumulation stage are condensed into the final product p15−p0 using a ripple carry adder. Fig.1(b) provides a numerical example to demonstrate the DDM architecture. It shows that three rows of PPs are generated by the decoder blocks based on the input select bits, while the remaining two rows are produced using conventional AND gate logic. These rows are then reduced to a final product using a reduction structure. The result obtained with DDM is almost as accurate as using a traditional, more resource-intensive method.

The circuit-level implementation of the proposed DDM architecture for an 8x8 multiplier. The decoder logic block functions based on the 2-bit select input ax+1.

- For ax+1="00", the output is zero.
- For ax ax+1="01", the partial product (PP) is directly formed by the multiplicand b7−b0.
- For ax ax+1="10", the PP is (b7−b0) left-shifted by one bit.

The combinational circuit to implement these three cases requires a minimal number of gates, as it involves either the conditional transfer of input bits or producing a zero.

The final case, ax+1="11", is computed as the sum of the PPs generated when ax+1="01" and "10". The adder logic for this scenario is implemented using a simple OR gate.

The proposed DDM multiplier architecture is scalable and can be extended for higher-bit multiplication. This is achieved through parameterized Verilog HDL code, using two parameters: N and M. Here, N represents the total number of bits being multiplied, and M indicates the number of accurate blocks. For example, the initial 8x8 design would be denoted as DDM(8,2).The total number of decoder logic blocks is given by (N−M)/2, which also corresponds to the number of reduced partial product (PP) rows.

To further optimize the design, the output of solid OR gates is shared across multiple decoder logic blocks, reducing hardware requirements. This is possible because the multiplicand input remains constant across all blocks. This parameterized design offers flexibility, allowing users to balance between better accuracy and reduced hardware size, depending on application requirements.

## 4. Result Analysis

The proposed DDM (Dynamic Digital Multiplier) was tested against other existing approximate multipliers using 8-bit and 16-bit cases. Performance metrics like error rate, and MRED (Mean Relative Error Distance).

The DDM (Dynamic Digital Multiplier) was rigorously tested against existing 8-bit and 16-bit multipliers using Verilog HDL, Cadence simulators, and TSMC 180nm process node for hardware analysis. Metrics like area, delay, and power were evaluated.

For 8-bit multipliers

- DDM (8,2) showed 6.4% less area and 10% less power consumption compared to existing design.
- It achieved 17.7%, 11.8%, and 9.2% improvements in area, delay, and power compared to AWTM-4.
- The area was 30.13% less compared to DRUM4, although DRUM4 has lower power consumption but suffers from a high error rate.
- It also had 14.9% less area and 8.3% less power consumption compared to Xilin et al. [21] design, but a 13.46% increase in delay.

Overall, it achieved up to 30.13% area and 22.3% power improvements compared to existing designs, and 31.34% in area and 40.96% in power compared to an accurate multiplier.
For 16-bit multipliers

- DDM (16,M) generally consumed less power and occupied less area compared to most existing designs, except for Multiplier1, Xilin, and DRUM4.
- Designs which were more power and area-efficient than DDM (16,M) had lower accuracy.

In summary, DDM offers a superior trade-off between computation quality and resource utilization compared to existing designs.
The performance of the multiplier is evaluated using the quality metrics, mean square error (MSE) and peak signal-to-noise ratio (PSNR). The fig 2 compares the error rates of two types of multipliers: Xilin [21] has an error rate of 66.24, while DDM (N,N/4) has a lower error rate of 45.39. Based on error rates alone, DDM (N,N/4) appears to be more accurate.

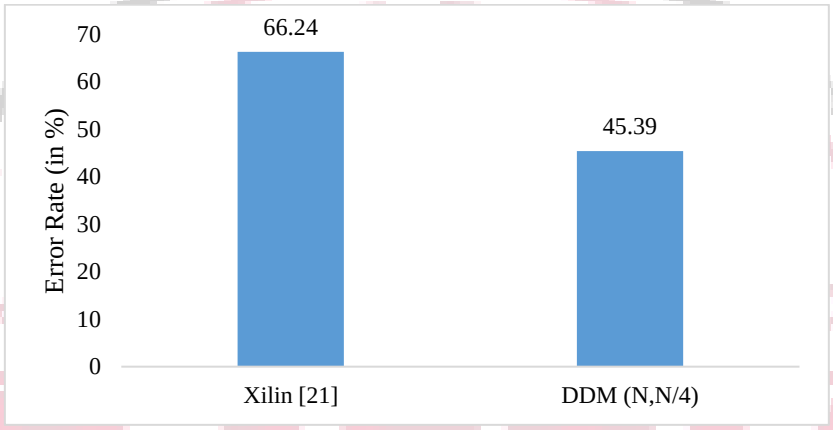


Fig. 2: Error Analysis of Approximate Multipliers

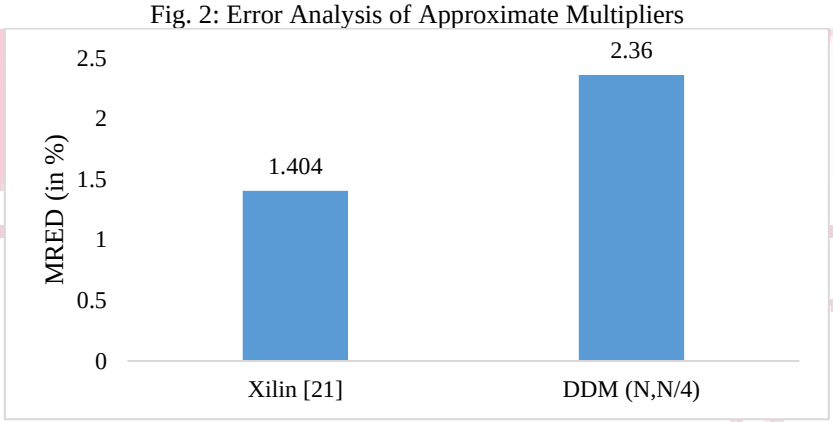


Fig. 3: MREL Analysis of Approximate Multipliers

The fig 3 shows MRED values for two multipliers: Xilin [21] has an MRED of 1.404, while DDM (N,N/4) has an MRED of 2.36. Based on these values, Xilin [21] appears to be more accurate in terms of error distance.
The fig 4 shows PSNR values for two multipliers: Xilin [21] has a PSNR of 33.51, while DDM (N,N/4) has a higher PSNR of 46.46. Based on these values, DDM (N,N/4) appears to offer higher signal quality and lower noise compared to Xilin [21].

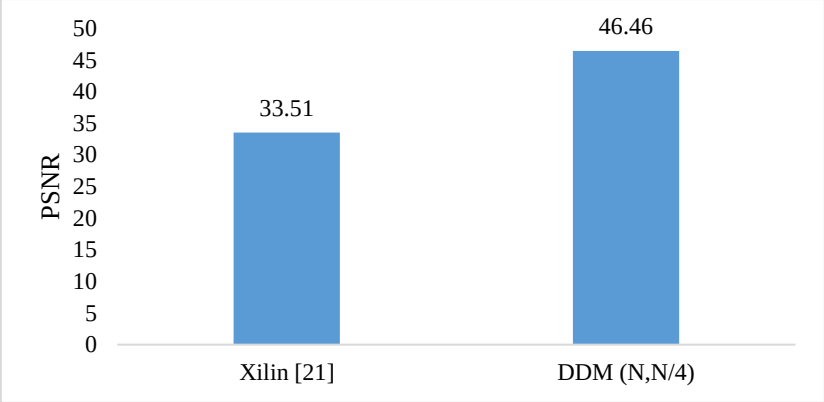


Fig. 4: PSNR of Approximate Multipliers

## 5. Conclusion

Approximate computing is a computational paradigm that presents several advantages, including reductions in area, delay, and power consumption. This particular research focuses on harnessing the inherent error-resilient nature of image processing applications. The proposed DDM (N,M) architecture is designed to decrease the number of Partial Products (PPs) generated during calculations, simplifying the subsequent PP accumulation stage. Through extensive testing of error and hardware performance, the research demonstrates that this architecture is more energy-efficient and requires less physical space compared to existing designs. Importantly, this approach achieves a superior balance between computational quality and resource usage, as evidenced by its successful application in image sharpening and compression algorithms. The future scope of this research lies in further optimizing and adapting architecture for a broader range of computational tasks beyond image processing, exploring its potential applications in fields like machine learning, signal processing, and scientific simulations. Additionally, the development of automated techniques for determining when and where to employ approximate computing in different contexts can be an avenue for future investigation, enhancing its practicality and impact in resource-constrained computing environments.

## References

[1] S. Nambi, U. A. Kumar, K. Radhakrishnan, M. Venkatesan, and S. E. Ahmed, "DDM: Decoder-Based Approximate Multiplier for Low Power Applications," IEEE Embed. Syst. Lett., vol. 13, no. 4, pp. 174–177, Dec. 2021, doi: 10.1109/LES.2020.3045165.

[2] V. A and D. S. K. G. A, "Approximate Multiplier for Low Power Applications," Int. J. Eng. Res. Technol., vol. 4, no. 14, Jul. 2018, doi: 10.17577/IJERTCONV4IS14039.

[3] H. Liu, M. Wang, L. Yao, and M. Liu, "A Piecewise Linear Mitchell Algorithm-Based Approximate Multiplier," Electron. 2022, Vol. 11, Page 1913, vol. 11, no. 12, p. 1913, Jun. 2022, doi: 10.3390/ELECTRONICS11121913.

[4] N. Arya, T. Soni, M. Pattanaik, and G. K. Sharma, "READ: A fixed restoring array based accuracy-configurable approximate divider for energy efficiency," Integration, vol. 76, pp. 1–12, Jan. 2021, doi: 10.1016/J.VLSI.2020.08.002.

[5] K. K. Senthilkumar et al., "Approximate Multiplier based on Low power and reduced latency with Modified LSB design", doi: 10.1051/e3sconf/202339901009.

[6] Y. Zhu, W. Liu, P. Yin, T. Cao, J. Han, and F. Lombardi, "Design, evaluation and application of approximate-truncated Booth multipliers," IET Circuits, Devices Syst., vol. 14, no. 8, pp. 1305–1317, Nov. 2020, doi: 10.1049/IET-CDS.2019.0398.

[7] L. Qiu, "Designing Approximate Computing Circuits with Scalable and Systematic Data-Driven Techniques Recommended Citation", Accessed: Sep. 19, 2023. [Online]. Available: https://tigerprints.clemson.edu/all_theses

[8] Da. Devi, P. Sai Kumar, R. Swapna, and A. Siddhartha, "International Journal of Computer Science and Mobile Computing Design Analysis of Approximate Redundant Binary Multipliers," Int. J. Comput. Sci. Mob. Comput., vol. 11, no. 1, pp. 74–94, 2022, doi: 10.47760/ijcsmc.2022.v11i01.010.

[9] S. Mittal, "A Survey of Techniques for Approximate Computing," ACM Comput. Surv., vol. 48, no. 4, Mar. 2016, doi: 10.1145/2893356.

[10] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," IEEE Trans. Very Large Scale Integr. Syst., vol. 23, no. 6, pp. 1180–1184, Jun. 2015, doi: 10.1109/TVLSI.2014.2333366.

[11] H. Jiang, F. Lombardi, and J. Han, "60 A Review, Classification and Comparative Evaluation of Approximate Arithmetic Circuits," 2017, doi: 10.1145/3094124.

[12] S. E. C O N D E D I T I O N Behrooz Parhami and N. York Oxford, "COMPUTER ARITHMETIC Algorithms and Hardware Designs," 2010, Accessed: Sep. 19, 2023. [Online]. Available: http://www.oup.com

[13] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Trans. Comput., vol. 62, no. 9, pp. 1760–1771, 2013, doi: 10.1109/TC.2012.146.

[14] X. Yi, H. Pei, Z. Zhang, H. Zhou, and Y. He, "Design of an energy-efficient approximate compressor for error-resilient multiplications," Proc. - IEEE Int. Symp. Circuits Syst., vol. 2019-May, 2019, doi: 10.1109/ISCAS.2019.8702199.

[15] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient Approximate Wallace Tree Multiplier for error-resilient systems," Fifteenth Int. Symp. Qual. Electron. Des., pp. 263–269, 2014, doi: 10.1109/ISQED.2014.6783335.

[16] E. J. Rao, K. T. Rao, K. S. Ramya, D. Ajaykumar, and R. Trinadh, "Efficient Design of Rounding-Based Approximate Multiplier Using Modified Karatsuba Algorithm," J. Electron. Test., vol. 38, no. 5, pp. 567–574, Oct. 2022, doi: 10.1007/S10836-022-06029-4.

[17] Z. Yang, J. Han, and F. Lombardi, "Approximate compressors for error-resilient multiplier design," Proc. 2015 IEEE Int. Symp. Defect Fault Toler. VLSI Nanotechnol. Syst. DFTS 2015, pp. 183–186, Nov. 2015, doi: 10.1109/DFT.2015.7315159.

[18] HaMinho and LeeSunggu, "Multipliers With Approximate 4–2 Compressors and Error Recovery Modules," IEEE Embed. Syst. Lett., Mar. 2018, doi: 10.5555/3195730.3195769.

[19] S. Hashemi, "DRUM: A Dynamic Range Unbiased Multiplier for Approximate Applications," 2015. doi: 10.1109/ICCAD.2015.7372600.

[20] S. Venkatachalam and S.-B. Ko, "Design of Power and Area Efficient Approximate Multipliers," 2017, doi: 10.1109/TVLSI.2016.2643639.

[21] Yi, Xilin, et al. "Design of an Energy-Efficient Approximate Compressor for Error-Resilient Multiplications." 2019 IEEE ISCAS. IEEE, 2019.