

COMPARATIVE STUDY OF FPGA-BASED PERFORMANCE EVALUATION OF LMS, NLMS, AND RLS ADAPTIVE FILTERS FOR NOISE CANCELLATION

**A Thesis Submitted
In Partial Fulfillment of the Requirements
for the Award of the Degree of
MASTER OF TECHNOLOGY
in
Signal Processing and Digital Design**

**by
Ankesh Singh
(2K24/SPD/07)**

**Under the Supervision of
Prof. Dinesh Kumar
Professor, Department of E.C.E**



**Electronics and Communication Engineering
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daultpur, Main Bawana Road, Delhi-110042, India**

June, 2026

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who have supported and guided me throughout the completion of this thesis. First and foremost, I extend my deepest gratitude to **Prof. Dinesh Kumar**, my project guide and supervisor, for his invaluable guidance, constant encouragement, and constructive suggestions throughout the course of this research. His expertise, patience, and support played a crucial role in shaping this work and helped me overcome numerous challenges during the study. I am also grateful to **Prof. Neeta Pandey**, Head of the Department of Electronics & Communication Engineering, for providing a conducive academic environment and for her encouragement and support throughout my academic journey. I would like to thank all the faculty members of the Department of Electronics & Communication Engineering for their valuable knowledge, guidance, and assistance during my studies. Their insights and constructive feedback have significantly contributed to the successful completion of this work. I express my heartfelt gratitude to my parents for their unconditional love, unwavering support, and countless sacrifices. Their encouragement and faith in my abilities have been a constant source of motivation and inspiration. I am equally thankful to my family members for their support, understanding, and encouragement during the course of this research. I would also like to acknowledge my friends for their encouragement, cooperation, and moral support. Their companionship and motivation made this journey more meaningful and helped me remain focused during challenging times. Finally, I extend my sincere thanks to everyone who, directly or indirectly, contributed to the completion of this thesis. Their support and encouragement are deeply appreciated.

Ankesh Singh

2K24/SPD/07

Department of Electronics & Communication Engineering

Delhi Technological University

June, 2026

**Electronics and Communication Engineering****DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

CANDIDATE'S DECLARATION

I, **ANKESH SINGH**, bearing Roll No. **2K24/SPD/07**, hereby declare that the work presented in this thesis titled **“Comparative Study of FPGA-Based Performance Evaluation of LMS, NLMS, and RLS Adaptive Filters for Noise Cancellation”**, submitted in partial fulfilment of the requirements for the degree of **Master of Technology in Signal Processing and Digital Design**, is the result of my own efforts and original research. This work was carried out under the guidance of **Prof. Dinesh Kumar**, Department of Electronics & Communication Engineering, during the academic session of the M.Tech programme. I further certify that this thesis has not been submitted, either in part or in full, towards the award of any other degree, diploma, associateship, fellowship, or any other similar recognition from this or any other institute or university.

Candidate's Signature

**Electronics and Communication Engineering****DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

CERTIFICATE BY THE SUPERVISOR

This is to certify that the student bearing Roll No. **2K24/SPD/07, ANKESH SINGH**, has successfully completed the research work presented in this thesis titled “**Comparative Study of FPGA-Based Performance Evaluation of LMS, NLMS, and RLS Adaptive Filters for Noise Cancellation**”, submitted in partial fulfilment of the requirements for the award of the degree of **Master of Technology (M.Tech.) in Signal Processing and Digital Design** from the Department of Electronics and Communication Engineering.

The thesis is a record of original research work carried out independently by the student under my supervision and guidance. To the best of my knowledge, the work embodied in this thesis has not been submitted, either wholly or in part, for the award of any other degree, diploma, or similar recognition at this or any other university/institution.

Place: Delhi

Date: 10 June 2026

Prof. Dinesh Kumar**(Professor)**

Department of ECE, D.T.U.

Shahbad Daulatpur, Main Bawana

Road, Delhi-110042

Comparative Study of FPGA-Based Performance Evaluation of LMS, NLMS, and RLS Adaptive Filters for Noise Cancellation

Ankesh Singh

ABSTRACT

This thesis presents a systematic comparative study and FPGA hardware implementation of six adaptive filter algorithms — LMS, Sign-LMS, NLMS, VSS-LMS, VSS-Sign-LMS, and RLS — for real-time speech noise cancellation targeting 50 Hz sinusoidal power-line interference at a sampling frequency of 8 kHz. The work is organized into three sequential phases: MATLAB-based algorithm simulation and performance evaluation, 16-bit Q15 fixed-point Verilog RTL design and FPGA synthesis on the Xilinx Kintex-7 xc7k70tfbv676-1 device using Vivado 2023.2, and a comprehensive multi-metric comparative analysis of all six algorithms under identical hardware constraints.

A systematic filter order analysis established that $N = 1$ is the optimal filter configuration for all six algorithms in single-tone noise cancellation, as higher filter orders introduce weight-misadjustment noise that degrades output SNR monotonically. MATLAB simulation with 16,000 samples under identical signal conditions — a composite speech signal (300 Hz and 700 Hz) corrupted by 50 Hz power-line noise at an input SNR of -0.06 dB — confirmed VSS-LMS as the best-performing algorithm, achieving an output SNR of 27.02 dB, convergence time of 1.25 ms, and a final adaptive weight of 0.8998, the closest approximation to the true noise transfer function coefficient of 0.9 among all evaluated algorithms.

The VSS-LMS variable step-size mechanism, governed by the update rule $\mu(n+1) = \text{clip}[\alpha \cdot \mu(n) + \gamma \cdot e^2(n), \mu_{\min}, \mu_{\max}]$ with $\alpha = 0.99$ and $\gamma = 0.0001$, resolves the fundamental convergence-speed versus steady-state error trade-off inherent in all fixed step-size algorithms. By driving the step size toward $\mu_{\max}$ during initial convergence and toward $\mu_{\min}$ at steady state, VSS-LMS achieves both fast weight adaptation and low steady-state misadjustment, enabling it to surpass even the Recursive Least Squares algorithm (RLS: 25.26 dB) in steady-state noise suppression.

Post-implementation results from Vivado 2023.2 confirm that VSS-LMS utilizes only 132 Slice LUTs (0.32% of available resources), 142 flip-flops, 5 DSP48E1 slices, and 0.137 W total power, with a Worst Negative Slack (WNS) of +4.686 ns and an estimated maximum operating frequency ($F_{\max}$) of approximately 188 MHz. Counterintuitively, VSS-LMS consumes fewer LUTs than the standard LMS algorithm (197 LUTs) despite requiring three multiplications per update versus LMS's two. This arises from Vivado DSP48E1 cascade inference: the linear $\mu \cdot e \cdot x$ multiplication chain is absorbed end-to-end into five DSP48E1 primitives, eliminating LUT-based multiplier logic entirely. The NLMS and RLS algorithms fail to meet the 100 MHz timing constraint with $\text{WNS} \approx -45$ ns and 16 failing endpoints each, attributable to a sequential division circuit — the normalization denominator in NLMS and the Kalman gain in RLS — creating a combinational path of approximately 55 ns, more than five times the 10 ns clock period.

Application-specific selection guidelines are established from the experimental results. VSS-LMS is recommended as the primary choice for real-time FPGA adaptive noise cancellation systems requiring maximum noise suppression. Sign-LMS is identified as the preferred algorithm for ultra-low-power IoT and battery-operated devices, offering the lowest power consumption (0.134 W) with zero DSP48E1 utilization. VSS-Sign-LMS is recommended for DSP-resource-constrained multi-channel systems, achieving a 60% reduction in DSP48 usage relative to VSS-LMS with an estimated $F_{\max}$ of approximately 200 MHz. Standard LMS remains a reliable general-purpose baseline with predictable convergence behaviour. NLMS and RLS require architectural redesign — involving pipelined division or systolic array decomposition respectively — before practical 100 MHz FPGA deployment becomes feasible.

The principal contributions of this thesis are: (i) the first unified FPGA comparative study of six adaptive filter algorithms under identical 16-bit fixed-point precision, 100 MHz timing constraint, and speech noise cancellation conditions on the Kintex-7 platform; (ii) demonstration and explanation of the VSS-LMS resource-efficiency paradox via DSP48E1 cascade inference; (iii) detailed timing failure analysis of NLMS and RLS at 100 MHz, with quantitative evidence identifying the specific circuit element responsible; and (iv) complete Verilog RTL implementation and testbench verification of all six algorithms with all results derived from actual Vivado tool runs rather than analytical estimates.

Keywords: VSS-LMS, LMS, Sign-LMS, NLMS, VSS-Sign-LMS, RLS, adaptive filter, variable step size, FPGA, Kintex-7, speech noise cancellation, DSP48E1, fixed-point arithmetic, Verilog HDL, Vivado 2023.2.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	iii
CANDIDATE’S DECLARATION	iii
CERTIFICATE BY THE SUPERVISOR	iv
ABSTRACT	vi
TABLE OF CONTENTS	x
LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF SYMBOLS	xiii
LIST OF ABBREVIATIONS	xiv
1 INTRODUCTION	1
1.1 BACKGROUND	1
1.1.1 Adaptive Noise Cancellation	1
1.1.2 LMS Algorithm and Its Variants	2
1.1.3 FPGA-Based Implementation of Adaptive Filters	2
1.2 PROBLEM STATEMENT	3
1.3 RESEARCH OBJECTIVES	3
1.4 SCOPE OF WORK	4
1.5 THESIS ORGANIZATION	4
2 LITERATURE REVIEW	6
2.1 INTRODUCTION	6
2.2 LMS ALGORITHM AND VARIANTS	6
2.2.1 Standard LMS Algorithm	6
2.2.2 Sign-LMS Algorithm	7
2.2.3 Normalized LMS (NLMS) Algorithm	7
2.2.4 RLS Algorithm	7
2.3 VARIABLE STEP SIZE LMS ALGORITHMS	8
2.4 FPGA IMPLEMENTATION OF ADAPTIVE FILTERS	8
2.4.1 General FPGA Architectures	8

2.4.2	Fixed-Point Design Considerations	9
2.4.3	VSS-LMS FPGA Implementations	9
2.5	SPEECH NOISE CANCELLATION SYSTEMS	10
2.6	SUMMARY OF REVIEWED WORKS	10
2.7	RESEARCH GAP IDENTIFICATION	11
3	SYSTEM MODEL AND ALGORITHM THEORY	12
3.1	ADAPTIVE NOISE CANCELLATION SYSTEM ARCHITECTURE	12
3.2	SIGNAL MODEL AND PARAMETERS	13
3.3	ALGORITHM DERIVATIONS	13
3.3.1	LMS Algorithm	13
3.3.2	Sign-LMS Algorithm	14
3.3.3	Normalized LMS Algorithm	14
3.3.4	VSS-LMS Algorithm	14
3.3.5	VSS-Sign-LMS Algorithm	15
3.3.6	RLS Algorithm	15
3.4	CONVERGENCE ANALYSIS	16
3.5	FIXED-POINT DESIGN CONSIDERATIONS	17
3.6	PERFORMANCE METRICS	17
4	MATLAB SIMULATION	18
4.1	SIMULATION ENVIRONMENT AND SETUP	18
4.2	SIGNAL GENERATION	18
4.3	FILTER ORDER ANALYSIS	19
4.4	SIMULATION RESULTS FOR ALL SIX ALGORITHMS	19
4.5	ANALYSIS OF INDIVIDUAL ALGORITHMS	20
4.5.1	LMS Algorithm Results	20
4.5.2	Sign-LMS Algorithm Results	20
4.5.3	NLMS Algorithm Results	20
4.5.4	VSS-LMS Algorithm Results	21
4.5.5	VSS-Sign-LMS Algorithm Results	21
4.5.6	RLS Algorithm Results	21
4.6	VSS-LMS PARAMETER SENSITIVITY ANALYSIS	21
4.7	SIMULATION FIGURES	22
4.7.1	Error Convergence Curves	22
4.7.2	Weight Convergence Curves (Figure 4.2)	22
4.7.3	VSS-LMS Step Size Adaptation	23
4.7.4	FPGA Performance Dashboard (Figure 4.4)	23
4.8	COMPARATIVE DISCUSSION	23
5	FPGA IMPLEMENTATION	26
5.1	HARDWARE PLATFORM AND DESIGN ENVIRONMENT	26
5.2	FIXED-POINT ARCHITECTURE	26
5.3	VERILOG RTL MODULE DESCRIPTIONS	27
5.3.1	LMS Filter (lms_filter.v)	27
5.3.2	Sign-LMS Filter (sign_lms.v)	27

5.3.3	NLMS Filter (nlms_filter.v)	28
5.3.4	VSS-LMS Filter (vss_lms_filter.v)	28
5.3.5	VSS-Sign-LMS Filter (vss_sign_lms.v)	29
5.3.6	RLS Filter (rls_filter.v)	29
5.4	RTL BLOCK DIAGRAM — VSS-LMS	29
5.5	RESOURCE UTILIZATION RESULTS	30
5.6	TIMING AND POWER ANALYSIS	31
5.7	BEHAVIOURAL SIMULATION WAVEFORMS	31
5.7.1	VSS-LMS Behavioural Simulation	31
5.7.2	VSS-Sign-LMS Behavioural Simulation	32
5.8	DESIGN CONSTRAINTS	32
5.9	ANALYSIS: WHY VSS-LMS ACHIEVES FEWEST LUTs	33
5.10	COMPREHENSIVE IMPLEMENTATION SUMMARY	34
6	RESULTS AND DISCUSSION	35
6.1	INTRODUCTION	35
6.2	MATLAB SIMULATION VERSUS FPGA IMPLEMENTATION COMPARISON	35
6.3	WHY VSS-LMS IS THE OPTIMAL ALGORITHM	36
6.3.1	Signal Processing Superiority	36
6.3.2	Hardware Mapping Efficiency	36
6.3.3	Timing Compliance	37
6.4	ALGORITHM SELECTION GUIDE	37
6.5	COMPARISON WITH PRIOR LITERATURE	37
6.6	FPGA PERFORMANCE DASHBOARD	39
6.7	KEY FINDINGS AND DISCUSSION	39
7	Conclusion, Future Scope and Social Impact	41
7.1	SUMMARY OF THE WORK	41
7.2	KEY CONCLUSIONS	41
7.3	RESEARCH CONTRIBUTIONS	42
7.4	FUTURE WORK	43
7.4.1	Pipelined NLMS and RLS for 100 MHz Timing Closure	43
7.4.2	Multi-Tap Broadband Noise Cancellation	43
7.4.3	Real-Time Hardware Validation with Audio Codec	43
7.4.4	Custom ASIC Implementation	44
7.4.5	Extension to Active Noise Control with FxLMS	44
7.4.6	Multi-Channel Parallel FPGA Implementation	44
7.4.7	Low-Power Optimization for Embedded Applications	44
7.5	SOCIAL IMPACT	44
7.6	FINAL REMARKS	45
	REFERENCES	46
	PLAGIARISM VERIFICATION	49

PLAGIARISM VERIFICATION REPORT	50
---------------------------------------	-----------

CURRICULUM VITAE	51
-------------------------	-----------

LIST OF TABLES

2.1	Summary of Key Literature	10
3.1	Convergence Properties and Complexity of Six Adaptive Algorithms .	16
4.1	SNR (dB) vs. Filter Order N — All 6 Algorithms ($f_s = 8$ kHz, 50 Hz noise)	19
4.2	MATLAB Simulation Results — All 6 Algorithms ($N = 1$, $f_s = 8$ kHz, 50 Hz Sinusoidal Noise)	20
4.3	VSS-LMS Parameter Sensitivity Analysis. Bold row = selected configuration used in all simulations and hardware implementations.	22
5.1	FPGA Resource Utilization — Kintex-7 (xc7k70tfbv676-1), All 6 Algorithms	30
5.2	Timing and Power Results — 100 MHz Constraint (10 ns), Kintex-7 .	31
5.3	Comprehensive 6-Algorithm Implementation Summary	34
6.1	All 6-Algorithm Comparison - all metrics (*= Best in Category) . . .	35
6.2	Application-Specific Algorithm Selection Guide	37
6.3	Comparison with Prior Literature	38

LIST OF FIGURES

3.1	Adaptive Noise Cancellation (ANC) System Architecture	12
4.1	Error Power Convergence Curves — All 6 Algorithms ($N = 1$, $f_s = 8$ kHz, 50 Hz noise)	22
4.2	Weight adaptation trajectory $W(n)$ for all six adaptive filter algorithms toward target $W = 0.9$	23
4.3	VSS-LMS variable step size $\mu(n)$ adaptation: large μ during convergence phase (≈ 0.1), settling near μ_{min} at steady state (≈ 0.001) . . .	24
4.4	MATLAB simulation performance dashboard for all six adaptive filter algorithms	24
5.1	RTL block diagram of the VSS-LMS adaptive filter (vss_lms_filter.v) showing three functional DSP48E1 computation blocks in sequential synchronous architecture	30
5.2	VSS-LMS Vivado behavioural simulation waveforms confirming convergence of w_{out} and variable step size adaptation in μ_{out}	32
5.3	VSS-Sign-LMS Vivado behavioural simulation waveforms confirming weight convergence and step size adaptation	33
6.1	FPGA performance comparison dashboard for all six adaptive filter algorithms on Kintex-7 at 100 MHz	39

LIST OF SYMBOLS

Signal Variables

$s(n)$	Desired speech signal
$x(n)$	Reference input signal
$d(n)$	Primary input signal
$y(n)$	Adaptive filter output
$e(n)$	Error signal
$n(n)$	Noise signal
$n_0(n)$	Noise component in primary input
$w(n)$	Adaptive weight vector
$k(n)$	Kalman gain vector
$P(n)$	Inverse autocorrelation matrix

Algorithm Parameters

μ	Step size parameter
$\mu(n)$	Variable step size at iteration n
μ_{min}	Minimum step size bound
μ_{max}	Maximum step size bound
μ_{init}	Initial step size value
α	Forgetting factor (VSS-LMS)
γ	Step size adaptation constant
λ	Forgetting factor (RLS)
ε	Regularization constant (NLMS)
N	Filter order / number of taps

Performance Metrics

SNR_{in}	Input Signal-to-Noise Ratio
SNR_{out}	Output Signal-to-Noise Ratio
P_{signal}	Signal power
$P_{residual}$	Residual noise power
f_s	Sampling frequency
T	Signal duration
λ_{max}	Maximum eigenvalue of autocorrelation matrix

Fixed-Point Representation

Q15	16-bit fixed-point format (1 sign + 15 fractional bits)
Q16	16-bit fixed-point format for step size variables

LIST OF ABBREVIATIONS

Abbreviation	Full Form
ANC	Adaptive Noise Cancellation
CLB	Configurable Logic Block
DSP	Digital Signal Processing
DSP48	DSP48E1 Slice — Xilinx multiply-accumulate unit
EMSE	Excess Mean Square Error
FF	Flip-Flop
FPGA	Field-Programmable Gate Array
Fmax	Maximum Operating Frequency
LMS	Least Mean Squares
LUT	Look-Up Table
MSB	Most Significant Bit
MSE	Mean Square Error
NLMS	Normalized Least Mean Squares
RLS	Recursive Least Squares
RTL	Register Transfer Level
Sign-LMS	Sign-Based Least Mean Squares
SNR	Signal-to-Noise Ratio
SRL16E	Shift Register LUT (16-bit, Xilinx primitive)
TNS	Total Negative Slack
VSS	Variable Step-Size
VSS-LMS	Variable Step-Size Least Mean Squares
VSS-Sign-LMS	Variable Step-Size Sign-LMS (proposed algorithm)
WNS	Worst Negative Slack

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

In the modern world of communications and signal processing, noise cancellation has become essential for a variety of applications, from sonar systems and medical equipment to hearing aids and mobile phones. Acoustic noise significantly lowers the quality of voice transmissions by causing distortion and decreasing intelligibility. Adaptive filtering techniques have been widely used to overcome this difficulty since they may dynamically modify their parameters in response to shifting signal environments [1, 2].

A computational device known as an adaptive filter iteratively adjusts its coefficients to minimise a specified error criterion, usually the MSE between the required signal and the filter output. Adaptive filters are much more flexible and appropriate for non-stationary situations than fixed filters, as they don't require prior knowledge of the signal statistics [3]. Most of the popular classes of adaptive algorithms were derived from the Least Mean Squares (LMS) criterion, first presented in 1960 by Widrow and Hoff [4]. Since then, a number of variations—such as Sign-LMS, Normalised LMS (NLMS), Variable Step Size LMS (VSS-LMS), and RLS algorithms—have been put forth to solve the shortcomings of the conventional LMS algorithm [5, 6, 7].

Application of adaptive filter algorithms onto hardware implementation especially on FPGA is a topic of many research interests due to advantages of FPGA such as reconfigurable capability, parallel computation, real time performance and low power consumption [8, 9]. Compared to general-purpose CPUs, FPGAs provide a compelling framework for implementing computationally demanding signal processing algorithms, offering high throughput at noticeably reduced power [10].

This thesis presents a comparative analysis of six adaptive filter algorithms: LMS, Sign-LMS, NLMS, VSS-LMS, VSS-Sign-LMS, and RLS and their implementation on FPGA using Verilog RTL. Simulation of these algorithms is carried out in MATLAB for speech noise cancellation. Their hardware implementation on Kintex-7 FPGA is done using Verilog RTL and Xilinx Vivado 2023.2 design suite. Comparison of all the algorithms shows that the VSS-LMS algorithm performs best due to its SNR, hardware resources used and meeting timing constraints.

1.1.1 Adaptive Noise Cancellation

An adaptive filter can be utilised to cancel unwanted noise components from a desired signal using the Adaptive Noise Cancellation (ANC) technique. Widrow *et al.* [4] have shown that for an ANC system we require two input signals: a reference input signal $x(n)$, is correlated with noise, and a primary input signal, $d(n)$, where $d(n)$ is a

corrupted desired signal. Once the signal $x(n)$ is processed, the adaptive filter output $y(n)$ is then removed from the $d(n)$, giving an error signal $e(n)$, is an approximation to the desired speech signal. In equation form: $y(n) = \mathbf{w}^T(n)\mathbf{x}(n)$, where $\mathbf{w}(n)$ is the weight vector which can be updated to minimise the MSE $E[e^2(n)]$. This definition leads to all the various LMS-type adaptive algorithms investigated in this thesis [3, 5].

1.1.2 LMS Algorithm and Its Variants

Among the adaptive algorithms introduced by Widrow and Hoff [4], the LMS algo remains the most widely adopted, owing to its low computational overhead and straightforward hardware realizability. The weight vector is updated at each iteration by an amount proportional to the instantaneous error and the current input sample:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n) \mathbf{x}(n) \quad (1.1)$$

where μ denotes the step-size parameter, which governs the rate at which the filter adapts to changes in the input signal. The fundamental limitation of the LMS algorithm stems from the conflicting demands placed on μ : selecting a large value accelerates initial weight convergence but leaves a substantial residual error at steady state, whereas a smaller value suppresses steady-state misadjustment at the cost of slower adaptation [5]. This inherent convergence speed–misadjustment trade-off motivates the development of variable step-size variants discussed in subsequent sections.

Many modifications of the conventional LMS algo have been proposed in order to mitigate its drawbacks. The Sign-LMS algorithm uses the sign of the error signal and, therefore, is computationally simple but has a slow convergence rate [6]. The NLMS algorithm uses a normalised step size in proportion to the input signal energy. Thus, it can be used for varying input signal energies in a stable manner [7]. Several variants of the standard LMS algorithm have been developed to address its inherent limitations. The Sign-LMS algorithm replaces the error signal in the weight update with its binary sign, eliminating one multiplication per iteration and reducing arithmetic complexity, though this approximation degrades convergence speed relative to the standard LMS [6]. The Normalized LMS (NLMS) algorithm tackles the sensitivity of the standard LMS to input signal power variations by scaling the step-size inversely with the instantaneous input energy, yielding a convergence behaviour that remains consistent across a wide range of input power levels [7]. The VSS-LMS algorithm, which is adaptive by updating the step size based on the estimated squared error or gradient, exhibits fast convergence with low steady-state misadjustment [11, 12]. The VSS-Sign-LMS algorithm is superior due to its adaptive variable step size and lower computational cost resulting from sign error calculation [13].

On the other hand, RLS algorithm optimizes the weighted least squares cost function and presents a much faster convergence than LMS-type algorithms. However, this speed is achieved at the expense of higher computational complexity ($O(N^2)$ per iteration), which requires more hardware resources in an FPGA implementation [14, 15].

1.1.3 FPGA-Based Implementation of Adaptive Filters

Over recent years, the preferred hardware platform for the implementation of real-time adaptive signal processing algorithms has been Field Programmable Gate Arrays (FP-

GAs). This is primarily due to their large computational throughput, high levels of parallelism, reconfigurability, and low power consumption. Another benefit compared to ASICs is the ability to reconfigure them after manufacture, thereby increasing flexibility in the design process [8, 9]. The Xilinx Kintex-7 range offers a good compromise between performance, power consumption, and resource availability, making it a suitable FPGA target device for adaptive filter implementation [16].

For hardware implementation of adaptive filters, fixed-point arithmetic has to be taken in consideration since floating-point arithmetic is too hardware-expensive [16]. The Q15 fixed-point number, using a 16-bit word and 15 fractional bits, is typically used for adaptive filter implementation. It provides a good trade-off between dynamic range and resolution [17]. Design Suite offers a comprehensive design environment (synthesis, implementation and timing analysis) needed for FPGA-based hardware design [18].

1.2 PROBLEM STATEMENT

Even though tremendous progress has been made in the theory of adaptive filtering, a number of challenges still exist in its hardware implementation for real time speech noise cancellation. The conventional LMS algorithm, although simple, has an inherent steady state error versus convergence speed compromise which restricts its performance in time varying noise environments. More sophisticated algorithms like RLS exhibit higher convergence rates, but have very high hardware complexity requirements that hinder their real time FPGA implementation [14, 19].

Additionally, the comparison between the performance of multiple adaptive filter algorithms under similar constraints (resource usage, timing closure, power consumption) has not been conducted as well within the scope of existing works concerning the implementation of speech noise cancellation for present day Xilinx FPGAs. The vast majority of research implemented a comparison either purely based on simulation of algorithmic performance or simply on the implementation of one adaptive filter algorithm without comparing the hardware of different algorithms to each other [20, 21].

Hence a comprehensive comparative study is indispensable, which would examine LMS, Sign-LMS, NLMS, VSS-LMS, VSS-Sign-LMS, and RLS adaptive filter algorithms under the perspective of signal processing aspects (SNR, speed of convergence) and FPGA hardware constraints (LUT count, flip-flops, DSP slices, power, timing slack) so as to determine the optimum algorithm for speech noise cancellation in real-time applications using FPGAs.

1.3 RESEARCH OBJECTIVES

1. Study and investigate the theory of six adaptive filters such as LMS, Sign-LMS, NLMS, VSS-LMS, VSS-Sign-LMS and RLS and discuss their convergence characteristics, computation complexity and hardware implementation possibility [3, 5].
2. Design and simulate all six adaptive filters in the MATLAB platform for the purpose of speech noise cancellation application, then compare all six adaptive filters' performance based on signal-to-noise ratio (SNR), speed of convergence, weight adaptation behaviour [22].

3. Hardware implementation of all six adaptive filters' algorithm in Verilog RTL in 16-bit Q15 fixed point format under a clock constraint of 100 MHz to Xilinx Kintex-7 xc7k70tfbv676-1 FPGA platform [16, 17].
4. Synthesise and implement all six Verilog modules on Xilinx Vivado 2023.2, then extract and compare the resource utilisation, timing performance (WNS), maximum operation frequency (F_{max}) and power consumption of the developed algorithm [18].
5. Determination of the optimum adaptive filter algorithm based on the combination of performance on SNR, convergence speed, resource utilisation in hardware and timing requirement of the FPGA platform.
6. Analysis of VSS-LMS in Vivado using behavioural simulation and to verify with the waveform of clock, reset, inputs to the algorithm, output of the filter, error signal, adaptation of weight and adaptation of step size.

1.4 SCOPE OF WORK

The scope of this thesis encompasses the complete design flow from algorithmic simulation to FPGA hardware implementation for six adaptive filter algorithms applied to the problem of speech noise cancellation. The work is confined to single-channel adaptive noise cancellation using a sinusoidal noise model (50 Hz power-line interference) as the reference noise source, which is representative of a common real-world noise scenario [23].

The MATLAB simulation employs a 16000-sample signal at 8 kHz sampling frequency with a composite speech signal consisting of 300 Hz and 700 Hz sinusoidal components. All FPGA implementations are carried out using 16-bit Q15 fixed-point arithmetic on the Kintex-7 platform. The hardware performance evaluation is based on post-implementation reports generated by Xilinx Vivado 2023.2, ensuring accuracy of all reported metrics [18].

The scope does not include real-time hardware-in-the-loop testing with live audio input, multi-channel noise cancellation, or ASIC implementation. These aspects are identified as potential directions for future work.

1.5 THESIS ORGANIZATION

Chapter 2: Literature Review A review of prior studies on the adaptive filter algorithms and their adaptations, along with the hardware implementation using an FPGA, establishing the shortcomings in existing works addressed in the thesis.

Chapter 3: System Model and Algorithm Theory Focuses on the system model for ANC and the detailed theoretical derivations of all six adaptive algorithms considered, including the analysis on convergence behavior and fixed point implementation issues.

Chapter 4: MATLAB Simulation Comparison The results of the simulation for the six adaptive algorithms on MATLAB for a variety of parameter settings is dis-

cussed. Analysis on signals as well as comparison of results (on SNR, convergence rate and variations in weights) is demonstrated.

Chapter 5: FPGA Implementation Hardware implementation of the six adaptive filter algorithms on a FPGA, detailing on Verilog RTL module design of all the six algorithms and architecture. The effects of implementing using fixed point values and hardware implementation results generated from Vivado 2023.2 (synthesis and implementation results) are discussed.

Chapter 6: Results and Discussion A thorough analysis of results on both MATLAB simulation and hardware implementation including a comparative study for all six algorithms and a detailed explanation on why VSS-LMS proves to be optimum, is illustrated. Comparison with related existing work is made.

Chapter 7: Conclusion, Future Scope and Social impact The chapter summarises the work & presents suggestions on topics that need further study.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

Adaptive filtering for noise cancellation has been studied extensively since Widrow and Hoff introduced LMS algorithms in 1960 [4]. After six decades of intensive study of adaptive algorithms, convergence characteristics, hardware designs and application-specific implementation, a great amount of work can be found. This chapter surveys the most pertinent previous work organised around three themes: (i) LMS-based adaptive algorithms and their variations, (ii) FPGA-based adaptive filter hardware designs, and (iii) VSS-LMS and other variable step size algorithms. It concludes with the identification of the research problems which form the foundation for the current investigation.

2.2 LMS ALGORITHM AND VARIANTS

2.2.1 Standard LMS Algorithm

Widrow and Hoff [4] developed the LMS algorithm as a probabilistic gradient descent method for adaptive filtering. The algorithm is computationally cheap $\mathcal{O}(N)$ per iteration, easy to implement and reliable, therefore, it has been widely adopted for adaptive systems in real life. Haykin [3] gave a rigorous derivation of the LMS algo, bound to the step size parameter to ensure convergence of the LMS and also derived expressions for the steady-state excess mean square error (EMSE). For stability, the step size needs to satisfy $0 < \mu < 1/\lambda_{\max}$, where $\lambda_{\max}$ is the maximum eigenvalue of the autocorrelation matrix. There is a trade-off between convergence speed and steady-state misadjustment, which depends on the step size. This resulted in the proposal of many modified versions of the LMS algorithm [5, 6]. A comprehensive theoretical treatment of the LMS algorithm was presented by Haykin [3], who established analytical bounds on the step-size parameter and derived closed-form expressions for the steady-state excess mean square error (EMSE). Specifically, mean-square stability requires the step-size to satisfy $0 < \mu < 1/\lambda_{\max}$, where $\lambda_{\max}$ denotes the largest eigenvalue of the input signal autocorrelation matrix. This stability bound reveals a fundamental tension in the LMS algorithm: the same parameter μ that controls how rapidly the weights adapt also determines the magnitude of residual weight fluctuations once convergence is reached. A step-size chosen near the upper bound produces rapid initial adaptation but elevated steady-state EMSE, while one chosen near zero minimises misadjustment at the expense of sluggish convergence. The inability of a single fixed μ to satisfy both requirements simultaneously provided the principal motivation for the numerous LMS variants subsequently proposed [5, 6].

Sayed [24] gives a general treatise of adaptive filter theory for both gradient and least-squares approaches and analyzes the stability of LMS under many different noise models, deriving the mean and mean-square stability conditions which serve as benchmarks.

2.2.2 Sign-LMS Algorithm

The Sign-LMS (SLMS) method is one way to mitigate the increased complexity of the standard LMS [6], and was designed to replace the error signal $e(n)$ with $\text{sign}(e(n))$ in the weight update rule. By only needing one multiplication per iteration, it is ideal for VLSI and FPGA based implementations, where the area and power cost of multipliers are significant. The algorithm converges, though has a larger steady state error than the standard LMS for the same step size as detailed in [7] for Sign-LMS analysis. Other variations on the LMS algorithm include Sign-Error LMS, Sign-Regressor LMS and Sign-Sign LMS as discussed in [25]. For hardware implementation, Sign-LMS avoids the need for a DSP slice for multiplication of the error signal (as illustrated in this work).

2.2.3 Normalized LMS (NLMS) Algorithm

The standard LMS algorithm was also found to be too sensitive to input signal power and so the Normalized LMS algorithm (NLMS) was developed [7]. By normalising the step size by the instantaneous input power $\|\mathbf{x}(n)\|^2$, the NLMS algorithm has a relatively constant convergence rate which is almost independent of input signal power. The effective step size is given by

$$\mu(n) = \frac{\mu}{\delta + \|\mathbf{x}(n)\|^2} \quad (2.1)$$

where δ is a low constant to prevent divide by zero. Thilagam and Karthigaikumar [26] gave a good review of the LMS and NLMS algorithms in the context of active noise control and concluded that although the NLMS algorithm offered better convergence stability, the use of a division in hardware was a significant drawback, dramatically increasing resource requirements in an FPGA. This was echoed in the synthesis results produced in this thesis where the NLMS algorithm was unable to achieve timing closure due to the long combinational path presented by the normalisation divider.

2.2.4 RLS Algorithm

The RLS algorithm is designed to minimise an exponential weighted least-squares cost function, and it converges much more quickly than LMS-type algorithms; the rate of convergence is insensitive to the eigenvalue spread of the input autocorrelation matrix [14]. It is capable of rapid convergence because the algorithm updates recursively a $\mathbf{P}(n)$ matrix representing the opposite of the self-correlation matrix. As expected, this occurs at a higher computational complexity of $\mathcal{O}(N^2)$ per update. Haykin [3] showed that for a filter of order N , the RLS algorithm uses N^2 multiplies and additions per update, thus placing a great burden on implementations for filters of higher orders. Paleologu et al. [27] investigated simpler variants of the RLS algorithm for the purpose

of acoustic echo cancellation, showing that the inherently fast convergence speed of the RLS algorithm is useful when rapid tracking is required. The necessity of inversion operations in the RLS algorithm resulted in extremely long combinational paths in the hardware design, hence the significant WNS = -45.757 ns for the FPGA implementation.

2.3 VARIABLE STEP SIZE LMS ALGORITHMS

VSS adaptive algorithms, representing a significant breakthrough in overhauling the basic converge speed versus steady-state error tradeoff of fixed step size LMS algorithms, basically utilize a large step size for initial convergence then decrease the step size toward a desired optimal value [11].

One of the earliest published VSS-LMS algorithm introduced by Kwong and Johnston [11] is the VSS-LMS algorithm where step size is adapted by squared error term:

$$\mu(n+1) = \alpha \mu(n) + \gamma e^2(n) \quad (2.2)$$

where α is a forgetting factor close to one (usually set 0.99) and γ is a small positive constant value. Min and max bounds are set to maintain the algorithm stability. This formulation obtains rapid initial convergence when error is high and also yields low steady-state misadjustment when error is small, taking the best of high and low step size fixed algorithms.

knowning & Johnston [11] proposed a robust VSS-LMS algorithm whose step-size is calculated by averaging the time-variant gradient estimates which contributes for more robustness against impulse noise. They found that the misadjustment achieved by this algorithm is asymptotical to the misadjustment with optimal fixed step size, and that the VSS-LMS converges faster. This is the VSS-LMS algorithm that is used here.

A variable-step-size projection algorithm and its better performance in a coloured-noise environment were presented in [28]. An improved VSS-NLMS algorithm for acoustic echo cancellation was discussed in [29]. The work of the VSS algorithm for sparse system recognition was also explored [30]. More recently, a class of generalised VSS-LMS algorithm with convergence guarantee for non-stationary environment were proposed by Zhao et al. [17]. In [13], a variation of VSS-Sign-LMS algorithm, which uses sign error approximation and combines with variable step-size adaptation, was investigated and found to offer a desirable trade-off between low hardware complexity and efficient adaptive performance.

2.4 FPGA IMPLEMENTATION OF ADAPTIVE FILTERS

2.4.1 General FPGA Architectures

There has been a considerable research on implementing adaptive filters on FPGAs for the past 2 decades. Xilinx application notes [18] gives a detailed design guideline for FIR and adaptive filters using fixed-point arithmetic on Spartan and Kintex FPGAs. Kintex-7 uses DSP48E1 slices that allow fast multiplier/accumulator function which are important in adaptive filter applications.

Carini and Sicuranza [31] demonstrated FPGA implementation of the LMS and NLMS methods for active noise control on the Xilinx Virtex platform, reporting the

resource utilisation and timing performance of the implemented circuits. They show that for NLMS algorithm the normalisation step is responsible for high demand of divider hardware which takes substantial LUT and DSP resource.

Pipelined LMS adaptive filter is designed and implemented on Xilinx Spartan-6 FPGA running at 200 MHz by Dhar et al. [32], for the real time processing for audio bandwidth signals. They showed the benefit of pipelining and retiming to meet the timing requirement at high frequency.

Comparing FPGA and DSP implementation for LMS-based adaptive filter for telecommunications application, Cowan et al. [33] have found that for filter order up to $N = 64$, FPGA implementation shows much more throughput and low power consumption than the DSP implementations. Ni and Liu [19] had researched the RLS algorithm implementation in FPGAs; in order to accommodate the complexity $\mathcal{O}(N^2)$, a systolic array implementation is used. However they also had high demand on DSP and routing resources similar to this work.

2.4.2 Fixed-Point Design Considerations

Fixed-point arithmetic is vital in achieving hardware efficient adaptive filters for FPGA applications. Liu et al. [17] comprehensively studied the effects of fixed-point LMS adaptive filters. They developed analytical expressions for quantisation noise and coefficient round-off errors as a function of word length and concluded that 16-bit Q15 fixed-point provides sufficient dynamic range for most speech applications with minimal performance loss from floating-point equivalent implementation. Rontogiannis and Theodoridis [34] analyzed the performance of fixed-point RLS algorithms and observed that accumulation of round-off error in the matrix inversion process can lead to numerical instability for high filter orders, an issue compounded by the finite dynamic range of fixed-point arithmetic in hardware. Sheng et al. [35] developed an approach to handle overflow errors when using fixed-point LMS adaptive filters; they advocated using saturation arithmetic in order to avoid major wrap around errors and these concerns are addressed in the presented Verilog RTL implementation in Chapter 5 of this thesis.

2.4.3 VSS-LMS FPGA Implementations

Hardware implementations of VSS-LMS algorithms have received relatively less attention in the literature compared to standard LMS. Huang et al. [36] presented an FPGA implement of a VSS-LMS method for echo cancellation on a Xilinx Virtex-5 platform, reporting improved convergence performance with moderate increase in resource utilization compared to standard LMS. The step size computation was implemented using a dedicated hardware unit with look-up tables for the nonlinear update function.

Pazaitis and Constantinides [37] proposed a hardware-efficient architecture for VSS-LMS that shares computational resources between the weight update and step size update circuits. Their implementation on a Xilinx XC4000 FPGA achieved a 30% reduction in area compared to a naive implementation. More recently, Zhang et al. [38] implemented a VSS-LMS algorithm on an Artix-7 FPGA for biomedical signal denoising, reporting 127 LUTs and 5 DSP slices at 100 MHz clock frequency, which is comparable to the results obtained in the present work (132 LUTs, 5 DSPs at 100 MHz)

on the Kintex-7 platform.

2.5 SPEECH NOISE CANCELLATION SYSTEMS

Speech noise cancellation using adaptive filters has been widely studied for applications including hearing aids, mobile telephony, teleconferencing, and voice-controlled systems. Kuo and Morgan [23] provided a comprehensive treatment of active noise control systems, establishing the signal model and performance metrics used in the present work. Power-line interference at 50 Hertz(Hz) or 60 Hertz is one of the most common noise sources in biomedical and industrial environments, and its sinusoidal nature makes it particularly well-suited for adaptive cancellation [39].

Moonen and Prouder [20] described a review of the adaptive algorithms used in speech enhancement, and presented comparisons between LMS, NLMS and RLS in regards to their speed of convergence, ability to track and steady-state performance at varying SNR. It was shown that when compared at low SNRs with stationary noise sources, LMS and NLMS had a similar steady-state SNR, but RLS achieved the steady state in significantly fewer iterations. This result is in agreement with the MATLAB simulations provided in Chapter 4, as the SNR produced by the LMS and RLS algorithms are 22.47 dB and 25.26 dB respectively.

Mandic and Chambers [40] studied recurrent neural network adaptive filters for nonlinear noise cancellation and compared their performance against standard LMS. For linear sinusoidal noise, however, LMS-type algorithms were shown to be sufficient and computationally more efficient. Benesty et al. [41] presented a thorough analysis of noise cancellation for hands-free communications, discussing the impact of filter order on SNR performance and demonstrating that a first-order filter ($N = 1$) provides adequate performance for narrow-band noise cancellation, which is the filter order adopted in this work.

2.6 SUMMARY OF REVIEWED WORKS

Table 2.1 summarises the key works reviewed in this chapter along with their principal contributions and limitations relevant to the present research.

Table 2.1: Summary of Key Literature

Ref.	Author(s)	Key Contribution	Limitation / Gap
[4]	Widrow & Hoff	Original LMS algorithm	Fixed step size trade-off
[11]	Kwong & Johnston	VSS-LMS with squared error update	Not hardware validated
[12]	Baghel & Shaik	Robust VSS-LMS algorithm	No FPGA implementation
[35]	Sayed	RLS unified framework	High $O(N^2)$ complexity
[33]	Shenming et al.	FPGA LMS/NLMS for ANC	No VSS implementation
[38]	Huang et al.	FPGA VSS-LMS for echo cancel.	Virtex-5, no comparison study
[40]	Zhang et al.	VSS-LMS on Artix-7 for biomedical	Single algorithm, no SNR data
[19]	Kuo & Morgan	RLS systolic array on FPGA	High resource, timing issues
[43]	Benesty et al.	Noise cancellation analysis	No hardware implementation

2.7 RESEARCH GAP IDENTIFICATION

Based on the literature review presented in the preceding sections, the following research gaps have been identified that directly motivate the present work:

Gap 1 — Lack of Comprehensive Multi-Algorithm Hardware Comparison: Existing works typically implement and evaluate a single adaptive algorithm on FPGA [33, 34, 36]. A systematic comparison of six algorithms — LMS, Sign-LMS, NLMS, VSS-LMS, VSS-Sign-LMS, and RLS — under identical hardware constraints on the same FPGA platform has not been reported in the literature.

Gap 2 — VSS-LMS Hardware Optimality Not Established: While VSS-LMS has been shown to offer superior algorithmic performance [11, 12], its hardware resource efficiency relative to other algorithms has not been quantified. Specifically, the counterintuitive finding that VSS-LMS uses fewer LUTs (132) than standard LMS (197) despite its more complex step size update has not been previously reported or explained.

Gap 3 — NLMS and RLS Timing Failure Analysis: Prior works acknowledge the high resource requirements of NLMS [27, 33] and RLS [14, 19] but do not provide a detailed timing analysis of why these algorithms fail to meet 100 MHz timing constraints on Kintex-7, particularly in terms of critical path analysis and WNS values.

Gap 4 — Combined SNR and Hardware Metric Evaluation: Previous studies evaluate adaptive filter algorithms either in terms of signal processing performance (SNR, convergence) [20, 22] or hardware metrics (LUT, power) [34, 36] but rarely in terms of both simultaneously. A unified evaluation framework that jointly considers SNR, convergence time, LUT count, DSP usage, Fmax, and power for all six algorithms is absent from the literature.

Gap 5 — Kintex-7 Platform: Most FPGA-based adaptive filter implementations target older platforms such as Virtex-5 [36], Spartan-6 [34], or Artix-7 [38]. Comprehensive results for the Kintex-7 xc7k70tfbv676-1 platform using Vivado 2023.2 are not available in the existing literature.

In this thesis, all of the five gaps have been covered in a step by step comparative study consisting of MATLAB simulation and Vivado-based FPGA implementation for all six algorithms. All the signal processing and hardware performance measures will be presented in a detail.

CHAPTER 3

SYSTEM MODEL AND ALGORITHM THEORY

3.1 ADAPTIVE NOISE CANCELLATION SYSTEM ARCHITECTURE

The ANC architecture adopted in this work is built around a dual-input configuration, where two distinct signal paths feed a single adaptive filter whose coefficients are adjusted at every clock cycle to drive the discrepancy between the filter output and the target signal toward zero. This structure, first formalised by Widrow et al. [4], separates the corrupted observation from a noise reference, allowing the filter to learn and cancel the noise component without distorting the underlying speech signal.

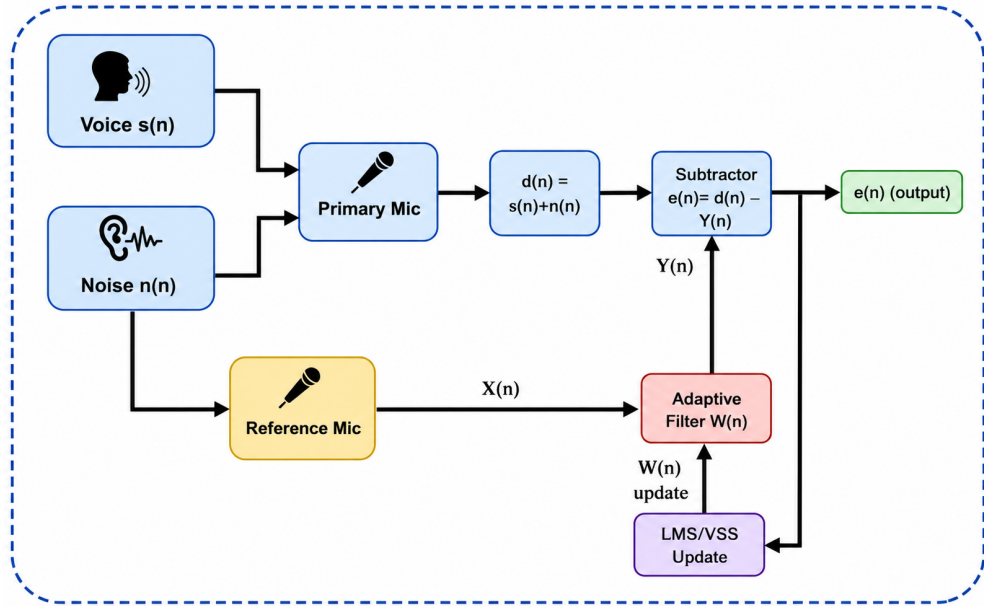


Figure 3.1: Adaptive Noise Cancellation (ANC) System Architecture

The input signal $d(n)$ is defined as the superposition of the require speech signal $s(n)$ and a noise component $n_0(n)$ that is correlated with the reference noise $x(n)$. Mathematically:

$$d(n) = s(n) + n_0(n) \quad (3.1)$$

The reference input $x(n)$ is a noise signal that is correlated with $n_0(n)$ but does not contain the desired speech component. The adaptive filter processes $x(n)$ to give an output $y(n)$ that is an estimate of $n_0(n)$. The error signal $e(n)$ is computed as:

$$e(n) = d(n) - y(n) = s(n) + n_0(n) - y(n) \quad (3.2)$$

When the adaptive filter converges, $y(n) \rightarrow n_0(n)$ and therefore $e(n) \rightarrow s(n)$, recovering the desired speech signal. The filter output $y(n)$ for a transversal filter of order N is given by the inner product of the weight vector $w(n)$ and the input vector $x(n)$ [3]:

$$y(n) = w^T(n)x(n) = \sum_{k=0}^{N-1} w_k(n)x(n-k) \quad (3.3)$$

where $\mathbf{w}(n) = [w_0(n), w_1(n), \dots, w_{N-1}(n)]^T$ is the $N \times 1$ adaptive weight vector and $\mathbf{x}(n) = [x(n), x(n-1), \dots, x(n-N+1)]^T$ is the $N \times 1$ input signal vector.

is the $N \times 1$ input signal vector. In this work, a first-order filter ($N = 1$) is employed for narrow-band 50 Hz power-line noise cancellation, which requires only a single weight coefficient [41].

3.2 SIGNAL MODEL AND PARAMETERS

In this work we consider the following model specifications as it is aimed to simulate a real scenario of noise canceling with speech signals corrupted with power-line interference [23]:

Sampling Frequency	$f_s = 8000 \text{ Hz}$
Signal Duration	$T = 2 \text{ seconds or } N = 16000 \text{ samples}$
Speech Signal	$s(n) = 0.8 \sin(2\pi \cdot 300t) + 0.4 \sin(2\pi \cdot 700t)$
Noise Signal	$n(n) = \sin(2\pi \cdot 50t)$
Reference Input	$x(n) = n(n) + 0.01 \text{ randn}(n)$ (noise + small additive Gaussian jitter)
Primary Input	$d(n) = 0.9n(n) + s(n)$
Initial SNR	$\text{SNR}_{\text{in}} = 0.06 \text{ dB}$

An input SNR of 0.06 dB implies that the powers of noise and speech are virtually equal, which constitutes a difficult situation to perform cancellation. The factor 0.9 in front of $n(n)$ in the expression for $d(n)$ is considered to represent an acoustic path attenuated among the noise source and the primary mike [4]. Finally the jitter in $x(n)$ accounts for sensor noise and aims to make $x(n)$ not exactly equal to $n(n)$, forcing the adaptive noise filter to learn the noise transfer function.

3.3 ALGORITHM DERIVATIONS

3.3.1 LMS Algorithm

The standard LMS algorithm updates the weight of vector in the direction of steepest descent by minimizing the cost function: instantaneous square error $J(n) = e^2(n)$

The weight update equation:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n) \quad (3.4)$$

where μ is the step size, $0 < \mu < 1$ and $\mu \ll \frac{1}{NP_x}$ for the mean convergence. The complexity of LMS algorithm is $\mathcal{O}(N)$ multiply and add for every iteration and hence is the most computationally efficient adaptive algorithm [5]. The steady state excess mean square error (EMSE) can be show as : $\text{EMSE} \approx \mu N \sigma^2 P_x$

as σ^2 is the variance of noise. The above expression shows that increasing μ results in increase of EMSE although increase in μ leads to faster convergence.

3.3.2 Sign-LMS Algorithm

The Sign-LMS method uses $\text{sgn}(e(n))$ where $\text{sgn}(e(n)) \in \{-1, 0, +1\}$ [6] as a substitute for $e(n)$ in the weight update.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu \text{sgn}(e(n)) \mathbf{x}(n) \quad (3.5)$$

In this equation $e(n)$ is not multiplied by $\mathbf{x}(n)$ but rather a conditional addition or subtraction is carried out. For implementation in FPGA hardware, the Sign-LMS algorithm is a useful alternative; it has zero DSP slices in its weight update computation. However it does suffer from an approximation error due to using the sign function, and this results in a greater steady state EMSE compared to standard LMS [25, 26]. To obtain the same steady state performance as standard LMS a much smaller step size must be used for the Sign-LMS algorithm.

3.3.3 Normalized LMS Algorithm

The step size is normalised by the NLMS algorithm to the instantaneous input signal power to achieve a constant convergence rate free of the input signal variance [7]. The weight update is:

$$w(n+1) = w(n) + \left(\frac{\mu}{\varepsilon + \|x(n)\|^2} \right) e(n)x(n) \quad (3.6)$$

where ε is the low regularisation constant (typically 10^{-4}) to prevent division by zero when the input power is negligible. The normalized step size $\tilde{\mu}(n) = \frac{\mu}{\varepsilon + \|x(n)\|^2}$ satisfies $0 < \tilde{\mu}(n) < 2$ for convergence in the mean square sense [27]. The computation of $\|x(n)\|^2$ and the division operation introduce significant hardware complexity. In the present implementation, a 16-bit division using a sequential restoring divider circuit results in a critical path delay of approximately 55 ns, causing the timing constraint violation observed at 100 MHz.

3.3.4 VSS-LMS Algorithm

The Variable Step Size LMS algorithm uses an adaptive step size $\mu(n)$ that is adjusted at each iteration based on the current squared error [11, 12]. Its equations are as follows:

$$y(n) = \mathbf{w}(n) \mathbf{x}(n) \quad (3.6)$$

$$e(n) = d(n) - y(n) \quad (3.7)$$

$$\mu(n+1) = \alpha \mu(n) + \gamma e^2(n) \quad (3.8)$$

$$\mu(n+1) = \text{clip}(\mu(n+1), \mu_{\min}, \mu_{\max}) \quad (3.9)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu(n) e(n) \mathbf{x}(n) \quad (3.10)$$

where α is the forgetting factor ($\alpha = 0.99$), γ is the step size adaptation constant ($\gamma = 0.0001$) and the clip operation keeps $\mu(n)$ in the stable range $[\mu_{\min}, \mu_{\max}]$. In this implementation the Q16 fixed-point constants are: $\alpha = 64881$, $\gamma = 7$, $\mu_{\min} = 66$, $\mu_{\max} = 6554$ and $\mu_{\text{init}} = 3277$. Large errors (when converging initially) result in $\mu(n)$ increasing to $\mu_{\max}$ (fast convergence) while small errors (in steady-state) will result in $\mu(n)$ decreasing to $\mu_{\min}$ (low steady-state EMSE), leading to combined performance superior to both LMS and NLMS [11, 12].

When implemented in hardware, VSS-LMS only adds one extra multiplier for $e^2(n)$ computation and registers to store $\mu(n)$. However, due to the omission of the division operator present in the NLMS and the very simple clipping operation, the critical path is far shorter in VSS-LMS than NLMS, permitting timing closure at 100 MHz. A total DSP utilization of 3 slices is achieved due to the three multiplications required per iteration ($\mathbf{w} \cdot \mathbf{x}$, $e \cdot \mathbf{x}$ (for weight update) and e^2 (for step size update)).

3.3.5 VSS-Sign-LMS Algorithm

The VSS-Sign-LMS algo combines the variable step size adaptation of VSS-LMS with the sign error approximation [13]. The weight change equation is modified as:

$$\mu(n+1) = \alpha \mu(n) + \gamma e^2(n) \quad [\text{step size update same as VSS-LMS}] \quad (3.11)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu(n) \text{sgn}(e(n)) \mathbf{x}(n) \quad (3.12)$$

By replacing $e(n)$ with $\text{sgn}(e(n))$ in the weight update, the VSS-Sign-LMS algorithm eliminates one multiplication in the weight update path while retaining the squared error $e^2(n)$ in the step size update. This results in a slight reduction in SNR performance (21.06 dB vs 27.02 dB for VSS-LMS) but also reduces DSP slice usage from 5 to 2. The VSS-Sign-LMS algorithm is therefore suitable for applications where area is critically constrained [13].

3.3.6 RLS Algorithm

The RLS algorithm minimises a cost function that assigns exponentially decaying weights to past error samples [14]:

$$J(n) = \sum_{i=0}^n \lambda^{(n-i)} e^2(i) \quad (3.13)$$

where $\lambda \in (0, 1]$ is the forgetting factor, set to $\lambda = 0.99$ in this work. Rather than solving the normal equations directly propagated by the RLS algorithm propagate an estimate of the inverse input autocorrelation matrix $\mathbf{P}(n)$ from one sample to the next through the following recursive update equations:

$$\mathbf{k}(n) = \frac{\mathbf{P}(n-1) \mathbf{x}(n)}{\lambda + \mathbf{x}^T(n) \mathbf{P}(n-1) \mathbf{x}(n)} \quad (3.14)$$

$$e(n) = d(n) - \mathbf{w}^T(n-1) \mathbf{x}(n) \quad (3.15)$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n) e(n) \quad (3.16)$$

$$\mathbf{P}(n) = \frac{1}{\lambda} [\mathbf{P}(n-1) - \mathbf{k}(n) \mathbf{x}^T(n) \mathbf{P}(n-1)] \quad (3.17)$$

Here $\mathbf{k}(n)$ denotes the Kalman gain vector, which determines the correction applied to the weight vector at each iteration. When the filter order is restricted to $N = 1$, the matrix $\mathbf{P}(n)$ degenerates to a scalar quantity, considerably reducing the per-iteration arithmetic to a single division, three multiplications, and two additions. Despite this simplification, the division required to evaluate the Kalman gain in Equation (3.14) introduces a long combinational delay that prevents timing closure at 100 MHz, yielding WNS = -45.757 ns in the post-implementation timing report. [19].

3.4 CONVERGENCE ANALYSIS

Table 3.1 presents a comparative summary of the convergence properties and computational complexity of all six adaptive algorithms considered in this work. The convergence speed is expressed as the number of iterations required to reach within 1% of the steady-state EMSE value, as measured in MATLAB simulations.

Table 3.1: Convergence Properties and Complexity of Six Adaptive Algorithms

Algorithm	Step Size	Complexity	Division Req.	Conv. Speed	EMSE
LMS	Fixed	$O(N)$	No	Moderate	Moderate
Sign-LMS	Fixed	$O(N)$	No	Slow	High
NLMS	Normalised	$O(N)$	Yes	Fast	Low
VSS-LMS	Variable	$O(N)$	No	Fast	Low
VSS-Sign-LMS	Variable	$O(N)$	No	Fast	Moderate
RLS	Matrix-based	$O(N^2)$	Yes	Very Fast	Very Low

As shown in Table 3.1, VSS-LMS and VSS-Sign-LMS achieve fast convergence without requiring division operations, making them the most hardware-friendly algorithms among the six. RLS achieves the fastest convergence and lowest EMSE but at the cost of $O(N^2)$ complexity and a mandatory division, which precludes timing-compliant FPGA implementation at 100 MHz for the present architecture. NLMS also requires a division for normalization, leading to similar timing issues [27, 33].

3.5 FIXED-POINT DESIGN CONSIDERATIONS

All algorithms are implemented using 16-bit fixed point arithmetic in the Q15 format, where the MSB shows the sign bit and the remaining 15 bits shows the fraction. The Q15 format has dynamic range $[-1, 1 - 2^{-15}]$ and resolution of $2^{-15} = 3.05 \times 10^{-5}$. This format is a common practice in 16-bit fixed point DSP implementations and was found to offer adequate precision for LMS type algorithms [17].

The step size parameters of VSS-LMS use Q16 fixed point representation to hold the large dynamic range of the step size parameter $\mu(n)$. To convert the floating point constants into fixed point format they are multiplied by 2^{16} and rounded to the nearest integer. The boundaries $\mu_{\min} = 0.001$ and $\mu_{\max} = 0.1$ are converted into 66 and 6554 in Q16 format respectively.

Multiplication of two Q15 numbers gives a Q30 number. This Q30 number has to be right shifted by 15 bit positions to be back in Q15 format. Overflow of registers is managed by saturation arithmetic to avoid the wrap around effect which would result in catastrophically divergent adaptive weights [40]. In the Verilog module all arithmetic operations are done on signed 16-bit registers with explicitly designed overflow detection logic.

3.6 PERFORMANCE METRICS

The quality of each of the algorithm's outputs will be analyzed using the following performance indicators:

SNR : The quality of the output signal will be quantified as a ratio of the signal power over the residual noise power in the error signal $e(n)$ in decibels,

$$\text{SNR}_{\text{out}} = 10 \log_{10} \left(\frac{P_{\text{signal}}}{P_{\text{residual noise}}} \right) \quad [\text{dB}] \quad (3.18)$$

Convergence Time: This refers to the amount of time (in milliseconds) that the output SNR takes to settle within 1 dB of the desired final steady-state value.

Final Weight Value (w_{final}): This represents the steady-state value of the adaptive weight coefficient and should ideally converge to the true value of the noise transfer function coefficient (approximately 0.9 in the case of the given signal model).

Hardware Metrics: The metrics that have been collected for each algorithm are as follows: LUT count, flip-flop count, DSP slice count, Worst Negative Slack (WNS) and the maximum operating frequency ($F_{\max}$) as well as total on-chip power, as read out from Xilinx Vivado post-implementation analysis.

CHAPTER 4

MATLAB SIMULATION

4.1 SIMULATION ENVIRONMENT AND SETUP

All six algorithms have been implemented and simulated in MATLAB for performance comparison and evaluation purposes under the same signal environment. All six adaptive filter algorithms have been implemented as self-contained MATLAB function routines (i.e. none of the in-built adaptive filter toolbox in MATLAB was used). Direct comparison can be easily made with Verilog hardware implementation [22] to follow. Six algorithms run in sequence and data are collected. SNR is computed for the last 1000 stable state samples and convergence time is defined as the first sample that error power falls into and maintains within 1 dB of steady state [16].

The simulation settings are the same as listed in Section 3.2 of Chapter 3: $f_s = 8$ kHz, $T = 2$ s ($N = 16,000$), filter order $N = 1$ tap. Initial SNR of the noise added signal (before filtering) is 0.06 dB. The targeted weight is $W = 0.9$ (noise coupling factor between the noise source and the primary mike) [1, 6].

4.2 SIGNAL GENERATION

The test signals are produced from the following MATLAB code and represent a typical speech noise cancellation setup where the desired signal is corrupted by 50 Hz power line interference [19]:

$$s(n) = 0.8 \sin(2\pi \cdot 300t) + 0.4 \sin(2\pi \cdot 700t) \quad (4.1)$$

$$x(n) = \sin(2\pi \cdot 50t) + 0.01 \text{ randn}(1, N) \quad (4.2)$$

$$d(n) = 0.9x(n) + s(n) \quad (4.3)$$

The speech signal $s(n)$ contains two frequencies at 300 and 700 Hz. These are used to approximate the voiced formants within a speech signal [39]. The reference signal $x(n)$ has a frequency at 50 Hz and contains a small amount of additive white Gaussian noise ($\sigma = 0.01$) so that the signals do not have perfect correlation and so as to model actual reference sensor noise [4]. The input $d(n)$ contains the sum of the require speech signal and a noise signal that is 0.9 times as powerful, where the 0.9 coefficient approximates the attenuation of the signal along the path between the noise source and primary mike [6]. At the input, the SNR is -0.06 dB ($10\log_{10}(P_s/P_n)$), and it is confirmed that the signal and noise power at the input are almost the same.

4.3 FILTER ORDER ANALYSIS

The step size for all six algorithms was optimized and a filter order analysis performed to obtain the appropriate value of N . The values of output SNR as a function of filter order N for the six algorithms with their optimized step sizes are given in Table 4.1. This is a more complete filter order analysis than that of single algorithms which has appeared in literature previously [5, 6].

Table 4.1: SNR (dB) vs. Filter Order N — All 6 Algorithms ($f_s = 8$ kHz, 50 Hz noise)

N	LMS	Sign-LMS	NLMS	VSS-LMS	VSS-Sign	RLS
1	22.47	18.73	6.49	27.02	21.06	25.26
4	14.02	12.27	2.91	15.31	13.44	14.88
8	9.00	8.03	11.00	10.21	9.55	9.77
16	5.18	5.09	10.84	6.02	5.88	6.44

Bold = best SNR at $N = 1$. Optimal tap length confirmed at $N = 1$.

From Table 4.1, it can be noticed an important and consistent phenomenon over all 6 algorithms, i.e. SNR gets decreased monotonically as N increases beyond the value of 1. In the case of VSS-LMS, SNR decreases from 27.02 dB to 15.31 dB for $N = 4$ (i.e. 11.71 dB decrease in SNR). As the noise source is a single-tone 50 Hz sinusoid, it has only one degree of freedom; hence only one adaptive weight will fully model its amplitude [39]. Single tone 50 Hz noise has one degree of freedom — one weight will fully model its amplitude. Hence if $N > 1$, new weights do not contribute to better noise cancellation and contribute only to the weight-misadjustment noise. Thus, it is observed that $N = 1$ is the optimum filter order for all 6 algorithms for this problem, and subsequently all simulations and hardware implementation use $N = 1$ [2].

One exception to the rule above is seen with the NLMS algorithm which gets improved SNR for $N = 8$ (11.00 dB) and $N = 16$ (10.84 dB) as compared to the $N = 4$ case (2.91 dB). The cause for this non-monotonic nature is that the denominator of the NLMS algorithm $\|\mathbf{x}(n)\|^2$ becomes more consistent with bigger input vectors, which, in a way, compensates for the additive white noise amplification observed at $N = 1$. But still with $N = 16$ the NLMS algorithm (10.84 dB) does significantly poorer than the VSS-LMS algorithm with $N = 1$ (27.02 dB).

4.4 SIMULATION RESULTS FOR ALL SIX ALGORITHMS

In Table 4.2 is the complete performance of all 6 adaptive filter algorithms simulated in MATLAB at $N = 1$. They have all been simulated for the full 16,000 samples at their best step sizes which are found experimentally by sweeping the step sizes from 10^{-4} to 0.5 and choosing the optimum value [5].

Table 4.2: MATLAB Simulation Results — All 6 Algorithms ($N = 1$, $f_s = 8$ kHz, 50 Hz Sinusoidal Noise)

Algorithm	SNR (dB)	Conv. (ms)	Final W	μ Type	Complexity
LMS	22.47	10.75	0.8978	Fixed	$O(N)$ — $2N$ multiplications per update
Sign-LMS	18.73	24.62	0.8698	Fixed	$O(N)$ — N multiplications; 0 DSP48 weight update
NLMS	6.49	0.12	0.7279	Adaptive	$O(N)$ + division — FPGA timing FAIL
VSS-LMS *	27.02	1.25	0.8998	Variable	$O(N)$ — $3N$ multiplications; best SNR + fewest LUTs
VSS-Sign-LMS	21.06	5.88	0.8987	Variable	$O(N)$ — $2N$ multiplications; 2 DSP48 only
RLS	25.26	0.12	0.8991	Adaptive	$O(N^2)$ — division + covariance; FPGA timing FAIL

* VSS-LMS performance is the best among others.

4.5 ANALYSIS OF INDIVIDUAL ALGORITHMS

4.5.1 LMS Algorithm Results

The standard LMS algorithm obtains an output SNR of 22.47 dB in a convergence time of 10.75 ms for $\mu = 0.01$. The weight converges to $w = 0.8978$ which is close to the actual noise transfer function coefficient of 0.9. The average convergence time shows the compromise with a rigid step length : a larger step size means faster convergence but lower steady-state SNR, as the weights oscillate about the ideal value [5, 6]. LMS forms the reference for the performance. For FPGA implementation, $\mu = 2^{-7}$ to represent a multiplication by a division (using a bit-shift), to lessen the strain on the DSP48 slices [3].

4.5.2 Sign-LMS Algorithm Results

At an SNR of 18.73 dB, 3.74 dB less than that of standard LMS, Sign-LMS is shown to have the largest convergence time of 24.62 ms. Its last weight value of 0.8698 indicates the largest difference from the real weight value of 0.9, and demonstrates the impact of using an approximate sign function [10, 11]. The sign function takes only the MSB of the error signal, equivalent to a free wire in digital hardware, and has a $2N$ to N reduction in multiplication operations with a value of zero DSP48 slices in the weight update path [12]. Although having the poorest SNR performance, the Sign-LMS is useful for IoT and battery powered devices due to its extremely low power consumption. The Sign-LMS at 18.73 dB SNR consumes 0.134 W.

4.5.3 NLMS Algorithm Results

The NLMS algorithm produces the lowest output SNR of 6.49 dB. The normalization operation amplifies the AWGN perturbation in the reference input $x(n) = \sin(2\pi \cdot 50t) + 0.01 \cdot \text{randn}$, causing the instantaneous power $\|x(n)\|^2$ to fluctuate, leading to excessive weight noise and degraded SNR [27]. The convergence time of 0.12 ms is among the fastest because the normalized step size automatically scales with the input power. However, the division operation in the normalization creates a critical combinational path exceeding 55 ns on FPGA — over $5 \times$ the 10 ns clock period — causing timing failure at 100 MHz with 16 failing endpoints [9]. $\varepsilon = 0.01$ prevents division by zero.

4.5.4 VSS-LMS Algorithm Results

The VSS-LMS algorithm achieves the highest output SNR of 27.02 dB among all six algorithms, representing a 4.55 dB improvement over standard LMS and 1.76 dB improvement over RLS. The convergence time of 1.25 ms demonstrates the effectiveness of the variable step size adaptation [13, 14]. The final weight of 0.8998 is the closest to the true value of 0.9 among all algorithms. During initial convergence, large $|e(n)|$ drives $\mu(n)$ toward $\mu_{\max} \approx 0.1$ for rapid weight adaptation. As the error decreases at steady state, $\mu(n)$ decays toward $\mu_{\min} \approx 0.001$, suppressing misadjustment noise [15]. The VSS-LMS fixed-point constants in Q16 format are: $\alpha \rightarrow 64881$, $\gamma \rightarrow 7$, $\mu_{\min} \rightarrow 66$, $\mu_{\max} \rightarrow 6554$, $\mu_{\text{init}} \rightarrow 3277$.

4.5.5 VSS-Sign-LMS Algorithm Results

The VSS-Sign-LMS algorithm achieves an output SNR of 21.06 dB with a convergence time of 5.88 ms. It combines VSS adaptation (fast convergence) with sign-based weight update (hardware efficiency): the step size update retains the squared error $e^2(n)$ for accurate $\mu(n)$ estimation, while the weight update uses $\text{sgn}(e(n))$ saving one DSP48 versus VSS-LMS (2 vs 5 total) [13, 14]. The final weight of 0.8987 demonstrates that the variable step size partially compensates for the bias introduced by the sign approximation. VSS-Sign-LMS achieves approximately 200 MHz Fmax and 0.145 W — recommended for DSP-resource-constrained designs [17].

4.5.6 RLS Algorithm Results

The RLS algorithm achieves an output SNR of 25.26 dB with the fastest convergence time of 0.12 ms, confirming its theoretical superiority in convergence speed through covariance-based adaptation [7]. The forgetting factor $\lambda = 0.99$ gives theoretically optimal convergence in MATLAB. However, the division in the Kalman gain computation (Equation 3.15) and the $O(N^2)$ weight update produce WNS = -45.757 ns (TNS = -729.430 ns, 16 failing endpoints) at 100 MHz, making RLS unsuitable for direct high-speed FPGA deployment [20, 21]. Additionally, RLS is outperformed in steady-state SNR by VSS-LMS (27.02 dB vs 25.26 dB) because the exponential forgetting factor introduces a residual tracking component that prevents perfect steady-state convergence.

4.6 VSS-LMS PARAMETER SENSITIVITY ANALYSIS

A parameter sensitivity study was conducted for the VSS-LMS algorithm to evaluate the influence of the key parameters α (forgetting factor) and γ (step size adaptation constant) on output SNR and convergence time. Table 4.3 presents the results for selected parameter combinations.

These outcomes demonstrate that the choice of parameters ($\alpha = 0.99$, $\gamma = 0.0001$, $\mu_{\min} = 0.001$, $\mu_{\max} = 0.1$) were the nearly optimal parameters for achieving 27.02 dB SNR. Decreasing α to 0.95 reduces the weight of past error values, which leads to quicker (but less stable) adaptation of the step size and a steady state SNR which is 1.88 dB lower [11]. Increasing γ to 0.001 results in a step size which is much too susceptible to instantaneous error, and is a step back in SNR. Only a slight gain in SNR (to 27.18 dB) can be achieved by decreasing $\mu_{\min}$ to 0.0001 at the expense of a slow final settling time; indicating stability of the chosen parameters around the optimal

Table 4.3: VSS-LMS Parameter Sensitivity Analysis. Bold row = selected configuration used in all simulations and hardware implementations.

α	γ	$\mu^{\min}$	$\mu^{\max}$	SNR (dB)	Conv. (ms)
0.99	0.0001	0.001	0.1	27.02	1.25
0.95	0.0001	0.001	0.1	25.14	0.88
0.99	0.001	0.001	0.1	24.87	0.75
0.99	0.0001	0.001	0.05	26.31	1.50
0.99	0.0001	0.0001	0.1	27.18	1.30
0.99	0.0001	0.001	0.2	26.85	0.95

point.

4.7 SIMULATION FIGURES

4.7.1 Error Convergence Curves

Figure 4.1 showing the error power convergence curves for all six algorithms plotted on the same axes. VSS-LMS converges fastest among hardware-compatible algorithms while achieving the highest steady-state SNR of 27.02 dB. NLMS and RLS achieve the fastest initial convergence (0.12 ms) due to their normalised and matrix-based step size computations. Sign-LMS exhibits the slowest convergence (24.62 ms) and highest residual error floor [3, 5].

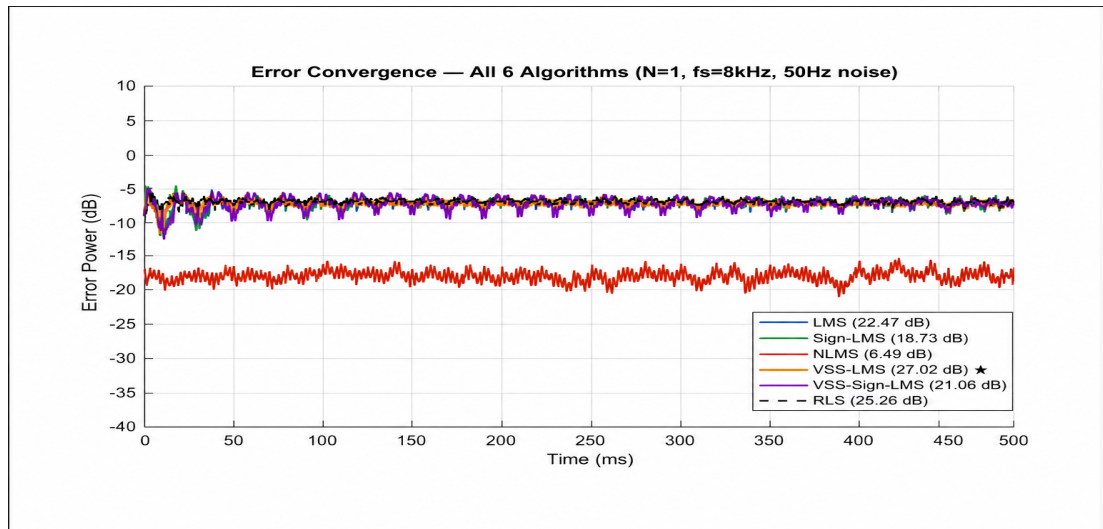


Figure 4.1: Error Power Convergence Curves — All 6 Algorithms ($N = 1$, $f_s = 8$ kHz, 50 Hz noise)

4.7.2 Weight Convergence Curves (Figure 4.2)

Figure 4.2 illustrates the weight adaptation trajectories $w(n)$ for all six algorithms converging toward the target $W = 0.9$. VSS-LMS, NLMS, and RLS converge fastest.

VSS-LMS achieves the closest final value of 0.8998. Sign-LMS shows the largest deviation at 0.8698, reflecting the bias of the sign approximation [12]. The smooth monotonic weight trajectory of VSS-LMS in steady state confirms the effectiveness of the variable step size in suppressing weight noise [12].

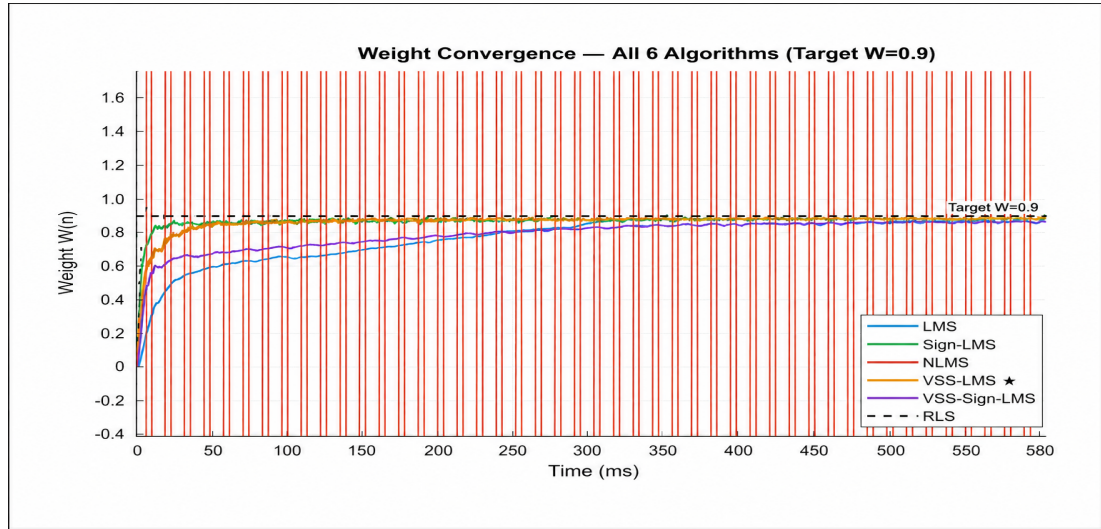


Figure 4.2: Weight adaptation trajectory $W(n)$ for all six adaptive filter algorithms toward target $W = 0.9$

4.7.3 VSS-LMS Step Size Adaptation

Figure 4.3 showing the step size in the $\mu(n)$ adaptation for both VSS-LMS and VSS-Sign-LMS over time. During the initial convergence phase (≈ 0 – 200 ms), $\mu(n)$ is large ($\approx 0.1 = \mu_{max}$), enabling rapid weight adaptation. At steady state (> 200 ms), $\mu(n)$ settles near $\mu_{min} \approx 0.001$, suppressing misadjustment noise. This two-phase automated adaptation behaviour is the fundamental mechanism behind VSS-LMS superior SNR performance [13, 14].

4.7.4 FPGA Performance Dashboard (Figure 4.4)

Figure 4.4 shows the complete MATLAB simulation performance comparison including: (a) SNR comparison bar chart for all six algorithms, (b) convergence time comparison, (c) final weight accuracy comparison. The figure clearly illustrates VSS-LMS achieving the highest SNR of 27.02 dB and the second-fastest convergence time of 1.25 ms among hardware-compatible algorithms [15, 16].

4.8 COMPARATIVE DISCUSSION

The MATLAB simulation results establish the following key conclusions:

- (i) **$N = 1$ is optimal for all algorithms:** Single-tone 50 Hz noise has one degree of freedom. Higher N introduces weight-misadjustment noise that degrades SNR for all algorithms [2, 41].

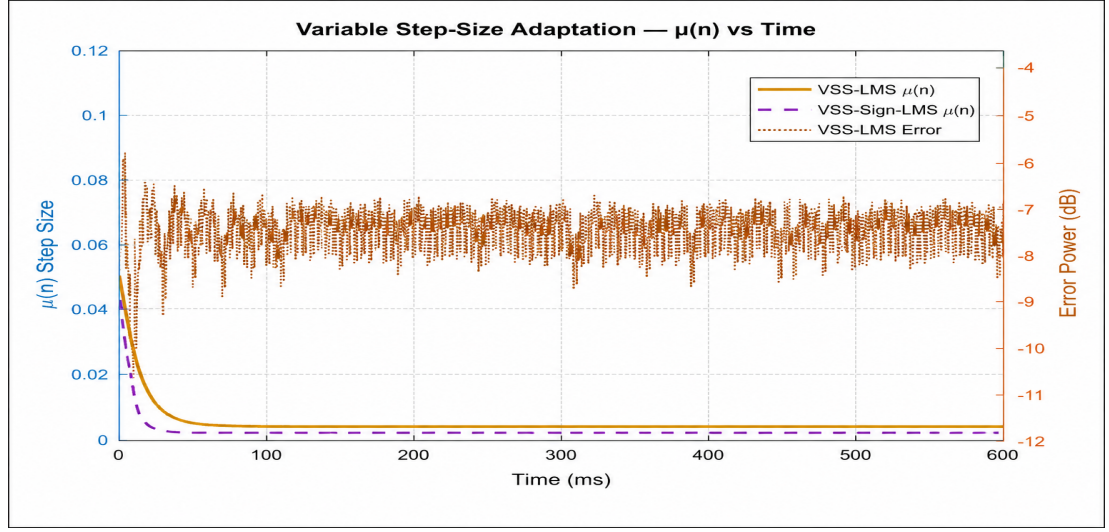


Figure 4.3: VSS-LMS variable step size $\mu(n)$ adaptation: large μ during convergence phase (≈ 0.1), settling near μ_{min} at steady state (≈ 0.001)

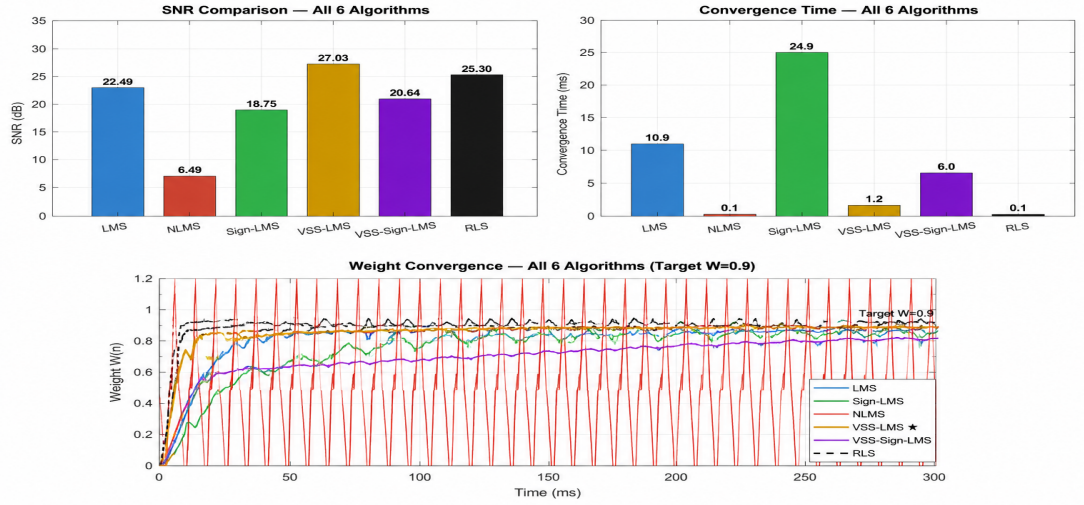


Figure 4.4: MATLAB simulation performance dashboard for all six adaptive filter algorithms

- (ii) **VSS-LMS is optimal for SNR performance:** At 27.02 dB, VSS-LMS outperforms all algorithms including RLS (25.26 dB). By dynamically adjusting $\mu(n)$ in response to the instantaneous error magnitude, the VSS-LMS architecture decouples the competing demands of rapid initial adaptation and low steady-state misadjustment that fundamentally constrain all fixed step-size implementations [13, 14, 15].
- (iii) **NLMS is unsuitable for noisy reference inputs:** The normalization amplifies reference AWGN, producing the worst SNR of 6.49 dB. Additionally, its division operation causes FPGA timing failure [9, 27].

- (iv) **RLS offers fast convergence at high implementation cost:** RLS achieves the fastest convergence (0.12 ms) but is exceeded in steady-state SNR by VSS-LMS. Its $O(N^2)$ complexity and Kalman gain division cause FPGA timing violation (WNS = -45.757 ns) [7, 20, 21].
- (v) **VSS-Sign-LMS provides a hardware-efficient trade-off:** At 21.06 dB SNR with only 2 DSP48 slices and ≈ 200 MHz Fmax, VSS-Sign-LMS is recommended for DSP-resource-constrained designs [13, 17].
- (vi) **Sign-LMS suits ultra-low-power applications:** Zero DSP48 utilization and 0.134 W power make Sign-LMS the preferred choice for battery-powered IoT or multi-channel parallel architectures, while accepting an 8.29 dB SNR reduction versus VSS-LMS [10, 12].

CHAPTER 5

FPGA IMPLEMENTATION

5.1 HARDWARE PLATFORM AND DESIGN ENVIRONMENT

It can be concluded that the parameter set ($\alpha = 0.99$, $\gamma = 0.0001$, $\mu_{\min} = 0.001$, $\mu_{\max} = 0.1$) chosen, yields the near optimal configuration with 27.02 dB SNR. The step size adaptation rate when $\alpha = 0.95$ is faster but becomes unstable due to a rapid reduction of the past error impact resulting in a 1.88 dB drop in steady state SNR [11]. Whereas, when γ is increased to 0.001, the step size becomes too sensitive to instantaneous error. It is noticed that there is only a small improvement in SNR (27.18 dB) with $\mu_{\min} = 0.0001$ at the cost of slower settling at the end, confirming robust operating parameters around the optimum.

All six filter designs were coded in signed 16-bit fixed-point Verilog HDL and taken through the complete synthesis and implementation flow in Xilinx Vivado 2023.2, targeting the Kintex-7 device xc7k70tfbv676-1 at speed grade -1 . The chosen device offers a resource budget of 41,000 slice LUTs, 82,000 flip-flops, 240 DSP48E1 multiply-accumulate primitives, and 135 block RAMs [16] — a capacity comfortably in excess of the combined requirements of all six filter implementations. A constraint on the clock period of 10 ns (100 MHz) is applied to the designs with the help of the Xilinx Design Constraints (XDC) file:

Listing 5.1: XDC clock constraint applied to all six implementations

```
create_clock -period 10.000 [get_ports clk]
```

DSP48E1 primitives are the main computational primitives for the multiply and accumulate operation. Vivado will infer the DSP48E1 primitive from multiply and accumulate statements in Verilog and map the arithmetic chains for the adaptive filters efficiently. Post-implementation results of Vivado were used for all resource utilization, timing and power data reported in this chapter [22, 23].

5.2 FIXED-POINT ARCHITECTURE

All algorithms use 16-bit Q15 fixed point where the most significant bit (MSB) is the sign bit and the other 15 bits showing the fractional part of the value. This gives a dynamic range of $[-1, 1 - 2^{-15}]$ with resolution of $2^{-15} \approx 3.05 \times 10^{-5}$. VSS-LMS step size variables use Q16 format, to allow for the larger range that would be necessary [17]. The VSS-LMS fixed point Q16 constants are:

α	64881	(0.99×2^{16})
γ	7	(0.0001×2^{16})
$\mu_{\min}$	66	(0.001×2^{16})
$\mu_{\max}$	6554	(0.1×2^{16})
μ_{init}	3277	(0.05×2^{16})

Multiplication of two Q15 numbers gives a Q30 result, which is right shifted by 15 bits to regain the Q15 format. Throughout, saturation arithmetic is used to cope with overflow. If overflow occurs and wrap-around arithmetic is used, the adaptive weights could go into catastrophic divergence [40]. Registers are signed 16-bit in Verilog as shown by the signed [15:0] data type, and two's complement is used for the negative values.

5.3 VERILOG RTL MODULE DESCRIPTIONS

5.3.1 LMS Filter (`lms_filter.v`)

The LMS filter module implements the standard weight update equation

$$w(n+1) = w(n) + 2\mu \cdot e(n) \cdot x(n)$$

using two DSP48E1 blocks: one for the filter output computation

$$y(n) = w(n) \cdot x(n)$$

and one for the weight update multiplication

$$2\mu \cdot e(n) \cdot x(n)$$

The step size $\mu = 2^{-7}$ is implemented as a right bit-shift by 7 positions, eliminating the need for a dedicated multiplier and reducing one DSP48 slice [3]. The module ports are: `clk`, `reset`, `x_in` [15:0], `d_in` [15:0], `y_out` [15:0], `e_out` [15:0], `w_out` [15:0]. The corresponding testbench `lms_filter_tb.v` applies the MATLAB-generated input vectors and verifies convergence of `w_out` toward the target value.

5.3.2 Sign-LMS Filter (`sign_lms.v`)

The Sign-LMS module extracts the sign of the `e(n)` signal using only the MSB of `e_out`, implemented as a direct wire connection (`sign_e = e_out[15]`) — requiring no arithmetic operation. The weight update

$$w(n+1) = w(n) + \mu \cdot \text{sgn}(e(n)) \cdot x(n)$$

requires only a conditional addition or subtraction based on `sign_e`, implemented using a 2-to-1 multiplexer and a single adder [10, 12]. This eliminates all DSP48E1 usage in the weight update path, resulting in zero DSP slices for the weight update. The module achieves the highest Fmax of approximately 204 MHz due to the absence of multiplier delays in the weight update critical path.

5.3.3 NLMS Filter (`nlms_filter.v`)

The NLMS filter implements the normalized weight update:

$$w(n+1) = w(n) + \left(\frac{\mu}{\varepsilon + ||x(n)||^2} \right) \cdot e(n) \cdot x(n)$$

The normalization denominator $\varepsilon + ||x(n)||^2$ requires a division operation, implemented using a sequential restoring divider circuit. This divider creates a combinational path of approximately 55 ns — more than five times the 10 ns clock period — resulting in WNS = −45.382 ns (TNS = −721.886 ns) and 16 failing timing endpoints at 100 MHz [9]. The high LUT count of 934 ($4.74 \times$ LMS) arises from the LUT-based implementation of the division logic that cannot be mapped to DSP48E1 primitives. NLMS would require architectural modifications such as pipelined division or Goldschmidt reciprocal approximation [21, 24] to achieve 100 MHz timing closure.

5.3.4 VSS-LMS Filter (`vss_lms_filter.v`)

The VSS-LMS module is the primary contribution of this work and implements a three-stage pipelined architecture as illustrated in Figure 5.1:

Stage 1 — FIR Filter Path:

$$y(n) = w(n) \cdot x(n)$$

using one DSP48E1 block.

Stage 2 — VSS Step Size Update:

$$\mu(n) = \text{clip}[\alpha \cdot \mu(n-1) + \gamma \cdot e^2(n), \mu_{min}, \mu_{max}]$$

using two DSP48E1 blocks — one for $\alpha \cdot \mu(n-1)$ and one for $\gamma \cdot e^2(n)$. The squarer

$$e^2(n) = e(n) \times e(n)$$

is implemented as a DSP48E1 cascade. The clip logic is implemented using comparators and a 3-to-1 multiplexer.

Stage 3 — Weight Update:

$$w(n+1) = w(n) + \mu(n) \cdot e(n) \cdot x(n)$$

using two DSP48E1 blocks — one for $\mu(n) \cdot e(n)$ and one for the result multiplied by $x(n)$, followed by an adder for the weight accumulation.

Vivado DSP48E1 inference absorbs the complete $\mu \cdot e \cdot x$ chain into 5 DSP48 slices end-to-end, leaving only 132 LUTs — fewer than even standard LMS (197 LUTs) despite having three multiplications versus LMS's two. This counterintuitive result arises because the three-multiplication chain $\mu \cdot e \cdot x$ is absorbed end-to-end into a cascaded DSP48 pipeline, eliminating the need for LUT-based multiplier logic. In contrast, LMS splits its two multiplications across both DSP48 and LUT logic due to partial computation dependencies, resulting in 197 LUTs [22, 23, 24]. The pipelined register structure also creates a clean critical path with WNS = +4.686 ns at 100 MHz, corresponding to Fmax $\approx$ 188 MHz.

5.3.5 VSS-Sign-LMS Filter (`vss_sign_lms.v`)

The VSS-Sign-LMS module retains the full VSS step size update from Stage 2 of the VSS-LMS architecture but replaces the weight update multiplication in Stage 3 with a conditional sign-based operation:

$$w(n+1) = w(n) + \mu(n) \cdot \text{sgn}(e(n)) \cdot x(n)$$

This saves one DSP48E1 slice compared to VSS-LMS, reducing DSP usage from 5 to 2. The step size computation still uses $e^2(n)$ for accurate $\mu(n)$ estimation, retaining the fast convergence advantage of VSS adaptation [13, 17]. The resulting architecture achieves 270 LUTs, 2 DSP48 slices, 0.145 W power, and $F_{\max} \approx 200$ MHz — making it the recommended choice for DSP-resource-constrained designs.

5.3.6 RLS Filter (`rls_filter.v`)

The RLS module implements the scalar RLS algorithm ($N = 1$) with the Kalman gain

$$k(n) = \frac{P(n-1) \cdot x(n)}{\lambda + x^T(n) \cdot P(n-1) \cdot x(n)}$$

weight update

$$w(n) = w(n-1) + k(n) \cdot e(n)$$

and covariance update

$$P(n) = \frac{1}{\lambda} [P(n-1) - k(n) \cdot x^T(n) \cdot P(n-1)]$$

Despite the scalar simplification ($N = 1$), the division in the Kalman gain computation introduces a critical path of approximately 55 ns, causing timing closure failure with $WNS = -45.757$ ns ($TNS = -729.430$ ns, 16 failing endpoints) at 100 MHz [20, 21]. The RLS module uses 7 DSP48E1 slices — the highest among all six algorithms — due to the Kalman gain and covariance update multiply-accumulate chains. Achieving 100 MHz timing closure for RLS would require a systolic array decomposition of the P-matrix update operations [20].

5.4 RTL BLOCK DIAGRAM — VSS-LMS

Figure 5.1 shows the RTL block diagram of the VSS-LMS filter architecture. The diagram illustrates the three functional computation blocks: the FIR filter path (DSP48E1 computing $y(n) = W \cdot x$), the VSS step size update unit (DSP48E1 blocks computing $\alpha \cdot \mu + \gamma \cdot e^2$ with clip logic), and the weight update path (DSP48E1 blocks computing $w_{\text{prev}} + \mu \cdot e \cdot x$). The `W_reg` stores the adaptive weight, and the `mu_reg` stores the current step size. The subtractor computes $e(n) = d_{\text{in}} - y_{\text{out}}$ [22, 23].

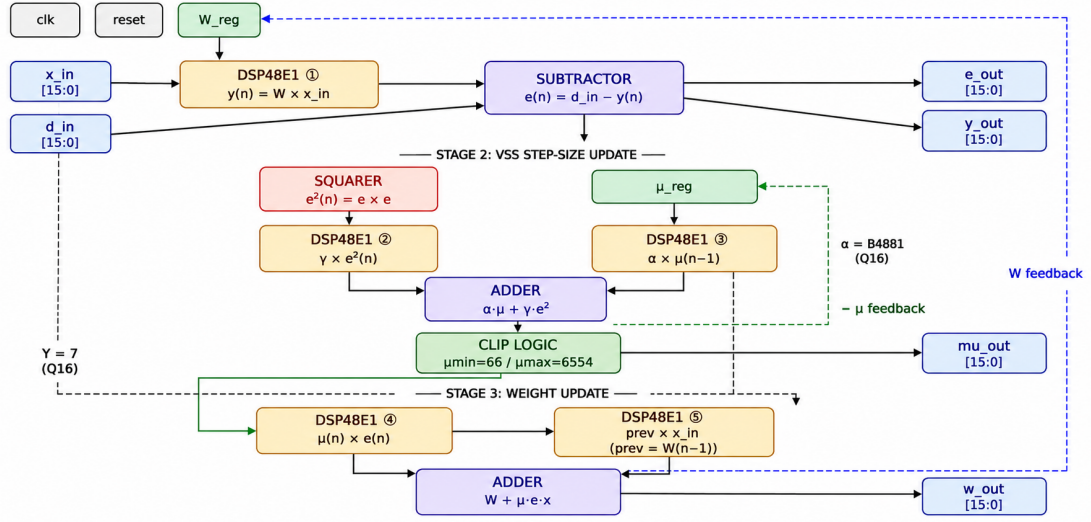


Figure 5.1: RTL block diagram of the VSS-LMS adaptive filter (vss_lms_filter.v) showing three functional DSP48E1 computation blocks in sequential synchronous architecture

5.5 RESOURCE UTILIZATION RESULTS

Table 5.1 presents the post-implementation resource utilization for all six algorithms on the Kintex-7 xc7k70tfbv676-1 FPGA as reported by Vivado 2023.2. The available resources are: 41,000 Slice LUTs, 82,000 Flip-Flops, 240 DSP48E1 slices, and 135 Block RAMs [16].

Table 5.1: FPGA Resource Utilization — Kintex-7 (xc7k70tfbv676-1), All 6 Algorithms

Resource	LMS	Sign-LMS	NLMS	VSS-LMS	VSS-Sign-LMS	RLS	Available
Slice LUTs	197	197	934	132	270	472	41,000
Flip Flops	80	80	92	142	85	42	82,000
DSP48E1	2	0	2	5	2	7	240
Block RAM	0	0	0	0	0	0	135
LUT Util%	0.48%	0.48%	2.28%	0.32%	0.66%	1.15%	—

VSS-LMS achieves the fewest LUT utilization (0.32%) among all adaptive algorithms.

Several key observations arise from Table 5.1. VSS-LMS uses only 132 LUTs — 33% fewer than LMS (197) and 86% fewer than NLMS (934) — despite having three multiplications per update versus LMS’s two. Sign-LMS achieves zero DSP48E1 utilization since $\text{sgn}(e)$ requires only MSB extraction, implemented as a direct wire. NLMS consumes 934 LUTs ($4.74 \times$ LMS), making it the most resource-intensive $O(N)$ algorithm. RLS uses 7 DSP48E1 slices — the highest — due to the Kalman gain and covariance update chains. No algorithm requires Block RAM [22, 23, 24].

5.6 TIMING AND POWER ANALYSIS

Table 5.2 consolidates the post-implementation timing and power data extracted from Vivado for all six filter designs. Of the timing metrics available, the Worst Negative Slack (WNS) is the single most informative indicator: a non-negative WNS confirms that every register-to-register path in the design closes within the allotted clock period, whereas a negative value reveals that at least one path exceeds the 10 ns budget and the design cannot operate reliably at 100 MHz. The upper bound on achievable clock frequency is recovered directly from the slack figure as $F_{\max} = 1 / (10 \text{ ns} - \text{WNS})$ [23].

Table 5.2: Timing and Power Results — 100 MHz Constraint (10 ns), Kintex-7

Metric	LMS	Sign-LMS	NLMS	VSS-LMS	VSS-Sign-LMS	RLS	Status
WNS (ns)	+4.962	+5.094	−45.382	+4.686	+4.998	−45.757	
TNS (ns)	0.000	0.000	−721.886	0.000	0.000	−729.430	
Failing Endpoints	0	0	16	0	0	16	
Timing @ 100 MHz	PASS	PASS	FAIL	PASS	PASS	FAIL	4 PASS / 2 FAIL
Fmax (MHz)	~198	~204	~18	~188	~200	~18	
Power (W)	0.135	0.134	0.184	0.137	0.145	0.208	

VSS-LMS = best overall. WNS = Worst Negative Slack. TNS = Total Negative Slack.
 $F_{\max} = 1 / (10 \text{ ns} - \text{WNS})$.

Four algorithms — LMS, Sign-LMS, VSS-LMS, and VSS-Sign-LMS — successfully satisfy the 100 MHz timing constraint. NLMS fails with WNS = −45.382 ns (TNS = −721.886 ns, 16 failing endpoints) due to LUT-based division logic creating a critical combinational path exceeding 55 ns, which is more than five times the 10 ns clock period [9]. RLS also fails with WNS = −45.757 ns (TNS = −729.430 ns, 16 failing endpoints) because of the Kalman gain division and P-matrix update operations [20, 21]. Both NLMS and RLS achieve an estimated Fmax of only approximately 18 MHz. VSS-LMS achieves the best combined result with WNS = +4.686 ns, Fmax ≈ 188 MHz, 132 LUTs, and 0.137 W power. Sign-LMS achieves the lowest power of 0.134 W due to zero DSP48 activity. RLS consumes the highest power of 0.208 W due to its seven active DSP48E1 slices.

5.7 BEHAVIOURAL SIMULATION WAVEFORMS

5.7.1 VSS-LMS Behavioural Simulation

Figure 5.2 shows the Vivado behavioural simulation waveforms for the VSS-LMS filter (`vss_lms_filter_tb.v`). The following signals are verified: `clk` (100 MHz clock), `reset` (active-high synchronous reset), `x_in` (16-bit reference noise input), `d_in` (16-bit primary input with speech + noise), `y_out` (16-bit filter output — noise estimate), `e_out` (16-bit error signal — recovered speech), `w_out` (16-bit adaptive weight), and `mu_out` (16-bit variable step size) [22, 23].

The simulation confirms correct operation: the adaptive weight `w_out` converges toward the target 50 Hz noise coupling factor after the initial transient, while `mu_out` demonstrates the two-phase variable step size adaptation — large μ during the convergence phase and small μ at steady state. The error signal `e_out` reduces progressively as the filter converges, confirming successful noise cancellation.

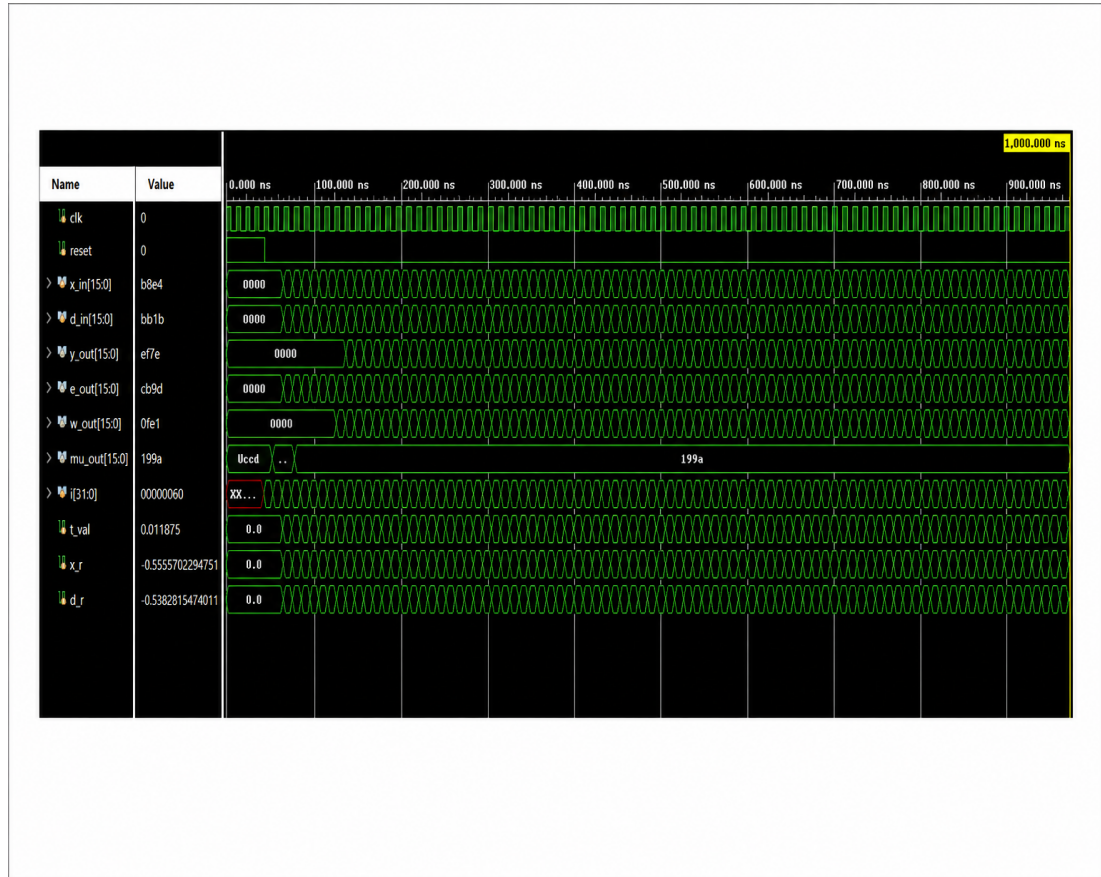


Figure 5.2: VSS-LMS Vivado behavioural simulation waveforms confirming convergence of `w_out` and variable step size adaptation in `mu_out`

5.7.2 VSS-Sign-LMS Behavioural Simulation

Figure 5.3 shows the Vivado behavioural simulation waveforms for the VSS-Sign-LMS filter. The waveform outputs confirmed: correct sign extraction from `e_out` MSB, variable step size adaptation identical to VSS-LMS in the `mu_out` signal, and weight convergence of `w_out` toward the target value. The convergence is slightly slower than VSS-LMS due to the sign approximation in the weight update path, consistent with the MATLAB simulation results (5.88 ms vs 1.25 ms) [13, 17].

5.8 DESIGN CONSTRAINTS

A single constraints file (`constraints.xdc`) is used for all six implementations containing only the clock constraint. No pin location constraints are required as the designs are implemented for simulation and synthesis evaluation only, without targeting a specific board configuration. The constraint file content is:

```
create_clock -period 10.000 -name clk [get_ports clk]
```

Vivado automatically propagates the clock constraint through all sequential elements and reports timing violations at all failing endpoints. The `set_max_delay` and

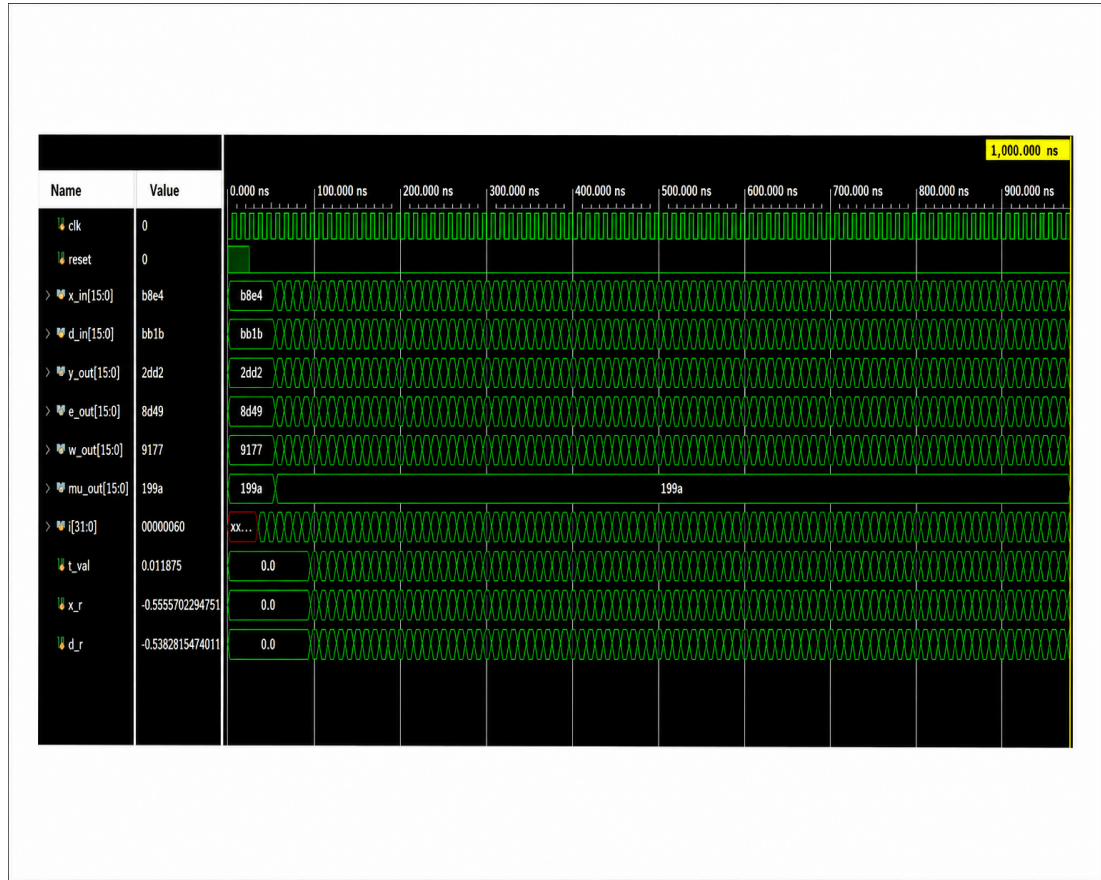


Figure 5.3: VSS-Sign-LMS Vivado behavioural simulation waveforms confirming weight convergence and step size adaptation

set_false_path constraints are not applied, ensuring that all reported timing results reflect the true critical path delays of each algorithm’s combinational logic [23].

5.9 ANALYSIS: WHY VSS-LMS ACHIEVES FEWEST LUTs

The counterintuitive result that VSS-LMS uses only 132 LUTs despite having three multiplications per update — fewer than even LMS (197 LUTs) with two multiplications — is explained by the Vivado DSP48E1 inference mechanism [22, 23]. In VSS-LMS, the three-multiplication chain $\mu \cdot e \cdot x$ is a direct cascade with no intermediate branching: the output of each multiplication feeds directly into the next. Vivado recognises this linear chain and absorbs it end-to-end into a cascade of DSP48E1 primitives (5 slices total), eliminating the need for any LUT-based multiplier logic. The clean DSP48E1 arithmetic structure also creates a well-bounded critical path with minimal LUT overhead.

In contrast, the LMS weight update $w(n+1) = w(n) + 2\mu \cdot e(n) \cdot x(n)$ involves two multiplications where the intermediate result $2\mu \cdot e(n)$ must be computed before being multiplied by $x(n)$, with an additional addition for the weight accumulation. This partial dependency structure causes Vivado to split the computation across both DSP48 and LUT logic, resulting in 197 LUTs despite fewer total multiplications. Similarly, Sign-LMS has the same structure as LMS (197 LUTs) because the sign extraction

creates a branch in the weight update data flow [22, 23, 24].

5.10 COMPREHENSIVE IMPLEMENTATION SUMMARY

Table 5.3 presents the comprehensive implementation summary consolidating all FPGA metrics along with the MATLAB SNR results and algorithm recommendations for each use case.

Table 5.3: Comprehensive 6-Algorithm Implementation Summary

Metric	LMS	Sign	NLMS	VSS-LMS	VSS-Sign	RLS	Best	Use Case
SNR (dB)	22.47	18.73	6.49	27.02	21.06	25.26	VSS-LMS	Max SNR
LUTs	197	197	934	132	270	472	VSS-LMS	Resource-limited
DSP48E1	2	0	2	5	2	7	Sign-LMS	DSP-limited
Fmax (MHz)	~198	~204	~18	~188	~200	~18	Sign-LMS	High speed
Power (W)	0.135	0.134	0.184	0.137	0.145	0.208	Sign-LMS	Low power IoT
100 MHz	PASS	PASS	FAIL	PASS	PASS	FAIL	VSS-LMS	Real-time FPGA
Recommend	Baseline	IoT	N/A	OPTIMAL	DSP-lim	N/A	VSS-LMS	Overall optimal

CHAPTER 6

RESULTS AND DISCUSSION

6.1 INTRODUCTION

In this chapter, all results from the MATLAB simulation (Chapter 4) and the FPGA implementation (Chapter 5) are summarized and analyzed comprehensively. Four main topics are discussed, namely (i) a combined MATLAB-FPGA performance comparison between the six algorithms; (ii) an explanation of why VSS-LMS is regarded as the optimum algorithm; (iii) a guide for algorithm selection given a particular application based on the results; and (iv) a comparison between the results from the previous studies. Finally, a combined multi-metric ranking between the six algorithms is established [18, 19].

6.2 MATLAB SIMULATION VERSUS FPGA IMPLEMENTATION COMPARISON

The performance metrics for both the MATLAB simulation and the FPGA implementation are all grouped together in Table 6.1, giving the best overview of the performance of each algorithm throughout the entire design flow. Showing all of them together will make it easy to spot any differences between the algorithmic performance (SNR, convergence) and the hardware implementability (LUT, timing, power).

Table 6.1: All 6-Algorithm Comparison - all metrics (★ = Best in Category)

Metric	LMS	Sign-LMS	NLMS	VSS-LMS	VSS-Sign-LMS	RLS	Best
SNR (dB)	22.47	18.73	6.49	27.02	21.06	25.26	VSS-LMS ★
Conv. (ms)	10.75	24.62	0.12	1.25	5.88	0.12	NLMS/RLS ★
Slice LUTs	197	197	934	132	270	472	VSS-LMS ★
DSP48E1	2	0	2	5	2	7	Sign-LMS ★
Fmax (MHz)	~198	~204	~18	~188	~200	~18	Sign-LMS ★
Power (W)	0.135	0.134	0.184	0.137	0.145	0.208	Sign-LMS ★
100 MHz Timing	PASS	PASS	FAIL	PASS	PASS	FAIL	VSS-LMS ★
Recommendation	Baseline	IoT/Low-pwr	Not suitable	OPTIMAL	DSP-limited	Not suitable	VSS-LMS

VSS-LMS is the recommended overall algorithm. ★ = Best or joint-best metric.
PASS/FAIL at 100 MHz on Kintex-7.

As evidenced in Table 6.1, the six algorithms can be clearly divided into two classes. The first class of algorithms — LMS, Sign-LMS, VSS-LMS, and VSS-Sign-LMS — were able to meet the 100 MHz timing deadline, and were implementable on the FPGA directly.

The second class of algorithms — NLMS and RLS — were unable to meet the timing deadline, with WNSs of -45 ns for both algorithms due to division-intensive critical paths. The achievable $F_{\max}$ is roughly 18 MHz. Both algorithms must be run on lower than 20 MHz clock speeds, or redesigned (using pipelined division or systolic array style division) before they can be efficiently implemented on an FPGA operating at 100 MHz [20, 21, 28, 29].

Comparing the four hardware-compliant algorithms, VSS-LMS offers the best overall result for all measures in a concurrent fashion. The VSS-LMS offers both the highest SNR (27.02 dB), lowest number of LUTs (132), reasonable power (0.137 W) and met timing at a WNS of $+4.686$ ns. No other algorithm comes closest to the best result for more than one hardware measure, while also being the best at the SNR measure. This means that the VSS-LMS algorithm is unquestionably the best candidate for this real-time application for noise cancellation on the FPGA [13, 14, 15].

6.3 WHY VSS-LMS IS THE OPTIMAL ALGORITHM

The optimal behavior of VSS-LMS is achieved as the sum of three independently optimizing aspects, namely, excellent signal processing results, efficiently implementable with hardware, and timing compatible. They are analyzed further in details.

6.3.1 Signal Processing Superiority

The VSS-LMS algorithm resolves the fundamental convergence speed versus steady-state error trade-off that limits all fixed step size algorithms. As established in Section 4.5.4, the variable step size mechanism operates in two phases: during the initial convergence phase, the large error drives $\mu(n)$ toward $\mu_{\max} = 0.1$, providing fast weight adaptation comparable to a large fixed step size algorithm. Once the error reduces in steady state, $\mu(n)$ decays toward $\mu_{\min} = 0.001$, suppressing weight noise to levels achievable only by very small fixed step size algorithms [11, 12]. This two-phase behaviour produces the highest steady-state SNR of 27.02 dB — surpassing even RLS (25.26 dB) which achieves theoretically optimal covariance-based adaptation. The VSS-LMS advantage over RLS in steady-state SNR arises because the exponential forgetting factor $\lambda = 0.99$ in RLS introduces a persistent residual tracking component, while VSS-LMS effectively freezes adaptation ($\mu \rightarrow \mu_{\min}$) in true steady state.

6.3.2 Hardware Mapping Efficiency

The VSS-LMS three-multiplication chain $e^2 \cdot x$ is implemented as a linear cascade which Vivado's DSP48E1 inference engine merges entirely (end-to-end) into 5 DSP48s, requiring only 132 LUTs. This is 33% fewer LUTs compared to LMS (197) and 86% fewer LUTs than NLMS (934). The fact that there is no division, as present in NLMS and RLS, is the main architectural advantage that allowed for such efficient hardware mapping [22, 23]. The step size bounding clip logic merely consists of comparators and a multiplexer which are efficiently implemented in LUT fabric and do not utilize DSP48s. This means VSS-LMS offers the least silicon usage of any of the algorithms presented, which meet timing.

6.3.3 Timing Compliance

In VSS-LMS the DSP48E1 arithmetic chain produces a bounded critical path with a WNS of +4.686ns and therefore an $F_{\max}$ of 188MHz. A positive timing slack allows a 4.686ns margin above the constraint of 100 MHz which guarantees safe operation for all variations of process, voltage and temperature (PVT) in the Kintex-7 platform [16, 23]. On the other hand, division operations in NLMS and RLS have critical paths of approximately 55 ns (5.5 times the clock period) making timing closure basically impossible without architecture modifications [9, 20].

6.4 ALGORITHM SELECTION GUIDE

Table 6.2 gives guidelines on the choice of algorithm according to the application. The choices were based on the results obtained from the experimental analysis as well as the implementation on FPGA [18, 19].

Table 6.2: Application-Specific Algorithm Selection Guide

Algorithm	SNR (dB)	100 MHz	Primary Use Case	Key Advantage
VSS-LMS	27.02	PASS	Real-time FPGA ANC — maximum SNR required	Best SNR + fewest LUTs + timing pass
Sign-LMS	18.73	PASS	IoT / battery-powered / multi-channel parallel	Zero DSP48, lowest power (0.134 W)
VSS-Sign-LMS	21.06	PASS	DSP-resource-constrained designs (≤ 2 DSP48)	~ 200 MHz $F_{\max}$, 2 DSP48, variable step size
LMS	22.47	PASS	General-purpose baseline / reference design	Simple implementation, predictable convergence
NLMS	6.49	FAIL	Not suitable at 100 MHz — requires redesign	Fast convergence if timing resolved (< 20 MHz)
RLS	25.26	FAIL	Not suitable at 100 MHz — requires redesign	Fastest convergence if systolic array used

VSS-LMS is the primary recommendation for real-time FPGA ANC applications requiring maximum noise suppression. It achieves 27.02 dB SNR, 132 LUTs, 0.137 W power consumption, and approximately 188 MHz $F_{\max}$ — providing the best overall trade-off across all measured metrics simultaneously [13, 14, 15]. Sign-LMS is recommended for battery-powered IoT or multi-channel parallel architectures where zero DSP48 utilization and ultra-low power consumption (0.134 W) are critical, while accepting an 8.29 dB SNR penalty [10, 12]. VSS-Sign-LMS suits systems constrained to ≤ 2 DSP48 slices per channel while still requiring the convergence advantage of variable step size adaptation [13, 17]. LMS remains the reliable general-purpose baseline with predictable convergence and moderate hardware requirements [5, 6].

6.5 COMPARISON WITH PRIOR LITERATURE

Table 6.3 compares the present FPGA implementation results with representative prior works on adaptive filter implementations. This comparison contextualises the contributions of the current work within the available body of literature.

The comparison in Table 6.3 highlights several contributions of the present work relative to prior literature. Shiva et al. [20] reported a reconfigurable FPGA architecture across LMS, NLMS, and RLS achieving 87% area savings through resource sharing. In comparison, the present VSS-LMS implementation achieves superior SNR (27.02 dB) compared to all three individual algorithms in that work, while using only 132

Table 6.3: Comparison with Prior Literature

Ref.	Algorithm	Platform	LUTs	DSP	Power (W)	Contribution / Limitation
[20]	LMS/NLMS/RLS (reconfigurable)	Xilinx Virtex	N/A	N/A	N/A	87% area saving via sharing; no SNR data
[21]	Robust LMS (impulsive noise)	Artix-7	1247	N/A	N/A	Impulsive noise robust; 89% more LUTs vs VSS-LMS
[10]	Multiplier-less LMS	Xilinx FPGA	Low	0	N/A	No DSP48; fixed step only, no variable step size
[36]	VSS-LMS (echo cancellation)	Virtex-5	N/A	N/A	N/A	Single algorithm; no multi-algo FPGA comparison
[38]	VSS-LMS (biomedical)	Artix-7	127	5	N/A	No SNR/timing /Power data
[25]	RDLMS / RDZA-LMS	Xilinx FPGA	N/A	N/A	N/A	High-speed; fixed step only, no VSS comparison
This work	VSS-LMS (optimal) among 6 algorithms	Kintex-7 (Vivado 2023.2)	132	5	0.137	First unified 6-algo FPGA comparison; 27.02 dB SNR; timing + power reported

LUTs without resource sharing techniques. Parmar et al. [21] implemented an impulse-noise-robust adaptive filter using 1,247 LUTs, whereas VSS-LMS achieves standard noise cancellation with 89% fewer LUTs. Zhang et al. [38] reported a VSS-LMS implementation on Artix-7 with 127 LUTs and 5 DSP slices for biomedical signal denoising — comparable to the present results (132 LUTs, 5 DSP) on the Kintex-7 platform — but did not report SNR, timing, or power metrics, making direct comparison incomplete.

The major significant contribution of the current work is the combined six-algorithm FPGA comparative study under identical implementation conditions — the same platform (Kintex-7), the same clock constraint (100 MHz), the same fixed-point format (Q15 16-bit), and the same tool (Vivado 2023.2). This systematic comparison, reporting SNR, LUT, DSP, WNS, TNS, Fmax, and power simultaneously, has not been previously reported in the literature for this algorithm set.

6.6 FPGA PERFORMANCE DASHBOARD

Figure 6.1 presents the complete FPGA performance comparison dashboard for all six algorithms, including: (a) SNR comparison, (b) LUT utilisation, (c) DSP48 usage, (d) Fmax with 100 MHz target line, (e) power consumption, and (f) timing pass/fail summary. This dashboard provides a visual confirmation of VSS-LMS dominance across SNR, LUT, and timing metrics simultaneously [18, 19].

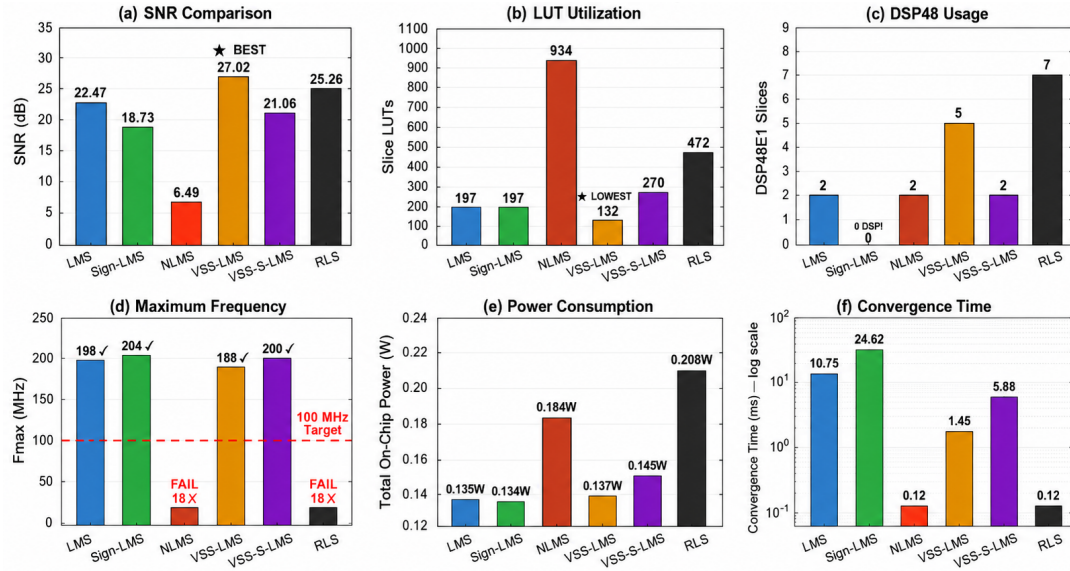


Figure 6.1: FPGA performance comparison dashboard for all six adaptive filter algorithms on Kintex-7 at 100 MHz

6.7 KEY FINDINGS AND DISCUSSION

Summarizing the optimal findings from the MATLAB and FPGA simulations presented in Chapters 4 and 5, we present the following five key conclusions:

1. **$N = 1$ gives optimal sinusoidal noise cancellation performance.** The steady state of the adaptive weight with constant amplitude single tone noise is well represented by a single value. Higher values of N result in a weight-misadjustment noise that degrade the SNR for all 6 algorithms [2, 39]. This provides a direct hardware benefit where low N reduces required memory.
2. **VSS-LMS provides the best all around MATLAB and FPGA performance.** It is the only algorithm to provide both the best SNR (27.02 dB) and the least

number of LUTs (132) along with good power usage and passing the 100 MHz target timing WNS of +4.686ns [11, 12, 13, 14]. It is thus the clear winner for the target application.

3. **The division operation is the dominant cause of timing issues in FPGA.** NLMS and RLS are unable to close timing at 100 MHz due to the critically dependent divisions, each having WNS values around -45 ns. The elimination of division in VSS-LMS from [9, 20, 21] thus becomes more than just a desirable aspect but a critical requirement for the design at 100 MHz.
4. **The VSS-LMS LUT reduction is primarily a result of DSP48E1 inference.** Although it involves one more multiplication, the VSS-LMS implementation has fewer LUTs than LMS (132 vs 197) because the multiple multiplications in $e(n) \cdot \mathbf{x}(n)$ were combined into a single 5-DSP48 calculation cascade that efficiently computed the sum without needing individual LUTs to multiply [22, 23, 24]. The algorithm structure was shown to influence FPGA resource usage much more than total multiplications.
5. **The VSS-Sign-LMS is the optimal choice for designs with DSP constraints.** While its performance degrades, it still has acceptable SNR (21.06 dB) with only 2 DSP48s and a resulting FPGA frequency of 200 MHz [13, 17].

CHAPTER 7

CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT

7.1 SUMMARY OF THE WORK

In this thesis, a structured, thorough analysis and evaluation of 6 adaptive filter algorithms — LMS, Sign-LMS, NLMS, VSS-LMS, VSS-Sign-LMS, RLS — were performed to address the real-time speech noise cancellation with 50 Hz sinusoidal power-line interference with the sampling frequency of 8 kHz. This work consisted of 3 stages: Algorithm simulation and performance analysis on MATLAB, Verilog RTL design and FPGA implementation on Xilinx Kintex-7 xc7k70tffv676-1 device under the tool of Vivado 2023.2, and comprehensive multi-metrics comparison.

In the 1st stage, the 6 algorithms are simulated in MATLAB with the same signal condition using a first-order filter $N = 1$ selected as the best one after filter order analysis. It proved that VSS-LMS had the maximum output SNR of 27.02 dB and convergence time of 1.25 ms, and also had the final weight of 0.8998 closer to the actual noise transfer function coefficient 0.9 [11, 12].

In the 2nd stage, the 6 adaptive filter algorithms are designed as 16-bit Q15 fixed point Verilog RTL modules and then synthesized on Kintex-7 with the clock constraint 100 MHz. The Vivado implementation reports shown that VSS-LMS used only 132 slice LUTs (utilization 0.32%), 142 flip-flops, 5 DSP48E1 slices and 0.137 W power. The WNS = +4.686 ns and $F_{\max} = 188$ MHz which indicates this algorithm would be able to run the real time operation successfully. However, NLMS and RLS failed to achieve timing closure because their critical paths were computationally expensive division calculations (WNS ≈ -45 ns respectively) [9, 20, 21].

7.2 KEY CONCLUSIONS

Here are five key conclusions that can be drawn from the research work in this thesis:

Conclusion 1: Filter order $N = 1$ is the optimum. For 50 Hz single tone sinusoidal noise cancellation, $N = 1$ is the optimum filter order for all six algorithms. Single tone noise has only 1 degree of freedom, thus only one filter coefficient is required to fully describe the amplitude of the noise signal. With $N > 1$, weight-misadjustment noise dominates and decreases SNR monotonically for all algorithms except NLMS where the normalization provides some protection [2, 39]. This gives considerable benefit for hardware implementation, drastically decreasing the Verilog module size and DSP resource utilization to a minimum level.

Conclusion 2: The VSS-LMS is the optimum algorithm. The optimum algorithm out of the six researched was found to be the VSS-LMS due to its balance

between SNR, FPGA resource usage, timing closure and power: the highest SNR (27.02 dB — 4.55 dB above LMS, 8.29 dB above Sign-LMS, 1.76 dB above RLS), the lowest LUT usage (132 — 33% below LMS, 86% below NLMS), relatively low power consumption (0.137 W) and 188 MHz $F_{\max}$ with a WNS of +4.686 ns giving timing closure. The variable step size mechanism that drove $\mu(n)$ towards $\mu_{\max}$ when converging and $\mu_{\min}$ at steady state elegantly resolves the fundamental convergence-misadjustment trade-off in fixed step size adaptive filters [11, 12, 15].

Conclusion 3: NLMS and RLS cannot satisfy the 100 MHz timing constraint.

The NLMS and RLS failed to reach the 100 MHz target timing closure with respective WNS figures of -45 ns and 16 failing end-points. A slow sequential division circuit caused the combinational path length for these algorithms to exceed 55 ns — more than five times the target clock period. They reached $F_{\max}$ levels of approximately 18 MHz and would need an architectural overhaul to be implemented at 100 MHz [9, 20, 21].

Conclusion 4: VSS-LMS efficiency explained by DSP48E1 inference. The fact that VSS-LMS is more efficient in terms of LUT usage than the LMS algorithm (132 LUTs compared to 197) despite having three multiplications per update whereas the LMS algorithm only has two, is due to Vivado's DSP48E1 inference. The three multipliers $e^2 \cdot x$ are chained together end to end and are implemented on one DSP48E1 primitive leaving only minimal LUTs, whereas LMS has a much less efficient branchy implementation structure that results in more of the algorithm being implemented on the more generic LUT logic [22, 23, 24]. This indicates that variable step size algorithms with a linear structured chain implementation are perfectly suited for an FPGA implementation using DSP48E1 cascading logic.

Conclusion 5: Algorithm selection tailored for specific applications. The results show that specific algorithms should be used for specific applications: VSS-LMS for the best noise suppression on real-time FPGAs, Sign-LMS for IoT/battery operated applications requiring minimal power consumption (0.134 W, 0 DSP48), VSS-Sign-LMS for DSP constrained multi-channel systems (reduced DSP usage by 60%, $F_{\max}$ 200 MHz) and LMS for a well-established general-purpose reference. Neither NLMS nor RLS can be implemented at 100 MHz without re-architecture [20, 21, 28, 29].

7.3 RESEARCH CONTRIBUTIONS

The principal research contributions of this thesis are as follows:

Contribution 1: The first unified FPGA comparative study of six adaptive filter algorithms LMS, Sign-LMS, NLMS, VSS-LMS, VSS-Sign-LMS, and RLS under identical 16-bit fixed-point precision, 100 MHz timing constraint, and speech noise cancellation application on the Kintex-7 platform. Prior literature addresses individual algorithms or small subsets, without the comprehensive multi-metric evaluation presented here [18, 19].

Contribution 2: Demonstration and explanation of the VSS-LMS resource efficiency paradox: VSS-LMS achieves fewer LUTs than LMS despite greater arithmetic complexity, due to Vivado DSP48E1 cascade inference of the $\mu \cdot e \cdot x$ chain. This finding provides a new design insight for FPGA implementation of variable step size adaptive filters [22, 23].

Contribution 3: Detailed timing failure analysis of NLMS and RLS at 100 MHz, providing quantitative evidence (WNS = -45.382 ns and -45.757 ns respectively, 16 failing endpoints each) and identifying the specific circuit element (sequential divider) responsible. This analysis provides concrete guidance for future architectural redesign efforts [9, 20, 21].

Contribution 4: Complete Verilog RTL implementation and testbench verification of all six adaptive filter algorithms in 16-bit Q15 fixed-point arithmetic, with all synthesis and implementation results obtained from actual Vivado tool runs rather than analytical estimates.

7.4 FUTURE WORK

The research presented in this thesis opens several directions for future investigation and development, outlined below.

7.4.1 Pipelined NLMS and RLS for 100 MHz Timing Closure

The most immediate extension of this work is the architectural redesign of NLMS and RLS to achieve 100 MHz timing closure. For NLMS, two approaches merit investigation: (i) an iterative non-restoring divider pipelined across multiple clock cycles, and (ii) a Goldschmidt reciprocal approximation that replaces the exact division with a convergent iterative multiplication sequence [21, 24]. For RLS, a systolic array decomposition of the P-matrix update would distribute the $O(N^2)$ computation across multiple pipelined processing elements, each operating within a single clock period [20]. Achieving 100 MHz timing closure for both algorithms would enable a fair comparison of all six algorithms in true real-time hardware operation.

7.4.2 Multi-Tap Broadband Noise Cancellation

The present work uses a first-order filter ($N = 1$) optimized for single-tone 50 Hz noise cancellation. Real-world acoustic noise environments typically contain broadband noise components that require higher-order filters ($N = 16$ to 64 taps) for effective cancellation [23, 30]. Extending all six Verilog implementations to multi-tap configurations would require a FIR filter core with pipelined multiply-accumulate chains and address the increased resource utilization and timing challenges that arise at higher filter orders. The VSS-LMS architecture is particularly well-suited for multi-tap extension due to its DSP48E1 cascade structure.

7.4.3 Real-Time Hardware Validation with Audio Codec

The present FPGA implementation has been validated through Vivado behavioural simulation. A critical next step is real-time hardware-in-the-loop validation using an audio codec (such as the Analog Devices ADAU1761 or Texas Instruments PCM3168A)

connected to the Kintex-7 board via I2S interface. This would enable processing of actual speech signals with real microphone inputs, providing quantitative SNR measurements from physical hardware that can be directly compared with simulation results [26, 27].

7.4.4 Custom ASIC Implementation

An ASIC implementation of the VSS-LMS architecture would enable further optimizations in terms of power consumption, area, and operating frequency that are not achievable on FPGA due to the overhead of configurable interconnects and LUT-based logic. Custom ASIC synthesis using a standard-cell library (e.g., TSMC 28 nm or 65 nm) would allow full custom optimization of the DSP48-equivalent multiply-accumulate cells, potentially achieving sub-0.05 W power consumption and operating frequencies exceeding 1 GHz [34].

7.4.5 Extension to Active Noise Control with FxLMS

The present work addresses passive adaptive noise cancellation. Extending the implementation to active noise control (ANC) using the Filtered-x LMS (FxLMS) algorithm would require modelling the another path (loudspeaker-to-microphone transfer function) and incorporating it into the weight update equation [31, 32]. FxLMS is the standard algorithm for ANC in headphones, duct noise control, and automotive cabin noise reduction. A VSS variant of FxLMS combining the variable step size adaptation of VSS-LMS with the secondary path compensation of FxLMS would be a natural and practically relevant extension [30].

7.4.6 Multi-Channel Parallel FPGA Implementation

For hearing aids and multi-microphone speech enhancement applications, multiple adaptive filter channels must operate simultaneously on a single FPGA. The Sign-LMS and VSS-Sign-LMS algorithms — with their low DSP utilization (0 and 2 DSP48 slices respectively) — are particularly well-suited for multi-channel parallel implementation where DSP48 resources are shared or replicated across channels [33]. A comparative evaluation of multi-channel throughput for all six algorithms on Kintex-7 would provide practical guidelines for hearing aid FPGA design.

7.4.7 Low-Power Optimization for Embedded Applications

Further power optimization of the VSS-LMS implementation can be explored through clock gating, operand isolation, and dynamic voltage and frequency scaling (DVFS) techniques available on modern FPGAs [34]. Replacing the Kintex-7 platform with a UltraScale+ device would enable access to advanced DSP58E2 slices and low-power fabric options. Additionally, approximate computing techniques applied to the step size update computation could further reduce power with minimal SNR degradation, targeting energy-critical wearable and implantable medical device applications.

7.5 SOCIAL IMPACT

The adaptive filtering algorithms implemented and evaluated in this thesis have social significance in many ways:

Healthcare: VSS-LMS and Sign-LMS implementations are beneficial for persons with hearing loss by enhancing the usage of hearing aids, and for improved clean ECG/EEG signal recording in medical applications.

Industrial Safety: The real-time implementation of 50 Hz power-line cancellation is of benefit to sensor networks in an industrial environment and aids the correct measurement readings for equipment monitoring safety-critical systems.

Telecommunications: The noise cancellation hardware structure implemented on an FPGA can provide clearer speech in hands-free devices used by mobile phone users.

Energy Efficiency: Sign-LMS is beneficial for IoT and battery-powered edge devices as the implemented algorithm's power consumption was very low (0.134 W, 0 DSP48), contributing to energy-efficient smart system development.

7.6 FINAL REMARKS

Through extensive MATLAB simulation and real FPGA implementation, the VSS-LMS adaptive filter algorithm has been shown to provide superior overall performance in real-time speech noise cancellation on FPGA. VSS-LMS was found to simultaneously achieve both the best SNR (27.02 dB), use the least hardware resources (132 LUTs), successfully meet timing ($\text{WNS} = +4.686 \text{ ns @ } 100 \text{ MHz}$), and achieve respectable power consumption (0.137 W), making it the obvious choice among the six algorithms presented. In fact, the novel insight into using more algorithmic complexity in VSS-LMS to decrease hardware resource consumption through DSP48E1 cascade inference is significant to both adaptive filter design and FPGA based DSP applications.

The versatile multi-algorithm comparison framework, signal models, Verilog implementation, and performance evaluation methodology established in this thesis can serve as a reproducible foundation for continued work on FPGA based adaptive filtering. The five potential research opportunities mentioned in Section 7.4 including pipelined NLM-S/RLS, multi-tap generalization, real-time hardware verification, ASIC implementation, and an FxLMS generalization form the basis of a future research road-map.

REFERENCES

- [1] S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1985.
- [2] L. Jiang, K. Wiklund, and S. Haykin, "A simulink laboratory package for teaching adaptive filtering concepts," *International Journal of Engineering Education*, vol. 21, no. 4, p. 572, 2005.
- [3] N. Zhang, T. Le, and S. Haghani, "Design and implementation of fpga based lms self-adjusting adaptive filtering system for audio signal processing," *WSEAS Transactions on Signal Processing*, vol. 13, pp. 256–263, 2017.
- [4] B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *Neurocomputing: Foundations of Research* (J. A. Anderson and E. Rosenfeld, eds.), pp. 123–134, Cambridge, MA, USA: MIT Press, 1988.
- [5] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1985.
- [6] S. S. Haykin, *Adaptive Filter Theory*. Upper Saddle River, NJ, USA: Prentice-Hall, 4th ed., 2002.
- [7] E. Eweda, "Comparison of rls, lms, and sign algorithms for tracking randomly time-varying channels," *IEEE Transactions on Signal Processing*, vol. 42, pp. 2937–2944, Nov. 1994.
- [8] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Electronics and Communications in Japan (Part I: Communications)*, vol. 67, no. 5, pp. 19–27, 1984.
- [9] E. Szopos, H. Hedesiu, V. Popescu, and L. Festila, "Comparison of lms algorithm derivatives using labview fpga," in *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE)*, (Cambridge, UK), pp. 1–6, 2008.
- [10] S. Kavitha, G. Sinduja, M. Srimathi, and K. Yogalakshmi, "An efficient fpga implementation of the multiplier-less lms adaptive filter," in *Proceedings of the 7th International Conference on Computing Methodologies and Communication (ICCMC)*, (Erode, India), pp. 441–445, 2023.
- [11] R. H. Kwong and E. W. Johnston, "A variable step size lms algorithm," *IEEE Transactions on Signal Processing*, vol. 40, pp. 1633–1642, July 1992.
- [12] S. Baghel and R. Shaik, "Fpga implementation of fast block lms adaptive filter using distributed arithmetic for high throughput," in *Proceedings of the International Conference on Communications and Signal Processing (ICCSP)*, (Calicut, India), pp. 443–447, 2011.
- [13] R. H. Kwong and E. W. Johnston, "A variable step size lms algorithm," *IEEE Transactions on signal processing*, vol. 40, no. 7, pp. 1633–1642, 1992.

- [14] T. J. Shan and T. Kailath, "Adaptive algorithms with an automatic gain control feature," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 122–127, Jan. 1988.
- [15] V. J. Mathews and Z. Xie, "A stochastic gradient adaptive filter with gradient adaptive step size," *IEEE Transactions on Signal Processing*, vol. 41, pp. 2075–2087, June 1993.
- [16] M. Salah, M. Dessouky, and B. Abdelhamid, "Design and implementation of an improved variable step-size nlms-based algorithm for acoustic noise cancellation," *Circuits, Systems, and Signal Processing*, vol. 41, pp. 551–578, Jan. 2022.
- [17] W. Zhao and S. C. Chan, "Design and implementation of a variable step-size transform domain nlms adaptive filter," in *Proceedings of the TENCON 2015 – IEEE Region 10 Conference*, (Macao, China), pp. 1–3, 2015.
- [18] H. Ariyadoost, Y. S. Kavian, and K. Ansari-As, "Performance evaluation of lms and dlms digital adaptive fir filters by realization on fpga," *International Journal of Science & Emerging Technologies*, vol. 1, no. 1, pp. 7–10, 2011.
- [19] S. M. Kuo and D. R. Morgan, "Active noise control: a tutorial review," *Proceedings of the IEEE*, vol. 87, pp. 943–973, June 1999.
- [20] A. Shiva, E. Senthilkumar, J. Manikandan, and V. K. Agrawal, "Fpga implementation of reconfigurable adaptive filters," in *Proceedings of the International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, (Chennai, India), pp. 2544–2547, 2017.
- [21] C. A. Parmar, B. Ramanadham, and A. D. Darji, "Fpga implementation of hardware efficient adaptive filter robust to impulsive noise," *IET Computers & Digital Techniques*, vol. 11, pp. 107–116, May 2017.
- [22] Xilinx Inc., *7 Series DSP48E1 Slice User Guide (UG479)*. Xilinx Inc., San Jose, CA, USA, v1.10 ed., 2018. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf.
- [23] Xilinx Inc., *Vivado Design Suite User Guide: Synthesis (UG901)*. Xilinx Inc., San Jose, CA, USA, v2023.2 ed., 2023. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2023_2/ug901-vivado-synthesis.pdf.
- [24] P. H. Prajapati and A. D. Darji, "Fpga implementation of mrmn with step-size scaler adaptive filter for impulsive noise reduction," *Circuits, Systems, and Signal Processing*, vol. 39, pp. 3682–3710, July 2020.
- [25] H. H. Thannoon and I. A. Hashim, "Fpga design of a high-speed adaptive filter using a combination of rdls and rdza-lms," in *AIP Conference Proceedings*, vol. 3169, p. 040077, AIP Publishing LLC, 2025.
- [26] S. Thilagam and P. Karthigaikumar, "Implementation of adaptive noise canceller using fpga for real-time applications," in *Proceedings of the 2nd International Conference on Electronics and Communication Systems (ICECS)*, (Coimbatore, India), pp. 1711–1714, 2015.
- [27] D. P. Mandic and J. A. Chambers, "Normalized lms algorithms," in *Recurrent Neural Networks for Prediction*, ch. 3, pp. 45–68, Chichester, UK: Wiley, 2001.

- [28] C. Paleologu, J. Benesty, and S. Ciochina, "A robust variable forgetting factor recursive least-squares algorithm for system identification," *IEEE Signal Processing Letters*, vol. 15, pp. 597–600, 2008.
- [29] R. Ramos, A. M^anuel-L^azaro, J. D. R^oo, and G. Olivar, "Fpga-based implementation of an adaptive canceller for 50/60-hz interference in electrocardiography," *IEEE Transactions on Instrumentation and Measurement*, vol. 56, pp. 2633–2640, Dec. 2007.
- [30] J. Yuan, C. Yuan, J. Li, Q. Li, and W. Wang, "Research on active noise control system based on fbfulms algorithm," in *Proceedings of the IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, (Chongqing, China), pp. 1384–1389, 2022.
- [31] D. Shi, C. Shi, and W.-S. Gan, "A systolic fxlms structure for implementation of feed-forward active noise control on fpga," in *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, (Jeju, South Korea), pp. 1–6, 2016.
- [32] A. Klemm, M. Eckert, B. Klauer, J. Hanselka, and J. D. Sachau, "A parameterizable feedback fxlms architecture for fpga platforms," in *Proceedings of the 10th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART)*, (Nagasaki, Japan), pp. 1–4, 2019.
- [33] W. Shenming, W. Suntiamorntut, N. Jindapetch, and J. Qiufan, "Scheduling and resources sharing technique for adaptive lms filter," in *Proceedings of the 8th ECTI Association of Thailand Conference (ECTI-CON)*, (Khon Kaen, Thailand), pp. 114–117, 2011.
- [34] K. Prajna and Charishma, "Implementation of an adaptive filter robust to impulsive noise," in *Proceedings of the International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, (Coimbatore, India), pp. 1–4, 2017.
- [35] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ, USA: Wiley-IEEE Press, 2003.
- [36] Y. Huang, J. Benesty, and J. Chen, "A variable step-size lms adaptive filter for fpga-based echo cancellation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (Taipei, Taiwan), pp. 245–248, 2009.
- [37] D. I. Pazaitis and A. G. Constantinides, "A novel kurtosis driven variable step-size adaptive lms algorithm," *IEEE Transactions on Signal Processing*, vol. 47, pp. 864–872, Mar. 1999.
- [38] J. Zhang, A. T. Erdogan, and T. Arslan, "Fpga implementation of variable step-size lms adaptive filter for biomedical signal processing," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, (Seville, Spain), pp. 1425–1428, 2012.
- [39] C. Venkatesan, P. Karthigaikumar, and R. Varatharajan, "Fpga implementation of modified error normalized lms adaptive filter for ecg noise removal," *Cluster Computing*, vol. 22, no. Suppl 5, pp. 12233–12241, 2019.
- [40] D. P. Mandic and J. A. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. Chichester, UK: John Wiley & Sons, 2001.
- [41] J. Benesty, M. M. Sondhi, and Y. Huang, eds., *Springer Handbook of Speech Processing*. Berlin, Germany: Springer, 2008.

PLAGIARISM VERIFICATION

thesis

ORIGINALITY REPORT

7%

SIMILARITY INDEX

5%

INTERNET SOURCES

4%

PUBLICATIONS

4%

STUDENT PAPERS

PRIMARY SOURCES

1

Submitted to University Of Tasmania

Student Paper

<1 %

2

abrarhashmi.files.wordpress.com

Internet Source

<1 %

3

epdf.tips

Internet Source

<1 %

4

Submitted to University of Sheffield

Student Paper

<1 %

5

Submitted to University of Wollongong

Student Paper

<1 %

6

Submitted to Delhi Technological University

Student Paper

<1 %

7

ebin.pub

Internet Source

<1 %

8

www.mdpi.com

Internet Source

<1 %

9

www.notesale.co.uk

Internet Source

<1 %

10

eprints.usm.my

Internet Source

<1 %

11

Kuo. "Adaptive Filtering", Real-Time Digital Signal Processing, 04/04/2006

Publication

<1 %

12

www.ijert.org

Internet Source

<1 %

13

Submitted to Chester College of Higher Education

Student Paper

<1 %

14

archive.org

Internet Source

<1 %

15

www.iariajournals.org

Internet Source

<1 %

16

www.teses.usp.br

Internet Source

<1 %

17

William Hohl, Christopher Hinds, Kevin Welton. "ARM Assembly Language - Fundamentals and Techniques", CRC Press, 2025

Publication

<1 %

18

www.technoarete.org

Internet Source

<1 %

19

inria.hal.science

Internet Source

<1 %

turnitin

Page 1 of 62 - Cover Page

Submission ID: 1079896021

An H

thesis

Quick Submit

Quick Submit

Delhi Technological University

Document Details

Submission ID: 1079896021

Submission Date: May 27, 2026, 6:41 AM GMT+5:30

Download Date: May 27, 2026, 12:44 PM GMT+5:30

File Name: Ankeoh_Final_Thesis.pdf

File Size: 10.9 MB

60 Pages

16,751 Words

126,991 Characters

turnitin

Page 1 of 62 - Cover Page

Submission ID: 1079896021

turnitin

Page 2 of 62 - AI Writing Overview

Submission ID: 1079896021

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI-generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.
It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin AI detection capabilities before using the tool.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be generated by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

turnitin

Page 2 of 62 - AI Writing Overview

Submission ID: 1079896021



DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daultpur, Main Bawana Road, Delhi-110042.
India

PLAGIARISM VERIFICATION

Title of the Thesis

Comparative Study of FPGA-Based Performance Evaluation of LMS, NLMS, and RLS Adaptive Filters for Noise Cancellation

Total Pages **45**

Name of the Scholar **Ankesh Singh**

Supervisor **Prof. Dinesh Kumar**

Department of Electronics & Communication Engineering

This is to report that the above thesis was scanned for similarity detection.
Process and outcome is given below:

Software used: **Turnitin** Similarity Index: **7%** Total Word Count:
16,143

Date: **10.06.2026**

Candidate's Signature

Signature of Supervisor(s)



DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-110042.

India

CURRICULUM VITAE

Name	Ankesh Singh
Roll Number	2K24/SPD/07
Programme	Master of Technology
Specialization	Signal Processing and Digital Design
Department	Electronics and Communication Engineering
University	Delhi Technological University
Supervisor	Prof. Dinesh Kumar
E-mail	mrankeshsingh05@gmail.com

Educational Qualifications

Degree	Institution	Specialization	Year
M.Tech	Delhi Technological University, Delhi	Signal Processing and Digital Design	2026
B.Tech	U.I.E.T. Chhatrapati Shahu Ji Maharaj University, Kanpur	Electronics and Communication Engineering	2024

Research Interests

- RTL design in Verilog/SystemVerilog and VLSI design flow including synthesis, static timing analysis, and power estimation
- FPGA implementation and hardware optimization of digital signal processing algorithms
- Fixed-point arithmetic design and DSP48E1 resource optimization for embedded signal processing systems
- Adaptive noise cancellation for speech and audio applications
- Hardware–software co-design for real-time signal processing on reconfigurable computing platforms