Transport and Communications Science Journal

LOW-POWER CORDIC MULTIPLIER DESIGN USING APPROXIMATE ARITHMETIC FOR ENERGY-EFFICIENT COMPUTING

Thanh-Dat Nguyen[1], Van-Vu Luyen[1], Quang-Thai Pham[1], Duy-Anh Nguyen[1*], Khanh N. Dang[2], Nguyen Khanh Linh[3,4], Dao Thanh Toan[1]

[1]Department of Electronic Engineering, University of Transport and Communications, No.3 Cau Giay Street, Hanoi, Vietnam
[2]School of Computer Science and Engineering, University of Aizu, Japan
[3]Chu Van An High School, No 10, Thuy Khue Street, Hanoi, Vietnam
[4]High School Collaboration Program, Department of Electronic Engineering, University of Transport and Communications, No.3 Cau Giay Street, Hanoi, Vietnam

ARTICLE INFO

TYPE: Research Article
Received: 08/09/2025
Revised: 02/10/2025
Accepted: 07/10/2025
Published online: 15/01/2025
https://doi.org/10.47869/tcsj.77.1.3
* *Corresponding author:*
Email: anhnd@utc.edu.vn

Abstract. In digital IC design, low-power CORDIC-based multipliers have attracted significant attention due to their potential to integrate approximate adders for reducing energy and area costs. While CORDIC is hardware-efficient, its precise design still has room for improvement, particularly in terms of power consumption and area overhead. To address this, we present an approach to enhance the CORDIC multiplier using Approx. Adders from the EvoApproxLib library. The proposed design offers multiple variants with different optimization targets: up to 19.3% power reduction in CORDIC + Approx. Adder, 11.5% area savings in CORDIC + Approx. Adder, and 14.7% frequency improvement in CORDIC + Approx. Adder compared to the conventional exact CORDIC multiplier. When applied to Gaussian filtering and Sobel edge detection, optimal variants such as CORDIC + Approx. Adder and CORDIC + Approx. Adder yield PSNR values of 60 and 48 dB respectively, with SSIM values exceeding 0.990, indicating minimal quality loss. The evaluation shows that 8–10 iterations provide the best efficiency-accuracy trade-off, enabling designers to select appropriate variants based on specific application requirements. These results demonstrate the effectiveness of the proposed method for energy-constrained, error-tolerant systems in IoT devices, edge computing, and image processing applications.

Keywords: Approximate computing, CORDIC, low-power, multiplier, Gaussian filter, edge detection.

1. INTRODUCTION

Low-power design is essential in today's electronics landscape, particularly for portable gadgets like smartphones, wearables, IoT sensors, and medical implants, where constantly recharging or swapping out batteries just is not feasible [1]. It helps cut down on energy costs for both personal devices and larger systems, like data centers, while also reducing the need for complicated cooling systems. By consuming less power, we not only generate less heat but also boost system reliability and prolong the lifespan of the hardware. Moreover, it plays a vital role in supporting environmental sustainability by lowering energy consumption and carbon emissions. In the fiercely competitive market we have today, being energy-efficient is a significant edge, making low-power design crucial for both technical performance and commercial success [2].

The CORDIC (COordinate Rotation DIgital Computer) algorithm is an efficient method for implementing multiplicative functions in digital hardware [3]. By replacing multipliers with shift-add operations, it significantly reduces both power and area costs, up to 76.69% in power and 63.64% in area in various FPGAs [4]. These gains come from lower switching activity and simpler arithmetic units. CORDIC supports linear, hyperbolic, and rotational modes, enabling it to perform multiplication, division, square roots, and complex functions without multipliers. Its efficiency and flexibility make CORDIC well suited for FFTs, DSPs, graphics, and communication systems, especially in power-constrained environments [3].

Although CORDIC is hardware-efficient, its iterative nature leads to high latency and energy consumption, and the need for more iterations and complex control for high precision and limited convergence range makes it less suitable for modern, low-power, and precision-critical applications like neural networks and edge devices [5]. Conventional CORDIC is limited in neural network applications due to its fixed-iteration and sequential design. It applies uniform high-precision computation to all activation functions, even when lower precision is sufficient. This leads to unnecessary energy consumption and hardware overhead [6]. Its sequential nature also conflicts with the parallel structure of modern neural accelerators, especially in MAC units. As a result, additional synchronization is required, increasing power and area costs [7]. CORDIC also lacks scalability and flexibility for large models like transformers, which demand dynamic precision and adaptive computation. These limitations reduce its suitability for modern AI systems [3].

Approximate computing reduces energy use and improves performance by allowing small errors in computation [8]. Many AI and signal processing tasks, like image recognition and neural networks, tolerate such errors due to their statistical or perceptual nature [9]. This enables simpler hardware and lower power consumption. Approx. Adders and multipliers have shown good energy savings with minimal loss in accuracy, especially in MAC operations [10]. Recent work also links approximate computing to better noise resilience. These benefits make it well-suited for low-power, error-tolerant applications [11], [12].

There is a clear gap in research on applying Approx. Adders within CORDIC for multiplication. While CORDIC and approximate arithmetic have been studied independently, their integration remains largely unexplored. This represents a missed opportunity to enhance energy efficiency and noise tolerance in low-power systems such as IoT and edge devices [13]. CORDIC's shift-add structure and inherent error resilience make it a good candidate for

approximate computing. Incorporating Approx. Adders could enable more efficient and power-aware designs for AI and signal processing applications [14].

The main contributions of this paper are as follows:

- A novel CORDIC-based multiplier architecture that integrates Approx. Adders to improve power and area efficiency.
- A comprehensive evaluation of the trade-offs between computational accuracy and hardware efficiency (power, area, and delay).

The organization of the paper is as follows: Section 2 shows our proposed design. Then, section 3 presents the evaluation results and section 4 concludes the paper.

2. PROPOSED FRAMEWORK

2.1. CORDIC Algorithm Overview

The COordinate Rotation DIgital Computer (CORDIC) algorithm is a shift-add method that enables efficient hardware implementation of mathematical functions without multipliers [3]. CORDIC computes vector rotations through iterative micro-rotations using only addition, subtraction, bit-shifting, and table lookups, significantly reducing hardware complexity and power consumption [15].

CORDIC operates in rotation mode (computing sine, cosine, complex multiplication) and vectoring mode (computing magnitude and phase). It supports three coordinate systems: circular (trigonometric functions), linear (multiplication/division), and hyperbolic (exponentials/logarithms), each sharing a unified iterative structure [16].

2.2. Approximate Computing Fundamentals

Approximate computing is a design paradigm that intentionally introduces controlled inaccuracies into computations to reduce energy consumption, delay, and hardware complexity. This approach is particularly effective in applications that exhibit inherent error resilience, such as image processing and neural networks, where perfect accuracy is not always required.

Approx. Adders are key components in approximate arithmetic units. They are broadly categorized as follows [17]:

1) *Speculative adders:* These designs predict carry signals over a truncated bit-slice to reduce carry propagation delay. Examples include the Almost Correct Adder (ACA) and Error-Tolerant Adder (ETA) families [18].

2) *Segmented adders:* The adder is divided into accurate and approximate segments, with the carry chain truncated between them. This segmentation shortens the critical path while maintaining partial accuracy [19].

3) *Carry-select Approx. Adders:* These compute partial sums under both possible carry-in conditions and select the result using a predicted carry signal. The Speculative Carry-Select Adder (SCSA) is a well-known example [20].

4) *Approximate full-adder based designs:* These modify sum or carry logic in the least significant bits. Examples include the Lower-part OR Adder [19] and Heterogeneous Block-

based Approx. Adders (HBAA) [21], which use varied sub-adder configurations to achieve Pareto-optimal trade-offs between accuracy and efficiency.

2.3. CORDIC Multiplier Architecture

Our CORDIC multiplier architecture is shown in Fig. 1 and Algorithm 1. We use a basic sequential architecture to update y and $z\_\{scaled\}$ through multiple iterations, where the updated values are determined by the current iteration number and the sign of $z\_\{scaled\}$.

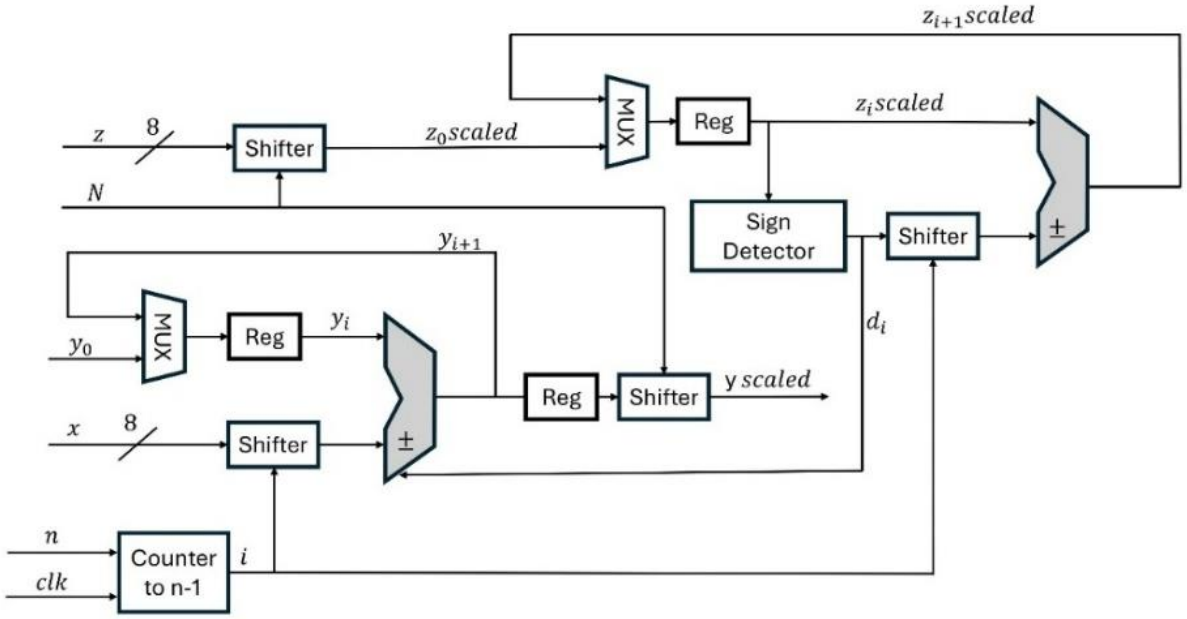

Figure 1. CORDIC Multiplier Architecture.

Algorithm 1: CORDIC-based Multiplication of $x.z$

INPUT: values x, z; number of iterations n; number of bits for representation N.

OUTPUT: approximate product $y_{scaled} \approx x.z$.

1 CALCULATE NORMALIZATION FACTOR: $k = 2^{N-1}$

2 INITIALIZE: $z_{scaled} = \frac{z}{2^k}$

3 INITIALIZE: $y = 0$

4 FOR $i = 0$ TO $n - 1$ DO:

$d = sign(z_{scaled})$

$y = y + d.(x.2^{-1})$

$z_{scaled} = z_{scaled} - d.2^{-i}$

5 RESCALE: $y_{scaled} = y.2^k$

6 RETURN: y_{scaled}

To help illustrate the algorithm, consider a simple example where $x = 5, z = 3$, and the number of iterations is $n = 3$.

Initialization:

- $k = 4$ (normalization factor)
- $z_{scaled} = \frac{3}{4} = 0.75$
- $y = 0$ (accumulator)

Iteration 1 (i = 0):

- $d = sign(0.75) = +1$
- $y = 0 + 1 \cdot (5 \cdot 20) = 0 + 5 = 5$
- $z_{scaled} = 0.75 - 1 \cdot 20 = 0.75 - 1 = -0.25$

Iteration 2 (i = 1):

- $d = sign(-0.25) = -1$
- $y = 5 + (-1) \cdot (5 \cdot 2 - 1) = 5 - 2.5 = 2.5$
- $z_{scaled} = -0.25 - (-1) \cdot 2 - 1 = -0.25 + 0.5 = 0.25$

Iteration 3 (i = 2):

- $d = sign(0.25) = +1$
- $y = 2.5 + 1 \cdot (5 \cdot 2 - 2) = 2.5 + 1.25 = 3.75$
- $z_{scaled} = 0.25 - 1 \cdot 2 - 2 = 0.25 - 0.25 = 0$

Final Result:

- $Rescale: y_{scaled} = 3.75 \times 4 = 15$
- $Expected\ result: 5 \times 3 = 15$

This example demonstrates how the algorithm approximates multiplication using only shift and add operations.

2.4. Approx. Adder Integration

1) Cordic Multiplier with Approx. Adder: In this paper, we proposed using an Approx. Adder to calculate the update of y in each iteration. Algorithm 2 shows the proposed multiplier.

Since the signed adders for the updates of y are restricted to 16-bit adders only, we restricted the multiplier to use an 8-bit input. In this study, we used a set of 16-bit signed Approx. Adders from EvoApproxLibLITE, focusing on key metrics such as mean absolute error (MAE), error probability (EP), and hardware cost. We evaluated all the 16-bit signed adders from the library.

Algorithm 2: CORDIC-based Multiplication of $x.z$ with Approx. Adder

INPUT: values x, z; number of iterations n; number of bits for representation N.

OUTPUT: approximate product $y_{scaled} \approx x.z$.

1 CALCULATE NORMALIZATION FACTOR: $k = 2^{N-1}$

2 INITIALIZE: $z_{scaled} = \frac{z}{2^k}$

3 INITIALIZE: $y = 0$

4 FOR $i = 0$ TO $n - 1$ DO:

$d = sign(z_{scaled})$;

$y = ApproximateAdd(y, d \,.\, (x \,.\, 2^{-1}))$;

$z_{scaled} = z_{scaled} - d \,.\, 2^{-i}$; //with exact adder

5 RESCALE: $y_{scaled} = y \,.\, 2^{k}$

6 RETURN: y_{scaled}

2) The effect of the number of iterations to the accuracy: In regular CORDIC algorithms, more iterations usually mean better accuracy. But when we use Approx. Adders, this relationship becomes unclear. Each iteration should reduce the error, but approximate arithmetic adds its own errors that might change how this works.

To better understand this, we look at how changing the number of CORDIC iterations affects accuracy when using different Approx. Adders. This helps us answer two questions: do traditional CORDIC accuracy improvements still work with approximate arithmetic, and is there a point where more iterations don't help or even make things worse.

These results will help designers choose the right number of iterations for different Approx. Adders and applications.

3. EVALUATION

3.1. Evaluation Methodology

1) Hardware Evaluation Methodology: The 8-bit CORDIC multiplier with Approx. Adder design was exhaustively tested using all possible input combinations (256 × 256 test cases). The 16-bit signed adders of EvoApproxLib [22] were evaluated and compared with the baseline CORDIC multiplier using the exact adder design. The average error rate was calculated for each design.

The hardware modules were implemented in Verilog and synthesized using Design Compiler in a 45 nm process technology. Functional and netlist simulations were performed using Questasim. The estimated maximum operating frequency and power consumption were extracted using PrimeTime.

2) Software Evaluation Methodology: Based on the hardware results, several designs with optimal area cost and power consumption were selected to evaluate their impact on basic software applications, specifically the Sobel filter and Gaussian filter.

To investigate the effects of increasing iteration count on result accuracy, the CORDIC multiplier with Approx. Adder was tested on squared functions with varying numbers of iterations. For this experiment, a Python script was used to simulate the values of x^2 using the CORDIC multiplier with different adders, and the results were compared with exact values.

Testing was conducted for 256 test cases (for 8-bit CORDIC multiplier design), ranging from 5 to 16 iterations. The average errors for all test cases were recorded.

To assess the impact of using the CORDIC multiplier with Approx. Adder, experiments were conducted in Python on Gaussian filter and Sobel edge detection. Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) were used as key evaluation metrics.

To enhance numerical precision in these image processing applications, fixed-point representations were adopted and tailored to each computational component. Pixel values were represented using the Q8.0 format, while the Gaussian kernel employed the Q0.8 format to accommodate its fractional characteristics. The Sobel kernel utilized the Q2.6 format, which provided an optimal balance between dynamic range and precision for effectively capturing spatial gradients. This configuration ensured adequate accuracy in intermediate computations, particularly when combined with approximate arithmetic units.

3.2. Hardware Evaluation Results

Table 1 shows the comparative results from each variant of the design.

From Table 1, significant improvements in hardware area cost, power consumption, and maximum operating frequency were observed when Approx. Adders were applied in the CORDIC multiplier design. Compared to the baseline design with exact adders, area savings of up to 11.5% were achieved (for the 32T variant), and power consumption was reduced by up to 19.3% (for the 2YM variant). Several variants demonstrated improved maximum operating frequencies, with the 2YM variant achieving the highest frequency of 562MHz (14.7% improvement over the exact design) and the 2UB variant reaching 535MHz (9.2% improvement).

Table 1. Comparison of Multiplier CORDIC Multiplier Approx (CMA) with ITERATIONS = 16, f = 100MHz.

Design	Area(μm^2)	Power(μW)	Avg error(%)	F_{max}(MHz)
Exact	617	114	0.24%	490
2TN	615	112	0.39%	490
2U6	593	108	0.44%	476
2UB	555	99	1.88%	535
2UY	606	111	0.09%	474
2YM	562	92	3.76%	562
32T	546	96	4.48%	513
36D	556	98	1.62%	508

Regarding accuracy trade-offs, the average error rate was slightly increased in the worst case 32T variant, up to 4.48%, though some variants demonstrated excellent accuracy preservation (e.g., 2UY variant with only 0.09% error rate). The 2TN and 2U6 variants provide a good balance between hardware efficiency and accuracy, with error rates below 0.5% while achieving notable power and area reductions.

In conclusion, the experimental results demonstrated that incorporating Approx. Adders into the CORDIC multiplier design yielded favorable gains in area cost, power consumption,

and maximum operating frequency. The 2YM variant offers the best overall hardware efficiency, while the 2UY variant provides superior accuracy with moderate efficiency gains. Based on specific application requirements, suitable variants can be selected to optimize the design for either maximum performance or highest accuracy.

Table 2 summarizes the comparison between the proposed CMA and previously reported CORDIC-based designs. As observed, the CMA achieves up to 39× smaller area (615 vs. 8578 µm2 in Lyu et al. [4]) and 47× lower power (112 vs. 5240 µW), while also delivering explicit error control with 0.39% MRE compared to 6.03% MRE in Khurshid et al. [23]. In contrast, Lyu et al. [4] emphasize high operating frequency (6.25 GHz) at the cost of larger area and power, while Nair et al. [24] focus on FPGA implementations with significant resource usage (16,579 LUTs). Unlike these broader architectures, the proposed CMA is specialized for multiplication, making it a compact and energy-efficient solution particularly suited for low power and error-tolerant domains.

Table 2. Comparison between the proposed CMA and previously proposed architectures.

Design	Area (µm²)	Power (µW)	Avg. error (%)	Fmax (MHz)
2TN (Proposed)	615	112	0.39 (MRE)	490
Khurshid et al. [23].	420 LUTs	3470	6.032 (MRE)	667.55
Lyu et al. [4].	8578	5240	–	6250
Nair et al. [24].	16,579 LUTs	82,000	–	8.26

3.3. Software Evaluation Results

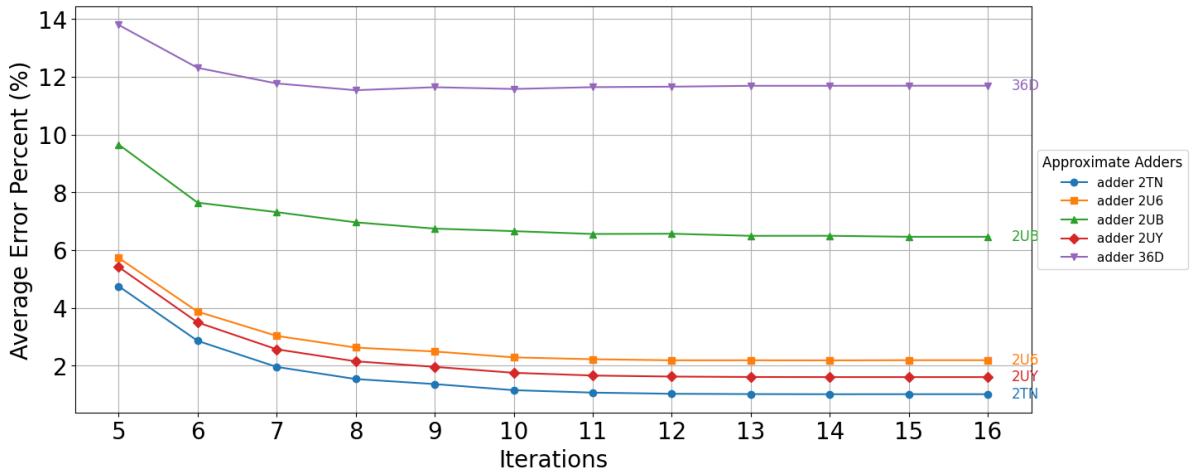

Figure 2. CORDIC Squaring Average Error vs Iterations for Different Approx. Adders.

This section presents the evaluation results from both squared functions and applications using Gaussian Filter and Sobel Edge Detection. Based on the hardware evaluation results, five variants of the Approx. Adder were evaluated: 2TN, 2U6, 2UB, 2UY, and 36D, each offering different trade-offs in terms of area, latency, and power consumption. One group offers better accuracy (2TN, 2U6, 2UY), one group have better area and power consumption rate but slightly worse accuracy (2UB, 36D)

1) Evaluation results for the squared functions: Fig. 2 shows the average error percentages for different variants of the CORDIC multiplier with varying numbers of iterations. The software simulation demonstrated similar error rates to the hardware results, with the higher accuracy group (2TN, 2UY, 2U6) clearly showing better performance than the other group (2UB, 36D). The iteration number exhibited a similar trend across all adders, where accuracy tended to saturate after 9-10 iterations.

2) *Evaluation results for the applications:* Table 3 shows the PSNR and SSIM results for different CORDIC-based designs using five selected 16-bit Approx. Adders across iteration counts ranging from 5 to 10. Overall, the results demonstrate a consistent trend where increasing the number of iterations improves image quality across both Gaussian and Sobel filters.

For Gaussian filtering, all designs achieve relatively high SSIM scores and PSNR values that increase sharply with more iterations. Notably, even at low iteration counts, the degradation is minimal, confirming that approximation errors introduced by both the CORDIC multiplier and the Approx. Adder exert limited impact on the output of this smoothing operation. Fig. 3 and Fig. 4 illustrate the effects of applying CORDIC Multiplier with and without Approx. Adders for Gaussian filter and Sobel Detection tasks.

In contrast, the Sobel filter exhibits greater sensitivity to approximation, especially at 5–6 iterations. PSNR values at iteration 5 are around 9–19 dB, depending on the design, but improve significantly with higher iterations, reaching 44–49 dB at iteration 10 for most configurations. SSIM follows a similar pattern, rising from 0.42 to 0.98, highlighting the importance of numerical precision in edge detection tasks.

Among the evaluated designs, the group with better accuracy (2TN, 2U6, 2UY) consistently outperforms the group optimized for area and power (2UB, 36D) in terms of image quality. However, the gap narrows as the number of iterations increases. All designs exhibit diminishing returns after iteration 9, suggesting that this is a reasonable saturation point for balancing performance and hardware complexity.

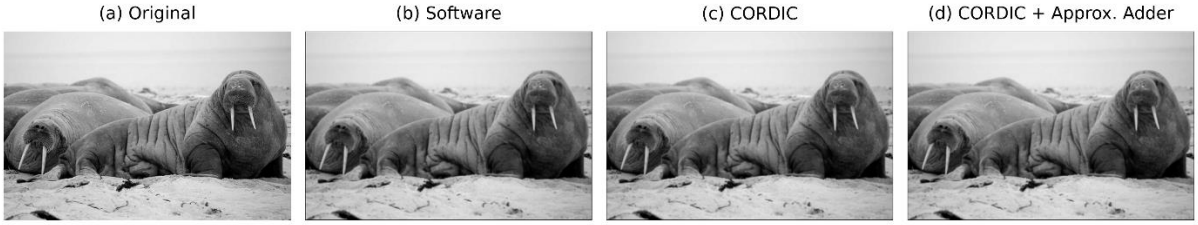

Figure 3. Comparison of processing results for Gaussian Filter: original, software, CORDIC multiplier, and approximate CORDIC multiplier.

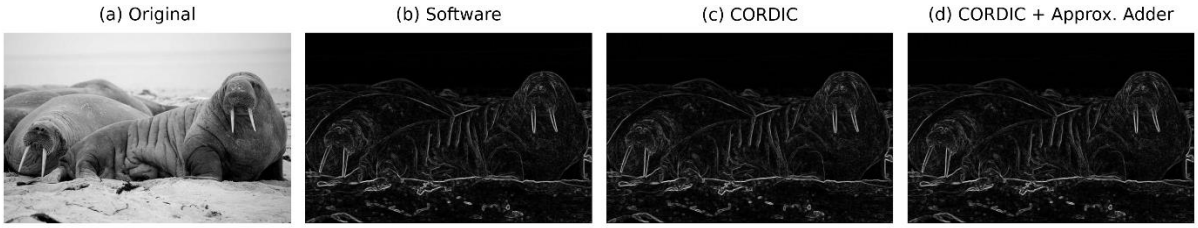

Figure 4. Comparison of processing results for Sobel Filter: original, software, CORDIC multiplier, and approximate CORDIC multiplier.

Table 3. PSNR and SSIM Comparison for Selected Approx. Adders with Varying CORDIC Iterations.

Design	#Iter.	Gaussian PSNR	Gaussian SSIM	Sobel PSNR	Sobel SSIM
CORDIC Multiplier	5	29.98	0.997	19.45	0.4216
	6	45.02	0.9987	25.45	0.5849
	7	50.87	0.9985	31.66	0.7206
	8	53.91	0.9985	37.98	0.8339
	9	51.28	0.9984	44.22	0.9316
	10	56.15	0.999	49.73	0.9815
CORDIC + Approx. Adder (2TN)	5	30.03	0.9971	9.16	0.4197
	6	45.19	0.9987	25.74	0.58
	7	50.63	0.9985	31.53	0.7258
	8	53.35	0.9984	37.69	0.8292
	9	51.77	0.9984	43.33	0.9184
	10	57.17	0.9992	48.75	0.979
CORDIC + Approx. Adder (2U6)	5	29.98	0.997	9.16	0.42
	6	45.1	0.9987	25.74	0.5802
	7	50.68	0.9985	31.54	0.7258
	8	53.29	0.9984	37.68	0.8294
	9	51.91	0.9984	43.22	0.9179
	10	60.17	0.9995	48.44	0.9782
CORDIC + Approx. Adder (2UY)	5	29.98	0.997	9.16	0.4198
	6	45.1	0.9987	25.74	0.5801
	7	50.68	0.9985	31.52	0.7255
	8	53.29	0.9984	37.55	0.827
	9	51.91	0.9985	43.15	0.9157
	10	57.97	0.9993	48.46	0.978
CORDIC + Approx. Adder (2UB)	5	30.77	0.9971	9.15	0.4126
	6	48.97	0.9984	25.57	0.5682
	7	44.96	0.9983	30.7	0.7021
	8	46.78	0.9986	35.57	0.7928
	9	51.13	0.9985	40.54	0.8896
	10	51.56	0.9984	44.29	0.9616
CORDIC + Approx. Adder (36D)	5	30.68	0.9972	9.15	0.4143
	6	49.38	0.9988	25.57	0.5703
	7	44.74	0.9982	30.75	0.7052
	8	46.06	0.9984	35.63	0.7942
	9	50.82	0.9984	39.81	0.8698
	10	47.58	0.999	42.5	0.9169

4. CONCLUSION

This paper proposes a low-power CORDIC multiplier using Approx. Adders to achieve energy-efficient computing. The design combines CORDIC iterations with approximate arithmetic, offering favorable trade-offs between accuracy and hardware efficiency.

Hardware evaluation demonstrates different optimization targets with variants achieving up to 19.3% power reduction, 11.5% area savings, and 14.7% frequency improvement compared to conventional exact CORDIC multipliers.

Software evaluation on Gaussian filtering and Sobel edge detection shows that the best-performing configurations—specifically those using adders such as 2U6, 2UY, and 2TN with 9–10 CORDIC iterations—achieve PSNR values exceeding 50 dB for Gaussian and 44 dB for Sobel, with SSIM values consistently above 0.998 and 0.97, respectively.

The results demonstrate effectiveness for energy-constrained, error-tolerant applications, enabling designers to select appropriate configurations based on specific requirements. Future work will focus on adaptive approximation strategies and extending this approach to other CORDIC-based functions.

ACKNOWLEDGMENT

This research was supported by the University of Aizu Competitive Research Funding P-21. This work was supported by the Minister of Education and Training (MOET), grant no. B2025-GHA-02.

REFERENCES

[1]. S. Frey, M. Guermandi, S. Benatti, V. Kartsch, A. Cossettini, L. Benini, BioGAP: A 10-core FP-capable ultra-low power IoT processor, with medical-grade AFE and BLE connectivity for wearable biosignal processing, 2023 IEEE International Conference on Omni-layer Intelligent Systems, (2023) 1-7. https://doi.org/10.1109/COINS57856.2023.10189286

[2]. C. Guo, F. Luo, Z. Cai, Z.Y. Dong, Integrated energy systems of data centers and smart grids: State-of-the-art and future opportunities, Applied Energy, 301 (2021) 117474. https://doi.org/10.1016/j.apenergy.2021.117474

[3]. A. Changela, M. Zaveri, D. Verma, A comparative study on CORDIC algorithms and applications, Journal of Circuits, Systems and Computers, 32 (2023) 2330002. https://doi.org/10.1142/S0218126623300027

[4]. F. Lyu, C. Wu, Y. Wang, H. Pan, Y. Wang, Y. Luo, An optimized hardware implementation of the CORDIC algorithm, IEICE Electronics Express, 19 (2022) 20220362. https://doi.org/10.1587/elex.19.20220362

[5]. N. Bai, R. Qu, Y. Xu, Y. Wang, X. Chen, L. Li, Low-iteration hybrid computing CORDIC architecture, Microelectronics Journal, 156 (2025) 106481. https://doi.org/10.1016/j.mejo.2024.106481

[6]. J. Deng, J. Yang, X. Wang, An Efficient FP16-Based Hardware Architecture with Enhanced CORDIC to Implement Sigmoid and Tanh Functions, 2024 IEEE 7th Information Technology, Networking, Electronic and Automation Control Conference, 7 (2024). https://doi.org/10.1109/ITNEC60942.2024.10733095

[7]. K. Li, H. Fang, Z. Ma, F. Yu, B. Zhang, Q. Xing, A Low-latency CORDIC algorithm based on pre-rotation and its application on computation of arctangent function, Electronics, 13 (2024) 2338. https://doi.org/10.3390/electronics13122338

[8]. Y. Wu, C. Chen, W. Xiao, X. Wang, C. Wen, J. Han, X. Yin, W. Qian, C. Zhuo, A survey on approximate multiplier designs for energy efficiency: From algorithms to circuits, ACM Transactions on Design Automation of Electronic Systems, 29 (2024) 1-37. https://doi.org/10.1145/3610291
[9]. R. Kobayashi, I. Okubo, K.N. Dang, ApproxiMorph: energy-efficient neuromorphic system with layer-wise approximation of spiking neural networks and 3D-stacked SRAM, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, (2025). https://doi.org/10.1109/TCAD.2025.3597251
[10]. V. Mishra, S. Mittal, S. Singh, D. Pandey, R. Singhal, Mega-mac: a merged accumulation based approximate mac unit for error resilient applications, Proceedings of the Great Lakes Symposium on VLSI, (2022) 1-4. https://doi.org/10.1145/3526241.3530384
[11]. A.M. Dalloo, D.A. Abdulhussein, M.M. Sabry, A.J. Humaidi, A.R.J. Almusawi, R. Woods, N. TaheriNejad, Approximate computing: Concepts, architectures, challenges, applications, and future directions, IEEE Access, (2024). https://doi.org/10.1109/ACCESS.2024.3467375
[12]. R. Kobayashi, K.N. Dang, An efficient hardware implementation of spiking neural network using approximate Izhikevich neuron, 9th International Conference on Integrated Circuits, Design, and Verification, (2024). https://doi.org/10.1109/ICDV61346.2024.10616602
[13]. G. Narmadha, S. Kanniyappan, V. Dhilip Kumar, P. Ramu, A low power and high speed Approx. Adder for image processing applications, Journal of Engineering Research, 10 (2022) 150-160. https://doi.org/10.36909/jer.10037
[14]. B. Khurshid, High-performance CORDIC-based approximate MAC architectures for FPGA platforms, Integration, 101 (2025) 102338. https://doi.org/10.1016/j.vlsi.2024.102338
[15]. Chung R.L., Hsueh Y., Chen S.L., Abu P.A.R., Efficient and accurate cordic pipelined architecture chip design based on binomial approximation for biped robot, Electronics, 11(11) (2022) 1701. https://doi.org/10.3390/electronics11111701
[16]. H. Chen, K. Cheng, Z. Lu, Y. Fu, L. Li, Low-complexity high-precision method and architecture for computing the logarithm of complex numbers, IEEE Transactions on Circuits and Systems I: Regular Papers, 68 (2021) 3293-3304. https://doi.org/10.1109/TCSI.2021.3081517
[17]. H.H. Que, H.Y. Lin, X. Yin, C. Zhuo, J. Han, A Survey of approximate computing: from arithmetic units design to high-level applications, Journal of Computer Science and Technology, 38 (2023) 251-272. https://doi.org/10.1007/s11390-023-2537-y
[18]. M.A. Hanif, R. Hafiz, M. Shafique, DAEM: A data-and application-aware error analysis methodology for Approx. Adders, Information, 14 (2023) 570. https://doi.org/10.3390/info14100570
[19]. P. Balasubramanian, R. Nayar, D.L. Maskell, Gate-level static Approx. Adders: a comparative analysis, Electronics, 10 (2021) 2917. https://doi.org/10.3390/electronics10232917
[20]. V. Lakshmi, V. Pudi, J. Reuben, In-memory implementation of an Approx. Adder with reduced latency and error, IEEE Transactions on Circuits and Systems I: Regular Papers, (2024). https://doi.org/10.1109/TCSI.2024.3511955
[21]. E. Farahmand, M.E. Salehi, S. Mohammadi, A. Yazdanbakhsh, Design and analysis of high performance heterogeneous block-based Approx. Adders, ACM Transactions on Embedded Computing Systems, 22 (2023) 1-32. https://doi.org/10.1145/3625686
[22]. M.Češka, J. Matyáš, V. Mrazek, L. Sekanina, Z.Vasicek, T.Vojnar, Designing approximate arithmetic circuits with combined error constraints, 25th Euromicro Conference

on Digital System Design, (2022). https://doi.org/10.1109/DSD57027.2022.00110
[23]. B. Khurshid, J.J. Khan, An efficient fixed-point multiplier based on CORDIC algorithm, Journal of Circuits, Systems and Computers, 30 (2021) 2150080. https://doi.org/10.1142/S0218126621500808
[24]. H. Nair, A. Chalil, FPGA implementation of area and speed efficient CORDIC algorithm, 6th International Conference on Computing Methodologies and Communication, (2022). https://doi.org/10.1109/ICCMC53470.2022.9753730