Semnan University

Journal of Modeling & Simulation in Electrical & Electronics Engineering (MSEEE)

Journal homepage: https://mseee.semnan.ac.ir/journal/about /

ISSN: 2821-0786

Approximate Computing on FPGAs for Edge Computing Applications: A Case Study of Adaptive Filters

Reza Omidi [1•]

***Abstract*-- The exponential growth of data and the paradigm shift towards edge computing have necessitated innovative approaches to energy-efficient computation, especially for resource-constrained IoT devices. Approximate computing, a paradigm that exploits the inherent tolerance of many applications to imprecision, has been extensively explored in the context of ASICs to reduce power consumption and area overhead. However, its potential in FPGA-based devices, which offer flexibility and rapid prototyping capabilities, remains largely untapped. This paper investigates the feasibility and performance implications of approximate computing techniques for FPGAs, with a focus on the implementation of FIR and adaptive filters as illustrative case studies. Specifically, we propose novel approximate multipliers based on the Reverse Carry Propagation (RCP) adders, which are evaluated through their integration into adaptive and finite impulse response filters. Simulation results demonstrate significant improvements in operating frequency, as well as substantial reductions in hardware area for FIR filters. While the area reduction for adaptive filters is less pronounced, the proposed multipliers still exhibit acceptable performance. Our findings highlight the potential of approximate multipliers for low-power computing systems, particularly in edge computing applications.**

***Index Terms*-- Approximate computing; Edge computing; FPGA; adaptive filter; multiplier; adder; inexact arithmetic.**

I. Introduction

With the proliferation of the Internet of Things (IoT) and the surge in big data, coupled with the limitations of cloud computing, the technological paradigm has shifted towards edge computing. The goal of edge computing is to perform data acquisition, processing, and generation at the source, thereby minimizing the need to transmit data to the cloud. A simple example would be implementing artificial intelligence directly on the source chip rather than relying on cloud-based AI[1]. However, the edge computing approach faces significant challenges due to limited resources, such as processing power and energy consumption, at the edge. Consequently, offloading computational tasks from the edge processor can be highly beneficial. One approach to mitigate computational overhead and consequently reduce power consumption is approximate computing. In this paradigm, approximate results are produced rather than generating exact outputs. While this inherent imprecision may degrade the quality of service, it is often tolerable in applications such as multimedia processing. Leveraging this concept, a plethora of approximate circuits have been proposed and extensively studied in the literature. Another critical consideration is the implementation platform for approximate circuits. Most existing research has focused on ASIC implementations. However, FPGAs are more commonly used in various applications. There is a relative dearth of studies on implementing approximate circuits on FPGAs [2, 3].

In this paper, we aim to evaluate and analyze the impact of employing approximate circuits in FPGA-based implementations. To this end, an adaptive filter based on the least mean squares algorithm is selected as a suitable case study. This filter comprises numerous computational units—multipliers and adders—making it an ideal candidate to demonstrate the effects of approximation. Moreover, due to its feedback mechanism, the filter can mitigate the errors introduced by approximation, thus preserving the required quality of service. Adaptive filters based on the least mean squares (LMS) algorithm are widely used in digital signal processing[4, 5]. As an approximation of the Wiener filter, the LMS algorithm is inherently inexact and serves as a natural candidate for exploring approximate hardware. However, this application is always accompanied by the challenge of a feedback path for updating filter coefficients. In this paper, the LMS adaptive filter is investigated by replacing exact multipliers with approximate multipliers. The proposed approach employs a carry-save adder to implement the

[1] Department of Electrical Engineering, University of Zanjan, Iran.
• Corresponding author Email: rezaomidi@znu.ac.ir

Cite this article as:
Omidi R,2025, Approximate Computing on FPGAs for Edge Computing Applications: A Case Study of Adaptive Filters. Journal of Modeling & Simulation in Electrical & Electronics Engineering (MSEEE), 5(1), pp. 1-9.

https://doi.org/10.22075/MSEEE.2025.37914.1215

approximate multiplier, which is then used in the LMS algorithm implementation. The final implementation results on an FPGA show up to a 29% reduction in the number of LUTs used for the adaptive filter with the proposed approximate multiplier. On the other hand, the results show no improvement in the operating frequency of this type of approximate filter. The novel contributions of this paper are as follows:

- Introduction of Three Novel Approximate Multiplier Architectures: Three new high-performance approximate multipliers have been proposed, leveraging the concept of the modified Booth algorithm. These novel architectures offer significant performance improvements in specific applications while maintaining acceptable accuracy.
- Comprehensive Performance Evaluation in Various Filters: The proposed multipliers have been rigorously evaluated in both adaptive and finite impulse response filters. Results demonstrate the significant superiority of these multipliers in improving operating frequency and reducing area occupancy in finite impulse response filters.
- FPGA Implementation: All proposed circuits have been implemented and evaluated on FPGA devices. This choice is motivated by the flexibility and widespread use of FPGAs in edge computing.

In the subsequent sections of this paper, we will first delve into the structure of adaptive filters and review prior research in this area. Next, the proposed approximate circuits and architectures, with a focus on adaptive filters as a case study, will be introduced. In the following section, simulation results and a comprehensive analysis will be presented. Finally, the paper will conclude by summarizing the contributions and providing suggestions for future research.

II. ADAPTIVE FILTER STRUCTURE AND LITERATURE REVIEW

Adaptive filters are a class of digital filters that can self-adjust their coefficients to optimize their performance according to a specific criterion. In the context of a Finite Impulse Response (FIR) filter with a fixed number of taps, the adaptation process involves updating these coefficients to minimize the difference between the filter's output and a desired signal. This difference is commonly referred to as the error signal. The adaptation algorithm, often based on techniques like the Least Mean Squares (LMS) or Recursive Least Squares (RLS), uses the error signal to iteratively adjust the filter coefficients. This process allows the adaptive FIR filter to track changes in the input signal or the desired response, making it suitable for applications such as noise cancellation, system identification, and channel equalization. The output of an FIR filter is given by:

$$y(n) = \sum_{\{k=0\}}^{\{M-1\}} w_k(n)x(n-k)$$

where, y(n) is the output signal at time n, x(n) is the input signal at time n, $w_k(n)$ are the filter coefficients, M is the number of taps (filter order). The error signal is defined as e(n) = d(n) - y(n); where, d(n) is the desired signal at time n, the LMS algorithm updates the filter coefficients as follows:

$$w_k(n+1) = w_k(n) + \mu e(n)x(n-k)$$

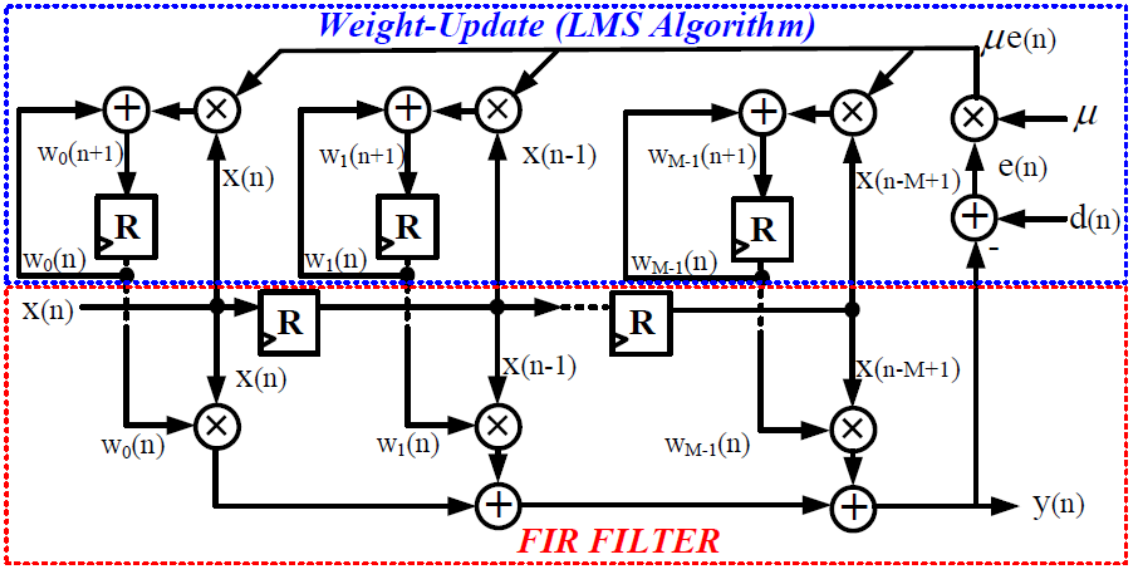

Fig. 1: A block diagram illustrating the structure of an adaptive filter [6].

where, $w_k(n+1)$ is the updated filter coefficient at time n+1, μ is the step size (adaptation rate). These equations illustrate the fundamental mathematical relationships governing the operation of an adaptive FIR filter. By iteratively adjusting the filter coefficients based on the error signal and the input signal, the filter adapts to changing conditions and optimizes its performance. The general structure of an adaptive filter is illustrated in Figure 1. The input signal is typically a sampled version of a continuous-time signal. The key characteristic of an adaptive filter lies in its ability to adjust its coefficients over time to minimize a predefined error criterion, such as the mean squared error (MSE). Various adaptive algorithms have been developed to update the filter

coefficients, each tailored to specific applications and performance requirements. According to Wiener-Hopf theory, the optimal filter coefficients are obtained when the MSE is minimized. The update process, represented by $w_k(n)$, continually adjusts the filter coefficients, driving the filter output towards the desired signal. Numerous practical applications require generating an output signal that mimics a desired reference signal in response to an input signal. Applications of adaptive filters can be broadly categorized into system identification, inverse modeling, linear prediction, and interference cancellation.

To implement this structure approximately, we must first examine its internal components. Assuming the main filter is an FIR filter, it consists of a series of multipliers and accumulators (MACs) as shown in Figure 1. The coefficient update structure, as depicted in Figure 1, also comprises a series of multipliers and accumulators. Literature review indicates that inexact multipliers and adders have been employed for the inexact implementation of this structure. This research focuses on the direct form FIR filter due to its faster convergence and reduced memory requirements. While numerous studies have investigated reducing the computational complexity of the Least Mean Squares (LMS) algorithm or other adaptive filtering algorithms, to the best of our knowledge, no research except [6] has explored hardware complexity reduction through approximate computing. The Least Mean Squares (LMS) algorithm is an iterative algorithm designed to minimize the mean squared error between the desired output of a finite impulse response (FIR) filter and its actual output. This minimization requires the calculation of the gradient of the mean squared error, which is approximated in the LMS algorithm. Consequently, the algorithm inherently contains a degree of noise, making it a promising candidate for exploring approximate computing techniques. Due to the algorithmic complexity and inherent approximations in the LMS algorithm itself, research has primarily focused on approximating the algorithm itself rather than investigating hardware-level approximations.

Moreover, the presence of numerous multipliers in filter coefficients and coefficient update circuits provides a suitable context for leveraging approximate computing, specifically approximate multipliers. This idea is the primary subject of the reference paper [6]. The methodology and concept presented in this paper have served as the primary guidelines for the approach adopted in this paper. In this paper, an approximate multiplier named Darjn is proposed, which is designed based on the DATA multiplier introduced in paper [7]. By reducing partial products and the number of required adders, the Darjn multiplier improves the implementation parameters of the approximate adaptive filter compared to other approximate multipliers. Finally, the implementation results of the adaptive filter, considering the system identification application, are compared for this multiplier with four other approximate multipliers, including the DATA multiplier.

This section provides a brief overview of research on reducing the complexity of least mean squares (LMS) adaptive filters. The LMS algorithm is an iterative algorithm designed to minimize the mean squared error between the desired signal and the output of a finite impulse response (FIR) filter. This minimization requires the calculation of the gradient of the mean squared error, which is approximated in the LMS algorithm, hence the term 'noisy gradient' [8]. Consequently, the LMS algorithm inherently contains a degree of noise, making it a suitable candidate for applying approximate computing techniques. In reference [8], a modified CORDIC-based LMS algorithm is proposed, reducing the number of multipliers by 50% and pipelining the CORDIC unit. References [9] and [10] explore using logarithmic number systems, along with serial bit-level operators and lookup tables, to reduce area and power consumption.

The critical paths of the Least Mean Squares (LMS) algorithm were analyzed in [11], where the author demonstrated that, in most cases, pipelining is unnecessary for LMS implementations. Moreover, the study revealed that the delayed LMS algorithm can enhance system performance for high sampling rates. Paper [12] explored the realization of multiplication operations using add-shift techniques within the signed LMS algorithm family. However, in all optimization cases, significant improvements in terms of latency or power consumption were not achieved through algorithmic manipulations.

III. PROPOSED APPROXIMATE CIRCUITS AND ARCHITECTURES: ADAPTIVE FILTER CASE STUDY

In this paper, similar to previous works, we have replaced exact multipliers and adders with their approximate counterparts to approximate the adaptive filter architecture. Initially, an approximate multiplier is developed based on the inexact Reverse Carry Propagation type II (RCP-II) adder. The proposed multiplier is more compatible with FPGA architectures, leading to better performance of the proposed adaptive filter compared to previous designs. In the following subsections, the proposed approximate multipliers are presented, followed by the proposed adaptive filter architecture. Additionally, the Darjn and DATA clustering technique is employed in conjunction with the RCP adder. Further details are provided in the subsequent sections. In the proposed approximate filter architectures, 32-bit adders and 16-bit multipliers are considered. In this paper, instead of exact adders in the adaptive filter structure - both in the main FIR filter section and in the coefficient update section - the proposed approximate adder in Figure 2 is used. In the proposed adders of Figure 2, 20 bits are approximated using the RCP structure, while the remaining 12 bits are implemented with an exact ripple carry adder. The RCP structure has three types: I, II, and III. In this figure, only the internal structure of type II is shown.

An array-based structure was employed for the multipliers. To approximate these structures, we considered clustering of bits using the DATE, Darjn, and conventional approaches as depicted in Figure 3. In the Darjn and DATE structures, OR gates were used instead of full adders (FAs). In this paper, for all three structures, we replaced the exact FAs and the OR gates in the Darjn and DATE structures with inexact single-bit RCP adders. It is worth noting that we evaluated all three types (I, II, and III), but since only type II yielded the desired output in the adaptive filter, this structure was considered for further analysis. In this paper, similar to previous studies, we have implemented approximate arithmetic units, specifically multipliers and adders, to approximate the adaptive filter.

As depicted in Figure 4, the multipliers and adders highlighted in orange are implemented approximately, while those highlighted in red are implemented precisely. The approximate units utilize the adders and multipliers presented in Figures 2 and 3. It is noteworthy that in this study, we have not modified the adder and multiplier that compute the error e(n) and the product of the update rate μ and the error, as approximating these units would lead to the divergence of the adaptive filter output, and these units are highly sensitive. The simulation results of the proposed structure, including convergence analysis as well as circuit parameters such as delay, area, etc., are reported in the subsequent sections.

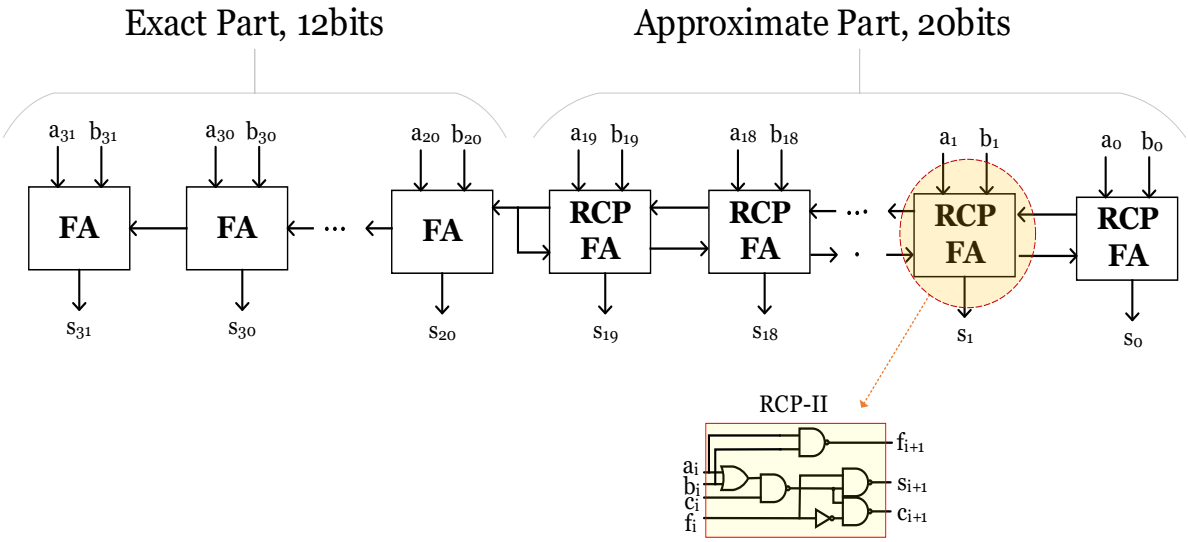

Fig. 2: Approximate 32-bit adder with a hybrid RCP (Type II) and ripple carry design.

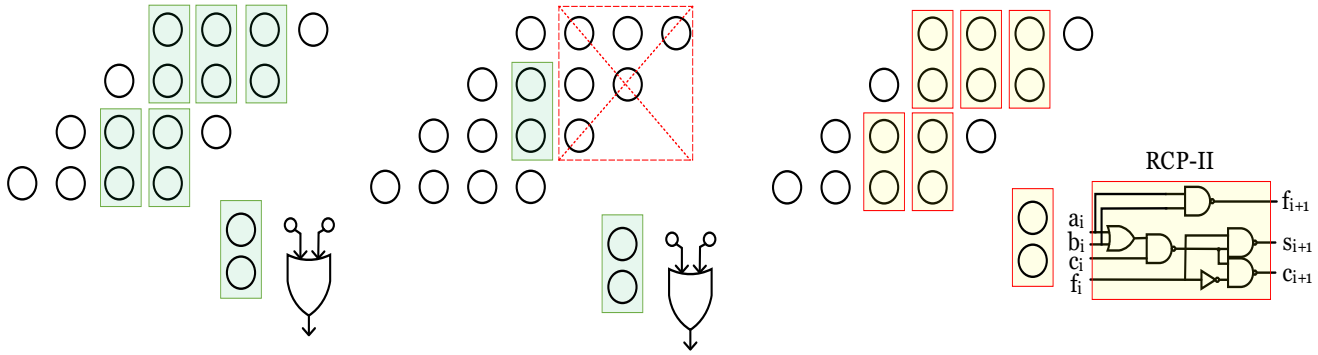

Fig. 3: Clustering strategies for approximate multipliers: DATE, Darjn, and conventional.

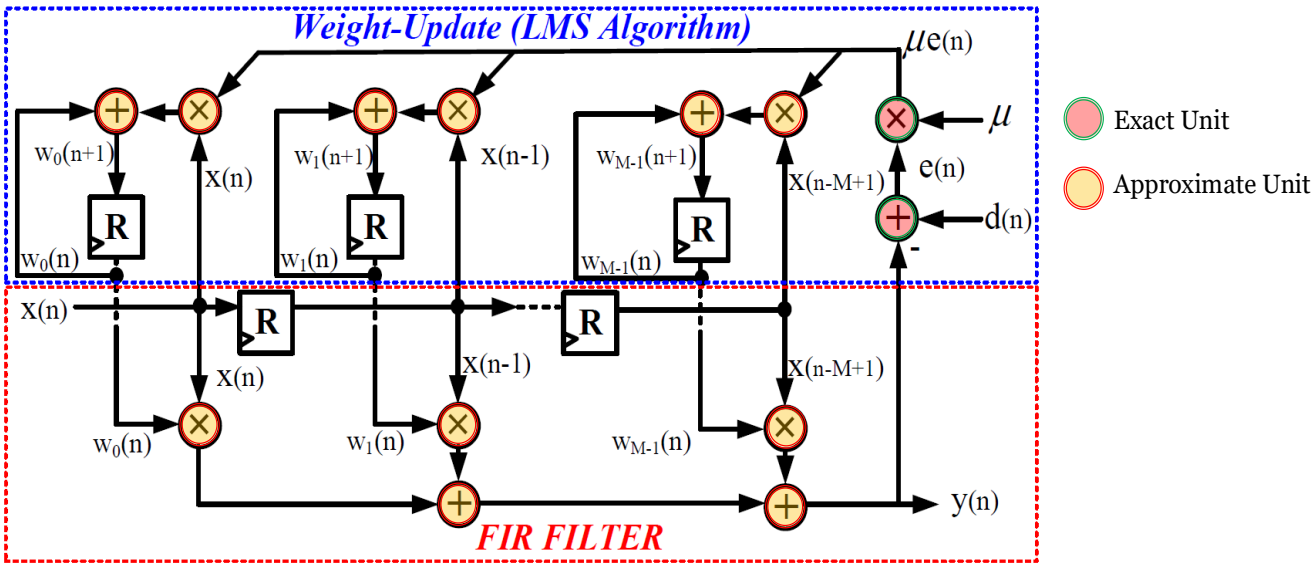

Fig. 4: Overview of the proposed approximate adaptive filter structure, highlighting the approximate (orange) and exact (red) arithmetic units.

IV. SIMULATION RESULTS AND ANALYSIS

Similar to previous studies, the Least Mean Squares (LMS) algorithm was employed to implement the adaptive

filter due to its computational simplicity, convergence properties, and stability. The intended application for the adaptive filter is system identification, as commonly adopted in related research. A low-pass filter was designed for the reference filter with a passband gain of 1 dB at 0.4π radians/sample and a stopband attenuation of -120 dB at 0.5π radians/sample. Random inputs were used for the simulations. MATLAB was utilized for implementation and convergence analysis, while the ISE software was employed to evaluate circuit parameters on the FPGA.

A. Convergence and Error

In approximate computing, the primary concern is the level of error in the output. For simple circuits like adders and multipliers, this error can be easily calculated by comparing the output of the approximate circuit to the exact circuit using the following Mean Error Distance (MED) equation:

$$MED = |Ex-Ax|/Ex$$

However, for a complex circuit such as an adaptive filter, the Mean Error Distance (MED) alone may not be a sufficient metric. In addition to the output error relative to the ideal adaptive filter, the output characteristic curve should also be examined. As mentioned, in this paper, we have employed inexact adder cells (RCP) in place of conventional OR gates in both the DATA and Darjn approximate multiplier structures. Furthermore, we have investigated three types of RCP adders (Type I, II, and III) within the conventional array multiplier structure. Among the proposed architectures, only those based on Type II RCP converged when used in adaptive filters. The structures implemented using Type I and Type III did not exhibit the necessary convergence, as illustrated in Figure 5. Consequently, the remainder of this paper focuses exclusively on Type II RCP-based architectures. Figure 6 illustrates the simulation results of the adaptive filter using various approximate multipliers. As shown in the figure, the inexact Darjn and DATA multipliers, as well as their modified structures utilizing RCP-II, and the proposed inexact multiplier based on RCP-II, all yield satisfactory convergence behavior. Furthermore, the average error distance of the modified coefficients in the adaptive filter implementation with approximate multipliers is 100.14 for the Darjn structure, 84.47 for the DATA structure, 116.51 for the modified Darjn structure, 118.88 for the DATA-RCPII structure, and 27.27 for the RCP-II array-based structure. The convergence behavior of the adaptive filter using different approximate multipliers is visually compared in Figures 5 and 6. To provide a quantitative measure of convergence performance, the Mean Error Distance (MED) of the final updated filter coefficients is calculated for each design and summarized in Table I. This metric effectively captures the steady-state error introduced by the approximate arithmetic. As the results indicate, the proposed RCP_II multiplier achieves the lowest MED value (27.27), signifying superior convergence accuracy closest to the exact filter's performance. While the other approximate multipliers (DATA_EX, Darjn_EX, DATA_RCPII, Darjn_RCPII) exhibit higher MED values, their convergence plots remain stable, indicating that the introduced error is bounded and does not lead to divergence for the given application and input data.

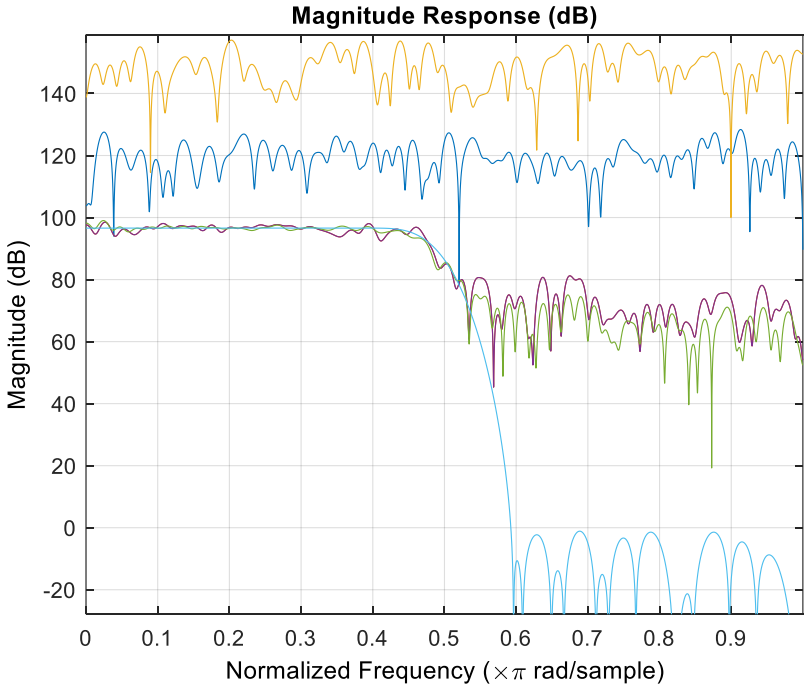

Fig. 5: Comparison of adaptive filter implementation results using RCP I, II, and III multipliers.

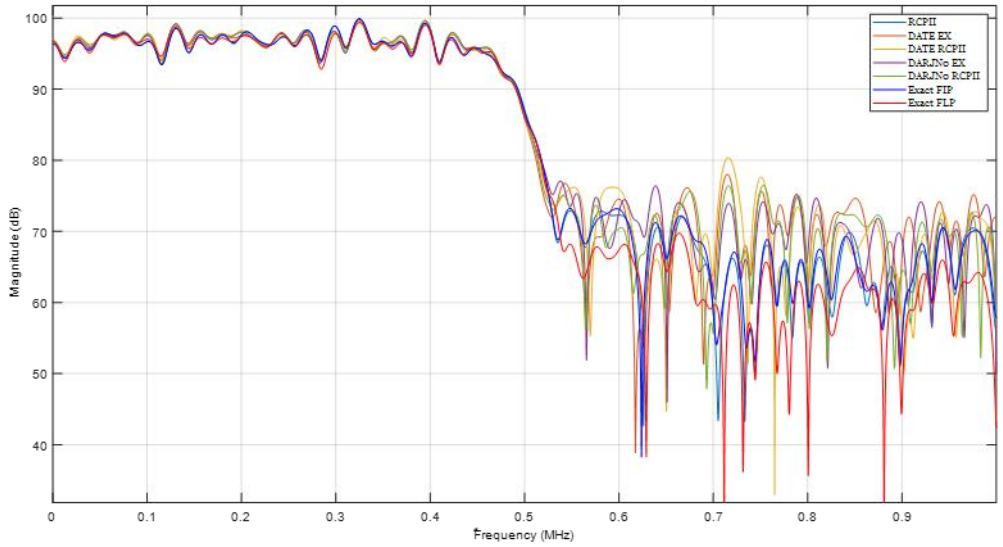

Fig. 6: Convergence behavior of an adaptive filter using different approximate multipliers.

TABLE I
MED Values for the Coefficients of the Approximate Adaptive Filter with Different Multipliers.

Multiplier Type	*Darjen_RCPII*	*Darjen_EX*	*DATE_RCPII*	*DATE_EX*	*RCPII*
MED	116.51	100.17	118.88	84.47	27.27

B. Hardware Parameters

As mentioned, only Type II converged in the approximate filter structure among the three types of Reverse Carry Propagation adders. Therefore, this paper does not report adders, multipliers, and filters designed based on Type I and Type III structures. The Type II Reverse Carry Propagation adder was first evaluated on an FPGA. Implementation results on the FPGA show that this type of adder reduces the delay from 9.87 ns to 5.469 ns (approximately 80% improvement) compared to the exact adder. However, the required number of LUTs remains unchanged at 31. Consequently, in the following subsection, we first evaluate and compare the proposed approximate multiplier with those from previous studies on an FPGA platform. Subsequently, the performance of FIR and adaptive filters based on these multipliers and adders is evaluated and reported. All hardware implementations and performance analyses were conducted on a Xilinx Virtex-6 XC6VCX57T FPGA.

Table II presents a comparison of maximum frequency and occupied LUTs for both approximate and exact multiplier implementations. The proposed multipliers in this paper, namely RCP_II, DATA_RCPII, and Darjn_RCPII, whose architectures were detailed in the previous section, are evaluated. As the results in the table demonstrate, the highest frequency improvement of 103% was achieved by the approximate multiplier DATA_RCP_II. This significant enhancement is attributed to the simultaneous use of clustering and the calculation of partial product sums using a hybrid carry-save adder. The reduction in the number of clusters implies fewer partial products, consequently requiring fewer adders to obtain the final product. Furthermore, the partial product sums were calculated using the approximate modified Booth recoding technique, which significantly contributed to the delay reduction. Moreover, based on the results from the adder section, the carry-save adder does not lead to a reduction in LUT usage. The 30% improvement compared to the DATA_EX multiplier confirms this finding, as this value is close to the approximate DATA multiplier that employs an exact adder for partial product summation. The greatest reduction in occupied LUTs was observed in multipliers utilizing clustering, and these values exhibit minimal variation across different designs due to the relatively consistent clustering algorithm employed in all cases.

Table III presents a comparison of the maximum frequency and utilized LUTs in the implementation process of the finite impulse response (FIR) filter using both approximate and exact multipliers. The proposed multiplier in this paper is referred to as RCP_II, DATA_RCPII, and Darjn_RCPII. According to the table results, none of the multipliers improved the maximum operating frequency of the circuit, and the only improvement was observed in the number of utilized LUTs. This is because, in coding these types of filters, the critical path delay, which determines the maximum operating frequency of the circuit, is not in the multipliers. This is because in none of the filter implementations with the mentioned multipliers, despite the improvement in the operating frequency of the approximate multipliers compared to the exact multiplier, an improvement in the

filter's operating frequency was observed. This indicates that the delay bottleneck of the circuit lies in other calculations, which may be the exact adders used in this structure. On the other hand, according to the obtained results, the improvement in the number of utilized LUTs is related to the structure of the approximate multipliers used, which was mentioned in the previous paragraph, due to the improvement in the number of LUTs resulting from the clustered structure of the multipliers. As seen in the results, the RCP_II multiplier with an 18% improvement and the Darjn_RCPII multiplier with an 11% improvement have achieved the highest reduction in LUT consumption.

Finally, we compare the implementation results of the adaptive filter discussed in this paper, using the proposed approximate multipliers. Due to software simulation limitations, a 10-tap adaptive filter was considered. As shown in Table V, the use of the proposed multipliers, RCP_II, DATA_RCPII, and Darjn_RCPII, does not improve the operating frequency of the circuit. In all cases, using reverse carry propagation adders has decreased the maximum operating frequency. On the other hand, in all cases using reverse carry propagation adders, a significant improvement in the number of occupied LUTs has been observed compared to the exact multiplier. This is due to the clustered structure of the DATA and Darjn multipliers, which reduces the number of partial products and consequently reduces the implementation complexity and the number of LUTs used. The greatest improvement in the number of LUTs consumed is related to the Darjn_RCPII multiplier, with a 29% improvement compared to the original version of this multiplier, which is based on an exact adder. On the other hand, using the reverse carry propagation adder in the DATA multiplier has resulted in an 18% improvement in the number of occupied LUTs, which is a 0.3% improvement compared to its exact counterpart. The frequency results provide a critical insight into applying approximate computing on FPGAs. While our combinational blocks (adders and multipliers) demonstrated significant frequency improvements of up to 80%, this benefit did not translate to the larger, sequential FIR and adaptive filter implementations. The complex feedback loops and sequential logic elements in these circuits became the new critical path, overriding the speed gains of the approximate multipliers. This result, while perhaps not what is commonly expected, is a crucial finding that highlights a key difference between theoretical gains at the component level and practical performance at the system level. We believe it is essential to present these realistic outcomes to prevent designers from making false assumptions about the scalability of approximate computing benefits on FPGAs for complex application.

While our study's scope was limited to FIR and adaptive filters, our findings provide a crucial perspective on applying approximate computing to FPGAs. Contrary to expectations from ASIC-based research, we observed that hardware gains from the base components did not scale linearly to the full-circuit implementations. This was particularly evident in the lack of frequency improvement in the full FIR and adaptive filters, despite gains in the proposed multipliers.

Furthermore, the power savings were on the order of microwatts, making them negligible compared to the overall power consumption of the FPGA. These results, while not as dramatic as those seen in some ASIC studies, are a significant contribution, highlighting that the benefits of approximate computing on FPGAs are highly application-dependent and not always a direct translation from core component improvements.

A key finding of our research, particularly relevant to edge devices, concerns the analysis of energy consumption. While approximate computing is highly effective in reducing power in ASIC designs, our results on the FPGA platform did not show a significant benefit. This can be attributed to the high static power consumption of FPGAs (in the order of watts), which dwarfs the minute dynamic power savings (in the order of microwatts) achieved by our approximate circuits. Reporting these negligible power savings, as demonstrated in related work (e.g., Hassan et al., 2023[13]), would provide a misleading representation of the true energy efficiency. Therefore, we focused on other hardware metrics where our designs showed measurable improvements. We believe this finding is a critical contribution, as it highlights a practical limitation of applying approximate computing on FPGAs for applications where overall power is the primary concern.

TABLE II
Implementation results of Approximate and Exact Multipliers.

MULTIPLIER	DELAY (NS)	FREQ. (MHZ)	# OF LUT	FREQ. %	# OF LUT %
PLAIN	16.367	61	313	-	-
RCP_II	9.491	105	293	+ 72%	+ 6%
DATA	14.287	69	218	+ 13%	+ 30%
DATA_RCPII	8.0263	124	216	+ 103%	+ 30%
DARJN	13.242	75	209	+ 22%	+ 33%
DARJN_RCPII	10.377	96	210	+ 57%	+ 32%

TABLE III
Implementation results of the FIR Filter with Approximate and Exact Multipliers.

FIR Filter	*Delay (ns)*	*Freq. (MHz)*	*# of LUT*	*# of FFs*	*Freq. %*	*# of LUT %*
Plain	18.212	54.910	4110	376	-	-
RCP_II	18.478	54.117	3356	378	+ 0%	+ 18%
DATA	18.816	53.147	4300	389	+ 0%	- 4%
DATA_RCPII	18.502	54.049	3715	393	+ 0%	+ 9.6%
Darjn	18.262	54.758	3750	382	+ 0%	+ 8.7%
Darjn_RCPII	18.022	55.487	3624	381	+ 1%	+ 11.8%

TABLE V
Implementation Results of Adaptive Filter with Approximate and Exact Multipliers

Adaptive Filter	*Delay (ns)*	*Freq. (MHz)*	*# of LUT*	*# of FFs*	*Freq. %*	*# of LUT %*
Plain	16.7	59.8	8017	511	-	-
RCP_II	19.01	52.5	7982	504	- 12.2%	+ 0.4%
DATA	16.18	61.79	6152	525	+ 3.3%	+ 23.2%
DATA_RCPII	19.04	52.5	6123	508	- 12.2%	+ 23.6%
Darjn_EX	16.59	60.27	6057	527	+ 0.7%	+ 24.3%
Darjn_RCPII	18.88	52.9	5650	542	- 11.5%	+ 29.5%

V. CONCLUSION

In this research, three novel approximate multipliers based on the modified Booth algorithm are proposed, and their performance is evaluated in adaptive and finite impulse response (FIR) filters. Simulation results demonstrate a significant improvement in operating frequency compared to existing designs. Additionally, these multipliers substantially reduce the hardware area in FIR filters and exhibit reasonably good performance in adaptive filters. All proposed circuits have been implemented on FPGA devices, enabling their application in edge computing. As a suggestion for future research, these multipliers can be utilized in audio and image processing blocks, and the resulting improvements can be assessed. Furthermore, investigating the use of modified Booth adders in implementing other approximate multipliers and comparing them with the proposed designs of this paper can be the subject of future studies. A limitation of our current study is using random input data for simulations. While this approach was effective for evaluating the direct hardware benefits of our approximate multipliers, it does not fully replicate the conditions of specific real-world applications. Future work will involve testing our proposed architectures using realistic datasets from edge computing applications, such as speech processing, biomedical signals (e.g., ECG), or IoT sensor data. This will provide a more comprehensive analysis of our designs' performance and error tolerance in practical scenarios.

REFERENCES

[1] H. J. Damsgaard, A. Ometov, and J. Nurmi, "Approximation Opportunities in Edge Computing Hardware: A Systematic Literature Review," ACM Comput. Surv., vol. 55, no. 12, p. Article 252, 2023, doi: 10.1145/3572772.

[2] S. S. Sahoo, S. Ullah, S. Bhattacharjee, and A. Kumar, "AxOCS : Scaling FPGA-Based Approximate Operators Using Configuration Supersampling," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. PP, pp. 1-14, 06/01 2024, doi: 10.1109/TCSI.2024.3385333.

[3] S. Ullah and A. Kumar, Approximate Arithmetic Circuit Architectures for FPGA-based Systems. 2023.

[4] V. Chowthri, A. Uma, and P. Kalpana, "Design and implementation of low complexity LMS adaptive filter," International Journal of Nonlinear Analysis and Applications, vol. 12, no. Special Issue, pp. 1827-1833, 2021, doi: 10.22075/ijnaa.2021.5893.

[5] V. B. A. Rasel, A. Uma, and P. Kalpana, "Design of window based FIR filter for electrocardiogram noise removal," International Journal of Nonlinear Analysis and Applications, vol. 12, no. Special Issue, pp. 1519-1528, 2021, doi: 10.22075/ijnaa.2021.5804.

[6] D. Esposito, G. D. Meo, D. D. Caro, N. Petra, E. Napoli, and A. G. M. Strollo, "On the Use of Approximate Multipliers in LMS Adaptive Filters," in 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 27-30 May 2018, pp. 1-5, doi: 10.1109/ISCAS.2018.8351089.

[7] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, and A. Yakovlev, Energy-efficient approximate multiplier design using bit significance-driven logic compression. 2017, pp. 7-12.

[8] M. Chakraborty, A. Dhar, and M. Lee, "A trigonometric formulation of the LMS algorithm for realization on pipelined CORDIC," Circuits and Systems II: Express Briefs, IEEE Transactions on, vol. 52, pp. 530-534, 10/01 2005, doi: 10.1109/TCSII.2005.850784.

[9] D. Allred, H. Yoo, V. Krishnan, W. Huang, and D. Anderson, "LMS adaptive filters using distributed arithmetic for high throughput," Circuits and Systems I: Regular Papers, IEEE Transactions on, vol. 52, pp. 1327-1337, 08/01 2005, doi: 10.1109/TCSI.2005.851731.

[10] M. T. Khan, S. R. Ahamed, and F. Brewer, "Low Complexity and Critical Path Based VLSI Architecture for LMS Adaptive Filter Using Distributed Arithmetic," in 2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems (VLSID), 7-11 Jan. 2017 2017, pp. 127-132, doi: 10.1109/VLSID.2017.16.

[11] P. K. Meher and s. y. Park, "Critical-Path Analysis and Low-Complexity Implementation of the LMS Adaptive Algorithm," Circuits and Systems I: Regular Papers, IEEE Transactions on, vol. 61, pp. 778-788, 03/01 2014, doi: 10.1109/TCSI.2013.2284173.

[12] S. Choudhary, P. Mukherjee, M. Chakraborty, and S. S. Rath, "A SPT Treatment to the Realization of the Sign-LMS Based Adaptive Filters," IEEE Transactions on Circuits and Systems I: Regular Papers,

vol. 59, no. 9, pp. 2025-2033, 2012, doi: 10.1109/TCSI.2012.2185300.

[13] M. Hassan, F. Awwad, M. Atef, and O. Hasan, "Approximate Computing-Based Processing of MEA Signals on FPGA," Electronics, vol. 12, no. 4, p. 848, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/4/848.