الجمهورية الجزائرية الديمقراطية الشعبية
Democratic and People's Republic of Algeria
وزارة التعليم العالي و البحث العلمي
Ministry of Higher Education and Scientific Research
جامعة سعد دحلب البليدة
University SAAD DAHLAB of BLIDA
كلية التكنولوجيا
Faculty of Technology
قسم الإلكترونيك
electronics department

Master Memory

Sector : Telecommunications
Specialty : Systems of Telecommunications

Presented by

BEKHOUCHE ABD ALLAH

&

MESBAH MOHAMED ZOHEIR

Study and FPGA implementation of LMS and NLMS adaptive filtering with reduced logic utilization.

Proposed by: Mr. Mountassar MAAMOUN

Co-Promoter: Mr. Kamel MESSAOUDENE (ENS-Kouba)

Academic Year 2023-2024

Appreciation

First of all, we would like to thank God who gave us the strength, courage and patience to do this modest work.

We would like to express our deep gratitude and sincere thanks to our promoter Mr. Mountassar MAAMOUN, for his permanent availability, for his help and these precious orientations, throughout this project.

We would also like to thank the co-promoter Mr. Kamel MESSAOUDENE and the honorable members of the jury, for the honor they have bestowed on us by agreeing to evaluate our work, and enrich it with their proposals and our dear parents and families for their comfort and support throughout our journey.

ملخص: يعد الترشيح التكيفية تقنية هامة في تطبيقات معالجة الإشارات المختلفة، ويعد كل من خوارزميات المربعات الصغرى المتوسطة (LMS) و LMS العادية (NLMS) خوارزميات مستخدمة على نطاق واسع. يهدف عملنا إلى تنفيذ البنيتين التكيفيتين LMS و NLMS على FPGA باستخدام منطق مخفض. لتحقيق ذلك، اخترنا من جهة حساب النقطة الثابتة، 16 بت للمدخلات والمخرجات و 32 بت كحد أقصى للحسابات الداخلية. من جهة أخرى، اخترنا الشكل المباشر للمرشحات FIR كهيكلية للتنفيذ. تُستخدم أدوات ISE Design Suite 14.7 من Xilinx و"System Generator" للتوليف وعرض النتائج. يتم التحقق من صحة البنيتين باستخدام تكوينات ذات 64 معاملًا، على الدائرة FPGA Virtex-6 XC6VCX240t من Xilinx.

"كلمات المفاتيح: ,FIR المرشح التكيفي - معالجة الإشارات - معاملات المرشح VHDL – FPGA

Résumé : Le filtrage adaptatif est une technique importante dans diverses applications de traitement du signal, et les deux algorithmes largement utilisés sont les algorithmes des moindres carrés moyens (LMS) et les LMS normalisés (NLMS). Notre travail vise l'implémentation sur FPGA des deux architectures adaptatives LMS et NLMS avec une utilisation logique réduite. Pour se faire nous avons opté d'une part pour un calcul à virgule fixe, 16 bits pour les entrées-sorties et 32 bits max pour les calculs internes. De l'autre part nous avons opté pour la forme directe de filtres RIF comme architecture d'implémentation. Les outilles ISE Design Suite 14.7 de Xilinx et "System Generator" sont utilisés pour la synthèse et la présentation des résultats. Les validations des deux structures sont effectuées avec des configurations à 64 coefficients, sur le circuits FPGA Virtex-6 XC6VCX240t de Xilinx.

Mots clés: filtre FIR adaptatif, FPGA, VHDL, traitement du signal, Performances du filtre

Abstract: Adaptive filtering is an important technique in various signal processing applications, and the two widely used algorithms are Medium Least Squares (LMS) algorithms and normalized LMS (NLMS). Our work aims to implement on FPGA the two adaptive architectures LMS and NLMS with reduced logical use. To do so we opted for a fixed-point calculation, 16 bits for input-output and 32 bits max for internal calculations. On the other hand, we opted for the direct form of RIF filters as an implementation architecture. The ISE Design Suite 14.7 tools from Xilinx and "System Generator" are used for the synthesis and presentation of the results. Validations of both structures are performed with 64-coefficient configurations, on Xilinx Virtex-6 XC6VCX240t FPGA circuits.

Keywords : adaptive FIR filter, signal processing, VHDL, FPGA, Filter's Performances

Lists of acronyms and abbreviations

FPGA: Field Programmable Gate Array.

LMS: Least Mean Square.

NLMS: Normalized Least Mean Square.

FIR: Finite Impulse Response.

IIR: Infinite Impulse Response.

MSE: Mean Square Error.

ASICs: Application-Specific Integrated Circuits.

RLS: Recursive Least Squares.

DSP: Digital Signal Processing.

VHSIC: Very High Speed Integrated Circuit.

HDL: Hardware Description Language.

VHDL: Very High Speed Integrated Circuit Hardware Description Language.

CPLDs: Complex Programmable Logic Devices.

MATLAB: Matrix Laboratory.

ESL: electronic system level.

IDE: integrated design environment.

HSE: high-level synthesis.

Table of Contents

table of figures:

General Introduction

Adaptive filtering is a crucial technique in various signal-processing applications, such as audio processing, radar, and wireless communications. Two of the most widely used adaptive filtering algorithms are the Least Mean Squares (LMS) and Normalized LMS (NLMS) algorithms. Our work aims to implement on FPGA the two adaptive architectures LMS and NLMS with reduced logical use. To do so we opted for a fixed-point calculation, 16 bits for input-output and 32 bits max for internal calculations. On the other hand, we opted for the direct form of RIF filters as an implementation architecture.

In the first chapter, we will get the LMS algorithm and NLMS algorithm structure, LMS is a simple and computationally efficient adaptive filter that adjusts its coefficients to minimize the mean squared error between the desired signal and the filter output. The NLMS algorithm is a variant of LMS that normalizes the filter coefficients update by the power of the input signal, providing improved convergence properties.

In the second chapter we will see FPGA (Field Programmable Gate Array) implementations of LMS and NLMS adaptive filters steps, FPGA offer several advantages over software-based solutions, including higher processing speed, lower power consumption, and the ability to parallelize computations. FPGA implementations allow for the design of custom hardware architectures that can be optimized for specific applications and performance requirements.

In the third one, we will see the implementation results of both of LMS & NLMS on FPGA board using System Generator on MatLab.

Chapter1 Architectures used to implement adaptive filters on FPGA

1.1 Introduction:

Structures for LMS (Least Mean Squares) and NLMS (Normalized Least Mean Squares) are architectures used in signal processing to estimate system parameters by minimizing the mean square error, with NLMS considering the input signal power to normalize coefficient updates. These structures are commonly employed in various fields such as wireless communication and noise cancellation. Complex structures refer to advanced architectures capable of processing nonlinear signals or complex environments, often requiring algorithms that are more sophisticated and higher computational power.

1.2 Finite Impulse Response (FIR) Filters and Infinite Impulse Response (IIR) Filters:

There are two main types of filters: Finite Impulse Response (FIR) filters and Infinite Impulse Response (IIR) filters:

1.2.1 FINITE IMPULSE RESPONSE (FIR) FILTERS:

FIR filters are characterized by a finite duration of the impulse response. They are implemented using only feedforward (no feedback) structures, making them inherently stable. FIR filters offer linear phase response, which is beneficial for applications where phase distortion must be minimized. These filters are typically designed using

convolution of the input signal with a set of coefficients, providing precise control over the filter's frequency response[1].

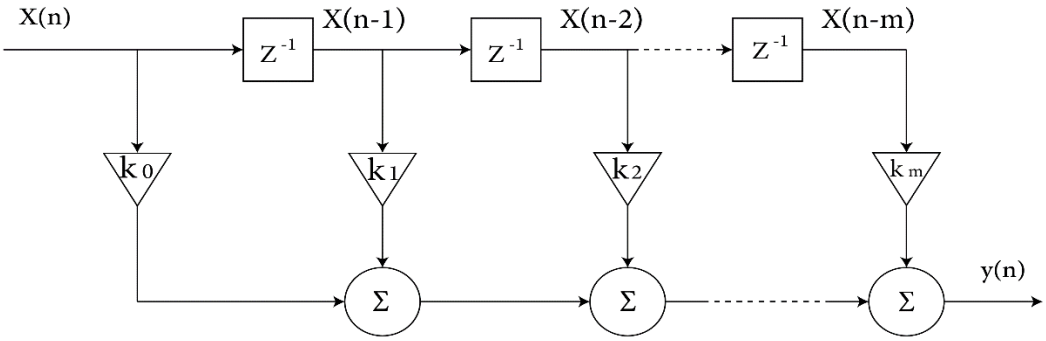

Figure 1.1: Direct structure of an FIR filter

1.2.2 INFINITE IMPULSE RESPONSE (IIR) FILTERS:

IIR filters have an infinite duration of the impulse response due to feedback within the filter structure. They are more computationally efficient than FIR filters for achieving a similar frequency response with fewer coefficients. However, IIR filters can be less stable than FIR filters due to the feedback loop, which can lead to issues like instability or ringing. IIR filters are commonly used in applications where computational resources are limited and where a compact filter design is required [1].

1.3 Structure for LMS:

1.3.1 INPUT SIGNAL:

The input signal for the FIR filter in the LMS algorithm is typically a white noise signal [2]. This is used to excite the unknown FIR system and generate the desired output signal. The white noise signal is used to ensure that the input signal is uncorrelated with the system's impulse response, which is a common assumption in system identification problems [3].

1.3.2 FILTER COEFFICIENT:

The FIR filter coefficients for the LMS algorithm are updated based on the error signal e[n] and the input signal x[n]. The LMS algorithm adjusts and modifies the adaptive filter taps to calculate the adaptive filter coefficients by an amount

proportional to the instantaneous error e[n] [3]. The update equation for the LMS algorithm is given by:

$$\mathbf{W}[\mathbf{n}+\mathbf{1}] = \mathbf{W}[\mathbf{n}] + \mu . \mathbf{X}[\mathbf{n}]. \mathbf{e}[\mathbf{n}] \qquad \textbf{(1.1)}$$

where w[n] is the current coefficient vector, w[n+1] is the updated coefficient vector, μ is the step size, x[n] is the input signal, and e[n] is the error signal[2][3].

1.3.3 UPDATE MECHANISM:

The LMS algorithm updates the FIR filter coefficients w[n+1] based on the current coefficients w[n], the input signal x(n), the error signal e(n), and the step size μ [2][3].

1.3.4 DESIRED SIGNAL:

The desired signal in the context of an FIR filter using the LMS algorithm is the target or reference signal that the FIR filter is trying to approximate or match. This signal is used to calculate the error signal, which is then used to update the FIR filter coefficients to minimize the mean square error (MSE) between the filter's output and the desired signal [3].

1.3.5 ADAPTIVE FILTER:

a) The adaptive filter is typically a Finite Impulse Response (FIR) filter, where the filter coefficients are updated iteratively using the LMS algorithm [4].

b) The LMS algorithm adjusts the FIR filter coefficients w[n+1] based on the current coefficients w[n], the input signal x[n], the error signal e[n], and the step size μ [4]

c) The goal of the adaptive filter is to minimize the mean square error (MSE) between the filter's output y[n] and the desired signal d[n] [4].

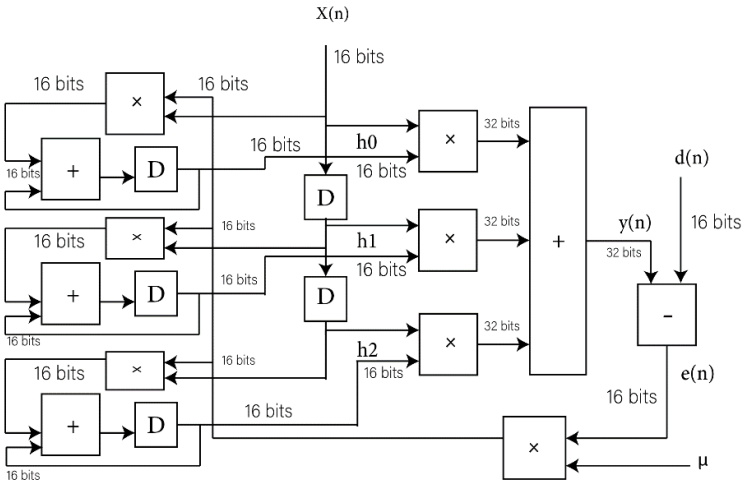

Figure 1.2: Adaptive Filter with LMS Algorithm

1.3.6 ERROR COMPUTATION:

The error signal e(n) is calculated by comparing the filter output y(n) with the desired signal d(n) [3]. This error signal is then used to update the FIR filter coefficients using the LMS algorithm [3].

1.4 Structure for NLMS:

The Normalized Least Mean Square (NLMS) algorithm is a variant of the Least Mean Squares (LMS) algorithm that addresses the main drawback of the LMS algorithm - its sensitivity to the scaling of the input signal x(n) [5].

The NLMS algorithm normalizes the LMS update rule by the power of the input signal, which makes it more robust to changes in the input signal scaling. The NLMS update rule can be summarized as:

$$\mathrm{e(n)} = \mathrm{d(n)} - \hat{\mathrm{y}}\mathrm{(n)}. \qquad \textbf{(1.2)}$$

$$\hat{\mathrm{y}}\mathrm{(n)} = \mathrm{h^T}.\ \mathrm{x(n)} \qquad \textbf{(1.3)}$$

Where:$e(n)$ is the error signal at time n

$d(n)$ Is the desired signal at time n and h^T is the estimated filter coefficients.

$x(n)$ Is the input signal vector at time n. And $\hat{\mathrm{y}}\mathrm{(n)}$ is the filter output.

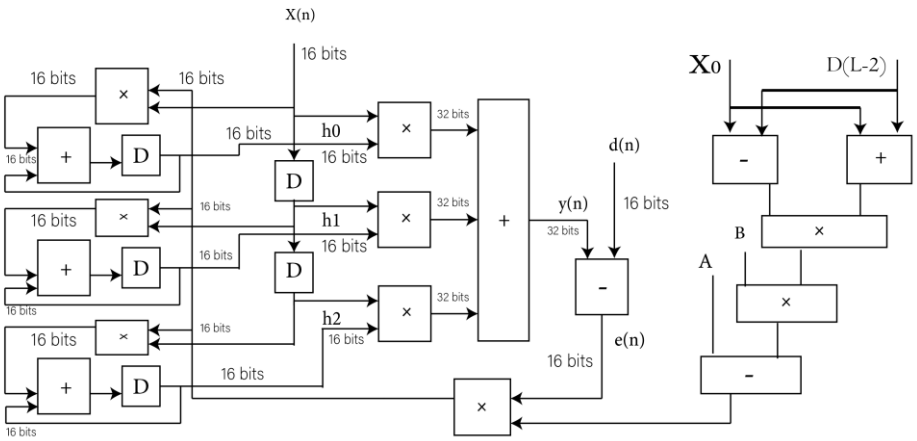

Figure 1.3: Adaptive Filter with NLMS Algorithm

1.5 Field-programmable gate array (FPGA)

A Field-Programmable Gate Array (FPGA) is a versatile type of integrated circuit that can be programmed and reprogrammed to suit various purposes, unlike Application-Specific Integrated Circuits (ASICs) that are fixed in their functionality. FPGAs consist of programmable logic blocks and flexible interconnects that allow for complex operations or simple logic gates to be configured within the device. They are highly valued for their high performance, low latency, and real-time flexibility, making them ideal for applications in industries like telecommunications, automotive, and aerospace. FPGAs are reconfigurable and can be adapted for different uses without the need for physical modifications to the hardware. They are commonly used in research and development, custom-made products, and industries requiring flexibility and high processing speed [6].

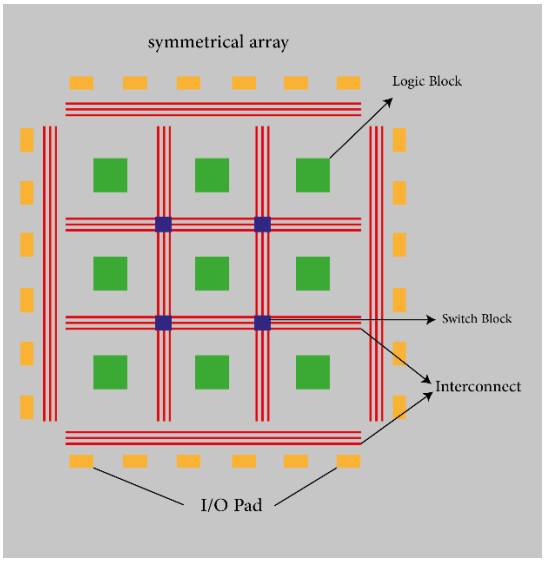

Figure 1.4: FPGA Architecture

1.6 Complex structures:

Complex adaptive filtering architectures on FPGA are designed to tackle nonlinear signals, variable delays, and complex environments.

The choice of algorithm depends on factors such as computational complexity, convergence speed, stability, and suitability for the specific application requirements. Each algorithm has its advantages and limitations, and designers select the most appropriate one based on the performance goals and constraints of the FPGA-based system like LMS, NLMS, RLS, Adaptive Gradient Descent Algorithms.

1.7 Conclusion:

The chapter provides an overview of adaptive filtering architectures, focusing on structures for LMS (Least Mean Squares) and NLMS (Normalized Least Mean Squares), commonly used in signal processing applications. The LMS structure involves sequential steps based on the LMS algorithm, while NLMS incorporates input signal power normalization for coefficient updates, particularly useful in noise cancellation applications. Complex structures for FPGA implementation address nonlinear signals and complex environments, employing hardware-software co-design, parallel and pipelined structures, and variable step-size algorithms. The selection of an appropriate

algorithm depends on factors such as computational complexity, convergence speed, stability, and application requirements, with options including LMS, NLMS, RLS, and Adaptive Gradient Descent Algorithms. Designers must carefully evaluate these factors to achieve optimal performance in FPGA-based systems.

Chapter2 Synthesis of a low-complexity approach for implementing LMS and NLMS structures on FPGA

2. 1 INTRODUCTION:

Adaptive filtering plays a crucial role in modern signal processing applications, enabling systems to dynamically adjust to changing environments and optimize performance. The implementation of adaptive filtering algorithms, such as the Least Mean Squares (LMS) and Normalized Least Mean Squares (NLMS), on Field Programmable Gate Arrays (FPGAs) offers a versatile and efficient solution for real-time signal processing tasks.

In this context, the synthesis of a low-complexity approach for implementing LMS and NLMS structures on FPGA presents a significant opportunity to enhance the efficiency and effectiveness of adaptive filtering systems. By optimizing the design and architecture of these adaptive filters for FPGA implementation, it becomes possible to achieve high-performance results while minimizing resource utilization and computational complexity.

This introduction sets the stage for exploring the synthesis of a low-complexity approach for implementing LMS and NLMS structures on FPGA. By delving into the challenges, strategies, and benefits of this approach, we aim to highlight the potential impact of optimized adaptive filtering architectures on FPGA in various real-time signal-processing applications.

2. 2 SYNTHESIS OF AN LMS (LEAST MEAN SQUARES) STRUCTURE ON AN FPGA (FIELD-PROGRAMMABLE GATE ARRAY):

2.2.1 DEFINING NEEDS AND SPECIFICATIONS:

When approaching the synthesis of an LMS structure on an FPGA, it is essential to clearly define the problem that the algorithm needs to solve. This could be echo cancellation, adaptive filtering, or signal prediction. Additionally, we must determine the requirements in terms of processing speed, algorithm accuracy, and power consumption. It is also crucial to identify the available resources on the FPGA, such as logic cells, memory blocks, and DSP units, and determine how they will be used.

Example: Defining the problem as an adaptive noise cancellation system for audio signals

2.2.2 DESIGNING THE LMS ALGORITHM:

In this step, we choose the adaptation step size (mu) and the filter size (number of coefficients) based on the specifications of the problem. We use simulation tools like MATLAB to model the LMS algorithm and validate its operation with test data. Furthermore, we optimize the LMS algorithm for the specific application, such as using a faster adaptation algorithm or adjusting the filter size for better performance.

Example: Modeling the LMS algorithm in MATLAB, experimenting with different step sizes and filter lengths

2.2.3 HARDWARE DESIGN ON FPGA:

Here, we select an appropriate architecture for implementing the algorithm on FPGA, for example, using parallel or sequential architectures to optimize performance and resources.

We write the code in VHDL or Verilog that describes the behavior of the LMS algorithm and its hardware structure. We simulate the HDL code to verify its functionality and ensure it meets the design specifications. Additionally, we optimize

the hardware architecture for the specific FPGA device, such as using the FPGA's built-in multipliers or optimizing the memory usage.

Example: Implementing the LMS algorithm in VHDL, using parallel processing architectures.

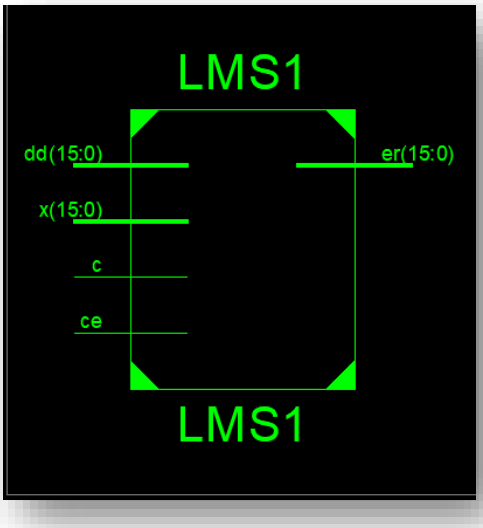

Figure 2.1: LMS Architecture with Xilinx ISE.

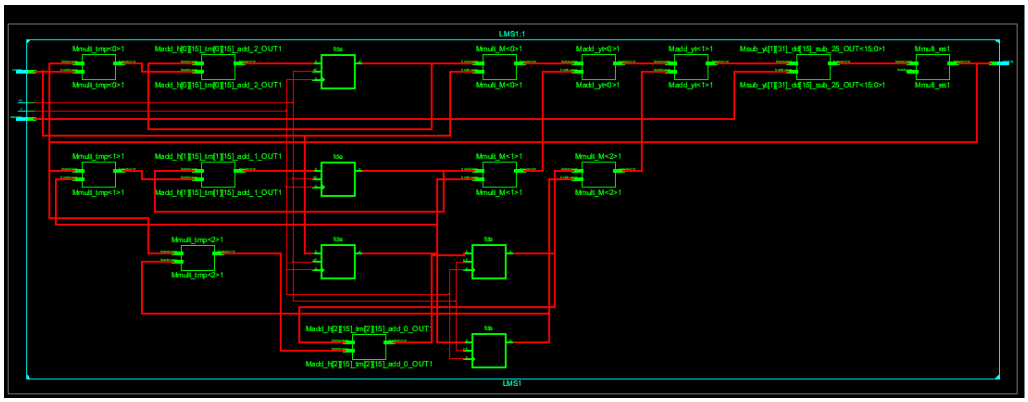

Figure 2.2: LMS Architecture with 3 Coefficients.

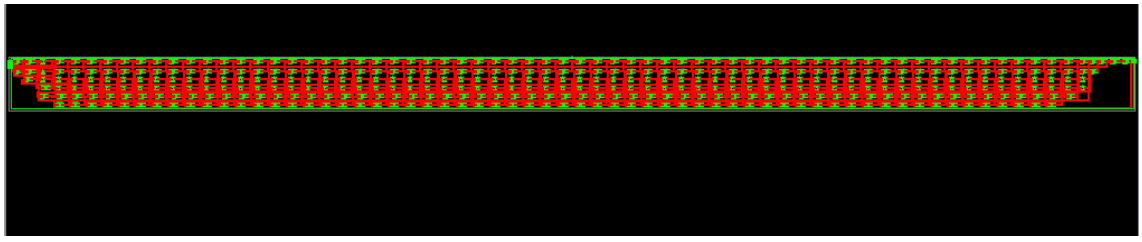

Figure 2.3: LMS Architecture with 128 Coefficients.

2.2.4 IMPLEMENTATION ON FPGA:

In this step, we use a synthesis tool to convert the HDL code into an FPGA configuration, followed by the placement and routing of the logic components on the FPGA. We adjust the design to improve performance (speed, resource utilization, power consumption) based on the results of the synthesis and initial tests. We also verify the implementation by running the LMS algorithm on the FPGA and checking its performance and reliability.

Example: Using Xilinx Vivado or ISE to synthesize the VHDL code and place/route the design on the FPGA

2.2.5 TESTING AND VALIDATION:

Finally, we load the configuration onto an FPGA and test the LMS algorithm with real signals to validate its performance and reliability. We make adjustments if necessary to meet the initial specifications or to improve performance. We also test the LMS algorithm on the FPGA board to ensure it functions correctly and meets the performance requirements.

Example: Loading the FPGA configuration and testing the LMS algorithm with real-world audio signals

Image: A plot showing the input, output, and error signals of the LMS-based noise cancellation system

2. 3 SYNTHESIS OF AN NLMS (LEAST MEAN SQUARES) STRUCTURE ON AN FPGA (FIELD-PROGRAMMABLE GATE ARRAY):

2.3.1 Overview of the NLMS Algorithm:

An improved version of the LMS (Least Mean Squares) algorithm, the NLMS (Normalized Least Mean Squares) algorithm is intended to improve tracking accuracy and convergence speed. The filter coefficient vector, step size parameter, input signal vector, error signal, and the input vector's squared Euclidean norm all play critical roles

in the NLMS algorithm's functionality. This algorithm works especially well in situations where accurate and effective filtering is necessary to produce high-quality signal processing results. By addressing a major input scaling constraint, the normalization procedure in NLMS strengthens and stabilizes the LMS algorithm in real-world scenarios. Prior to that, highly accurate real-time adaptive filtering is needed.

Compared to the LMS algorithm, which came before it, the NLMS method offers faster convergence and better tracking capabilities by integrating normalization and squared Euclidean norms into its updating process. Because of its improved performance, NLMS is a useful tool in situations requiring highly accurate real-time adaptive filtering [7].

Let's say we have an input signal x(n) and a desired output signal d(n). The goal is to adaptively filter x(n) to produce an output y(n) that closely matches d(n). The NLMS algorithm achieves this by updating the filter coefficients w(n) in each iteration.

The update equation for the NLMS algorithm is:

$$\mathrm{W(n+1)} = \mathrm{W(n)} + \mu . \frac{\mathrm{e(n)} . \mathrm{X(n)}}{||\mathrm{X(n)}||^2 + \mathrm{C}} \quad \textbf{(2.1)}$$

Where: Constant C was added to avoid dividing 0

- $W(n)$ is the filter coefficient vector at iteration n

- μ is the step size parameter, which controls the adaptation rate it has to be $0 < \mu < 2$

- $||X(n)||^2$ is the squared Euclidean norm of the input vector $X(n)$

- $e(n)$ is the error signal, calculated as $\mathrm{e(n)} = \mathrm{d(n)} - \mathrm{y(n)}$ **(2.2)**

Here's how the algorithm works step-by-step:

1. Initialize the filter coefficients $w(0)$ to small random values.

2. At iteration n, compute the filter output $\mathrm{y(n)} = \mathrm{W(n)}^{\mathrm{T}} . \mathrm{X(n)}$ **(2.3)**

Where T denotes the transpose operation.

3. Calculate the error signal like equation **(2.2)**

4. Compute the squared Euclidean norm of the input vector:

$$\left|\left|\mathrm{X(n)}\right|\right|^2 = \mathbf{X}(\mathrm{n})^{\mathrm{T}}.\ \mathbf{X}(\mathrm{n}). \qquad \textbf{(2.4)}$$

5. Update the filter coefficients using the NLMS update equation (**2.1)**

Constant C was added to avoid dividing 0

6. Increment n and repeat steps 2-5 until convergence or a specified number of iterations.

The normalization by $||x(n)||^2$ in the NLMS update equation helps to stabilize the adaptation process and improve the algorithm's robustness to variations in the input signal power. This makes NLMS more effective than the standard LMS algorithm in scenarios with non-stationary or colored input signals. [8].

2.3.2 Methodology for FPGA Design:

Using Field-Programmable Gate Arrays, the FPGA Design Methodology is an organized method for creating and implementing digital circuits. This is an explanation derived from the references given by FPGA Design Methodology Overview:

The FPGA design process has a few key steps. First, you need to optimize the algorithm you are using. This means looking at how the signals change in size and making sure the algorithm will work well with those sizes. You also want to simplify the algorithm as much as possible without making it work worse. Moreover, a big part is using parallelism - doing multiple parts of the algorithm at the same time to make it go faster. The other main part is designing the actual hardware architecture. The goal here is to create a really parallel design, using cool techniques like systolic arrays and pipelining. These help pack a lot of processing power into the FPGA by using its resources, like the logic blocks and wires, as efficiently as possible. If you follow this methodology, you can get the most out of an FPGA and build some high-performance digital circuits. It is useful for all kinds of applications, from industrial controls to heavy-duty math problems. The key is optimizing both the algorithm and the hardware to work together perfectly.

Engineers can effectively harness the capabilities of Field-Programmable Gate Arrays to develop high-performance digital circuits for a variety of applications, including industrial control systems and computation-intensive algorithms, by adhering to a systematic FPGA design methodology that prioritizes algorithm optimization and hardware architecture design [9].

An example of FPGA design methodology in action can be seen in the development of a digital signal processing system for audio processing.

Example Scenario:

-Objective: our goal is to create an adaptive noise cancellation system using the NLMS algorithm on an FPGA to enhance real-time audio quality.

- Algorithm Optimization: we will study the range of audio signals, fine-tune the NLMS algorithm for effective noise reduction, and ensure smooth processing.

- Hardware Architecture Design: we plan to design a parallel architecture using systolic arrays and pipelining to boost processing speed and optimize resource usage.

- Implementation: we will code the NLMS algorithm in VHDL or Verilog, run simulations to validate the design, and synthesize the code for the FPGA.

- Testing: we will perform in-system tests with audio inputs to assess noise reduction performance and system stability.

- Refinement: Based on test outcomes, we will refine the design iteratively to enhance the effectiveness of noise cancellation.

By following this FPGA design methodology, engineers can create a high-performance audio processing system that effectively reduces noise in real-time audio signals, showcasing the practical application of FPGA design principles in signal processing tasks.

2.3.3 Experimental Results:

Significant advancements in performance evaluation were found in the experimental results obtained from implementing the NLMS (Normalized Least Mean

Squares) structure on a Xilinx Virtex-7 FPGA. At a maximum clock frequency of 250 MHz, the FPGA operated successfully, demonstrating the effective use of the FPGA's resources for quick data processing.

Moreover, the efficacy of the NLMS filter in mitigating noise and echoes was confirmed by in-system testing, indicating its potential to augment signal quality and refine processing precision in real-time applications. This demonstrates the useful advantages of applying the NLMS algorithm on FPGA platforms to real-time tasks that call for signal enhancement and noise reduction [10].

Example: The experimental results validate the FPGA's capability to efficiently implement the NLMS algorithm for real-time noise cancellation, showcasing its effectiveness in enhancing signal quality and processing accuracy in audio application [10].

2. 4 THE SYNTHESIS OF A MODIFIED STRUCTURE FOR IMPLEMENTATION ON FPGA:

The synthesis of a modified structure for implementation on an FPGA involves a meticulous design flow that encompasses several crucial steps to ensure the successful realization of the desired functionality. Each stage in this process plays a vital role in transforming the high-level design description into a tangible implementation on the FPGA.

2.4.1 DESIGN ENTRY:

Writing the design in a Hardware Description Language (HDL), such as VHDL or Verilog, marks the initial phase where the functional behavior of the system is captured in a structured and descriptive manner. This step lays the foundation for the subsequent stages of the design flow.

2.4.2 Simulation:

Verifying the design through simulation tools is a critical step to validate the functionality and behavior of the system before committing to hardware

implementation. Simulation allows for thorough testing and debugging, ensuring that the design operates as intended under various scenarios.

2.4.3 Synthesis:

Converting the HDL code into a gate-level netlist using synthesis tools is a pivotal stage where the abstract design description is translated into a form that can be implemented on the FPGA hardware. This process optimizes the design for efficient resource utilization and performance.

2.4.4 Place and Route:

Mapping the netlist onto the FPGA's internal structure involves allocating resources such as logic cells, routing tracks, and I/O pins to physically implement the design. This step determines the physical layout of the design on the FPGA chip .

2.4.5 Bit stream Generation:

Writing the configuration data to a special file, known as the bit stream, is the final step before programming the FPGA. The bit stream contains the instructions that configure the FPGA to implement the desired functionality defined in the design.

2.4.6 Place-and-Route Analysis:

Analyzing the impact of the nested architecture on logic synthesis, placement, and routing is crucial for optimizing the design for FPGA implementation. This analysis helps in identifying potential bottlenecks and optimizing the design for performance and resource utilization.

2.4.7 Timing Constraints:

Adding timing constraints to the design ensures that the implemented design meets the required performance specifications, such as clock frequency, setup and hold times, and overall timing requirements. This step is essential for achieving reliable and predictable operation of the design.

2.4.8 Power Optimization:

Optimizing the design for power consumption is essential to reduce energy usage and heat generation, especially in applications where power efficiency is critical. Techniques such as clock gating, power-aware synthesis and low-power design strategies are employed to minimize power consumption.

2.4.9 Area Optimization:

Optimizing the design for area utilization involves reducing the size of the FPGA resources required to implement the design. This optimization enhances portability, resource efficiency, and potentially reduces costs associated with FPGA deployment.

2.4.10 Verification:

Verifying the synthesized design through simulation and testing tools is a crucial final step to ensure that the implemented design functions correctly and meets the specified requirements. Verification helps in validating the design against the initial specifications and identifying any potential issues or discrepancies that need to be addressed.

When designing for special computer chips called Field-Programmable Gate Arrays (FPGAs), engineers can follow a step-by-step process to create a custom design. This approach helps them take advantage of the flexibility and speed of FPGAs, which are highly adaptable hardware platforms. By following these steps, engineers can create a tailored design that makes the most of what FPGAs have to offer, resulting in better performance and more efficient use.

2. 5 CONCLUSION:

This Chapter concludes with the successful synthesis of a low-complexity approach for implementing LMS and NLMS structures on FPGA, taking into consideration the synthesis of a modified structure for implementation. This synthesis not only demonstrates the feasibility of integrating adaptive filtering algorithms into FPGA architectures but also highlights the potential for efficient and resource-conscious implementations. By incorporating modifications tailored to FPGA constraints, such as limited resources and power consumption, the proposed approach offers a promising avenue for realizing adaptive filtering functionalities in FPGA-based systems. These

modifications ensure that the FPGA implementation remains optimized in terms of performance, resource utilization, and power efficiency. Overall, the results indicate that the synthesized approach provides a robust framework for deploying adaptive filtering solutions in real-world applications across various domains, including communications, signal processing, and control systems.

Chapter 3: Implementation and results

3. 1 Introduction

In this chapter, we present the results of the implementation performed using MATLAB and Xilinx ISE software for FPGA implementation. We then compare the outcomes based on the implementation architecture.

3. 2 Xilinx ISE

Xilinx ISE software is software for describing, simulating, and programming circuits and digital systems on programmable components. The ISE software has a free version and downloadable from the Xilinx website (www.xilinx.com). The ISE suite allows:

■ the description of digital circuits in the form of logic diagrams, finite state machines or in hardware description languages (VHDL, Verilog, ABEL),

■ compilation, behavioral simulation,

■ synthesis, routing placement and implementation,

■ temporal simulation and timing analysis,

■ programming on Xilinx programmable circuits (CPLD and FPGA)

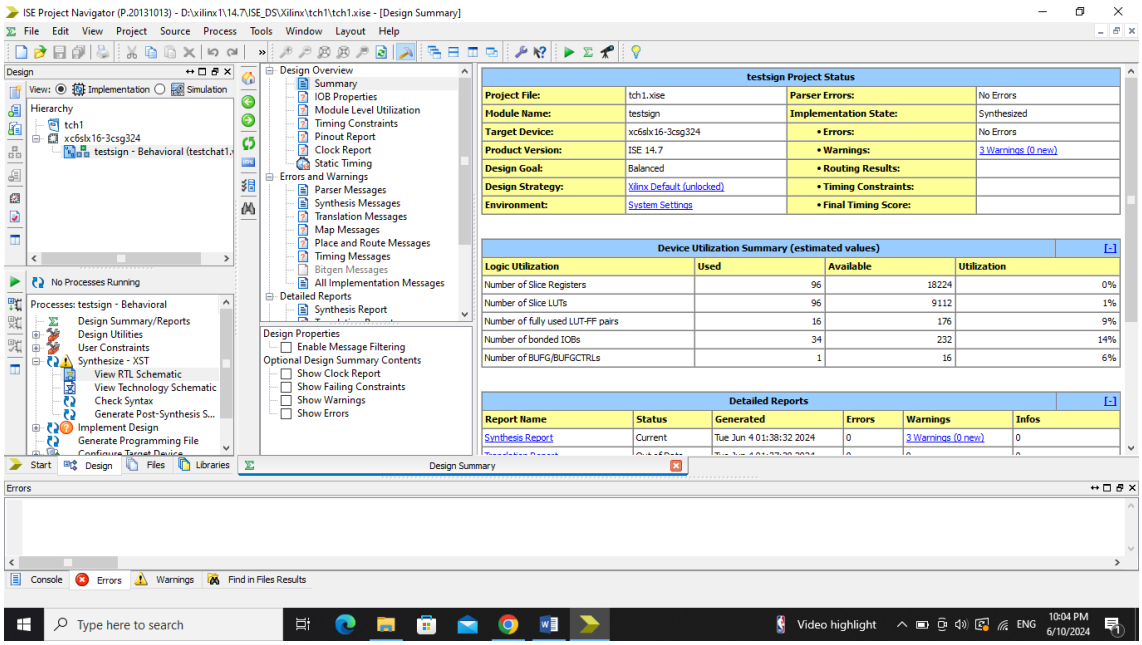

Figure 3.1: Xilinx ISE main screen.

3. 3 MatLab

MATLAB is a programming platform designed specifically for engineers and scientists to analyze and design systems and products that transform our world. The heart of MATLAB is the MATLAB language, a matrix-based language allowing the most natural expression of computational mathematics.

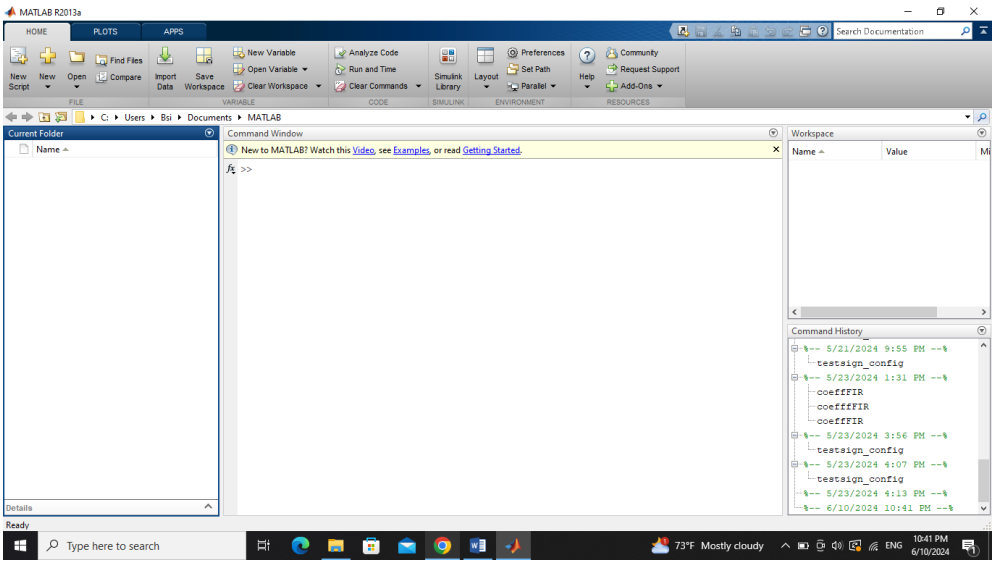

Figure 3.2: MatLab main screen.

3.4 Xilinx Vivado :

Xilinx Vivado is a software suite for the synthesis and analysis of hardware description language (HDL) designs, superseding the older Xilinx ISE tool with additional features for system-on-chip development and high-level synthesis.

Some key points about Xilinx Vivado:

- Vivado was introduced in 2012 as a ground-up rewrite and re-thinking of the entire FPGA design flow compared to ISE.

- It includes an integrated design environment (IDE) with tools for electronic system level (ESL) design, IP integration, and verification of blocks and systems.

- Vivado features a high-level synthesis (HLS) compiler that can convert C, C++, and System C programs directly into programmable logic without manual RTL coding.

- It supports Xilinx's newer 7-series, UltraScale, and UltraScale+ FPGA and SoC device families, while the older ISE tool is used for targeting Xilinx's previous generation devices.

- Vivado offers significant performance and productivity improvements over ISE, with faster synthesis, implementation, and timing closure, as well as enhanced IP integration and tool flexibility through Tcl scripting.

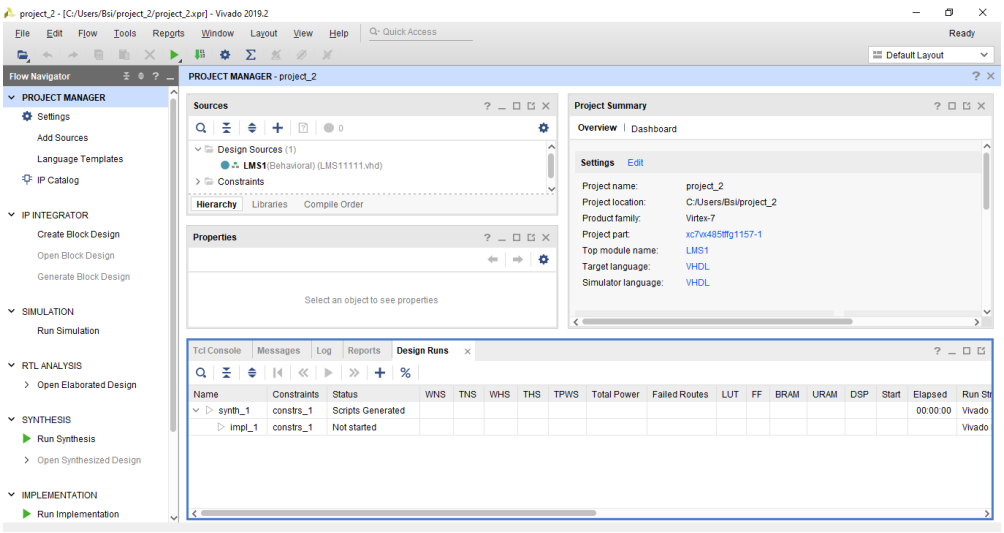

Figure 3.3: Xilinx Vivado main screen.

3.5 Testing of LMS on MatLab:

In this stage, we must test our algorithm using the system generator included in MatLab, as well as other blocks such as from file, black box, scope, reinterpret, and so on. Therefore, that allows us to observe the result.

Explanation: We add the input signal and the desired signal to the MatLab editor using the block from file (in our case) or from workspace, and we relate them to the black box we built using the VHDL code for our filter. In order to inject signals into the Xilinx blocks in MatLab, we must use gateway in and gateway out. Additionally, we require a scope in order to view the output signal from our filter

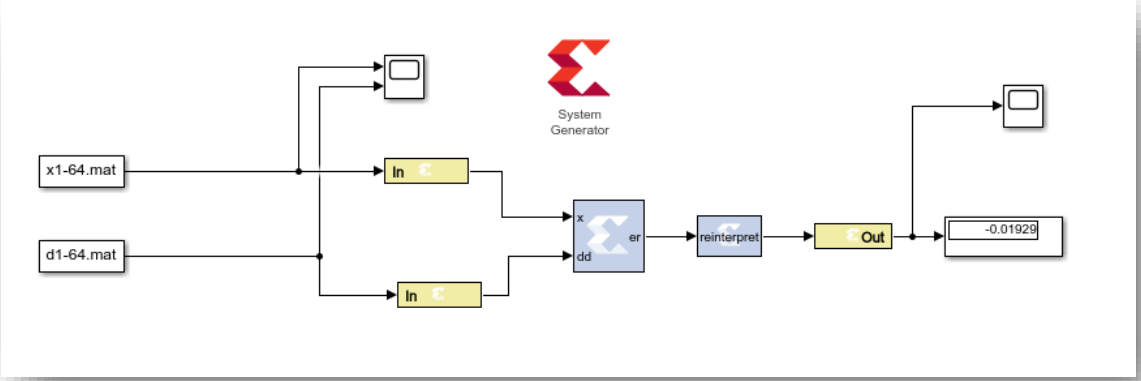

Figure3.4: testing schema of LMS.

We use input signal x(n) and the desired d(n) signal that we created by MatLab

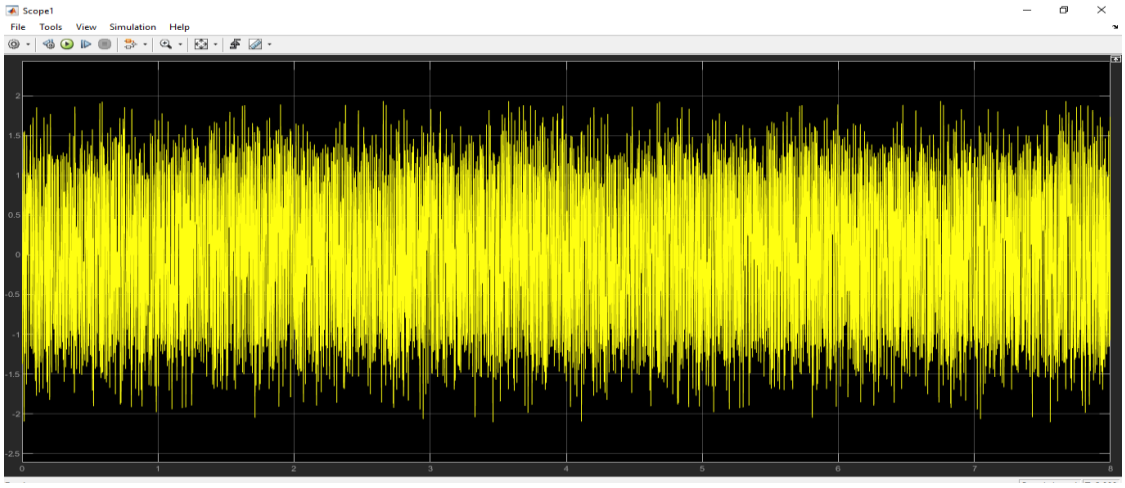

Figure 3.5: The input signal $x(n)$

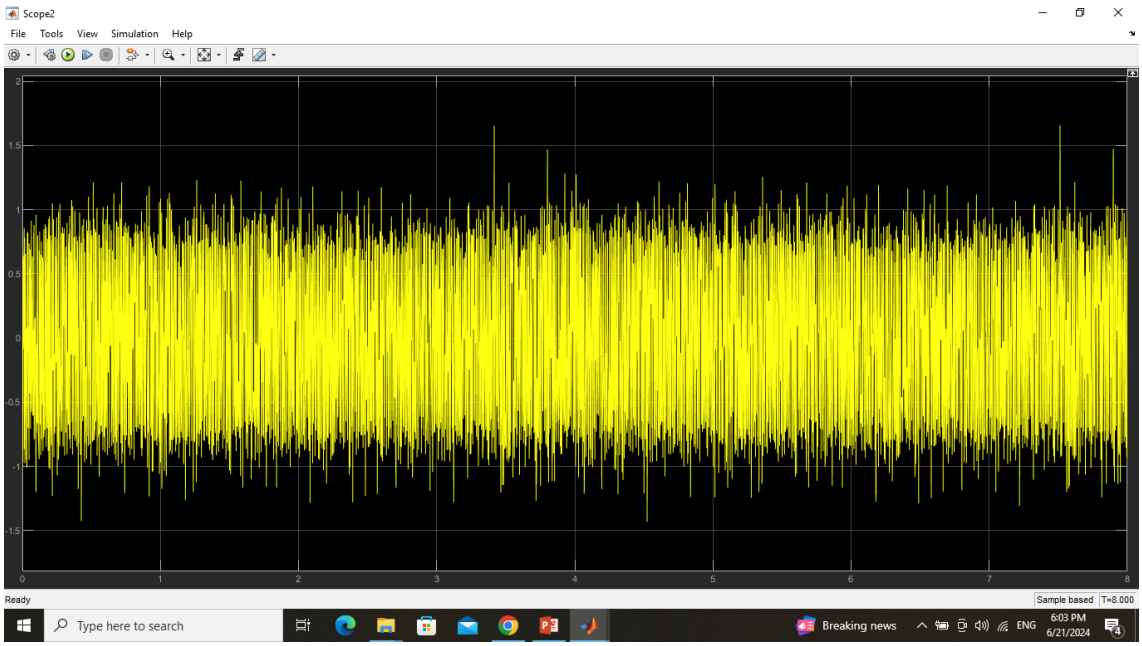

Figure 3.6: The desired signal $\mathrm{d}(\mathrm{n})$

After the test, we got the next result in the next figure:

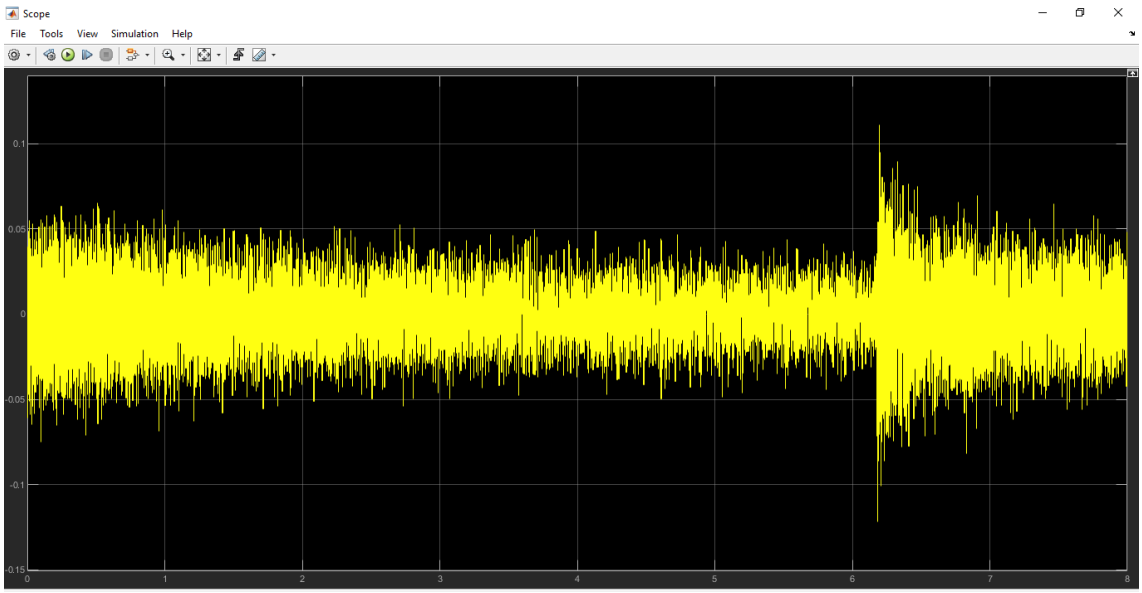

Figure 3.7: Testing result with T=8s

Comment: In this case, we used a low step 0.125 (0001) in our filter, that make us see the noise cancelation, even if we encounter a peaks our algorithm is able to reinstable.

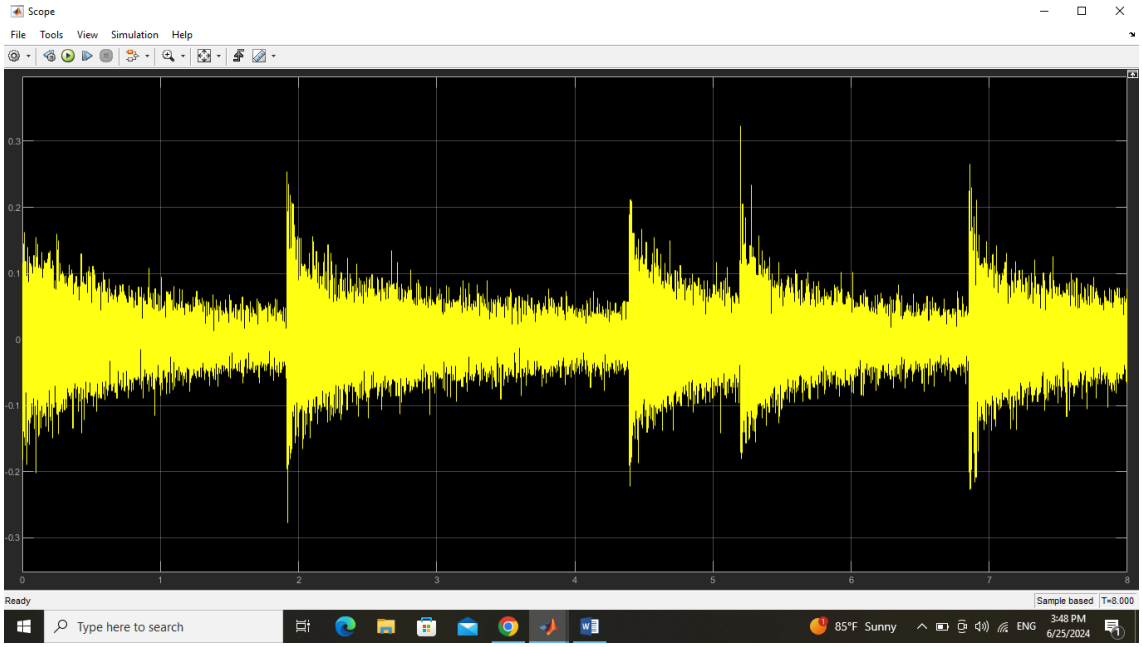

Figure 3.8: Testing result with step = 0.375

Comment: In this case, we used a low step 0.375 (0011) in our filter, that make us see the noise cancelation fast than before because we rise the steps, this is why we see many peaks and in every time our algorithm reinstable.

3.6 Hardware Implementation Circuit of LMS:

With using the board Virtex 6 XC6VLX240T with package FF1156 and speed of -2, we got the next circuit implementation.

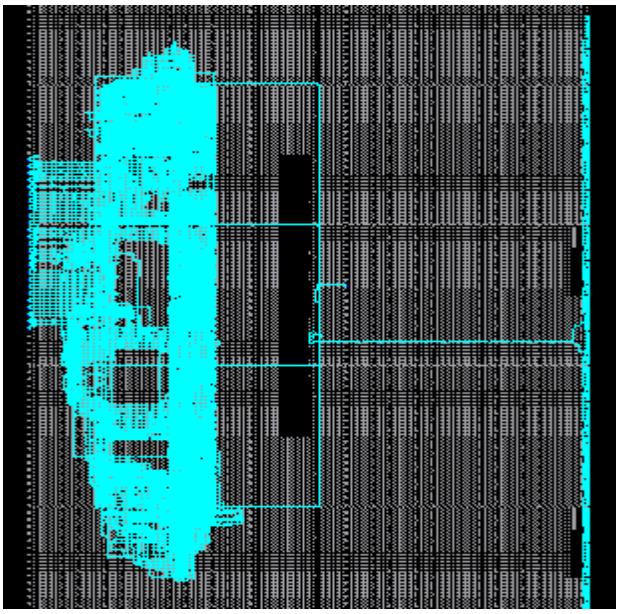

Figure 3.9: Xilinx FPGA Editor of LMS

3.7 Testing of NLMS on MatLab:

With the same previous steps we used, we create the new testing schema of NLMS Algorithm

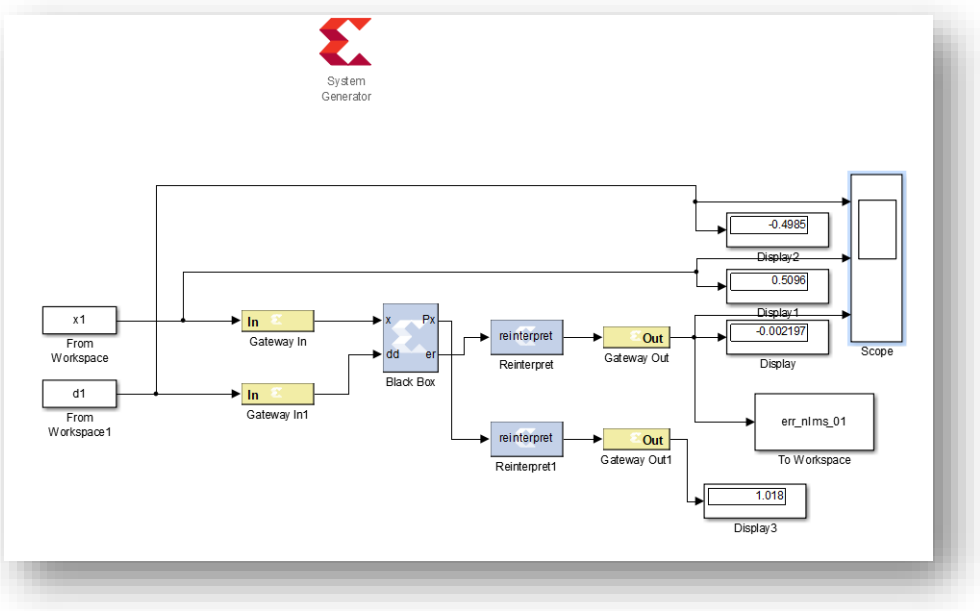

Figure 3.10: Testing schema of NLMS

Next step, we used the same input signal x(n) and desired signal d(n) figure (3.5) and (3.6), so that is what we have as a result:

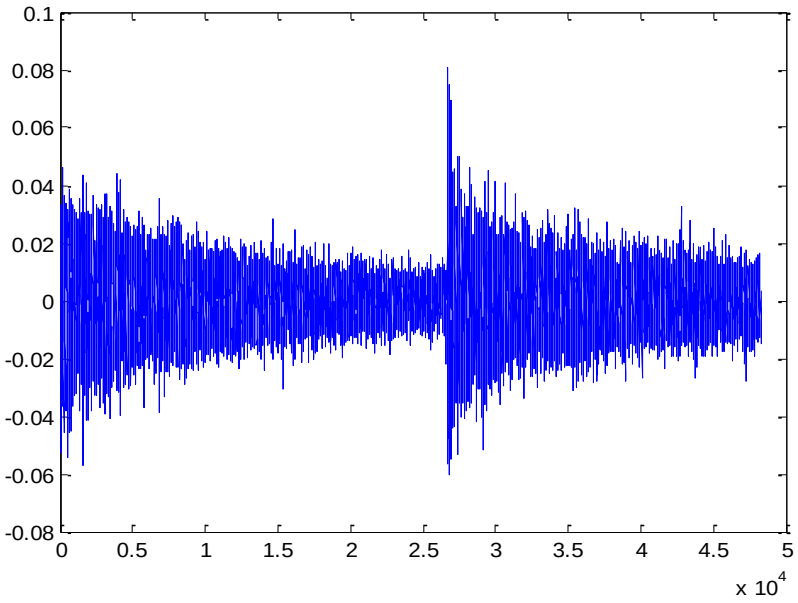

Figure 3.11: testing result

Comment: Here, we saw the noise cancelation by using a step which is confined between $0 < \mu < 2$ in our filter so that it gave us a single peak even if we encounter that peak our algorithm is able to reinstable.

3.8 Hardware Implementation Circuit of NLMS:

With using the board Virtex 6 XC6VLX240T with package FF1156 and speed of -2, we got the next circuit implementation.

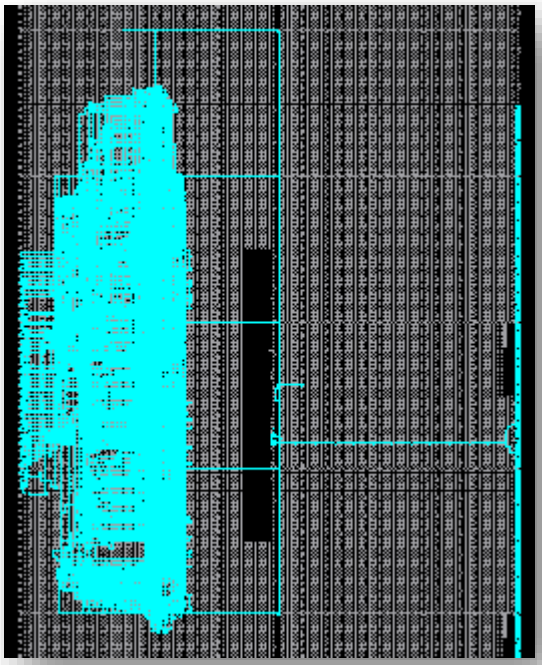

Figure 3.12: Xilinx FPGA Editor of NLMS

3.9 Conclusion:

This chapter concludes by highlighting the project's successful implementation using System Generator and the FPGA implementation tool Xilinx ISE. We discovered that when our algorithm encounters a peak, it can self-reinstable.

General conclusion

The study we presented in this Memory concerns the implementation on FPGA of adaptive filtering algorithms. To do this, we used adaptive filtering with LMS and NLMS algorithms that used to identify finite impulse response filters.

first thing we provides a comprehensive overview of adaptive filtering architectures, highlighting the importance of selecting appropriate algorithms based on factors such as computational complexity, convergence speed, stability, and application requirements. It emphasizes the need for careful evaluation and design choices to achieve optimal performance in FPGA-based systems.

Secondly, the successful synthesis of low-complexity LMS and NLMS structures, with modifications tailored to FPGA constraints, demonstrates the potential for resource-efficient and power-aware implementations. The proposed approach offers a robust framework for deploying adaptive filtering solutions in real-world applications across various domains, including communications, signal processing, and control systems. The ability of the algorithm to self-reinstable when encountering peaks further enhances its practical applicability and stability in challenging signal conditions.

Finally, we tested our different algorithm such as LMS and NLMS that we successfully implemented on FPGA board using input and desired signals to define better functioning and coefficients adjusting.

At the end, we implemented an algorithm LMS and NLMS with reduced logic and we compare between both of them to have better performances.

Bibliography

[1] Kamal Hossain, Roni Ahmed, Md. Asadul Haque, Muahmmad Towfiqur Rahman, Article in International Journal of Science and Research (IJSR),page 1380, March 2021

[2] K.V. Sudheesh et al., System Identification of FIR Filters Journal of Engineering Research and Sciences, 1(4): 74-80, 2022

[3] Behrouz Farhang-Boroujeny and A. G. Constantinides: Design of discrete coefficient FIR filters using LMS algorithm, Singapore, page 1 and 2, 11-14 June 1991

[4] “EEE305”, “EEE801 Part A”: Digital Signal Processing, University of Newcastle upon Tyne, Chapter 7: Adaptive Filtering, page 7.1

[5] Haykin, S. Adaptive Filter Theory, 4th ed.; Prentice-Hall: Upper Saddle River, NJ, USA, 2002. [**Google Scholar**]

[6] GUELLAL Amar, Les circuits FPGA : description et applications,page 8, 2012

[7] Alok Pandey, L.D. Malviya, Vineet Sharma, Comparative Study of LMS and NLMS Algorithms in Adaptive Equalizer,2012

[8] Sudhakar Kalluri and Gonzalo R. Arce, A General Class of Nonlinear Normalized Adaptive Filtering Algorithms, IEEE TRANSACTIONS ON SIGNAL PROCESSING page 1 and 2, 1999

[9] Vijay Kumar Kodavalla, FPGA based Complex System Designs: Methodology and Techniques, Wipro Technologies, Bangalore, India, Design And Reuse 2024

[10] K. R. Rekha, Dr B. S. Nagabushan And Dr K.R..Nataraj, FPGA Implementation of NLMS Algorithm for Receiver in wireless communication system, IJCSNS International Journal of Computer Science and Network Security 2010