Design and Implementation of Least Mean Square Adaptive Filter Using Verilog

Sandu Ajay Kumar, T. Satya Savithri

Abstract:** Adaptive filters are ability of adaptation to an unknown environment. These filters have been used widely because of its capable of operating in an unknown system and low power implementation of hardware. Adaptive filters have great range of signal processing and control operations for the tracking time variations of input statistics and Robust to the noise immunity. These filters used various Areas like Noise cancelling (interface cancelling), system identification, inverse modelling and echo predication. Adaptive filters structures have the adaptive algorithms to perform the time variations of the input statics and Robust to the noise cancelling. The most popular algorithm is LMS (Least Mean Square) it produces the least mean square of error signal in the adaptive filter to minimize noise power. Adaptive filter structures follow the two algorithms RLS and LMS. RLS algorithms excellent performance with increased complexity and the filter coefficients that minimize waited linear least squares cost function relating to the input signals. It requires infinite memory for error signal. LMS algorithms are simplest to understand and describe the hardware of the system compare to the RLS (Recursive Least Square). LMS algorithms are follows the stochastic gradient descent method to minimize the error signal and de-nosing task. It estimates the gradient vector from the input data and LMS incorporates an iterative procedure that makes successive corrections to the weight vector in the direction of the negative of the gradient vector which leads to minimum mean square error. It doesn't require correlation functions for calculations. The main aim of the project is to design the LMS algorithm based adaptive filter using Verilog HDL to reduce the power consumption, hardware complexity and improving the noise cancelling for the adaptive filter on the FGPA boards. An important challenge in the LMS adaptive filters design implementation of structural model in the Verilog HDL for image processing to target the noise cancelling, power and hardware complexity. Tool use for implementation the structural model of LMS filter is vivado tool and Xilinx software for FPGA board implementation.

***Keywords:** Adaptive Filters, FPGA, LMS, RLS, Increased Complexity*

I. INTRODCTION

An adaptive filter is a type of digital filter that adjusts its parameters in real-time to optimize its performance based on the input data and a desired output.

Manuscript received on 21 October 2024 | Revised Manuscript received on 10 December 2024 | Manuscript Accepted on 15 December 2024 | Manuscript published on 30 December 2024.
*Correspondence Author(s)

Sandu Ajay Kumar*, Department of Electronics & Communication Engineering, Jawaharlal Nehru Technological University, Hyderabad (Telangana), India. Email ID: ajaykumarsandu@gmail.com, ORCID ID: 0009-0009-2023-1395
Dr. T. Satya Savithri, Department of Electronics & Communication Engineering, Jawaharlal Nehru Technological University, Hyderabad (Telangana), India. Email ID: sanduajaykumar309@gmail.com

These filters find applications in various fields, including signal processing, communication systems, audio processing, image processing, and control systems [1].

Unlike traditional fixed filters with predetermined coefficients, adaptive filters are capable of modifying their coefficients iteratively to minimize some defined error or achieve a specific objective. The key idea behind adaptive filters is to continuously adapt to changing input signals and to converge towards a desired response. This adaptability makes them particularly useful in scenarios where the characteristics of the input data may vary over time. Here are some essential concepts related to adaptive filters an adaptive filter is a type of digital filter that adjusts its parameters in real-time to optimize its performance based on the input data and a desired output. These filters find applications in various fields, including signal processing, communication systems, audio processing, image processing, and control systems [2]. Unlike traditional fixed filters with predetermined coefficients, adaptive filters are capable of modifying their coefficients iteratively to minimize some defined error or achieve a specific objective [5]. The key idea behind adaptive filters is to continuously adapt to changing input signals and to converge towards a desired response [6]. This adaptability makes them particularly useful in scenarios where the characteristics of the input data may vary over time [7].

II. LITERATURE SURVEY

Adaptive filters have garnered substantial attention in the field of digital signal processing due to their capacity to dynamically adjust their coefficients in response to changing input signals. The concept of adaptivity was introduced by Widrow and Hoff in the late 1950s, leading to the development of adaptive filter algorithms. Widely used in applications such as echo cancellation, noise reduction, and channel equalization, adaptive filters have demonstrated their efficacy in real-world scenarios where signal characteristics evolve over time. The adaptive nature of these filters enables them to converge towards optimal coefficients, enhancing their performance compared to fixed filters.

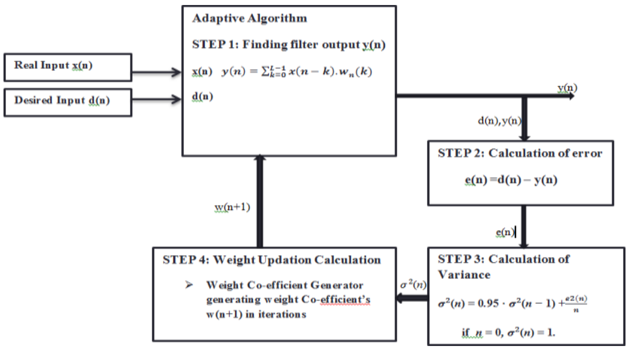
1. Adaptive Filter Theory (U.K Wiley & Chichester 1998): This paper introduced the concept of the adaptive filters, where they are useful and how to analyze the working of the adaptive filters.
2. Experiments with an adaptive filter noise cancellation (G.S Kang & L.J Fransen 2003): In this paper performed various experiments with the Adaptive (V.N. Raju, 2019, [1]) filters to use the application of the noise cancelling.
3. A novel adaptive filter implementation scheme using distributed arithmetic (R. Guo & L. S. DeBrunner): This paper contains the information about the how to implement the low

delay through the distributed arithmetic and Adaptive filter nature in the design using the MATLAB.

4. FPGA Implementation of Adaptive filter and its performance (J. Malarmannan & S.Malarvizhi, 2016): In this paper the designed adaptive filter using the FSM machine fashion and implemented in the MATLAB and compared area, power and delay.

III. METHODOLOGY AND METHODS

Adaptive filters are a class of digital filters that dynamically adjust their coefficients or parameters based on the input signal characteristics and desired output. Unlike conventional, fixed filters, adaptive filters continuously update their weights or coefficients to minimize the difference between the actual output and the desired output. This adaptive behavior allows them to effectively track and respond to changing signal conditions, making them valuable tools in various signal processing applications.

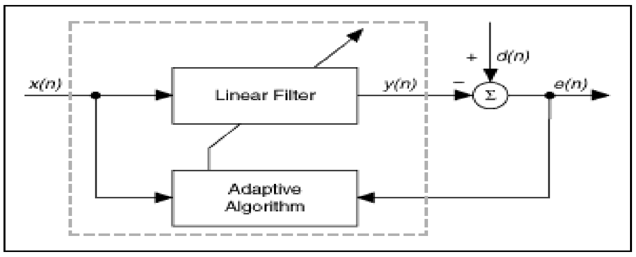

Real Input x(n)
Desired Input d(n)
Adaptive Algorithm
STEP 1: Finding filter output y(n)
x(n) $y(n) = \sum_{k=0}^{L-1} x(n-k).w_n(k)$
d(n)
y(n)
d(n),y(n)
STEP 2: Calculation of error
e(n) =d(n) – y(n)
e(n)
w(n+1)
STEP 4: Weight Updation Calculation
Weight Co-efficient Generator generating weight Co-efficient's w(n+1) in iterations
$\sigma^2(n)$
STEP 3: Calculation of Variance
$\sigma^2(n) = 0.95 \cdot \sigma^2(n-1) + \frac{e2(n)}{n}$
if n = 0, $\sigma^2(n) = 1$.

[Fig.1: Adaptive Filter Block Diagram]

IV. LMS ADAPTIVE FILTER

The Least Mean Squares (LMS) adaptive filter is a widely used algorithm in digital signal processing that enables the real-time adjustment of filter coefficients to minimize the difference between the desired output and the actual output. LMS adaptive filters are particularly effective in applications where signal characteristics change over time, such as communication, audio processing, and control systems [1]. Here's an overview of the LMS adaptive filter:

Algorithm Overview: The LMS adaptive filter operates by iteratively adjusting its filter coefficients based on the error signal between the desired output and the actual output. The goal is to minimize the mean squared error between these signals. The filter adjusts its coefficients by an amount proportional to the negative gradient of the error signal with respect to the coefficients.



[Fig.2: LMS Adaptive Filter Block Diagram]

LMS adaptive filter algorithm has Input Signal is Let x[n] be the input signal at time index n. Desired Output Signal is Let d[n] be the desired output signal at time index n. Filter Coefficients: is Let w[n] represent the filter coefficients at time index n. Estimated Output Signal is the estimated output signal y[n] of the adaptive filter is given by the inner product of the filter coefficients and the input signal:

$$y[n] = wT[n]\cdot x[n]$$

Error Signal is the error signal e[n] is the difference between the desired output and the estimated output:

$$e[n] = d[n]-y[n]$$

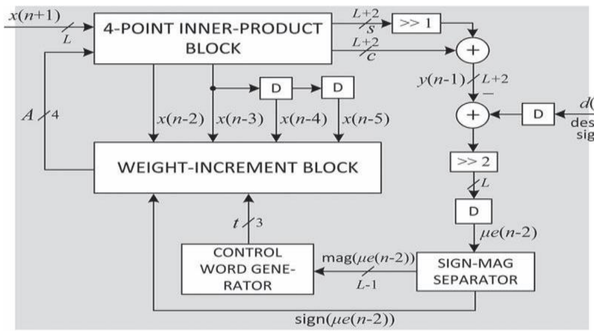
Coefficient Update Rule is the LMS algorithm updates the filter coefficients based on the error signal and the input signal:

$$w[n+1] = w[n]+\mu\cdot e[n]\cdot x[n]$$

Where w[n+1] is the updated filter coefficients at time index n+1. w[n] is the current filter coefficients at time index n. μ (mu) is the adaptation step size, controlling the speed of convergence. It's a positive constant typically chosen through experimentation [2].

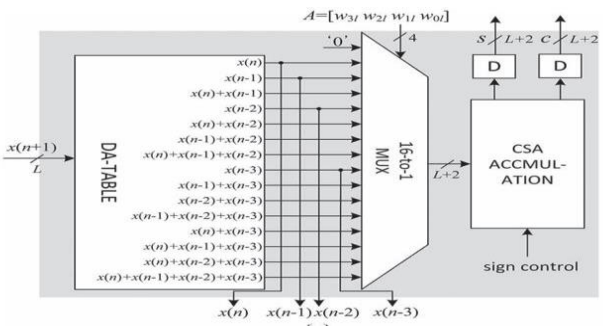
Algorithm Steps are Initialize filter coefficients: w [0]. For each time index n, compute estimated output y[n] and error e[n]. Update filter coefficients using the coefficient update rule. This process is repeated iteratively until the filter converges, and the error signal e[n] approaches zero or reaches a satisfactory level.

The LMS algorithm can have variations, such as the normalized LMS (NLMS) algorithm, which incorporates the input power to adjust the step size. The choice of step size, initialization, and other parameters depends on the specific application and desired performance.



[Fig.3: LMS Adaptive Filter Architecture Diagram]

Four Point Inner Product Block

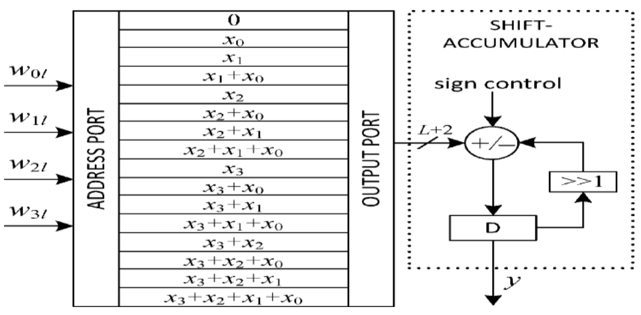


[Fig.4: Four Point Inner Product Block Diagram]

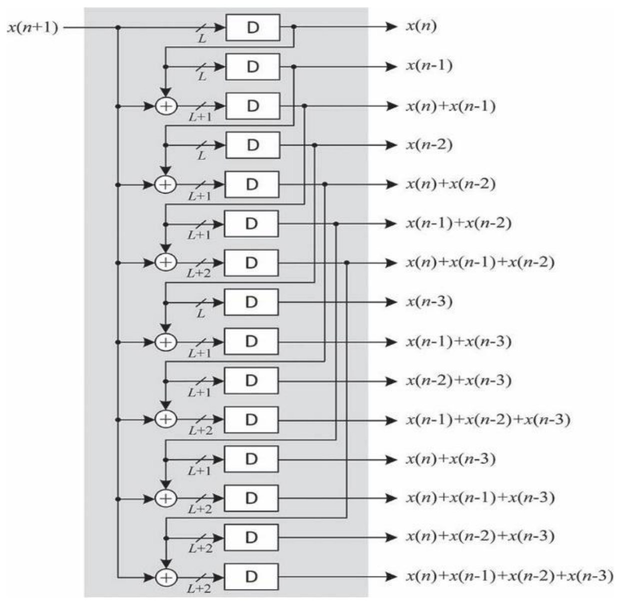
The "four-point inner product block" is a critical component within an adaptive filter architecture that facilitates the concurrent calculation of inner products for four data

points. Let's delve into its structure and operation in more detail: Structure of the Four-Point Inner Product Block: DA Table (Distributed Arithmetic Table) is
a vital component housing a set of 15 array registers. These registers play a pivotal role in storing the partial inner product values, each corresponding to a specific index n. These values are denoted as yn, where 0<n<15. The DA table serves as a repository for these intermediate results, contributing to the overall inner product computation.

Multiplexer (MUX) is A 16:1 multiplexer (MUX) is integrated into the architecture to facilitate the selection of content from one of the 15 array registers in the DA table. This MUX operates based on control signals and directs the chosen partial inner product value to subsequent processing stages. The MUX effectively enables the choice of which intermediate result to process further. Carry Save Adder (CSA): The content selected by the MUX is then fed into a Carry Save Adder (CSA). The CSA serves as a specialized arithmetic unit that predicts both the sum and the carry of the selected content. This prediction process leverages the characteristics of carry-save addition, which offers the advantage of efficient parallelized addition operations.



[Fig.5: Conventional DA-Based Implementation of Four-Point Inner Product]



[Fig.6: DA table for Generation of Possible Sums of Input Samples]

V. CALCULATION OF ERROR

The calculation of error in an adaptive filter using the formula e(n)=y(n)−d(n).

Calculation Explanation: The formula e(n)=y(n)−d(n) calculates the error by subtracting the desired output d(n)) from the estimated or actual output y(n)) at a specific time index n. The result of this subtraction quantifies how well the adaptive filter is currently performing. Here's By subtracting the desired output d(n) from the actual or estimated output y(n)), you obtain the error signal e(n)). This error signal indicates the extent to which the adaptive filter's current output deviates from the desired response. Ideally, the goal is to minimize this error over time by adjusting the filter's coefficients through an adaptive algorithm, such as the Least Mean Squares (LMS) algorithm.

In summary, the error calculation e(n)=y(n)−d(n) is a fundamental step in the operation of an adaptive filter. It provides a quantitative measure of how well the filter is approximating the desired output, guiding the adaptive algorithm's adjustment process to improve the filter's performance [3].

A control word generator using an 8:3 priority encoder is a digital logic circuit that generates control signals based on an input selection signal while prioritizing certain inputs over others. An 8:3 priority encoder takes 8 input lines and produces 3 output lines corresponding to the highest-priority active input line. This type of circuit is commonly used to generate control signals for multiplexers, memory addressing, and other applications where priority selection is crucial [4].

The Weight Increment Block within an adaptive filter algorithm plays a pivotal role in refining the filter's coefficients for optimal performance. This block involves a sequence of steps for the adjustment of filter weights based on the error signal e(n). Initially, the error signal e(n) is multiplied by a parameter known as the step size (μ). This step size controls the pace of convergence and is a critical factor that demands careful optimization. Subsequently, the Weight Increment Block adjusts each filter coefficient by adding the product of the error signal and the corresponding input value. This dynamic adjustment is geared towards minimizing the error signal, thereby enhancing the filter's performance in estimating desired outputs.

VI. TOOLS

A. Verilog

Verilog is a hardware description language (HDL) used for designing and modelling digital circuits and systems. It is widely used in the field of electronic design automation (EDA) for designing integrated circuits (ICs) and other digital hardware components. Verilog allows engineers to describe the behavior and structure of digital systems, which can then be synthesized into actual hardware implementations.

Key concepts and features of Verilog include:

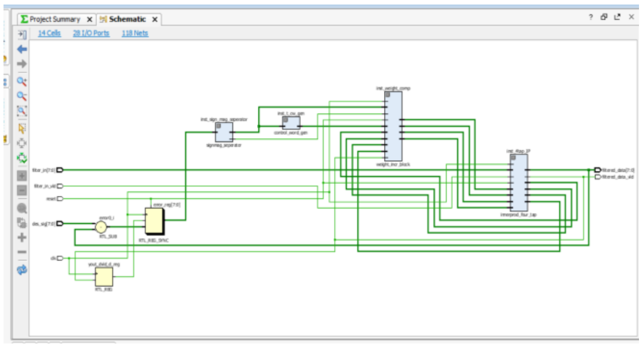
Verilog, a hardware description language, serves as a cornerstone in modern digital circuit design by offering a structured approach to creating complex digital systems. Central to Verilog's architecture are modules, akin to blocks of code or functions, which can be instantiated multiple times to form larger designs. These modules communicate via input and output ports, serving as the conduits for data flow between different components of the design. Signals, represented as variables, carry digital values or buses and are vital for describing the interconnections and operations within the design.

Verilog boasts two primary modes of description: behavioral and structural. Behavioral modelling employs procedural statements, mirroring programming languages, enabling the description one of circuit behaviors through tasks, functions, loops, conditionals, and more. In contrast, structural modelling delves into the design's hardware implementation, involving gates and lower-level components to define the circuit's architecture. Verilog supports a range of data types, encompassing wires for single-bit signals, registers for memory elements, and other data structures.

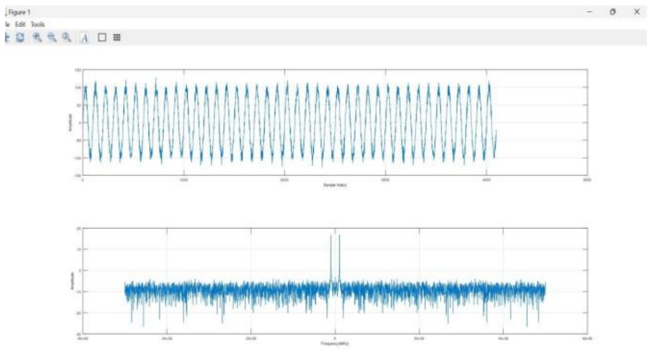
B. MAT LAB

MATLAB (Matrix Laboratory) is a high-level programming language and interactive environment widely used in various fields of engineering, science, and research. Known for its powerful numerical computation capabilities, MATLAB provides an extensive set of functions and tools for data analysis, visualization, and algorithm development. Its user-friendly interface allows users to perform complex computations without delving into low-level programming details. MATLAB's strength lies in its ability to handle matrices and arrays effectively, making it particularly well-suited for mathematical and linear algebra operations. Researchers, engineers, and students leverage MATLAB for a range of applications, including signal processing, image analysis, machine learning, and simulations. Its rich ecosystem of toolboxes and libraries extends its functionality, while its scripting and programming capabilities make it versatile for both prototyping and production-level code development. With its comprehensive documentation and active community, MATLAB remains a powerful choice for numerical computing and scientific research endeavors [3].

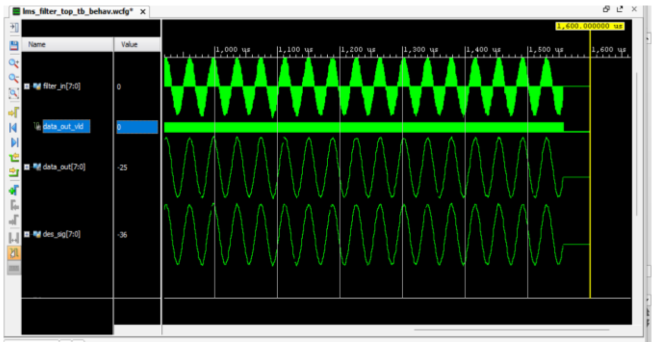
VII. RESULTS & SUMULATIONS



[Fig.7: TOP Level of LMS Adaptive Filter Schematic Diagram]



[Fig.8: Noise Signal]



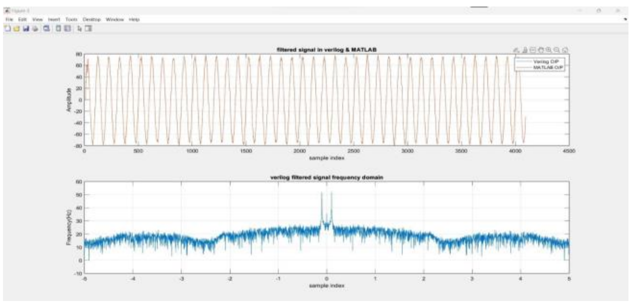
[Fig.9: Simulation of LMS Adaptive Filter]

The provided Verilog module designed for verifying the functionality of an LMS (Least Mean Squares) adaptive filter implementation. The testbench simulates the adaptive filter's behavior, including handling clock signals, initializing input data and weights, reading external files for data, and generating filter input validity signals. It manages the simulation process and monitors the adaptive filter's output, comparing it with desired signals.

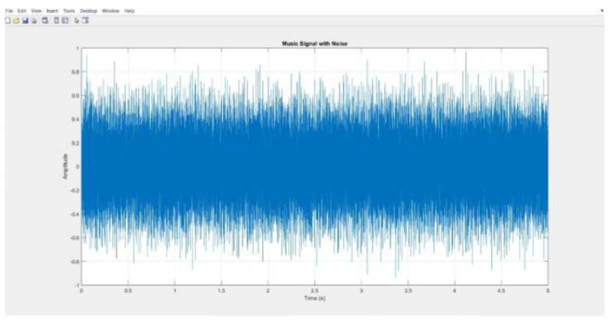
The first paragraph describes the core elements of the testbench. It generates a clock signal clk by toggling between high and low states with a delay of 10-time units. The testbench maintains various registers and signals, such as reset, cntr, filter_in_vld, start, x1, x2, x3, x4, disusing, the weights weight1 through weight4. The testbench uses integer variables like fp0, fp1, and fp2 to manage file operations for reading and writing data.

The second paragraph outlines the initialization process. It opens files for reading and writing, setting up connections to external data sources. The reset signal is initially set to bring the system to a known state. The weights for the adaptive filter are also initialized with specific values. After a delay of 200time units, the reset signal is de-asserted (reset <= 1'b0), and then, after another delay of 100-time units, the start signal is asserted (st). The third paragraph explains the data generation and simulation control aspects. The simulation includes the generation of filter input validity (filter_in_vld) signals and managing the simulation time using counters. Depending on the state of start, cntr, and cntr_inp, the filter input validity and counter values are adjusted. The adaptive filter input data is assigned based on the counter's value and the available input samples. The external files "filter_in_data.txt" and "desired_signal.txt" are used to provide input data and desired signals, respectively.

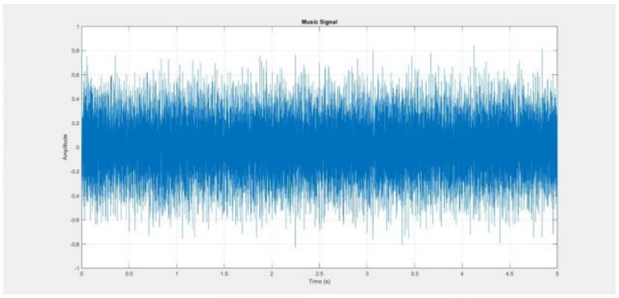
In summary, the testbench effectively simulates the behavior of an LMS adaptive filter, managing clock signals, initializing data and weights, controlling the simulation flow, and verifying the adaptive filter's output against desired signals. It provides a comprehensive framework for verifying the functionality and performance of the adaptive filter implementation.



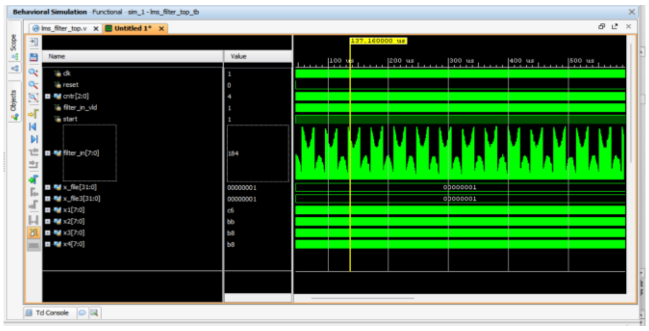
[Fig.10: Filter Signal in Matlab]



[Fig.11: Music Signal with Noise]



[Fig.12: Music Signal]

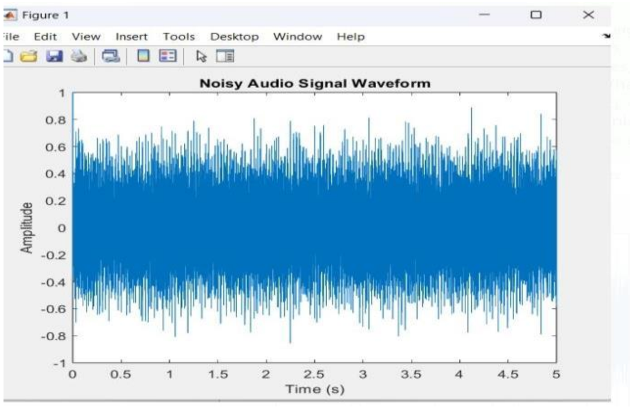


[Fig.13: Simulation of LMS Adaptive Filter with Noise Music Signal]

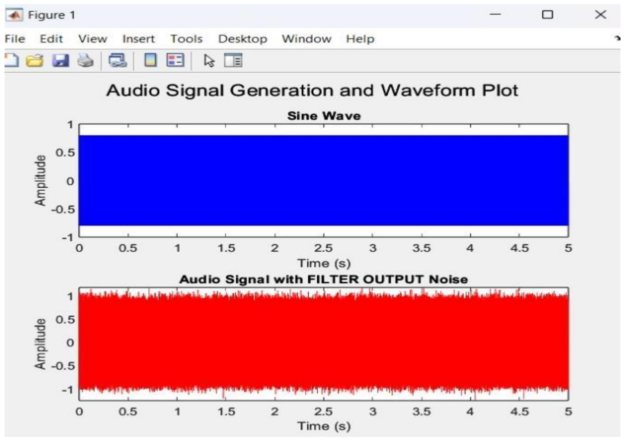
The code generate_music_signal_with_noise() that generates a musical signal with added random noise and then plays the resulting signal while also plotting its waveform. The code operates by defining various parameters and creating a time vector based on the specified sampling frequency and duration. It generates a musical note for each frequency in a C major scale using a combination of harmonic frequencies with different amplitude weights. Each harmonic contributes to the overall note signal, and these note signals are combined to create the complete music signal.

In more detail, the code first establishes the sampling frequency fs and the signal duration duration. It creates a time vector t to represent time instants over the specified duration. The musical note frequencies for a C major scale are provided in meshfree, and the harmonic weights for each harmonic are defined in harmonic weights. The script then initializes an empty music signal music signal.

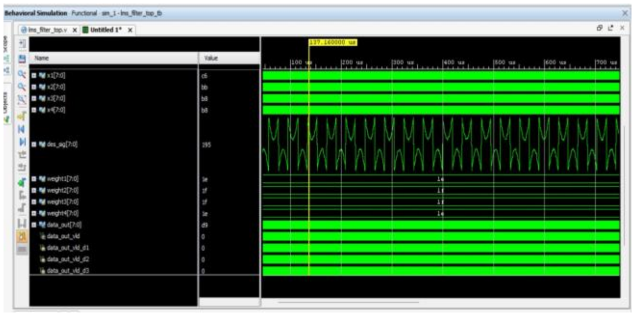
The code iterates through each note frequency, generating harmonic components using sinusoidal functions. Each harmonic frequency is scaled by its respective amplitude weight and added to the note signal. The note signal is normalized to prevent clipping, and it is then accumulated into the overall music signal. After processing all notes, random noise is generated using the random () function and added to the music signal. The resulting signal is again normalized to prevent clipping.



[Fig.14: Audio Signal with Noise]



[Fig.15: Audio with Noise Filtered Output]



[Fig.16: Simulation of LMS Adaptive Filter with Noise Audio Signal]

The code outlines musical note frequencies based on the C major scale via notes frequency, along with harmonic weights for each note as defined in harmonic weights. An empty audio signal music signal is initialized.

For each note frequency, the code generates harmonic components using sinusoidal functions. Each harmonic frequency is scaled by its corresponding harmonic weight and then integrated into the note signal. The note signal is normalized to prevent distortion and added to the overall music signal. After processing all the notes, random noise is generated using the random () function and added to the music signal. Again, the resultant signal is normalized to prevent distortion.

Subsequently, the code employs the sound function to play the music signal with added noise as audio. Additionally, it employs the plot() function to create a waveform plot, visually representing the signal's variations over time. The plotted waveform facilitates an intuitive understanding of the audio signal's characteristics and the impact of the added noise. Users can regulate the noise's amplitude to control its influence on the audio quality. The final result provides an auditory and visual representation of the generated audio, including music andthe the introduced noise. If the audio word needs to be filtered, the 'music signal' could be substituted with the corresponding audio word, and suitable filtering operations could be applied to the audio data.

VIII. CONCLUSION

The future scope of LMS adaptive filter architecture in Verilog structural level lies in its potential to enhance real-time signal processing applications across various domains. By leveraging advanced Verilog structures, such as pipelining, parallelism, and efficient memory utilization, LMS adaptive filters can achieve higher processing speeds and improved resource utilization, enabling their integration into complex systems like cognitive radios, smart sensors, and advanced communication systems. This architecture's adaptability and flexibility will enable the development of more sophisticated algorithms and the efficient implementation of adaptive filtering solutions, furthering the evolution of intelligent and responsive technological systems. By the replacing the filter circuit can be used in the various applications domains like radar signal, ECG signals in medical and audio speech noise elimination. This project may turn into machine learning algorithms.

ACKNOWLEDGEMENT

We would like to express our deep sense of gratitude to our, JNTUH who were very cooperative and them guidance was invaluable and inspiring in doing this research paper. We are very much thankful to who have extended their cooperation to finish this report. Professor Dr. T. Satya Savithri, Ece dept.

DECLARATION STATEMENT

After aggregating input from all authors, I must verify the accuracy of the following information as the article's author.

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Funding Support:** This article has not been sponsored or funded by any organization or agency. The independence of this research is a crucial factor in affirming its impartiality, as it has been conducted without any external sway.
- **Ethical Approval and Consent to Participate:** The data provided in this article is exempt from the requirement for ethical approval or participant consent.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Authors Contributions:** The authorship of this article is contributed equally to all participating individuals.

REFERENCES

1. Behrouz Farhang-Boroujeny, *USA*ADAPTIVE FILTERS THEORY AND APPLICATIONS, ISBN: 978-1-119-97954-8, © 1998, John Wiley & Sons, Ltd.,United Kingdom https://www.wiley.com/en-ie/Adaptive+Filters%3A+Theory+and+Applications%2C+2nd+Edition-p-9781119979548
2. G. Kang and L. Fransen, "Experimentation with an adaptive noise-cancellation filter," in IEEE Transactions on Circuits and Systems, vol. 34, no. 7, pp. 753-758, July 1987, DOI: https://doi.org/10.1109/TCS.1987.1086201
3. R. Guo and L. S. DeBrunner, "A novel adaptive filter implementation scheme using distributed arithmetic," *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, CA, USA, 2011, pp. 160-164, DOI: https://doi.org/10.1109/ACSSC.2011.6189976
4. J. Malarmannan et.al, FPGA Implementation of Adaptive Filter and its Performance Analysis, published in "International Journal of Engineering and Technology (IJET)", Vol 5 No 3 Jun-Jul 2013, ISSN : 0975-4024, page no 3056-3063 https://www.enggjournals.com/ijet/docs/IJET13-05-03-321.pdf
5. Umasankar, Ch. D., & Sairam, Dr. M. S. (2020). Performance Analysis of LMS, NLMS Adaptive Algorithms for Speech Enhancement in Noisy Environment. In International Journal of Innovative Technology and Exploring Engineering (Vol. 9, Issue 4, pp. 2330–2333). DOI: https://doi.org/10.35940/ijitee.d1864.029420
6. T Krishnarjuna Rao, M. Srinivasan, D. Lakshmaiah, Low Energy Less Area Less Delay Fixed Point LMS Method for Adaptive Noise Cancellation Filter. (2020). In International Journal of Recent Technology and Engineering (Vol. 8, Issue 6S, pp. 195–203). DOI: https://doi.org/10.35940/ijrte.f1038.0386s20
7. Pramil, Prateek, & Parashuram. (2020). Band Pass Semi Adaptive Digital Filters for Radar Applications. In International Journal of Engineering and Advanced Technology (Vol. 9, Issue 3, pp. 4184–4186). DOI: https://doi.org/10.35940/ijeat.c5017.029320

AUTHOR PROFILE

Dr. T. Satya Savithri is a distinguished professor in the Department of Electronics and Communication Engineering at JNTU Hyderabad, where she also served as the Head of the Department. She holds a Ph.D. and is a member of several prestigious organizations, including FIEI, FIETE, MIAE, MIEEE, and MISTE. With 140 publications, including 75 international journal papers, and 52 international conference papers, her research interests span digital image processing, VLSI design, antenna design, satellite communications, robotics, and channel models. Dr. Savithri has supervised 14 Ph.D. scholars and filed 2 patents. She is also a member of the board of directors at JNTU Hyderabad.

Sandu Ajay Kumar is an M.TECH. student in Digital Systems and Computers at the Department of Electronics and Communication Engineering, JNTU Hyderabad. Passionate about digital design and VLSI (Very Large-Scale Integration), he has gained industry experience through an internship in VLSI physical design, working with CAD tools. With a strong academic record, he completed his Bachelor's degree in Engineering from Osmania University. Ajay has also undertaken several projects in VLSI and nanotechnology. In addition, he has mentored several individuals, motivating them toward a career in the VLSI industry. He is eager to pursue research and development in the semiconductor sector.