Mehran University Research Journal of Engineering and Technology

https://doi.org/10.22581/muet1982.3220

2025, 44(2) 136-143

An optimized implementation of adaptive noise canceller based on proposed shift and add multiplier

Aneela Pathan [a, *], Khalil M. Zohaib [a], Rizwan Aziz [a], Adil Hussain Chandio [a], Syed Haseeb Shah [a]

[a] *Department of Electronic Engineering, The University of Larkana, Larkana, Pakistan*

[*] Corresponding author: Aneela Pathan, Email: pathan_aneela@uolrk.edu.pk

Received: 13 November 2024, Accepted: 27 March 2025, Published: 01 April 2025

K E Y W O R D S

DSP

FPGA

MATLAB

Shift and add

Optimization

A B S T R A C T

Information is deteriorated by communication channels in several ways. The most notable is the addition of noise to the signal during transmission. Noise is reduced by the use of adaptive filters. Wiener, Steepest, and LMS are the most often utilized. While in hardware translation on ASICS and FPGAs, adaptive filters require more resources than straightforward FIR or IIR designs. Reducing resources is necessary to optimize the implementation. The literature on resource-optimized filter implementation with multiplier optimization has been seen with a number of approaches. In this study, a new proposed shift and add multiplier is used to create an FPGA-based adaptive noise canceller based on the Steepest - descent algorithm, and its performance is compared with a traditional version. The adaptive noise canceller is first simulated in MATLAB and then designed in Xilinx Virtex 7 FPGA using the ISE 14.7 tool, but the proposed architecture is too flexible to be carried out on any FPGA board. The suggested shift and add multiplier consume less FPGA resources than the original shift and add multiplication scheme alone, and in designing an adaptive noise canceller. The proposed method also performs better than the conventional approach in terms of maximum frequency achieved. Therefore, it can be inferred that the proposed shift and add multiplier approach can be adapted for resource-optimized implementation in the communication domain and in DSP applications.

1. Introduction

Data gets distorted during transmission via a communication channel in the result of channel disturbances [1]. Creating an inverse filter that responds reciprocally to the channel and lessens its effects is the way to dispose of it. The Steepest-descent algorithm and other adaptive filters are commonly used in the design of inverse filters. In certain instances, channel degradations and noise may both harm the data. To achieve dual functionality, an additional factor is introduced to the inverse filter for noise removal [2]. Fig.1 elaborates on the idea.

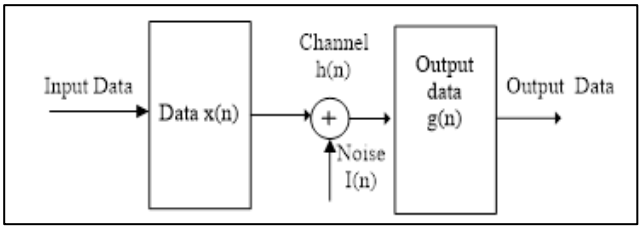

Fig. 1. Data Transmission over Communication Channel and Addition of Noise

The user data, we send over the channel is represented by f(n) in Fig.1, and the channel impulse response h(n). The presence of some random noise I(n) could tack onto the signal. Thus g(n) at the receiver end may be represented as Eq. (1).

$$g(n) = f(n) * h(n) + I(n) \quad (1)$$

In the time domain, f(n) and h(n) are convolved together, while convolution can occasionally prove difficult to handle. It is preferable to translate it into the multiplication in the frequency domain. The Eq. (2) is the frequency domain representation of Eq. (1).

$$G(N) = F(N).H(N) + I(N) \quad (2)$$

We have the degraded signal G(N) at the receiving end. However, by rearranging Eq. (2), we may obtain our desired term as we need the original data F(N) from that degraded data. Equation (3) represents the required noise term.

$$F(N) = \frac{G(N)}{H(N)} - \frac{I(N)}{H(N)} \quad (3)$$

The first term, G(N)/H(N) is referred to as an inverse filter since it represents the ratio of the received data to the channel impulse response. We are more interested in determining the noise term because the tests may yield the channel behavior. Another term is the ratio of noise to channel impulse response.

We must apply a Steepest-descent-based filter to adjust this noise term; this filter works better than an inverse filter since it takes into account both the statistical parameter of the noise (I(N)) and the degradation function (H(N)).

The class of adaptive filters known as Steepest-descent modifies its weight according to the error value computed between the desired and received data. The desired data is a delayed copy of the input data that is sent before to the receiver to compensate for any potential delays caused by the channel [2].

The error between the two sets of data (filtered and desired) can be computed using Eq. (4) where F^(N) represents the desired data and G(N) represents the filtered signal.

$$E^2 = E(G - F\hat{})^2 \quad (4)$$

Fig. 2 shows the block diagram for the traditional adaptive noise canceller to obtain the output data that is a replica of the desired data while minimizing the mean square error (the objective function).

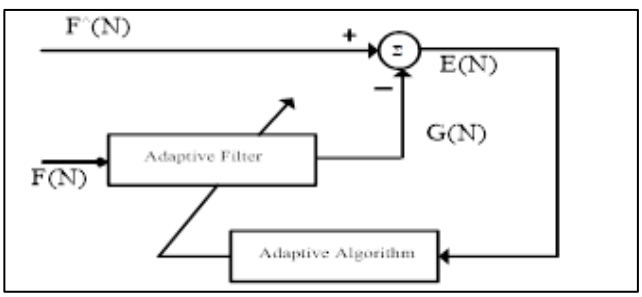

Fig. 2. Traditional Adaptive Noise Canceller

ASICs and FPGAs may be used in the hardware-based adaptive filter implementation. An FPGA-based design may significantly be impacted with the aid of using elements including deciding on the right FPGA boards, the automation tool (simulation) for Electronic Design (synthesis), and efficient programming techniques to go with resources efficient implementation. However, as an emerging technology, FPGA can implement resource-ambitious algorithms with more optimization than ASIC [3]. The optimization of the algorithm yields a more compact design in terms of both the obtained frequency and area.

The multiplier complexity [4], present in practically all DSP systems, including FIR, IIR, FFT, and others [5-7], is the primary problem when optimizing DSP algorithms. Various efforts have been undertaken to reduce multiplier complexity and create quick and effective DSP algorithms [8].

Several well-known techniques published for optimizing the multiply and accumulate (MAC) portion include: Booth's Algorithm [9, 10], Wallace Tree Multiplier [11], DADDA Multipliers [12], and Vedic Multipliers [13].

Besides optimized multiplier design, various other DSP systems have been implemented using FPGA. Some of them are real-time signal processing [14], wireless communications [15], image and video processing [16], radar and sonar systems [17], Machine Learning and AI Acceleration [18], Sensor Data Processing [19], and other Emerging Applications.

The authors have been focusing on hardware-based implementations of the multiplier optimization. Their work on the optimal implementation of multipliers is reported in [20-26]. One of the most recent publications in this domain is the FPGA-based implementation of modified shift and add multiplier [27]. This paper is an extension of the reported work that shows the optimized implementation of Steepest-descent-based adaptive noise canceller with modified shift and add multiplier. The performance of the proposed multiplier-based implementation is compared with traditional version. The adaptive noise canceller is first simulated in MATLAB and then designed in Xilinx Virtex 7 XC7vx330tffg1157 FPGA using ISE 14.7 tool. The proposed implementation, which utilizes a shift and add multiplier approach, requires fewer FPGA resources than traditional methods and demonstrates superior performance regarding achieved frequency. Consequently, it can be concluded that this approach is suitable for resource-

optimized applications in communication and digital signal processing (DSP) fields.

The paper continues as outlined below: the subsequent section delves into the architecture of the proposed shift and add multiplier, which is based on the FPGA architecture of the adaptive noise canceller. In the third section, simulation plots generated in MATLAB for adaptive noise cancellers are presented. The fourth section provides tables detailing the FPGA resource utilization for the proposed shift and add multiplier and its application in adaptive noise cancellers. Finally, the paper concludes in section five.

2. Proposed Architecture of Modified Shift and Add Multiplier and Adaptive Noise Canceller

When it comes to multiplication, the shift and add approach is among the simplest and most well-known. According to this strategy, the multiplicand's shifting or accumulation would be determined by the multiplier's LBS bit. This idea is demonstrated in Fig. 3, where the addition is done for the LBS to be one and the shift is done for the LBS bit of the multiplier to be zero. Since the new value stays in the accumulator for every number of bits, its size must be 2N (M+N), resulting in the total delay being of N cycles [28].

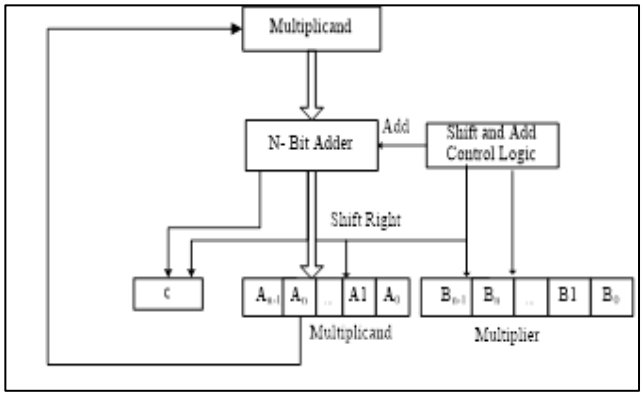

Fig. 3. Block Diagram of Shift and Add Multiplier

This approach is simple and easy to use, but it uses a lot of hardware because shift and add operations are used to implement the multipliers in logic [2]. Fig. 4 reproduces the suggested 8 × 8-bit shift and add multiplier architecture.

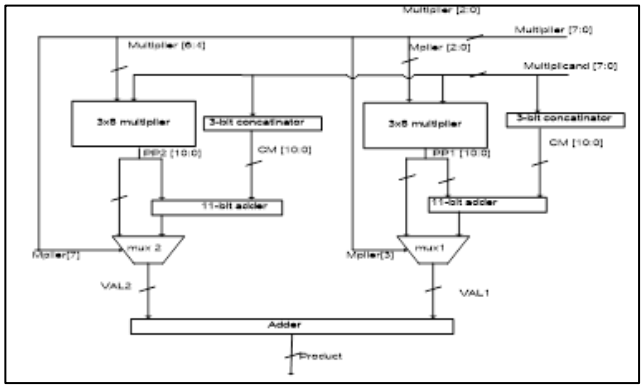

Fig. 4. Proposed Architecture of 8 × 8 Multiplier [27]

In the proposed 8 × 8 shift and add multiplier rather than checking each multiplicand bit (producing N cycle delay) the multiplier is checked (reducing N-1 cycles).

Following the modular approach, first, a 3 × 8 multiplier is designed consistent with the given algorithm and then two instants of it are called in the final implementation.

i. In the 3×8 multiplier, the values of the multiplier are 0-7 (3-bits), and 0-255 for multiplicand (8-bits).

ii. For zero multiplier value, the net product for any multiplicand will be zero.

iii. The produced output would be the same to multiplicand value for the case the multiplier is one.

iv. For other remaining possible multiplier values the output would be generated as mentioned below:

- Measure 1: Is it possible to write a multiplier in 2^n representation? For true, add n zeros to the multiplicand.
- Measure 2: Is it possible to write a multiplier in 2^n +1 representation? For true, add n zeros to the multiplicand. and add with actual multiplicand value.
- Measure 3: Is it possible to write a multiplier in 2^n +2^mrepresentation? For true, add n zeros to the multiplicand and add it with m zeros appended multiplicand.
- Measure 4: Is it possible to write a multiplier in 2^n -1 representation? For true, add n zeros to the multiplicand and subtract from it the original multiplicand.

The multiplier, designing this way reduces the resources and results in higher achieved frequency.

Filters require frequent multiplication of received data and filter weights. Also, the adaptive filter weights update equation undergoes multiplications. When directly translated on hardware, the implementation may cause huge resource utilization, while if the optimized multiplier is used instead a general optimized design is achieved.

The adaptive noise cancellers are implemented using FPGA with both the traditional and the suggested shift and add multiplier. Fig. 5 shows the FPGA-based design.

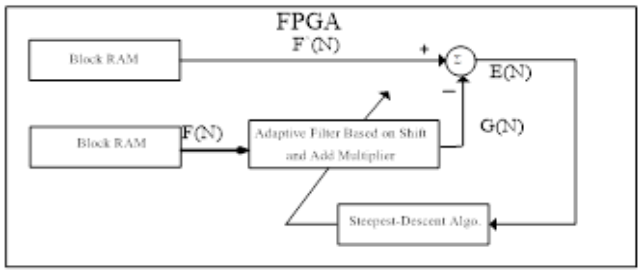

Fig.5. Adaptive Noise Canceller with Proposed Approach

The Steepest-descent adaptive filter requires the desired data and the noisy data for the filter to work [29].

One hundred samples of the desired data and the noisy input signal are stored in the FPGA's block-ram. The adaptive filter filters the noisy signal using the Steepest descent algorithm's weight update equation, provided below.

Autocorrelation matrix R, cross-correlation vector p, step-size μ, initial filter coefficients w (0), and maximum number of iterations N are the inputs.

For n = 1, ..., N do

$w(n+1) = w(n) - \mu[Rw-p]$.

end

Output: The filter w (0) at different time instants n.

3. Simulation Results of MATLAB

The Steepest descent-based adaptive noise canceller is simulated in MATLAB, and the results produced are reported in various proceeding graphs.

The adaptive filter is based on the error estimation between the filtered signal and the desired signal, it is required to produce the desired signal as a reference value at the receiver end. The desired signal is generated from the same input sound value producing a delay of ten thousand samples that would compensate for the real-time delay the signal may observe.

In MATLAB, eight thousand samples of the stored input voice signal, and the desired signal with amplitude between -0.8 and 0.8 root mean square (RMS) valu is plotted. The plots of the input and desired voice signals are represented in Fig.6 below.

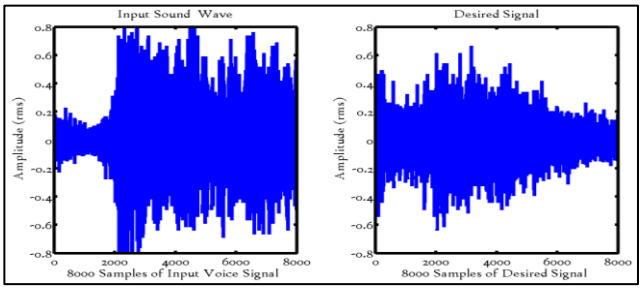

Fig. 6. 8000 Samples of Input and Desired Signal

A random Gaussian noise is generated using the Gaussian noise generator in MATLAB and is added to the input noise to see the channel effect. The rms value of the noise is kept between 0 to 0.035. Fig.7 shows the Gaussian noise and noisy input signal. The impact of noise may be seen in the input signal.

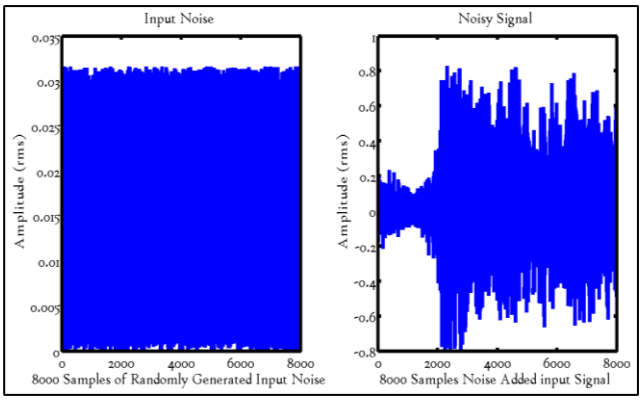

Fig. 7. 8000 Samples of Input Noise and 8000 Samples of Noise Added Input Signal

The noisy signal is filters through an adaptive filter for noise removal. The output results are shown in Fig.8.

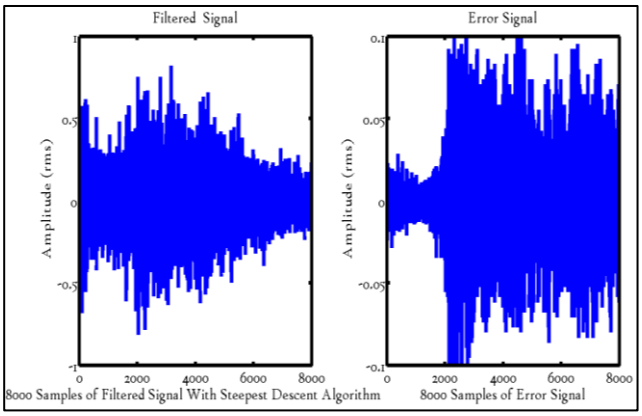

Fig. 8. 8000 Samples of Error Signal and Filtered Signal

The graph in Fig. 8 illustrates that the adaptive noise canceller based on the Steepest Descent method yields an output that matches the desired signal. To confirm this, an error graph comparing the filtered output to the desired signal has also been created. The resulting error value, measured in root mean square, is sufficiently low to deem the filter suitable for use in communication systems aimed at noise cancellation.

4. FPGA-Based Implementation and Results

The 8x8 shift and add multipliers utilizing the traditional method (Fig. 3) and the proposed method (Fig. 4) have been implemented on FPGA, and their architectures have been examined. The results are summarized in Table 1. Additionally, a comparison is conducted between the proposed approach and the conventional shift and add multiplier technique for the Steepest Descent-based adaptive noise canceller, with the outcomes detailed in Table 2.

Table 1

Proposed and typical shift and add multiplier on FPGA

Factors	Proposed Architecture	Conventional Architecture
LUTS	85	147
Adders/ Subtractors	2: [11-bit add/sub] 3:[12-bit adder]	7:[16-bit adders]
Multiplexers	2[1-bit 2-to-1 multiplexer] 24[11-bit 2-to-1]	8:[16-bit 2-to-1 multiplexer]
Logic Levels	12	20
Delay (ns)	2.163	3.830
Frequency (MHz)	462.235	261

The architectural elements under comparison include lookup tables, memory, and macro statistics (which indicate additional utilized components), while the efficiency metrics encompass logic levels, delay, and frequency. In the independent implementation of the proposed shift-and-add multiplier, resource consumption is lower than that of the traditional multiplier, and the observed delay is minimal, leading to an optimal frequency achievement.

Table 2

FPGA-based results of the proposed and conventional shift and add multiplier-based noise canceller

Factors	Proposed Architecture	Conventional Architecture
LUTS	577	1012
BROM	2	2
Macro Statistics	Add/Sub:23 Registers:176 Mux:104	Add/Sub:38 Registers:15 Mux:32
Logic Levels	23	38
Delay(ns)	4.623	5.62
Frequency (MHz)	216.293	177.81

Two architectures of a two-tap adaptive filter for noise cancellation have been implemented on the Xilinx Virtex 7 FPGA to filter one hundred samples of a noisy signal stored in the block RAM. The first design utilizes a conventional shift-and-add multiplier, while the second design incorporates a proposed multiplier. The resource utilization and the achieved frequency for both designs are summarized in the table above.

The lookup tables of the suggested design are nearly half than in the traditional design; additionally, the macro elements are reduced in the conventional method. When evaluating performance based on frequency, the proposed architecture performs well.

Various graphs in the section below provide the visual representation of the results discussed above to have more clear idea of the resource utilization and the performance achieved.

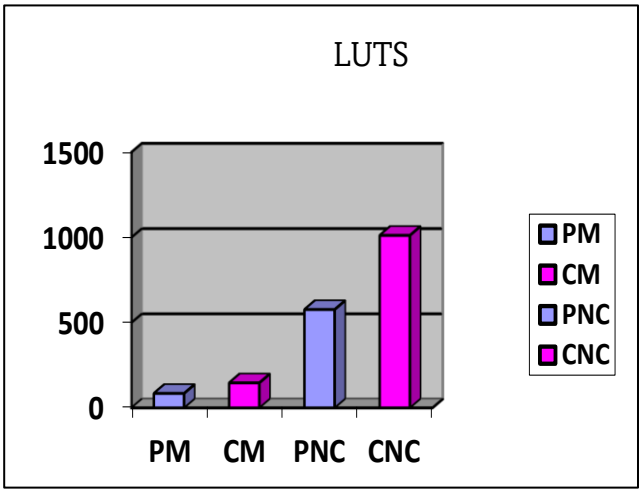

Fig. 9. Look Up Tables (LUTS) Consumed by The Convectional Multiplier (CM) and Proposed Multiplier (PM) Alone and in the Design of Adaptive Noise Canceller with a Proposed Multiplier (PNC) and The Conventional Multiplier (CNC)

Fig. 9 illustrates that the count of lookup tables in the proposed multiplier and the adaptive noise canceller utilizing the suggested multiplier is lower than that of the traditional method.

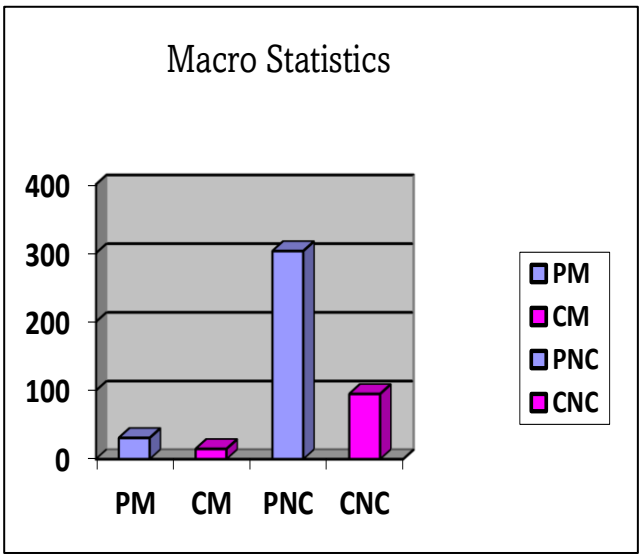

Fig. 10. Macro Statistics of The PM, CM, PNC and CNC

In Fig. 10, Macro Statistics, which comprise adder/subtractors, registers, multiplexers, and block RAM, are illustrated for both methods separately and in the design of the adaptive noise canceller. The utilization of these resources in the proposed design exceeds that of the traditional method, necessitating further optimization.

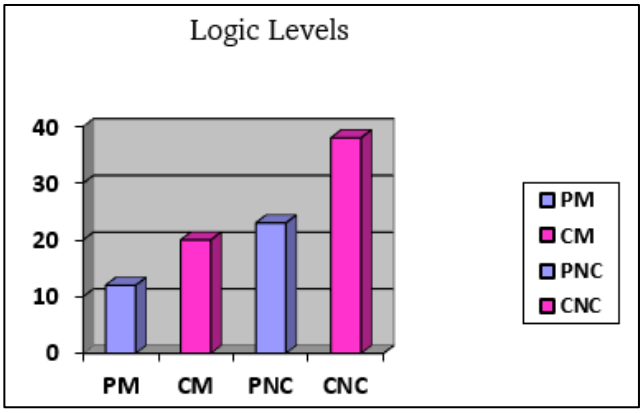

Fig. 11. Logic Levels of The PM, CM, PNC and CNC

The overall logic level in the design is illustrated in Fig. 11. The logic level is a performance parameter that has a direct effect on the delay and frequency of the entire system. The overall logic levels in the suggested designs are lower than anticipated.

The delay, another performance parameter is illustrated in Fig. 12 below. It is observed that the proposed design has a lower delay than the traditional design, thus rendering the proposed method more resilient and preferable.

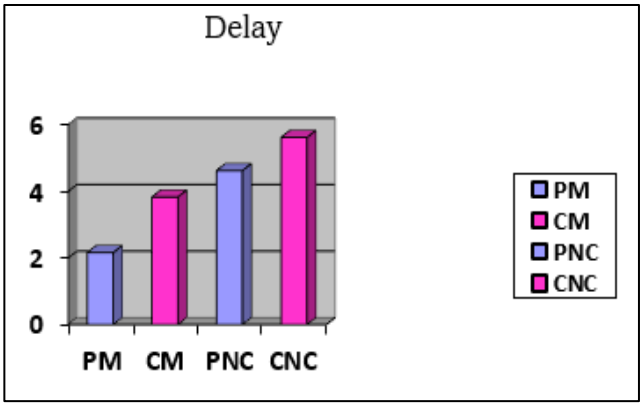

Fig. 12. Delay Observed by PM, CM, PNC, and CNC in Nanoseconds

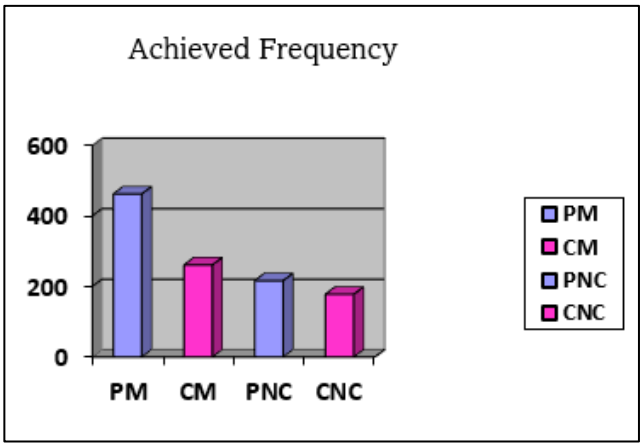

Fig. 13. Maximum Frequency Achieved by PM, CM, PNC, and CNC in MHz

The maximum achieved frequency of the suggested multiplier and adaptive noise canceller, as illustrated in Fig. 13, is considerably greater than that of the conventional design, thus making the design more feasible.

These outcomes of the above results suggest that the proposed method could be utilized as an optimized solution for the hardware implementation of resource-intensive adaptive filters.

5. Conclusion

This study utilizes a Xilinx Virtex 7 XC7vx330tffg1157 board in conjunction with the ISE 14.2 tool to develop an area-optimized version of an adaptive noise canceller that employs the Steepest Descent algorithm, with a newly proposed shift-and-add multiplier. The performance of this implementation is subsequently compared with the traditional approach.

A comparative analysis of different performance metrics indicates that the proposed shift and add multiplier consumes fewer FPGA resources compared to the conventional shift and add multiplier when utilized independently. This reduction in resource usage is similarly observed when the proposed multiplier is employed in the design of an adaptive noise canceller, demonstrating superior performance over traditional methods.

The design is founded on the shift and add multiplier, which is relatively easy to implement in practice but incurs higher resource consumption and takes more cycles. This drawback of the multiplier restricts the design's application in systems that need greater speed and fewer resources, such as satellite communications or other high-speed wireless systems. Although the suggested shift and add multiplier somewhat addresses this issue, further efforts are required in future to achieve a level of optimization comparable to other fast multipliers.

6. References

[1] N. K. Yadav, A. Dhawan, M. Tiwari, and S. K. Jha, "Modified Model of RLS Adaptive Filter for Noise Cancellation", Circuits, Systems, and Signal Processing, pp. 1-23, 2024.

[2] A. Pathan, "A Novel Approach Toward Algorithm Architecture Co-Optimization for the Application of Adaptive Noise Cancellation for Wireless Communication", Sukkur IBA Journal of Computing and Mathematical Sciences, vol. 7, pp. 51-59, 2023.

[3] J. A. Belloch, G. León, J. M. Badía, A. Lindoso, and E. San Millan, "Evaluating the computational performance of the xilinx ultrascale+ eg heterogeneous mpsoc", The

Journal of Supercomputing, vol. 77, pp. 2124-2137, 2021.

[4] A. Boutros and V. Betz, "FPGA architecture: Principles and progression", IEEE Circuits and Systems Magazine, vol. 21, pp. 4-29, 2021.

[5] M. H. Rais, "Efficient hardware realization of truncated multipliers using FPGA", International Journal of Applied Science, vol. 5, pp. 124-128, 2009.

[6] M. Boulasikis, M. Birbas, N. Tsafas, and N. Kanakaris, "Efficient Utilization of FPGA Multipliers for Convolutional Neural Networks", 2021 10th International Conference on Modern Circuits and Systems Technologies (MOCAST), 2021, pp. 1-5.

[7] T. Memon and P. Beckett, "The impact of alternative encoding techniques on field programmable gate array implementation of sigma-delta modulated ternary finite impulse response filters", Australian Journal of Electrical and Electronics Engineering, vol. 10, pp. 107-116, 2013.

[8] S. Ullah, S. Rehman, M. Shafique, and A. Kumar, "High-performance accurate and approximate multipliers for FPGA-based hardware accelerators", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, pp. 211-224, 2021.

[9] H. Wu and X. Gao, "Efficient multiplier and FPGA implementation for NTRU prime", 2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2021, pp. 1-5.

[10] K. VAMSI, "COMPARATIVE ANALYSIS OF VARIOUS TYPES OF MULTIPLIERS FOR EFFECTIVE LOW POWER AND TIME."

[11] C. WALLACE, "A Suggestion for a FAST Multipliers", IEEE Trans. on Computers, vol. 19, pp. 153-157, 1970.

[12] L. Dadda, Some schemes for parallel multipliers: IEEE Computer Society Press, 1990.

[13] S. Jagadguru and S. B. K. T. Maharaja, "Vedic Mathematics: Sixteen Simple Mathematical Formulae from the Veda", 2009.

[14] X. Liu, C. Jiang, S. Yang, B. Zhu, and Z. Zhao, "Design and Implementation of Real-time Signal Processing Heterogeneous System for Unmanned Platform", 2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP), 2023, pp. 340-345.

[15] J. Jin, Q. Shang, H. Zhang, and H. Lu, "FPGA-based adaptive rate-reduced visible light Ethernet communication system", Optics Continuum, vol. 4, pp. 522-534, 2025.

[16] T. Prabu and K. Srinivasan, "Design and Implementation of High-Performance FPGA Accelerator for Non-Separable Discrete Fourier Transform Optimizing Real-Time Image and Video Processing", Journal of Nanoelectronics and Optoelectronics, vol. 19, pp. 843-856, 2024.

[17] R. Wen, H. Zhang, and L. Xu, "An Efficient Dynamic Engineering Implementation Architecture for MIMO Radar System", Remote Sensing, vol. 17, p. 832, 2025.

[18] A. Mouri Zadeh Khaki and A. Choi, "Optimizing Deep Learning Acceleration on FPGA for Real-Time and Resource-Efficient Image Classification", Applied Sciences, vol. 15, p. 422, 2025.

[19] M. Vaithianathan, S. Udkar, D. Roy, M. Reddy, and S. Rajasekaran, "FPGA Design for Multimodal Sensor Data Fusion in Autonomous Robots", 2024 International Conference on Sustainable Communication Networks and Application (ICSCNA), 2024, pp. 237-242.

[20] A. Pathan, T. D. Memon, S. Keerio, and I. H. Kalwar, "FPGA Based performance analysis of multiplier policies for FIR filter", 2016 International Conference on Advances in Electrical, Electronic and Systems Engineering (ICAEES), 2016, pp. 17-20.

[21] T. D. Memon and A. Pathan, "An approach to LUT based multiplier for short word length DSP systems", 2018 International Conference on Signals and Systems (ICSigSys), 2018, pp. 276-280.

[22] A. Pathan and T. D. Memon, "An optimised 3× 3 shift and add multiplier on FPGA", 2017 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST), 2017, pp. 346-350.

[23] A. Pathan, T. D. Memon, F. K. Sohu, and M. A. Rajput, "Analysis of existing and proposed 3-bit and multi-bit multiplier algorithms for FIR filters and adaptive channel equalizers on FPGA", Quaid-E-Awam University Research Journal of Engineering, Science & Technology, Nawabshah., vol. 19, pp. 81-89, 2021.

[24] A. Pathan, T. D. Memon, and S. Memon, "A carry-look ahead adder based floating-point multiplier for adaptive filter applications",

International Journal of Computing and Digital Systems, vol. 7, pp. 95-102, 2018.

[25] A. Pathan, R. Balal, T. D. Memon, and S. A. Memon, "Analysis of booth multiplier based conventional and short word length FIR filter", Mehran University Research Journal of Engineering & Technology, vol. 37, pp. 595-602, 2018.

[26] A. Pathan, T. D. Memon, and F. Sohu, "A 3-Input Lookup Table Based Signed Multiplier For DSP Systems", Complement, vol. 7, p. 0111.

[27] A. Pathan, A. H. Chandio, and R. Aziz, "An Optimization in Conventional Shift &Add Multiplier for Area-Efficient Implementation on FPGA", 2022 International Conference on Emerging Technologies in Electronics, Computing and Communication (ICETECC), 2022, pp. 1-6.

[28] S. Mirzaei, A. Hosangadi, and R. Kastner, "FPGA implementation of high speed FIR filters using add and shift method", 2006 International Conference on Computer Design, 2006, pp. 308-313.

[29] T. D. Memon, A. Pathan, and P. Beckett, "FPGA based implementation and area performance analysis of sigma-delta modulated steepest algorithm for channel equalization", 2018 12th International conference on signal processing and communication Systems (ICSPCS), 2018, pp. 1-6.

