

Realization of Delayed Least Mean Square Adaptive algorithm using Verilog HDL for EEG signals

D M Ram Prakash



Department of Electronics and Communication Engineering
National Institute of Technology Rourkela

Realization of Delayed Least Mean Square Adaptive algorithm using Verilog HDL for EEG signals

Thesis submitted in partial fulfillment

of the requirements of the degree of

Master of Technology

in

Electronics and Communication Engineering

(Specialization: VLSI Design and Embedded systems)

by

D M Ram Prakash

(Roll Number: 214EC2164)

based on research carried out

under the supervision of

Prof. Upendra Kumar Sahoo



May, 2016

Department of Electronics and Communication Engineering
National Institute of Technology Rourkela



Department of Electronics and Communication Engineering
National Institute of Technology Rourkela

Prof. Upendra Kumar Sahoo

Assistant Professor

May 30, 2016

Supervisor's Certificate

This is to certify that the work presented in the thesis entitled *Realization of Delayed Least Mean Square Adaptive algorithm using Verilog HDL for EEG signals* submitted by *D M Ram Prakash*, Roll Number 214EC2164, is a record of original research carried out by him under my supervision and guidance in partial fulfillment of the requirements of the degree of *Master of Technology in Electronics and Communication Engineering*. Neither this thesis nor any part of it has been submitted earlier for any degree or diploma to any institute or university in India or abroad.

Prof. Upendra Kumar Sahoo

Dedication

*This thesis is dedicated to the memory of
my father;
R Mani,
who inspired me to achieve my goals
and providing me
unflagging patience and support.*

D M Ram Prakash

Declaration of Originality

I, *D M Ram Prakash*, Roll Number *214EC2164* hereby declare that this thesis entitled *Realization of Delayed Least Mean Square Adaptive algorithm using Verilog HDL for EEG signals* presents my original work carried out as a postgraduate student of NIT Rourkela and, to the best of my knowledge, contains no material previously published or written by another person, nor any material presented by me for the award of any degree or diploma of NIT Rourkela or any other institution. Any contribution made to this research by others, with whom I have worked at NIT Rourkela or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the section “Reference”. I have also submitted my original research records to the scrutiny committee for evaluation of my dissertation.

I am fully aware that in case of any non-compliance detected in future, the Senate of NIT Rourkela may withdraw the degree awarded to me on the basis of the present dissertation.

May 30, 2016
NIT Rourkela

D M Ram Prakash

Acknowledgment

This research work is one of the significant achievements in my life and is made possible because of the unending encouragement and motivation given by so many people in every part of my life. It is an immense pleasure to have this opportunity to express my gratitude and regards to them.

First and foremost, I would like to express my profoundest gratitude and sincere thanks to **Prof. Upendra Kumar Sahoo**, Asst. Prof., in Department of Electronics and Communication Engineering, for his esteemed supervision, support and guidance during the tenure of my project work. His invaluable advices have motivated me a lot when I felt saturated in my work. His impartial feedback in every walk of the research has made me to approach a right way in excelling the work. I would also like to thank him for providing best facilities in the department.

I would like to express my gratitude and respect to Prof. K. K. Mahapatra, Prof. D. P. Acharya, Prof. A.K. Swain, Prof. S. Sarkar, and Prof. M.N. Islam, for their guidance and suggestions throughout the M.Tech course. I take this opportunity to express gratitude to all of the Department faculty members for their help and support. Lastly, I would like to express my heartfelt wishes to my friends and classmates whose company and support made me feel much better than what I am.

May 30, 2016
NIT Rourkela

D M Ram Prakash
214EC2164

Abstract

An efficient architecture for the implementation of delayed least mean square (DLMS) adaptive filter is presented in this paper. It is shown that the proposed architectures reduces the register complexity and also supports the faster convergence. Compared to transpose form, the direct form LMS adaptive filter has fast convergence but both has most similar critical path. Further it is shown that in most of the practical cases, very small adaptation delay is sufficient enough to implement a direct-form LMS adaptive filter where in normal cases a very high sampling rate is required and also it shows that no pipelining approach is necessary. From the above discussed estimations three different architectures of LMS adaptive filter has been designed. They are, first design comprise of zero delays i.e., with no adaptation delays, second design comprises of only single delay i.e., with only one adaptation delay, and lastly the third design comprises of two adaptation delays. Among all the three designs zero adaptation delay structure gives efficient performance comparatively. Design with zero adaptation delay involves the minimum energy per sample (EPS) and also minimum area compared to other two designs.

The aim of this thesis is to design an efficient filter structures to create a system-on-chip (SoC) solution by using an optimized code for solving various adaptive filtering problems in the system. In this thesis our main focus is on interference cancellation in electroencephalogram (EEG) applications by using the proposed filter structures. Modern field programmable gate arrays (FPGAs) have the resources that are required to design an effective adaptive filtering structures. The designs are evaluated in terms of design time, area and delays.

Contents

Supervisor’s Certificate	ii
Dedication	iii
Declaration of Originality	iv
Acknowledgment	v
Abstract	vi
List of Figures	ix
List of Tables	xi
List of Acronyms	xii
List of Acronyms	xii
1 Introduction	1
1.1 Purpose	1
1.2 Overview	1
1.3 Literature Survey:	3
1.4 Organisation of Thesis	5
2 Least Mean Square Algorithm and its Implementation	7
2.1 Background:	7
2.2 Least mean square algorithm	7
2.3 Adaptive Filter Performance Parameters:	9
2.4 Analyzing FIR adaptive filter:	10
2.4.1 Applications:	10
2.5 Interference cancellation:	11
2.6 Simulation Results and Discussion:	12

3	The Delayed LMS Algorithm and its Implementation	15
3.1	Background	15
3.2	Introduction	16
3.3	The Delayed LMS Algorithm	17
3.3.1	Advantages:	17
3.3.2	Applications:	17
3.4	Implementation of Direct-form Delayed LMS Algorithm:	17
3.4.1	Modules Separation:	19
3.4.2	Adder tree and Subtractor:	19
3.4.3	Multiplexer:	19
3.4.4	De-Multiplexer:	20
3.4.5	Registers:	20
3.5	Proposed Structure:	20
3.5.1	Implementation of Zero Adaptation Delay:	21
3.5.2	Implementation of One Adaptation Delay:	23
3.5.3	Implementation of Two Adaptation Delay:	23
3.6	Simulation Results and Discussion:	24
4	FPGA Based Implementation for EEG Applications	29
4.1	EEG (Electro-Encephalogram):	29
4.2	Need for EEG Signal:	29
4.3	Artifacts in EEG Signal:	30
4.3.1	Motion Artifacts:	31
4.3.2	Sweat Artifact:	31
4.3.3	Power Line Interference:	31
4.3.4	ECG and EMG Artifacts:	31
4.3.5	Glossokinetic and Respiration Artifacts:	31
4.3.6	Electro-oculogram (EOG) Artifacts:	32
4.4	FPGA Implementation	32
4.5	Hardware Description Languages:	33
4.6	FPGA Design Flow:	33
4.7	Simulation Results and Discussion:	34
5	Conclusions	40
	References	41

List of Figures

2.1	Conventional LMS adaptive filter	8
2.2	Block diagram of FIR Adaptive filter.	10
2.3	Block diagram of Interference Cancellation concept.	11
2.4	Matlab Simulation of LMS adaptive filter.	12
2.5	Matlab Simulation of comparison between LMS and RLS adaptive filter. . .	13
2.6	Simulation of 4-tap FIR filter.	13
2.7	Matlab Simulation for noise cancellation using LMS adaptive filter.	14
3.1	Example of multiply-add operation to study delay.	16
3.2	Block diagram of direct-form DLMS adaptive filter.	18
3.3	Weight update block.	18
3.4	Error computational block.	18
3.5	Structure of Adder tree.	19
3.6	Structure of 2:1 Multiplexer.	20
3.7	Structure of 1:2 De-Multiplexer.	20
3.8	Structure for Zero adaptation delay time multiplexed direct-form LMS adaptive filter.	21
3.9	Structure for Two adaptation delay direct-form LMS adaptive filter.	23
3.10	RTL Schematic view of Zero-Adaptation-Delay.	24
3.11	Simulation of Zero-Adaptation-Delay.	25
3.12	RTL Schematic of One-Adaptation-Delay.	25
3.13	RTL Schematic inner view of One-Adaptation-Delay.	26
3.14	Simulation Result of One-Adaptation-Delay.	26
3.15	RTL Schematic of Two-Adaptation-Delay.	27
3.16	Simulation Result of Two-Adaptation-Delay.	27
4.1	Block diagram of EEG.	30
4.2	Block diagram of EEG Unit	32
4.3	SRAM based FPGA configuration.	33
4.4	FPGA Design Flow	34
4.5	RTL Schematic inner view of Two-Adaptation-Delay for EEG.	35
4.6	Simulation Results of Two-Adaptation-Delay for EEG signal.	35

4.7	RTL Schematic of Zero-Adaptation-Delay for EEG.	36
4.8	Simulation Results of Zero-Adaptation-Delay for EEG.	36
4.9	RTL Schematic of One-Adaptation-Delay for EEG.	37
4.10	Simulation Results of One-Adaptation-Delay for EEG.	37
4.11	ISE iMPACT window while FPGA implementation on SPARTAN 3E Board.	39

List of Tables

3.1	Summary of Zero-Adaptation-Delay	24
3.2	Summary of One-Adaptation-Delay	28
3.3	Summary of Two-Adaptation-Delay	28
4.1	Device Utilization Summary of Zero-Adaptation-Delay for EEG.	38
4.2	Device Utilization Summary of One-Adaptation-Delay for EEG.	38
4.3	Device Utilization Summary of Two-Adaptation-Delay for EEG.	38
4.4	Comparison Table for three different adaptive filtering techniques.	39

List of Acronyms

ASIC	Application Specific Integrated Circuit
CLB	Configurable Logic Block
DLMS	Delayed Least Mean Square
DSP	Digital Signal Processing
EEG	Electro - Encephalogram
ECG	Electro - Cardiogram
EMG	Electromyogram
EOG	Electro - oculogram
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
LMS	Least Mean Square
LUT	Look up Table
MSE	Mean Square Error
PROM	Programmable Read Only Memory
RLS	Recursive Least Square
RTL	Register Transfer Level
SoC	System - on - Chip
SRAM	Static Random Access Memory
VLSI	Very Large Scale Integration
UCF	User Constraints File

Chapter 1

Introduction

Performance is time and again restricted by the processing capabilities of the system that perform real-time processing of data. Therefore, it is an important task to evaluate different architectures to determine the most effective architectures. This chapter discusses the main purpose of the thesis, and gives a direction and an overview of the flow of thesis.

1.1 Purpose

Modern Field Programmable Gate Arrays (FPGAs) offers so much to explore the use of embedded system-on-chip (SoC) solutions. In this thesis, it will explore the use of modern FPGAs in effectively implementing the adaptive filtering applications. In this process many different technics for the adaptive filtering are examined and compared with their results to know the best out of it.

1.2 Overview

In signal processing, a finite impulse response (FIR) filter is a filter whose response to any finite length input (or the impulse response) is of finite duration. This is because, in finite time it settles to zero. Also the impulse response of an N th-Order discrete-time FIR filter keeps going precisely $N + 1$ sample initially before it goes and settles to zero [1]. The FIR filters can be continuous-time or it can be discrete-time, and analog or digital. This requires no feedback. It is understood that by summed cycles any adjusting errors are not exacerbated. Similar relative error happens in every count. This will contribute the execution to happen with less difficulty. Subsequent to the output is a whole of a finite number of finite products of the input, then this can be no more prominent than times the biggest value showing up in the input.

It can be done without much of a stretch be intended to be direct stage by grouping coefficient in systematic way. This property is now and again utilized for applications like phase-sensitive, for instance crossover filters, mastering and also data communications. In a very few digital signal processing (DSP) regions there are wide applications for Adaptive

digital filters. For example, echo and noise cancellation, channel equalization, channel estimation, and system identification. The simplest known adaptive filter which is famous for its process like where its weights are updated by the very well-known Widrow Hoff Least mean square (LMS) algorithm is called as tapped-delay-line finite-impulse-response (FIR) filter [2]. The LMS adaptive filter is well known because of its low-complexity, as well as because of its soundness and attractive meeting execution. Because of its few vital uses of current significance and expanding imperatives on region, time complexity, and also power complexity [3]. To execute the LMS calculation, one needs to update the weights of the filter amid every testing period utilizing the assessed error, where it rises to the distinction between the desired response and the present output of the filter.

In every cycle all the weights are updated simultaneously to calculate the output as per, the direct-form realization of the FIR filter is a characteristic possibility for execution. Notwithstanding, the direct-form LMS adaptive filter is regularly accepted to have a very long basic way because of an inward item calculation to acquire the output of the filter. This is mostly in view of the suspicion that a number juggling operation begins when the complete input operand words are accessible/created. Let us consider an instance, in the current writing on execution of LMS adaptive filters, it is accepted that the addition in a multiply-add operation can continue strictly when finish of the multiplication. Considering this presumption, the basic way of the multiply-add operation gets to be the time required for an addition and, a multiplication separately.

Sample period required in numerous reasonable circumstances, and requires a lessening of basic way delay by pipelined execution. In any case, the conventional LMS calculation does not help in pipelined execution. In this way, it is adjusted to a structure known as delayed LMS algorithm (DLMS), which permits pipelined execution of various areas of the adaptive filter [4], [5]. A few works have been accounted for in the writing in the course of the most recent a quarter century proficient usage of the DLMS algorithm. A fascinating systolic design has been proposed by Van and Feng [6], where they have utilized generally vast processing components (PEs) for accomplishing lower adjustment delay contrasted with different DLMS systolic structures with a basic way. And also a fine-grained pipelined configuration has been proposed by Yi et al [7]. for an adaptive filter in view of direct-form FIR filtering. In this they have utilized a completely pipelined parallel adder-tree execution of the considerable number of multiplications in the weight update way and error-calculation way to confine the basic way to a greater of one addition time.

In this engineering, it assists the high sampling frequency, however, includes substantial pipeline profundity, which has two antagonistic impacts. To start with, the register complexity, and subsequently the force dissemination, increments. Also, the adjustment delay increments and merging execution corrupts. Notwithstanding, in the accompanying talk, we set up that in such forceful pipelining is regularly not called for long time. Since the presumption that the number juggling operations begin strictly when the era of their

complete input operand words is not substantial for the execution of composite capacities in committed equipment. Such a suspicion can be legitimate if adders and multipliers are utilized as discrete segments, this is not the situation in FPGA and ASIC execution nowadays. Then again, we can expect that when the LSBs of the operands are accessible an arithmetic operation can begin.

1.3 Literature Survey:

- a. **Title:** The LMS algorithm with delayed coefficient adaptation

Authors: G. Long, F. Ling, and J. G. Proakis

Year: 2002

The delayed least mean square algorithm (DLMS) behavior has been studied in this paper [4]. For the stability and convergence of the above said algorithm, the step size plays a prominent role which is in the coefficient update. An upper headed for the progression size is inferred that guarantees the stability of the DLMS. Various fields has been studied here such as the delays effect on the convergence speed, and also the relationship between the convergence speed and step size is also studied. Computer simulations are used to study the analytical results.

Disadvantage:

- Only for simulations these analytical results are supported,
- Low stability of the DLMS.

- b. **Title:** A modular pipelined implementation of a delayed LMS transversal adaptive filter

Authors: M. D.Meyer and D. P. Agrawal

Year: 1990

In this paper the research has been carried out on the problem of effectively understanding a transversal adaptive filter [5]. A period moved form of the DLMS calculation is determined. Because of its request recursive nature, the rebuilt calculation is appropriate to parallel usage. Also, a new architecture known as pipelined systolic model has been introduced to implement the algorithm. The execution of the pipelined framework is broke down, and conditions for speedup are determined. The pipelined framework is able to do much more noteworthy throughput than existing ordinary least mean-square (LMS) executions, making it a decent possibility for ongoing applications where high sampling rates are required. Likewise, because of its high measured nature, the framework is effectively expandable.

Disadvantage:

- It requires high sampling rates.

- c. **Title:** A high-throughput DLMS adaptive algorithm

Authors: E. Mahfuz, C. Wang, and M. O. Ahmad

Year: 2005

This paper discuss about the effective implementation of high throughput DLMS algorithm [8]. Compared to LMS algorithm it experiences a slower convergence rate. Diverse renditions of the DLMS versatile calculation utilizing a change plan have been proposed to enhance the merging rate. This enhanced meeting was accomplished to the detriment of an expanded computational many-sided quality and a lower throughput rate than the first DLMS calculation. We propose another adjusted DLMS versatile calculation that, contrasted with the current transformation based DLMS calculation, gives a higher throughput rate to a comparative union rate. On the other hand, the proposed calculation gives a speedier merging to the similar throughput rate contrasted with that of transformation based DLMS calculation. In the above discussed two cases, the computational multifaceted nature of the proposed calculation is littler than the change based DLMS calculation. The estimated calculation utilizes the error from every phase of the versatile FIR channel autonomously to redesign the estimation of the comparing coefficient. Recreations outline the union execution of the new calculation. The execution of its design is assessed as far as computational many-sided quality, throughput, and inactivity. The proposed calculation gives a computational many-sided quality lower than that of the change and a superior throughput rate based DLMS calculation.

Disadvantage:

- Very high computational complexity for this design,
- Low throughput.

- d. **Title:** A High-Speed FIR Adaptive Filter Architecture using a Modified Delayed LMS Algorithm

Authors: Pramod K. Meher , Megha Maheshwari

Year: 2011

To achieve lower adaptation delay a modified DLMS adaptive algorithm has been proposed in this paper [9]. However, for this execution of adaptive filter a suitable pipelined architecture also has been proposed. This can be implemented for filter of order N by using weight update unit and also executed for calculating feedback error by using the inner product calculation unit. Here there are N parallel multiply accumulators in the weight update block. The main objective of this paper is to show that this modified algorithm can achieve less adaptation delay than the conventional algorithm. Even for higher order adaptive filters this modified DLMS algorithm can be used.

Disadvantage:

- Low Latency,
 - High ADP.
- e. **Title:** An Efficient Systolic Architecture for the DLMS Adaptive Filter and Its Applications

Authors: Lan-Da Van and Wu-Shiung Feng

Year: 2001

Based on an optimized tree level rule and a new tree systolic processing element (PE) a new model architecture has been proposed [6]. It is an efficient model for the delay LMS FIR filter. It is known as systolic architecture which is efficient in its process. The advantage of this model is that there is no need to lose the basic systolic array architecture properties during the process. And also a high convergence rate can be obtained for the same. This effective engineering amiable to VLSI execution outfits the same least basic time frame, with limited driving, neighborhood associations, and agreeable meeting at no additional range expense.

Disadvantage:

- In Real Time Design the implementation involves high cost,
- High Critical period.

1.4 Organisation of Thesis

Chapter 2

This chapter deals with the basics of incremental adaptive filter strategies for practical applications of the system. Adaptive filter algorithm such as least mean square (LMS) algorithm are discussed thoroughly in this chapter. Analyzes the adaptive filter performance parameters such as convergence speed, steady state error, stability, settling time and about learning curves which contribute to the overall performance of the adaptive filtering process. Also analyzes the adaptive FIR filter and discusses about its applications in various fields. Most importantly it discusses about the interference cancellation by using adaptive filter algorithms such as LMS algorithm. This chapter sums up with the simulation results which are simulated using Matlab tool and also Verilog implementation using Xilinx ISE 14.2 tool.

Chapter 3

This chapter explains the problems arising in the least mean square (LMS) and also suggests the best suitable solution to overcome the drawbacks of conventional LMS. The drift problems arising due to practical implementations of conventional LMS have been discussed

in this chapter. A modified algorithm is proposed with three different architecture such as zero adaptation delay, one adaptation delay, and two adaptation delay are explained in this chapter. Its advantages and applications are thoroughly explained. This chapter concludes with the simulation results for all the above discussed three different architectures.

Chapter 4

This chapter deals with the FPGA implementation of three proposed structure of the direct form delayed least mean square (DLMS) algorithm. It discusses about various artifacts that are present in the electroencephalogram (EEG) signals. It also explains the suitable approach to cancel the interference using the proposed adaptive filtering techniques. Finally, in this chapter a comparison is done among all the three proposed techniques and explain about the best suitable approach. This chapter sums up with the FPGA implementation for the proposed models for the application of interference cancellation in EEG signal. Simulation results and its discussions are done in this chapter.

Chapter 5

This chapter will draw conclusions of the thesis and discusses regarding the future work and any other extensions to the present work.

Chapter 2

Least Mean Square Algorithm and its Implementation

2.1 Background:

There are many different algorithms to update the weight coefficients of adaptive filter. Depending on the requirements and operating environment we chose the required algorithm. There are few parameters which we consider to before going for a particular algorithm. There are four different adaptive algorithms, they are the Least mean square (LMS), the Affine projection (AP), the Normalized Least mean square (NLMS), and the Recursive least square (RLS) algorithms [10]. However, all these algorithms eventually operate to minimize the error.

2.2 Least mean square algorithm

The LMS adaptive algorithm is very simple and most widely used algorithm. This LMS algorithm is much similar to that of steepest decent method. In the LMS algorithm it iteratively updates the coefficients of the filter. By this way it minimizes the Mean square error (MSE). Comparing LMS to other algorithms, no matrix operations are involved in this process. Because of its simplicity, the resources required for computation and also the storage are very less compared to other algorithms. The execution is very simple and very less complexity is involved [11]. Input signal, the step size parameter, and the error value are used by this LMS algorithm to repeatedly estimate the coefficients of the filter [12]. These are the operations that are performed by the LMS adaptive filter to update the coefficients of the filter,

- (a) From the adaptive filter it calculates the output signal $y(n)$ as follows,

$$y_n = W_n^T X_n \quad (2.1)$$

(b) The error signal $e(n)$ is calculated by using the following equation as,

$$e_n = d_n - y_n \quad (2.2)$$

(c) The coefficients of the filter is updated by following equation as,

$$W_{n+1} = W_n + 2\mu e_n X_n \quad (2.3)$$

Where with the weight vector W_n , and input vector X_n at the n th iteration is given by, respectively

$$W_n = [w_n(0), w_n(1), w_n(2), \dots, w_n(N-1)]^T \quad (2.4)$$

$$X_n = [x_n, x_{n-1}, x_{n-2}, \dots, x_{n-N+1}]^T \quad (2.5)$$

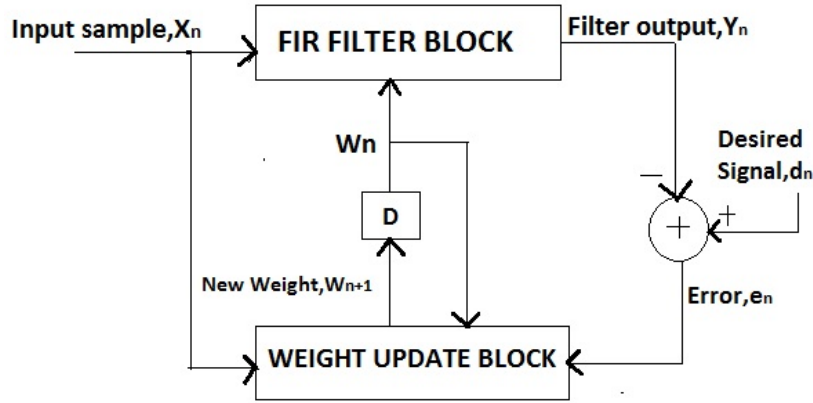


Figure 2.1: Conventional LMS adaptive filter

Where,

μ is the step size of adaptive filter,

$d(n)$ is the desired signal,

$e(n)$ is the error signal which estimates the difference between $d(n)$ and $y(n)$ and,

$y(n)$ is the filter output of the signal.

The adaptive filter performance parameters are to be analyzed before designing an adaptive filter to filter a signal. The convergence speed, settling time, steady state error, stability, and learning curve are the few performance parameters that ensures that the adaptive filter designed meets the requirements of the application.

2.3 Adaptive Filter Performance Parameters:

(a) Convergence Speed, Settling time:

Convergence speed is a very important factor before choosing the adaptive filter algorithm. Minimizing the power of the error signal is the functionality of adaptive filters. This can be achieved by repeatedly modifying the coefficients of the filter. This method of lessening the error signal power is known as convergence. This term convergence represents the time taken to estimate the coefficients of the filter to reduce the error signal power [13]. Fast convergence means that the filter took less time to estimate the coefficients of the filter while minimizing the power of the error [14]. One must take convergence speed into consideration while choosing the filter algorithm. By using the learning curves or the error signal of the filter we can estimate the convergence speed. The time period taken by the filter to converge is known as settling time. Settling time determines the convergence speed. If the settling time is very small then the convergence speed is very fast.

(b) Steady state error:

The stage where the filter converges and there are no changes to the coefficients of the filter is known as steady state. Due to the issues like optimization of filters and noise which is included along with signal, the error signals cannot be zero. This case is true even though the filter converges. Then this type of error is known as steady state error.

(c) Stability:

Stability of adaptive FIR filters always an issue due to its frequent adjustments of its coefficients. Whereas FIR filters are steady inherently. Adaptive FIR filters cannot be stable every time because it adjusts the coefficients of filter iteratively. Due to this process of iterative adjustments, the coefficients of filter can reach infinite. If this happens then we can say that the adaptive FIR filter is unstable [15].

(d) Learning curve:

To estimate the convergence speed, learning curves are used. Learning curve is the plot between Mean square error and number of iterations done. Usually error signal's amplitudes are displayed to calculate the convergence speed, but in addition to that learning curves are also used for the same purpose. Since it is the plot between MSE and number of iterations, evaluation of MSE of error signal is done by,

$$MSE = \frac{\sum_{k=0}^T e_n^2(k)}{T} \quad (2.6)$$

Where $e_n(k)$ is error signal at k th trail and T is number of times that a filter is executed. This figure below demonstrate the learning curve behavior with various number of

iterations,

2.4 Analyzing FIR adaptive filter:

In communication the most famous research topic is adaptive filtering. In general adaptive filters are widely used when a signal is corrupted by noise. Whenever noise corrupts the received signal, where both received and noise signals change consistently then the need of adaptive filter emerges. Adaptive filter is defined as the system which adjust the coefficients of the filter by its own with the help of optimizing algorithm. To calculate the error in noisy waves this filter are used. It includes change of channel parameter after some time [11]. It adjusts to change in signal attributes keeping in mind the end goal to reduce error. Adaptive filters doesn't have prior data of signals or know noise behaviors and also don't have consistent channel coefficients [16], [17]. If any interference occurs between noise and received signal or if it is useful for coefficient of filter to adapt to challenging conditions or if the unknown noise occupies the band completely then the use of adaptive filter are required.

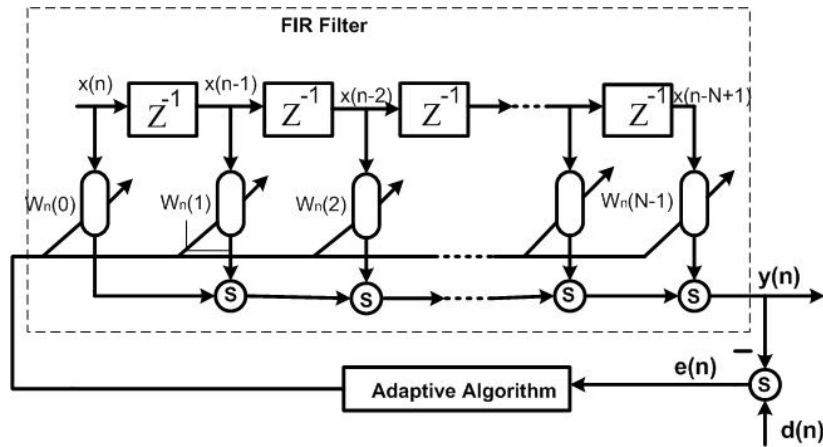


Figure 2.2: Block diagram of FIR Adaptive filter.

2.4.1 Applications:

The applications for FIR filters are as follows:

- Telecommunications,
- Echo cancellation,
- Data communications,
- Video processing,
- Wireless communications,

- Speech synthesis,
- High-speed modems,
- Filtering,
- Image enhancement,
- Digital Cameras,
- Video Processing,
- Digital Video Camcorders.

2.5 Interference cancellation:

The following figure (2.3) shows the basic problem also the solution for cancellation of adaptive noise present in the signal. A sensor receives a signal S that is transmitted through a proper channel, then along with signal S , a noise say n_0 which is uncorrelated with the input S is also transmitted to the sensor due to external resources. So the mixture of both the signals $S + n_0$ i.e., signal S and noise n_0 is the input given as the primary input to the canceler [18]. A noise n_1 which is correlated with the noise n_0 but uncorrelated with the signal S is received as input by the second sensor [19], [20]. This second sensor delivers the input received by it to the canceler as the reference input. This reference input which is fed into the filter is processed and produces a filtered output y_n which is very nearer to the estimated value of the noise n_0 . The filtered output y_n which is the output of the adaptive filter is subtracted from the primary input $S + n_0$ given to the first sensor. This gives the output of the system which is very close replica of the original signal S .

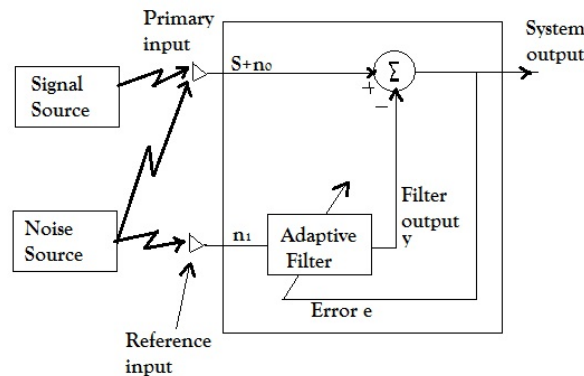


Figure 2.3: Block diagram of Interference Cancellation concept.

Theoretically it is possible to design a fixed filter only when the characteristics of the channels are known. The channels here refer to the channels over which the signal S is transmitted as primary input to the first sensor and also over which the noise n_1 is transmitted

as the reference input to the second sensor, it must be known to design a fixed filter model to cancel the interference. Then the signal which is obtained by the system as output by subtracting the reference input from the primary input is the required filtered signal. But here the characteristics of the channels are not known and the behavior is not a fixed one, so designing the fixed filter does not yield the optimum results [21], [22]. So, using an adaptive filter which adjusts its coefficients or the impulse responses automatically is optimum to design. The figure (2.3) shows that the reference input is processed by the adaptive filter. The adaptive algorithm does not change to sudden changes in the conditions and it adjusts iteratively by its own to reduce the error signal e .

In the Interference cancellation, the basic objective is to cancel the noise signal. The output of the system after cancelling noise that must be more suitable for the least square sense to the original signal S . This can be achieved by using a delayed LMS adaptive algorithm which iteratively adjusts the coefficients of the filter and thereby minimizing the error signal. As shown in the figure (2.3) of interference cancelling design, for the adaptive process the error signal is produced as the output of the system. In this process, a knowledge of noises n_0 and n_1 present in the system and also about the signal S has to be known in advance. This has to be known before designing the adaptive filter to produce the signal y which is used to cancel the noise in the system.

2.6 Simulation Results and Discussion:

Matlab simulations are done using Matlab R2012a version tool. FIR adaptive filter has been implemented in Verilog HDL and synthesis and simulation is done using Xilinx ISE 14.2.

Matlab Simulation of LMS adaptive filter:

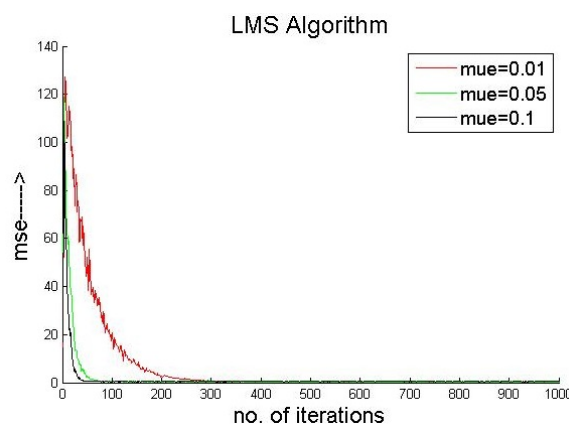


Figure 2.4: Matlab Simulation of LMS adaptive filter.

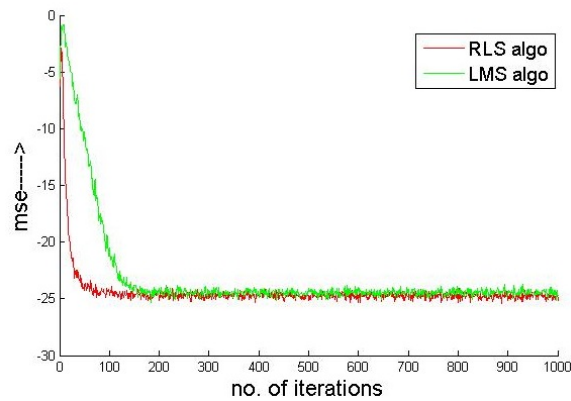
Matlab Simulation of comparison between LMS and RLS adaptive filter:

Figure 2.5: Matlab Simulation of comparison between LMS and RLS adaptive filter.

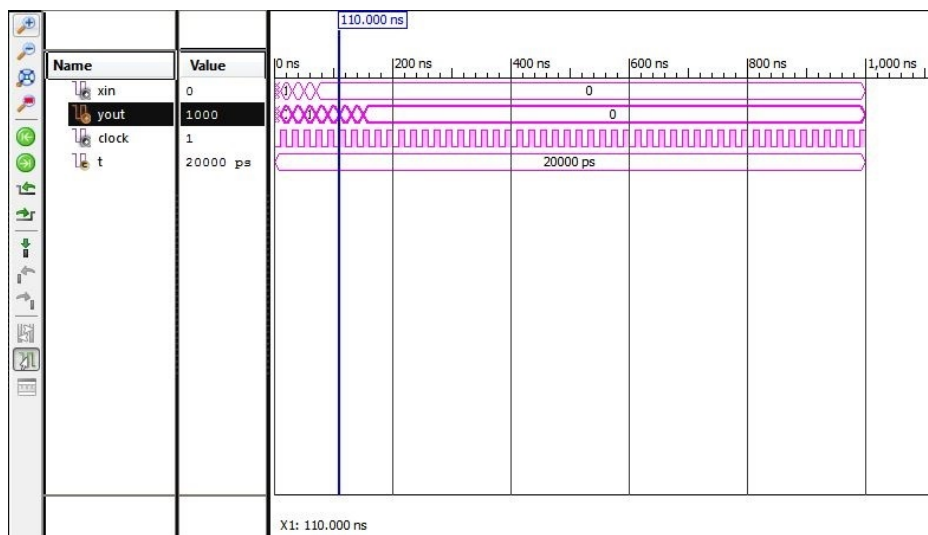
Simulation of 4-tap FIR filter:

Figure 2.6: Simulation of 4-tap FIR filter.

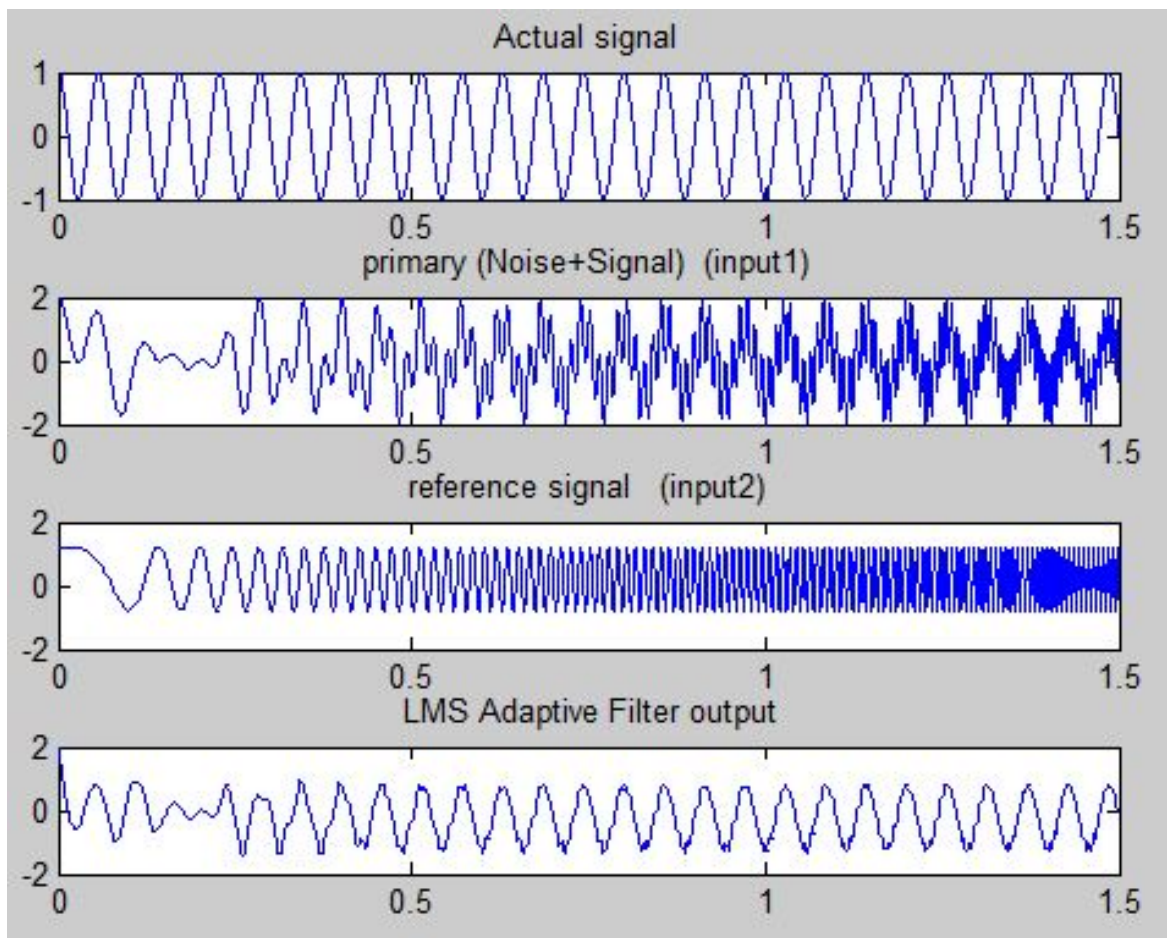
Matlab Simulation for noise cancellation using LMS adaptive filter:

Figure 2.7: Matlab Simulation for noise cancellation using LMS adaptive filter.

Chapter 3

The Delayed LMS Algorithm and its Implementation

3.1 Background

Further exertion has been made by Meher and Maheswari [9] to lessen the quantity of adjustment delays and in addition the basic way by an advanced usage of the internal item utilizing a bound together pipelined convey spare chain in the forward way. Meher and Park have proposed a 2-bit multiplication cell, and utilized that with an effective adder tree for the usage of pipelined inward item calculation to minimize the basic way and silicon territory without expanding the quantity of adaptation delays [23], [24]. Be that as it may, in these works, the basic way examination and fundamental outline contemplations are not considered. Because of that, the plans of still expend higher zone, which could be considerably lessened. Remembering the above perceptions, we display a systematic basic way investigation of the LMS adaptive filter, and taking into account that, we infer engineering for the LMS adaptive filter with negligible utilization of pipeline stages, which will bring about lower territory complexity and less power utilization without trading off the desired processing throughput.

Van and Feng have proposed a fascinating systolic design, where they have utilized generally extensive processing components (PEs) for accomplishing lower adjustment delay contrasted with different DLMS systolic structures with basic way of one MAC operation [6]. Yi et al. have proposed a fine-grained pipelined configuration of an adaptive filter in light of direct-shape FIR filtering, utilizing a completely pipelined parallel adder-tree execution of the considerable number of multiplications in the error-calculation way and weight update way to confine the basic way to a most extreme of one addition time [7]. This engineering bolsters high testing recurrence, yet includes vast pipeline profundity, which has two unfavorable impacts.

3.2 Introduction

In the first place, the register complexity, and consequently the force scattering, increments. Also, the adjustment delay increments and joining execution debases. Be that as it may, in the accompanying dialog, we set up that such forceful pipelining is regularly uncalled for, since the supposition that the number juggling operations begin simply after era of their complete input operand words is not legitimate for the usage of composite capacities in devoted equipment. Such a suspicion could be substantial when multipliers and adders are utilized as discrete segments, which is not the situation in ASIC and FPGA execution nowadays. Then again, we can expect that a number juggling operation can begin when the LSBs of the operands are accessible. In like manner, the engendering delay for the multiply-add operation in Fig. 3.1 could be taken to be the computational time of multiplier-add circuit is the sum of total path delay of multiplier, delay to produce the carry in a 1-bit full adder circuit, and propagation delay of sum generation in a 1-bit full-adder circuit. From this, we can likewise find that computational time of multiplier-add circuit is substantially less than sum of computational time of multiplier and computational time of adder.

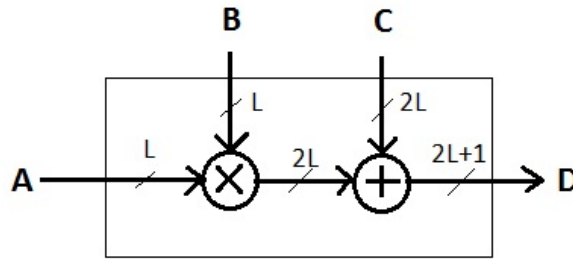


Figure 3.1: Example of multiply-add operation to study delay.

In addition, it has been demonstrated that no pipelining is required for executing the LMS calculation output, direct-form acknowledgment of the FIR filter is a characteristic contender for usage. Be that as it may, critical path is basically very long in the direct-form LMS adaptive filter because of an inward calculation to get the ouput of filter.

Under such presumption, the direct-form LMS adaptive filter which has long critical path will look at the union execution of LMS adaptive filters of various setups, we have reenacted the direct-form delayed LMS, and direct-form LMS, for the same system identification issue, where the system is characterized by utilizing the same reproduction arrangement. The expectations to absorb information accordingly got for filter length finding that the direct from LMS adaptive filter gives much quicker response than the transpose LMS adaptive filter in all cases [23]. The direct-form DLMS adaptive filter gives speedier meeting compared with the transpose-structure LMS adaptive filter even with additional 5 delays immediately. Be that as it may, the leftover mean-square error is observed to be about the same in all cases.

3.3 The Delayed LMS Algorithm

The finite impulse response (FIR) adaptive filter which is on LMS algorithm is very famous one because of its fast convergence performance and also of its simplicity. But this approach has a large critical path for the adaptive filtering input signal which has high sample-rate. This is due to its straight-forward implementation of least mean square adaptive filter. Therefore it is necessary to minimize the critical path. This can be done by using pipelining approach. But because of its recursive nature, the pipelining approach or implementation is not supported by the conventional least mean square (LMS) algorithm. Therefore, it is modified to a form which supports the pipelined implementation known as delayed least mean square algorithm (DLMS).

3.3.1 Advantages:

- Less area requirement,
- High speed,
- Less delay,

3.3.2 Applications:

- Signal processing and Digital communication applications,
- Software radio and Digital radio receivers,
- Down converters,

3.4 Implementation of Direct-form Delayed LMS Algorithm:

In the implementation of direct form delayed LMS algorithm, firstly it is presumed that there are m pipeline stages involved in the implementation of error computation block. So, here m cycles is the latency of that block. Then at n th cycle e_{n-m} is the error that is calculated by the structure. This error which is calculated at n th cycle is utilized along with the m cycles delayed input samples to produce the weight update term. For the DLMS algorithm the following gives the weight update equation which is as shown below,

$$W_{n+1} = W_n + 2\mu e_{n-m} X_{n-m} \quad (3.1)$$

The error is calculated by,

$$e_{n-m} = d_{n-m} - y_{n-m} \quad (3.2)$$

and

$$y_n = W_n^T X_n \quad (3.3)$$

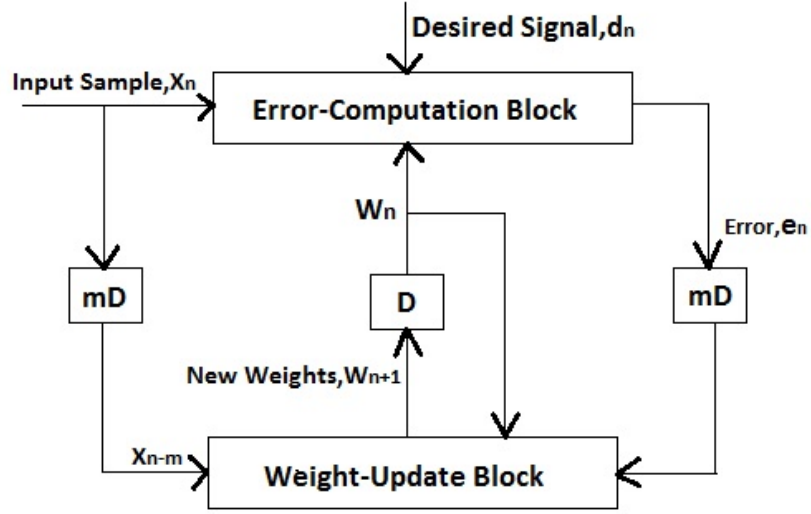


Figure 3.2: Block diagram of direct-form DLMS adaptive filter.

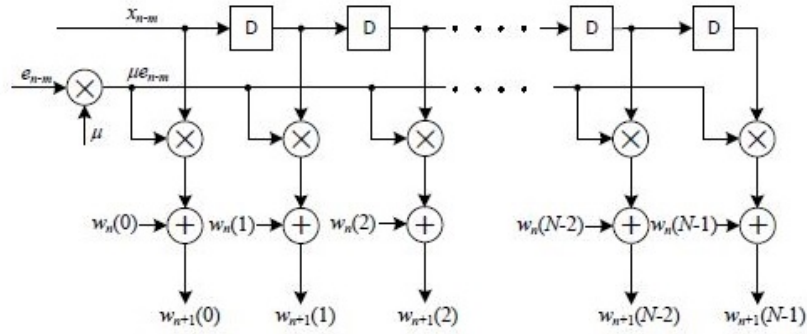


Figure 3.3: Weight update block.

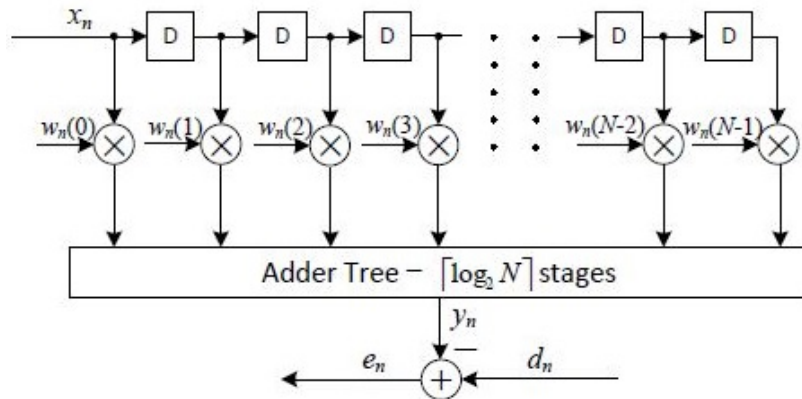


Figure 3.4: Error computational block.

The following figure (3.2) shows the summed up block diagram of the direct form DLMS adaptive filter. It has two computational blocks in its structure. They are weight update block

as shown in figure (3.3) and second is error computation block as shown in figure (3.4). As shown in the figure (3.1) in the block diagram, due to pipelining of error computation stage certain numbers of delays has been introduced. Here in this case m number of delays has been introduced which are also known as pipeline delays.

3.4.1 Modules Seperation:

- Error-computation,
- Weight-update block,
- Registers,
- Multiplexers and De-multiplexers,
- Zero-adaptation-delay LMS adaptive filter,
- One-adaptation-delay LMS adaptive filter,
- Two-adaptation-delay LMS adaptive filter.

3.4.2 Adder tree and Subtractor:

The adders which are used in weight update block for updating the weights and then the subtractor and adder-tree which are used in error computation block are different. Also the above said computing blocks takes very small part in the circuit design. In this implementation of zero adaptation delay, these two computational blocks are required to be computed in the same cycle.

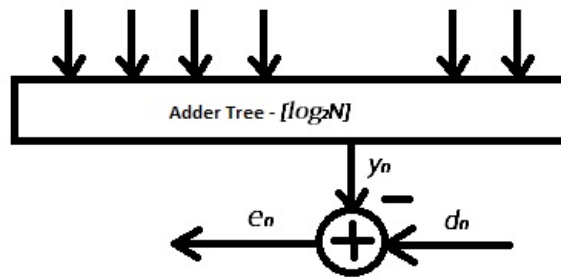


Figure 3.5: Structure of Adder tree.

3.4.3 Multiplexer:

Multiplexer is also known as mux. The operation of mux is defined as the process of choosing one of a few digital or analog input signal. This chosen signal is indeed forwarded to a single line. There are n select lines in a 2^n input multiplexer. These select lines are utilized to choose

which input line enter to give to the output. The primary purpose of these multiplexers is to maximize the data load that it is sent to the network keeping bandwidth and time constraints constant. Therefore, it is also known as data selector.

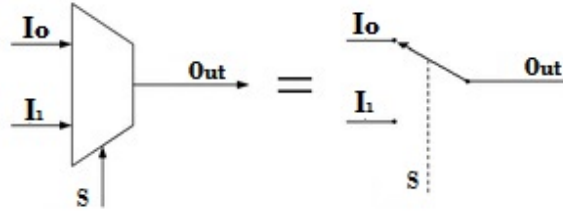


Figure 3.6: Structure of 2:1 Multiplexer.

3.4.4 De-Multiplexer:

De multiplexer is also known as demux. Unlike multiplexer, here in demux it takes a single input signal. Then it selects one of numerous output lines. All these are connected to a single input. On the receiving end a complementary de-multiplexer is utilized regularly by the multiplexer. A switch which acts as single input and multi-output is known as de-multiplexer.

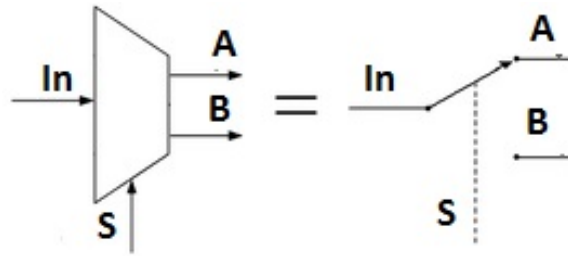


Figure 3.7: Structure of 1:2 De-Multiplexer.

3.4.5 Registers:

Larger than one bit of information can be stored using a register. Therefore, it is also known as memory device. The movement of data can be controlled to and from the register using a regular control signals. Therefore, these registers are normally acknowledged as a few flip-flops. These registers are also known as D registers because it comprises of D flip-flops.

3.5 Proposed Structure:

In pipelining approach, the error value e_n representing the n th iteration which is given as feedback to the adaptive filter is not available for updating the weights of the filter coefficients in the same cycle. After definite number of cycles it becomes available for

computing, the number of cycles here is called as the adaptation delays. Therefore, for updating the present weight the DLMS algorithm does not use the recent most error value instead uses the delayed error e_{n-m} which is corresponding to $(n - m)$ th iteration. Here m is the adaptation delay. The DLMS adaptive filter's weight update equation is given in the equation (3.1)

In this section, delay, power and area-efficient approaches are discussed for the implementation of direct-form LMS adaptive filters with zero, one, and two adaptation delays.

3.5.1 Implementation of Zero Adaptation Delay:

As shown in the below figure, in direct form LMS adaptive filter there are two most important computing blocks. They are (1) the error computation block which is shown in figure (3.4) and, (2) the weight update block which is shown in figure (3.3). It can be seen that most of the area intensive components in these two computational blocks (i.e., in both error computation block and weight update block shown in figure (3.4) and (3.3)) are common. The common components are the weight-registers, multipliers, tapped delay line. The adders which are used in weight update block for updating the weights and then the subtractor and adder-tree which are used in error computation block are different. Also the above said computing blocks takes very small part in the circuit design.

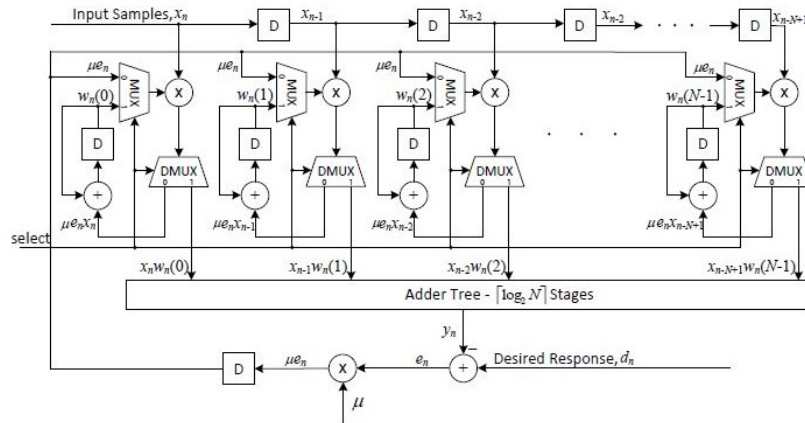


Figure 3.8: Structure for Zero adaptation delay time multiplexed direct-form LMS adaptive filter.

In this implementation of zero adaptation delay, these two computational blocks are required to be computed in the same cycle. The structure of this is non pipe lined design. Because of that, updating the weights and computing the error cannot be done simultaneously. In this way, the increases of both these stages could be multiplexed by the same arrangement of multipliers, while the same registers could be utilized for both these stages if computation of error is performed in the principal half cycle, while weight overhaul is performed in the second half cycle.

For zero adaptation delay the proposed model is shown in the below figure (3.8). This design is time multiplexed. This model is designed for a direct form N-tap LMS adaptive filter. It comprises of N multipliers. Firstly, through the multipliers the input samples are passed. These samples are sent from a regular tapped-delay line. The estimated error values after shifting it to right by a pre-determined number of locations and the weight values which are put away in registers are given to the multipliers as another input. This is done to acknowledge multiplication by the step size μ . This is done through a 2:1 multiplexer. The design requires an adder tree. This is because to sum the output of the multipliers for calculating the output of the filter. This design consists of N adders. These adders are required for alteration N weights. Additionally, a subtractor is necessary to process the error value. In this model N 2:1 de-multiplexers are used to send the result values to any one of either weight update or adder tree circuit. Clock signal is used to control the de-multiplexers and multiplexers.

At the rising edge of the clock pulse the registers in the deferral line are timed. Since the design take's one new test in each clock cycle, it will stay unaltered for a complete clock period. The values of the weight which are stored in various registers are sent to the multiplier. These are sent through the multiplexers to calculate the output of the filter. Then the product words are bolstered to adder-tree. This are sent through the de-multiplexers. The value of the error is calculated by a subtractor whereas output of the filter is calculated by the adder-tree. This calculated error value is then shifted towards right to get μe_n . This filter design needs at least one delay at an appropriate area to unchain the recursive circle. We can add this delay immediately after the adder-tree or after the calculation of error value e_n , or after the calculation of μe_n . Two cases are possible based on the placement of the delay in a structure. Firstly, suppose if the delay is put immediately after the adder-tree then the basic way moves to the weight-updating block. So it would be better to put delay after calculation of e_n or after the calculation of μe_n . Yet ideally placing delay after the calculation of μe_n minimizes width of the register.

The process of this computation is done in one complete cycle period, which further divided into two half-cycles. In this process, first half cycle of every clock period finishes by estimating the value of μe_n . Then the estimated value of μe_n is given as input to the multipliers in the second half cycle of the process. This is done through multiplexers to estimate the value of $\mu e_n x_n$. Then the de-multiplexers output is added to the put away weights to deliver new weights as indicated by equation (3.1). This is an iterative process where the second half cycle finishes its task once if its new weights are produced. As this is an iterative process this new weights are used as input in the next clock cycle's first half cycle to estimate the output of the filter and also for calculating value of error. At the point when the following cycle starts, the weight registers are additionally overhauled by the newly weighted values. In this manner, at the rising edge of every clock beat the weight registers are additionally timed.

Weight updating takes more time compared to time taken to compute error. It is ideal to do computation of error in first half cycle of the process and then followed by updating the weights in the next half cycle.

3.5.2 Implementation of One Adaptation Delay:

For a one-adaptation delay LMS adaptive filter the proposed structure is as shown in the below figure. Similar to the direct-form DLMS adaptive filter, one adaptation delay also has two computational blocks in its structure. They are weight update block as shown in figure (3.3) and second is error computation block as shown in figure (3.4). As shown in the figure, in this structure a pipeline latch has been added immediately after the calculation of μe_n . Because of μ is thought to be a power of 2 fraction, so just a hardwired shift is necessary for the multiplication with μ . Such that in pipelining there won't be any issues of register overhead. Likewise, the weights of the filter are used by the weight updating block and also the registers which are in tapped delay line are used by the error computation block.

3.5.3 Implementation of Two Adaptation Delay:

The figure (3.9) below, shows the LMS adaptive filter structure for the two adaptation delay model. Three pipeline stages are involved in this two adaptation delay process. Among them the very first stage in the error computation block stops immediately next to the primary level of adder tree. Then the remaining stages of this block consists of the following pipeline stages. Similar to error computation block, there are three pipeline stages in weight update block. There are additional registers in this two adaptation delay structure compared to one adaptation delay structure. This structure includes $N/2$ extra registers additional to previously discussed structure.

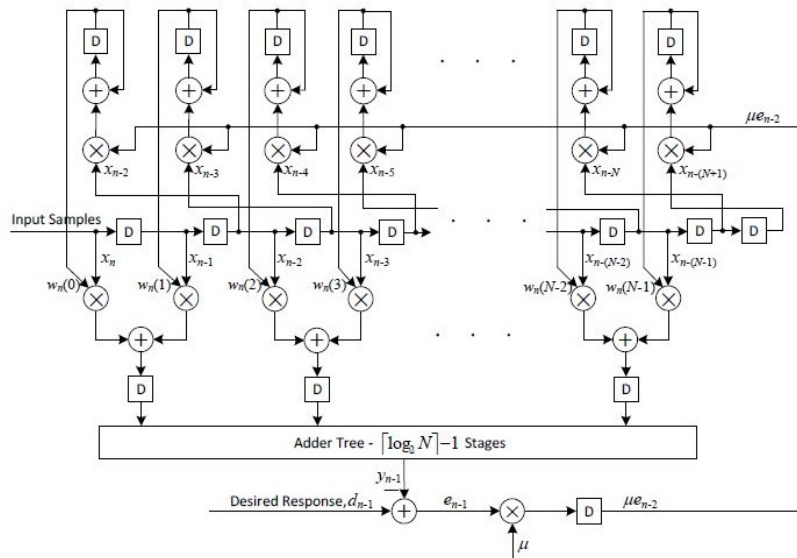


Figure 3.9: Structure for Two adaptation delay direct-form LMS adaptive filter.

3.6 Simulation Results and Discussion:

This has been implemented in Verilog HDL and synthesis has been done in Xilinx ISE 14.2 and simulation is done using ModelSim 10.3. FPGA implementation has been done successfully using SPARTAN-3E Xilinx board for all the three structures of direct-form LMS adaptive filter.

RTL Schematic view of Zero-Adaptation-Delay:

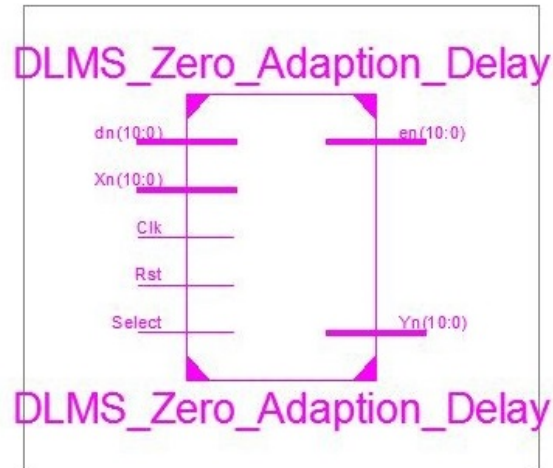


Figure 3.10: RTL Schematic view of Zero-Adaptation-Delay.

Summary of Zero-Adaptation-Delay:

Device Utilization Summary (estimated values)				[...]
Logic Utilization	Used	Available	Utilization	
Number of Slices	98	4656	2%	
Number of Slice Flip Flops	77	9312	0%	
Number of 4 input LUTs	167	9312	1%	
Number of bonded IOBs	47	232	20%	
Number of MULT18X18SIOs	4	20	20%	
Number of GCLKs	1	24	4%	

Table 3.1: Summary of Zero-Adaptation-Delay

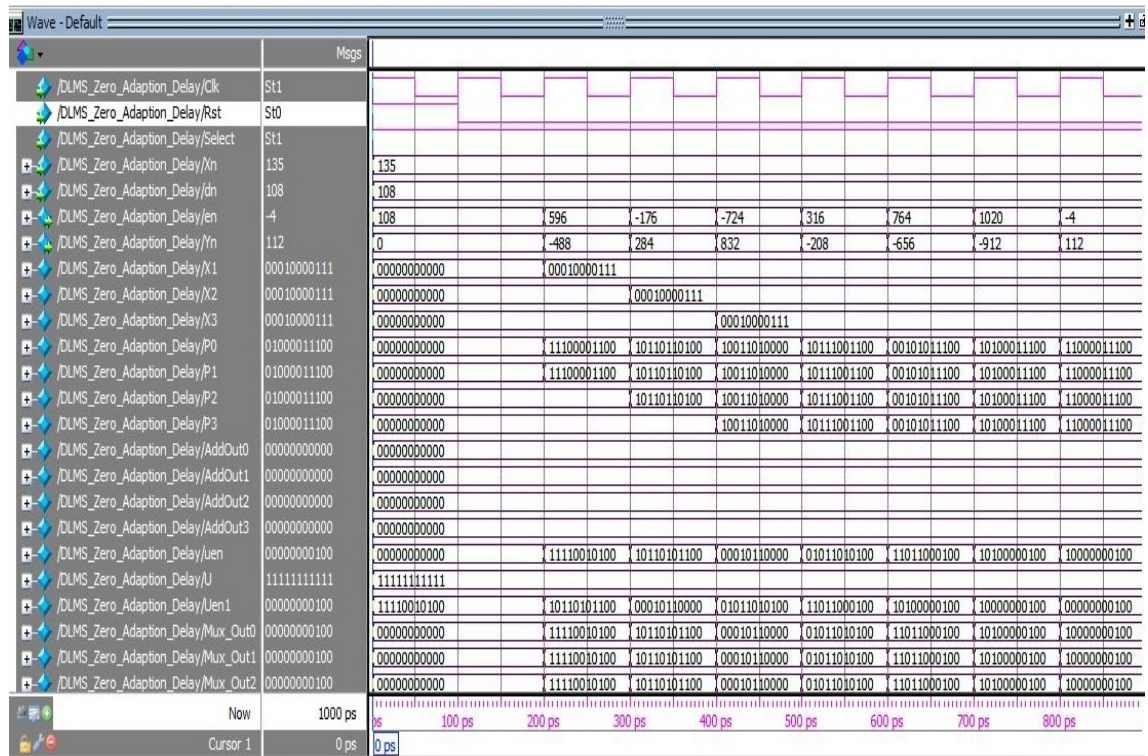
Simulation of Zero-Adaption-Delay:

Figure 3.11: Simulation of Zero-Adaption-Delay.

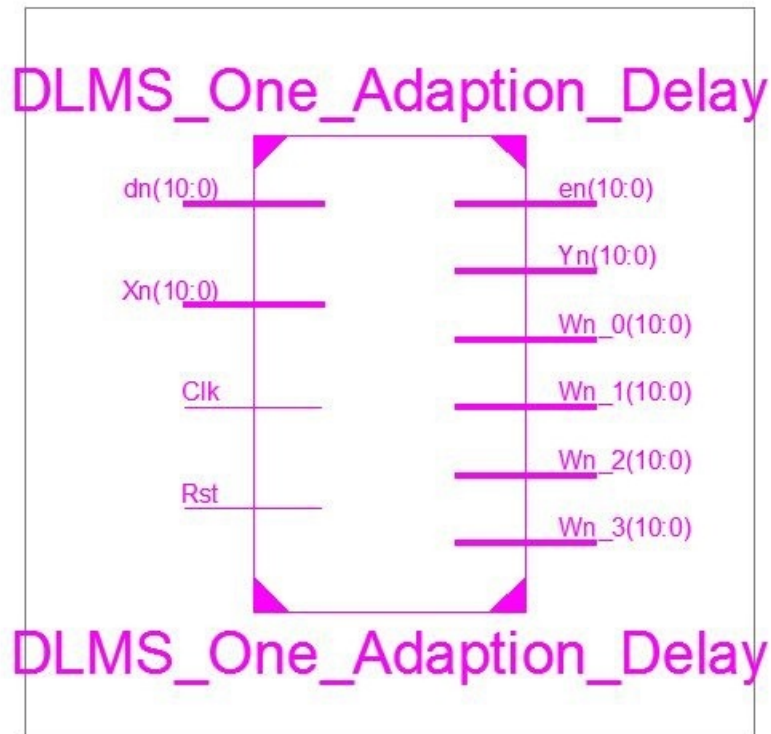
RTL Schematic of One-Adaption-Delay:

Figure 3.12: RTL Schematic of One-Adaption-Delay.

RTL Schematic inner view of One-Adaptation-Delay:

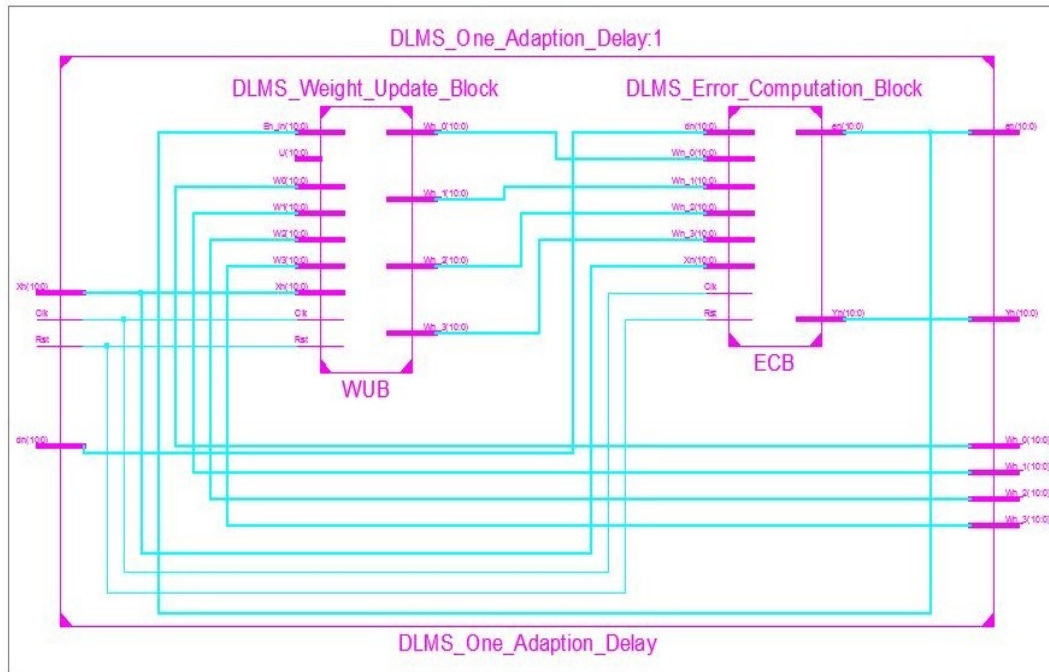


Figure 3.13: RTL Schematic inner view of One-Adaptation-Delay.

Simulation Result of One-Adaptation-Delay:

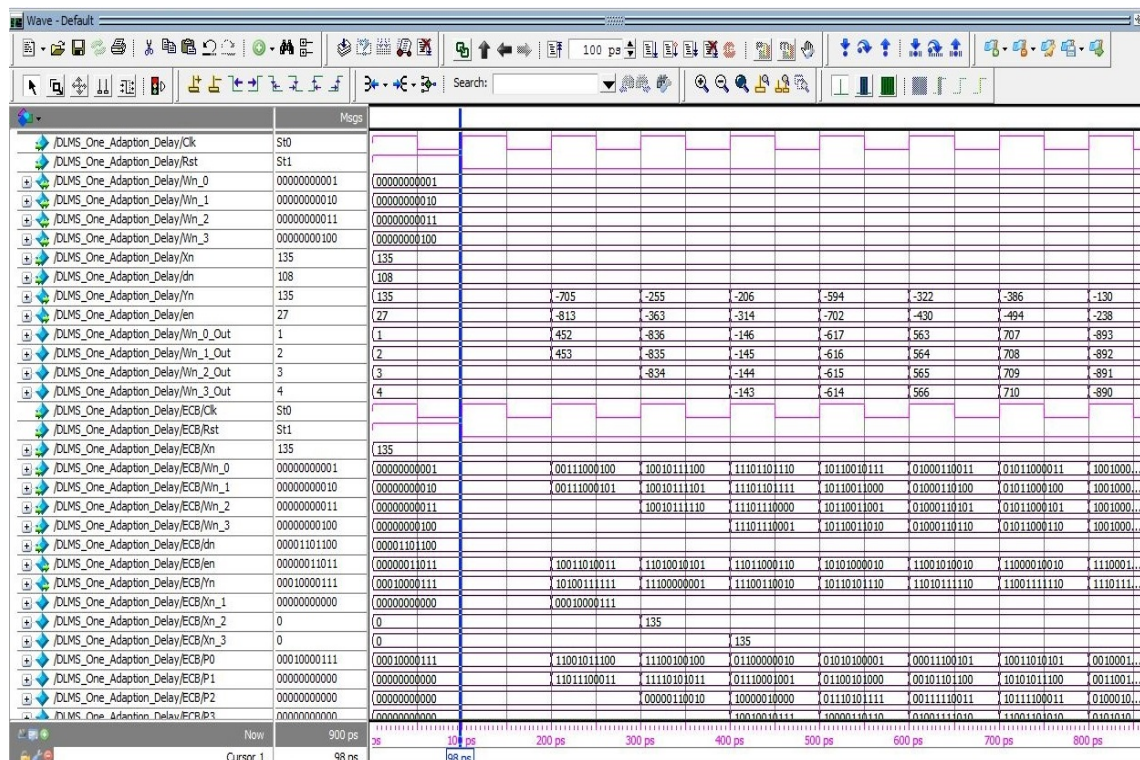


Figure 3.14: Simulation Result of One-Adaptation-Delay.

RTL Schematic of Two-Adaptation-Delay:

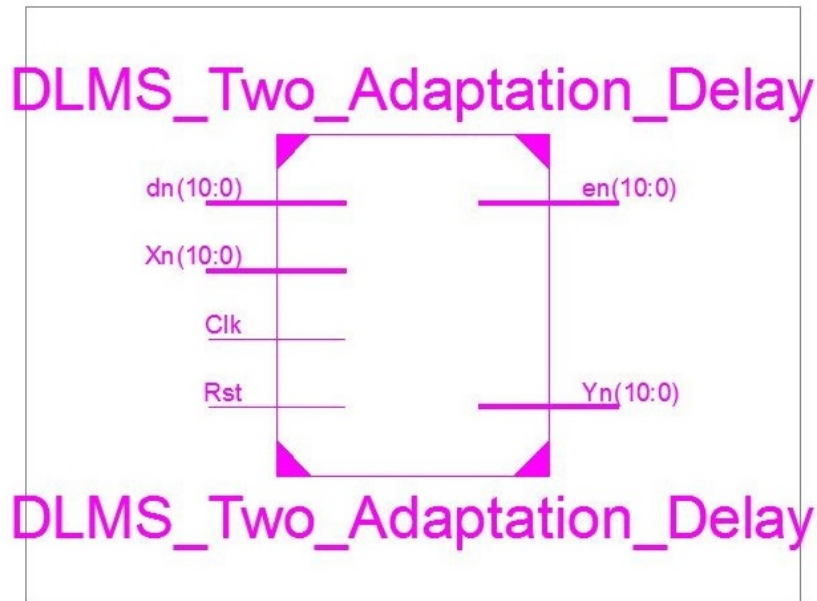


Figure 3.15: RTL Schematic of Two-Adaptation-Delay.

Simulation Result of Two-Adaptation-Delay:

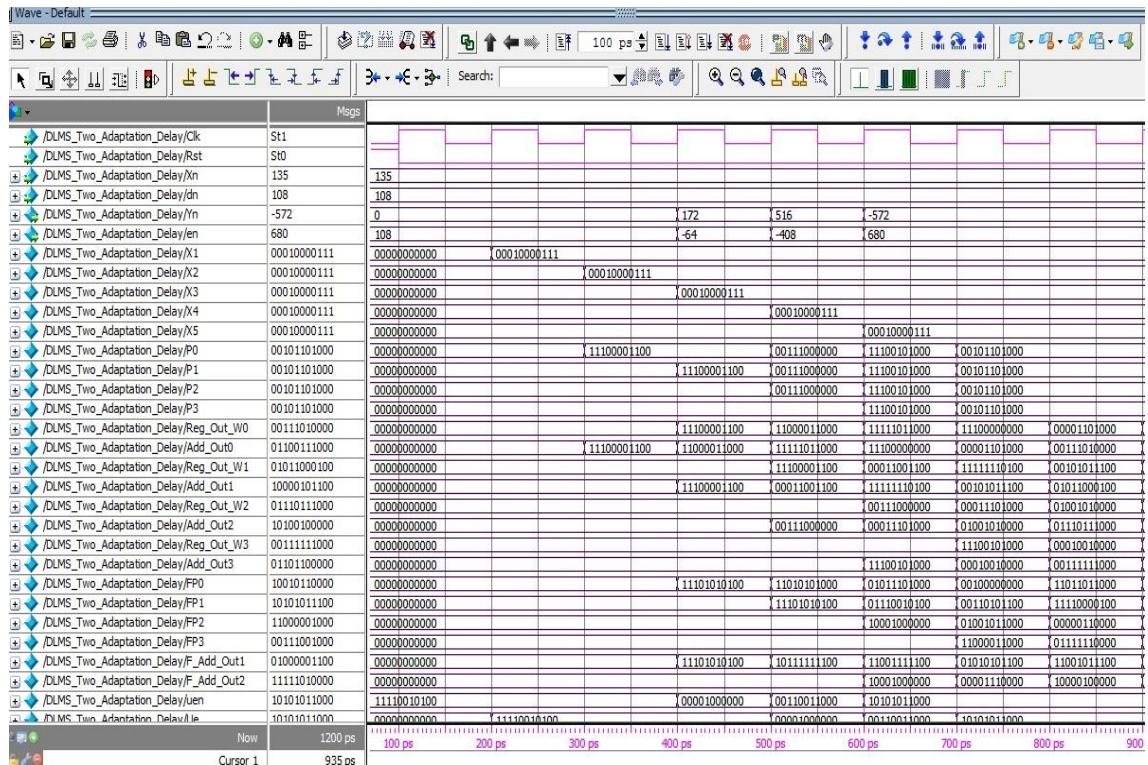


Figure 3.16: Simulation Result of Two-Adaptation-Delay.

Summary of One-Adaptation-Delay:

Device Utilization Summary (estimated values)				[-]
Logic Utilization	Used	Available	Utilization	
Number of Slices	62	4656	1%	
Number of Slice Flip Flops	33	9312	0%	
Number of 4 input LUTs	110	9312	1%	
Number of bonded IOBs	90	232	38%	
Number of MULT18X18SIOs	8	20	40%	
Number of GCLKs	1	24	4%	

Table 3.2: Summary of One-Adaptation-Delay

Summary of Two-Adaptation-Delay:

Device Utilization Summary (estimated values)				[-]
Logic Utilization	Used	Available	Utilization	
Number of Slices	79	4656	1%	
Number of Slice Flip Flops	88	9312	0%	
Number of 4 input LUTs	99	9312	1%	
Number of bonded IOBs	46	232	19%	
Number of MULT18X18SIOs	8	20	40%	
Number of GCLKs	1	24	4%	

Table 3.3: Summary of Two-Adaptation-Delay

Chapter 4

FPGA Based Implementation for EEG Applications

4.1 EEG (Electro-Encephalogram):

Electroencephalogram (EEG) are represented by the electrical impulses. To analyze the brain activities a device is used which is known as EEG or Electroencephalogram. In the brain by nerve firings electrical impulses are generated. These impulses will pass through the head. From head the brain activity generates a signal and this signal can be obtained and recorded using EEG. These signals are generated by shelling of neurons which are present in the brain. In the brain the electrical movement comprises of rhythms and using their frequency ranges they are divided into categories. Among them 0.5-4 Hz frequency range is named as delta rhythm, 4-8 Hz range is named as theta, 8-13 Hz range is alpha, 13-30 Hz range is beta, and greater than 30 Hz range is named as gamma rhythm.

By placing the electrodes on the scalp EEG measurements are recorded. Contamination of EEG signal is occurred due to various reasons. To name a few reasons like, the blinks, the eye movement, cardiac signals, line noise, and muscle signal are considered as a problem while recording the EEG signals. Because of externally generated artifacts and biologically generated artifacts, both the examination of EEG information and also drawing out of data from the signal are becoming difficult. When this artifacts passes through the body becomes immeasurable interference signal. Then overlap between this interference signal and the spectrum of EEG occurs. So, in this chapter we discuss about the importance of EEG signal, the artifacts present in EEG signal, and also the interference cancellation in EEG signal by using delayed LMS adaptive filter.

4.2 Need for EEG Signal:

EEG measurements are recorded and used to monitor alertness, brain death, and coma. And also to identify the damaged area which are caused by stroke, head injury, and tumor. These measurements are also required to examine the epilepsy and to test afferent nerves

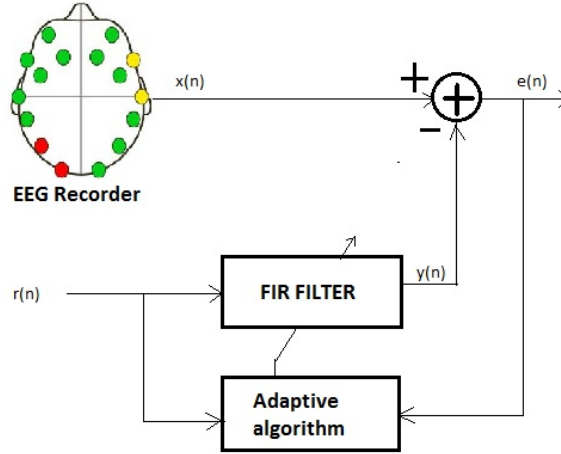


Figure 4.1: Block diagram of EEG.

for monitoring anesthesia depth. Normally, by placing the electrodes on the scalp EEG measurements are recorded. EEG signals have very small amplitude. These amplitudes are in the values of microvolts ranging from 0.6 to 100 μV .

4.3 Artifacts in EEG Signal:

Both from equipment interferences and patient are responsible for contamination of EEG data, this will lead to serious problems for EEG analysis and also for clinical applications. Artifacts which are generated biologically can be eliminated after recording the signals but artifacts generated externally can be reduced by using proper suitable techniques. As discussed in the need for EEG signal, amplitudes are in the values of microvolts ranging from 0.6 to 100 μV , but the artifacts which are generated through the movement of the eyes and also through the blinks are in the order of mV. In general, these artifacts in EEG can be controlled by removal of the segments of EEG which are affected. Suppose if an artifact is detected then discarding that particular length segment for a quick time of one second gives the suitable approach. But this method of approach has a problem of lose in data for analyzing the signal.

Physiological and non-physiological are the two differently categorized artifacts present in the EEG data. Physiological artifacts are the artifacts that are generated from inside the body [25]. This comprises the ordinary electrical activity of the eyes, muscles, and the heart. Whereas, Non-physiological artifacts are the artifacts such as sweat artifacts, movement artifacts, and power line interference. This artifacts are generated because of the movements of outside the body. This type of artifacts can be eliminated before the starting of recording process. This work predominantly focuses on the cancellation of physiological artifacts which are present in the electroencephalogram signal.

4.3.1 Motion Artifacts:

Motion artifacts are the signals which are generated with the movement of the patient. These signal has the frequency range in the order of 1 to 10 Hz and also has an amplitude of peak to peak value of 14 mV. Most of the cases these artifacts occurs due to stretching of the skin.

4.3.2 Sweat Artifact:

As name suggests this sweat artifacts are caused due to sweat that is produced in the scalp area while recording the EEG data. Large slow baseline sways are generated due to the chemical reaction of sweat and electrode. The electrodes metals react with the lactic acid and sodium chloride which is present in the sweat and generates the sways. This artifacts are ranging from 0.2 to 0.5 HZ frequency, so these artifacts are easy to eliminate and can be eliminated using high pass filter.

4.3.3 Power Line Interference:

Electrical equipment's activity which are very close the electrodes is the main reason for the occurrence of this type artifact. In EEG signal, power line interference is one among the foremost artifacts. These unwanted artifacts are removed by using the low pass filter because it operates at a low frequency range of below 50 Hz.

4.3.4 ECG and EMG Artifacts:

ECG artifacts are the artifacts that occur mostly with the people having wide and short neck [26]. These type of artifacts are commonly seen in babies with their heads close to the thorax and also in obese patients.

EMG artifacts are caused due to muscle activity. This includes the movement of different muscle groups such as facial and neck muscles [27]. The common cause for EMG artifacts in EEG signal is due to the clenching of muscles of the jaw. By making sure that the patient is relaxed this type of artifact can be eliminated.

4.3.5 Glossokinetic and Respiration Artifacts:

Glossokinetic artifacts are generated by the activity of tongue. It usually operates in the delta range but the frequency is variable. Similar artifacts are produced by sucking and chewing.

Respiration artifacts are the artifacts that are in the form of synchronous with the movements of the body, and of rhythmic and slow activity of respiration which affects the impedance of one electrode. With synchronous to exhalation and inhalation it can produce sharp and slow waves.

4.3.6 Electro-oculogram (EOG) Artifacts:

Human eye creates an electrical dipole which is caused by a negative retina and positive cornea. These two opposite charges has a 100mV of potential difference in between them. Blinks and movements of the eye changes the electric field around the eye causing an electrical signal which is known as electro-oculogram (EOG) [28], . These signals are recorded as artifact or noise in EEG when the signal spreads through the scalp and creates a serious problem for analyzing or interpretation of EEG. Two types of EOG artifacts are generated according to movement of eyes. Such as the artifact generated by the horizontal movement of eye is known as Horizontal EOG (HEOG) and the one generated by vertical movement of eye is known as Vertical EOG (VEOG). These are a low frequency signals which are less than 4 Hz but with high propagation. Whereas blinking of eye is another type of EOG artifact with less propagation but with high amplitude with same frequency as eye movement artifact. This work mainly focuses on the cancellation of ocular artifacts which are present in the electroencephalogram (EEG) signal.

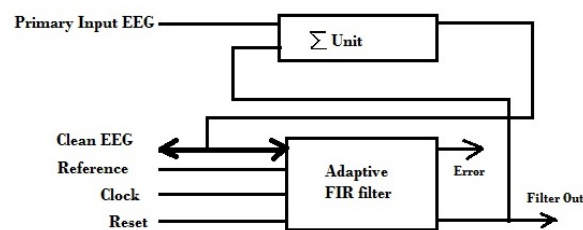


Figure 4.2: Block diagram of EEG Unit

In this process, a noise canceller is used with two reference inputs. EEG signal is given to the system as primary input which is picked up by a particular electrode. This signal is designed as a combination of both the electroencephalogram EEG $x(n)$ and a noise signal $r(n)$ [29]. The two reference inputs are, $v'(n)$ and $v(n)$ which are HEOG or VEOG, respectively. The desired output from the noise canceller $e(n)$ is the cleaned or corrected EEG.

4.4 FPGA Implementation

Most of the FPGAs are easily programmable because majority of them are SRAM-based. In the FPGA, the SRAM bits takes the decision to make connections or not as this bits are coupled to the Configuration points [30]. SRAM supplies the logic values to the passgate structure which in turn makes the connection on or off based on the information supplied by SRAM. FPGAs are volatile because they are based on SRAM. Each time power is functional they must be programmed accordingly. Generally this is accomplished by another part of circuit known as PROM which reloads the bit stream configuration [31]. The data to be stored in Look up tables (LUTs) and the connections to make are controlled by the SRAMs configuration bitstream. The arbitrary logic functions are performed by the LUTs which are

very small memory. Different names are used for basic blocks according to manufacturer but the common thing is LUTs are their functional units. Logic elements (LE) is the name used by Altera, whereas configurable logic blocks (CLBs) is the name used by the Xilinx FPGAs. Four logic slices are present in each CLB in a Xilinx FPGA. This in turn comprises of two 4-input function generators, arithmetic logic gates, two storage elements, carry logic and, function multiplexers.

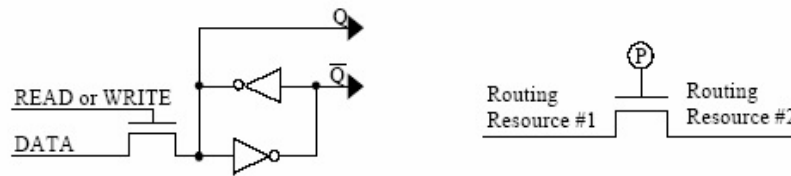


Figure 4.3: SRAM based FPGA configuration.

4.5 Hardware Description Languages:

Verilog and VHDL are the two most popular hardware description languages. Both these languages are based on text depictions of their behavior of the digital circuit. For representing time and concurrency its syntax contains various different notations. Around 1984, Gateway Design Automation Inc. started the Verilog language as a exclusive hardware modeling language. In 1990, the language went open to everyone and since then it is very popular in the semiconductor industry for ASIC and FPGA design.

4.6 FPGA Design Flow:

The design flow of FPGA is shown in figure 4.4. The design flow consists of design entry, design synthesis, design implementation, and programming the xilinx device. In the Design entry a user constraints file (UCF) is also included in the design file along with assigning pins, timing constraints, and area constraints. Device verification here comprises of both timing verification and functional verification. In functional verification, RTL simulation which is known as behavioral simulation has to be done before doing synthesis. It is followed by functional simulation as shown in figure 4.4. After this synthesis has to be done followed by design implementation which comprises of Translate, mapping, and finally place and route. Timing verification can be done after mapping and, place and route. To program FPGA a bit file is created by generating PROM file for debugging. Finally using iMPACT window the device is programmed with a programming cable.

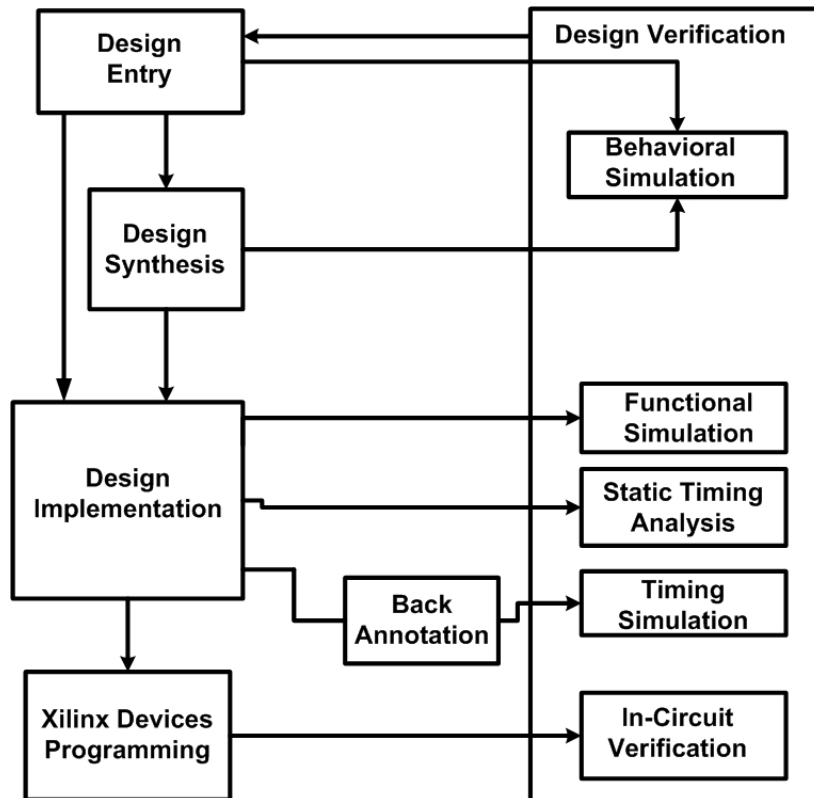


Figure 4.4: FPGA Design Flow

4.7 Simulation Results and Discussion:

Tools Used:

This has been implemented in Verilog HDL and synthesis has been done in Xilinx ISE 14.2 and simulation is done using ModelSim 10.3. FPGA implementation has been done successfully using SPARTAN-3E Xilinx board for all the three structures of direct-form LMS adaptive filter for the application of EEG signal.

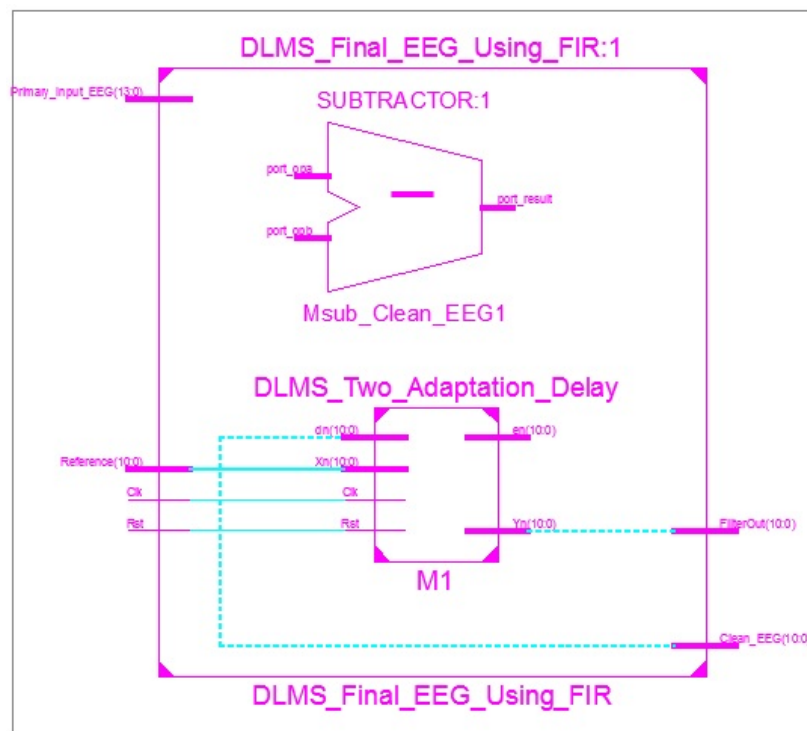


Figure 4.5: RTL Schematic inner view of Two-Adaptation-Delay for EEG.

Simulation Results of Two-Adaptation-Delay for EEG:

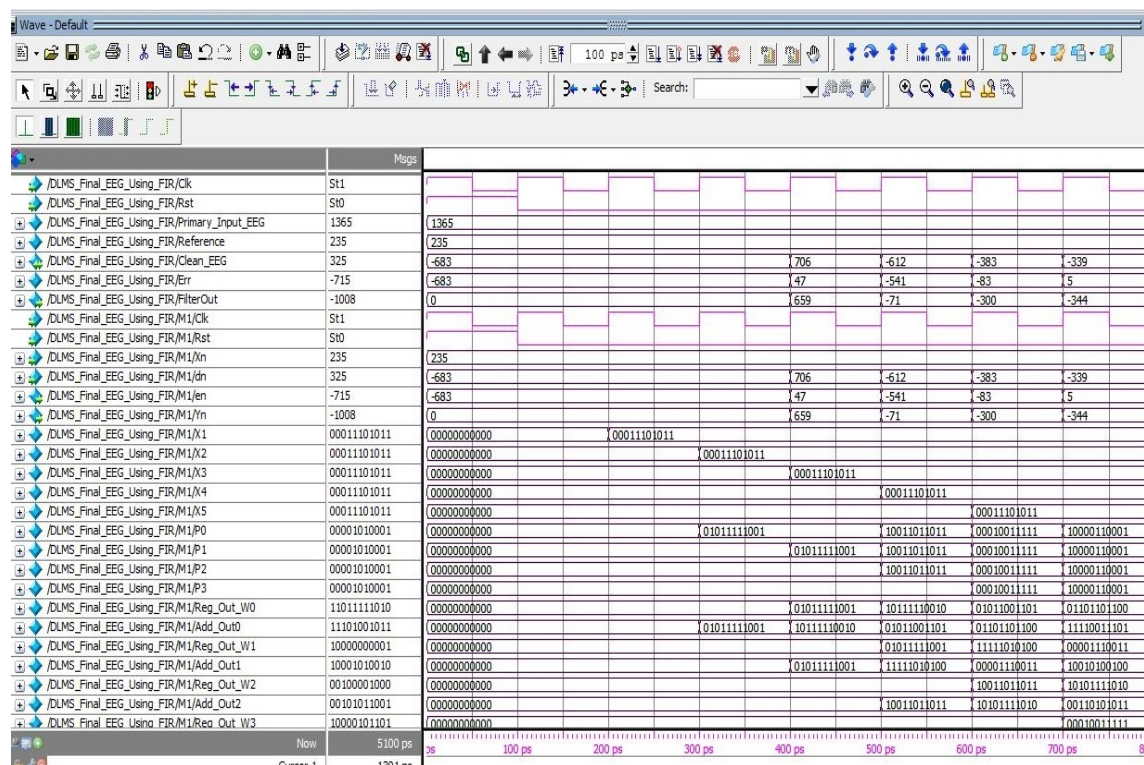


Figure 4.6: Simulation Results of Two-Adaptation-Delay for EEG signal.

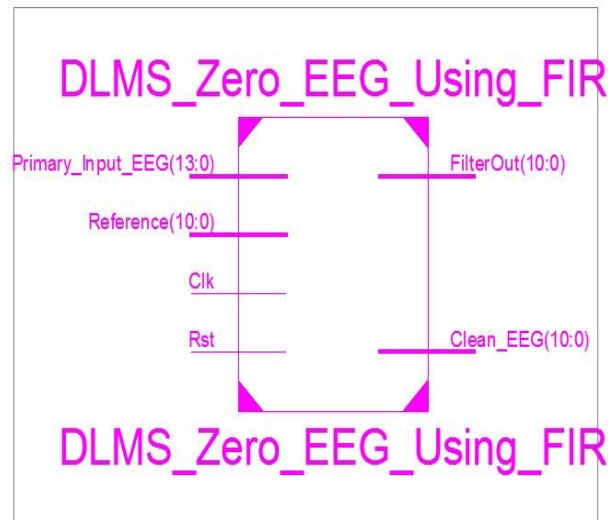
RTL Schematic of Zero-Adaptation-Delay for EEG:

Figure 4.7: RTL Schematic of Zero-Adaptation-Delay for EEG.

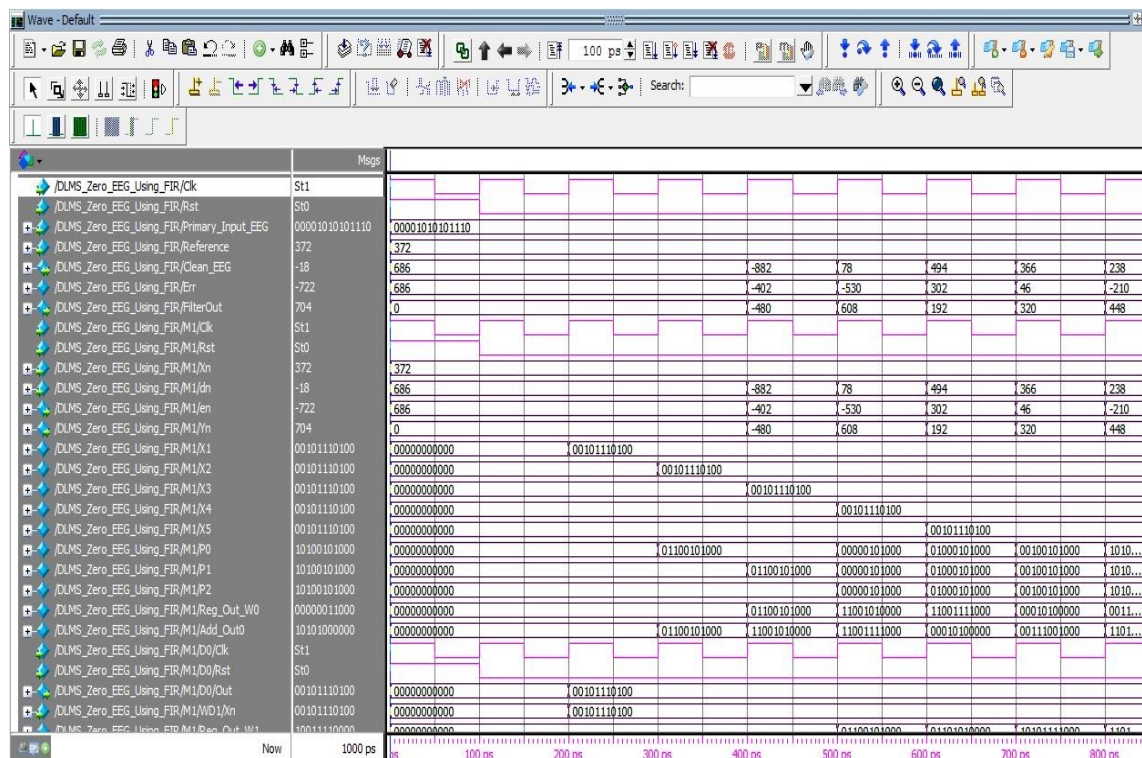
Simulation Results of Zero-Adaptation-Delay for EEG:

Figure 4.8: Simulation Results of Zero-Adaptation-Delay for EEG.

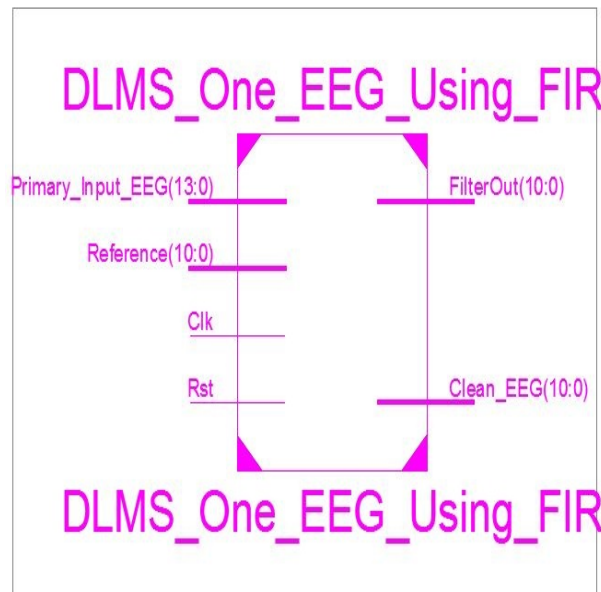
RTL Schematic of One-Adaptation-Delay for EEG:

Figure 4.9: RTL Schematic of One-Adaptation-Delay for EEG.

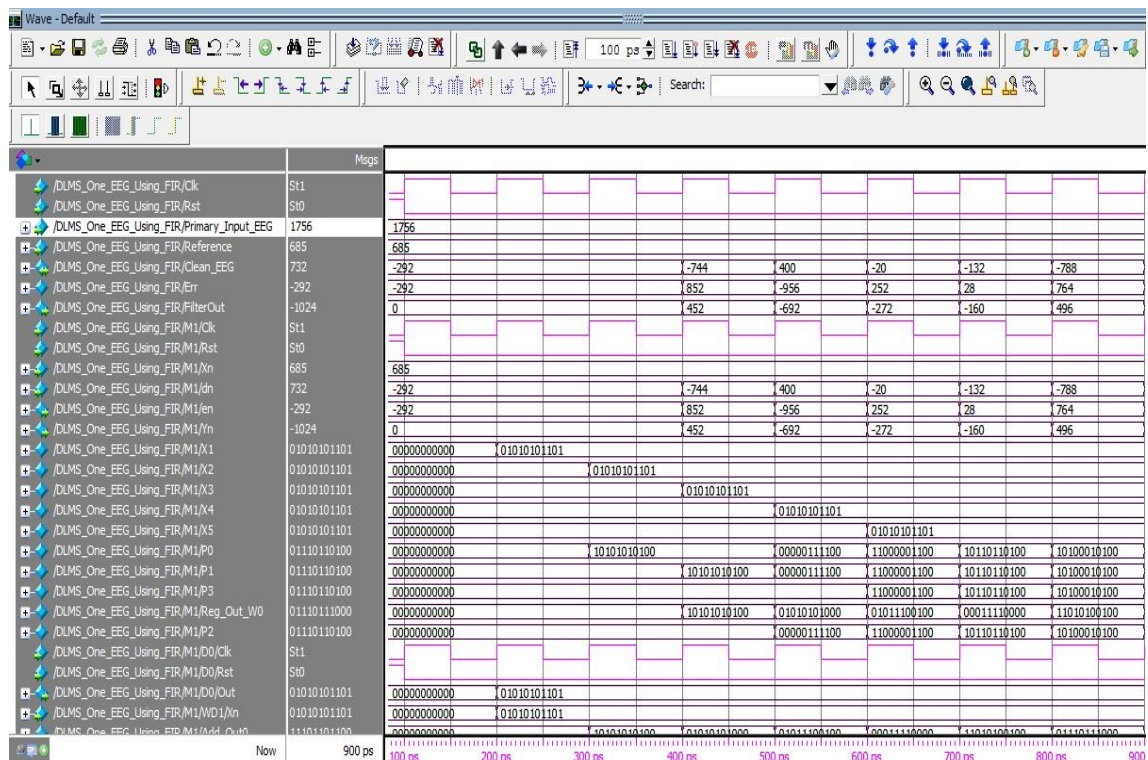
Simulation Results of One-Adaptation-Delay for EEG:

Figure 4.10: Simulation Results of One-Adaptation-Delay for EEG.

Summary of Zero-Adaptation-Delay for EEG:


Device Utilization Summary					
Logic Utilization	Used	Available	Utilization	Note(s)	
Total Number Slice Registers	296	9,312	3%		
Number used as Flip Flops	295				
Number used as Latches	1				
Number of 4 input LUTs	335	9,312	3%		
Number of occupied Slices	319	4,656	6%		
Number of Slices containing only related logic	319	319	100%		
Number of Slices containing unrelated logic	0	319	0%		
Total Number of 4 input LUTs	387	9,312	4%		
Number used as logic	255				
Number used as a route-thru	52				
Number used as Shift registers	80				
Number of bonded IOBs	24	232	10%		
Number of RAMB16s	1	20	5%		
Number of BUFGMUXs	2	24	8%		
Number of BSCANS	1	1	100%		
Number of MULT18X18SIOs	8	20	40%		
Number of RPM macros	13				
Average Fanout of Non-Clock Nets	2.38				

Table 4.1: Device Utilization Summary of Zero-Adaptation-Delay for EEG.

Summary of One-Adaptation-Delay for EEG:


Device Utilization Summary					
Logic Utilization	Used	Available	Utilization	Note(s)	
Number of Slice Flip Flops	33	9,312	1%		
Number of 4 input LUTs	109	9,312	1%		
Number of occupied Slices	74	4,656	1%		
Number of Slices containing only related logic	74	74	100%		
Number of Slices containing unrelated logic	0	74	0%		
Total Number of 4 input LUTs	110	9,312	1%		
Number used as logic	109				
Number used as a route-thru	1				
Number of bonded IOBs	90	232	38%		
Number of BUFGMUXs	1	24	4%		
Number of MULT18X18SIOs	8	20	40%		
Average Fanout of Non-Clock Nets	2.02				

Table 4.2: Device Utilization Summary of One-Adaptation-Delay for EEG.

Summary of Two-Adaptation-Delay for EEG:


Device Utilization Summary					
Logic Utilization	Used	Available	Utilization	Note(s)	
Total Number Slice Registers	336	9,312	3%		
Number used as Flip Flops	335				
Number used as Latches	1				
Number of 4 input LUTs	346	9,312	3%		
Number of occupied Slices	344	4,656	7%		
Number of Slices containing only related logic	344	344	100%		
Number of Slices containing unrelated logic	0	344	0%		
Total Number of 4 input LUTs	397	9,312	4%		
Number used as logic	266				
Number used as a route-thru	51				
Number used as Shift registers	80				
Number of bonded IOBs	46	232	19%		
Number of RAMB16s	1	20	5%		
Number of BUFGMUXs	2	24	8%		
Number of BSCANS	1	1	100%		
Number of MULT18X18SIOs	8	20	40%		
Number of RPM macros	13				
Average Fanout of Non-Clock Nets	2.52				

Table 4.3: Device Utilization Summary of Two-Adaptation-Delay for EEG.

ISE iMPACT window while FPGA implementation on SPARTAN 3E Board:

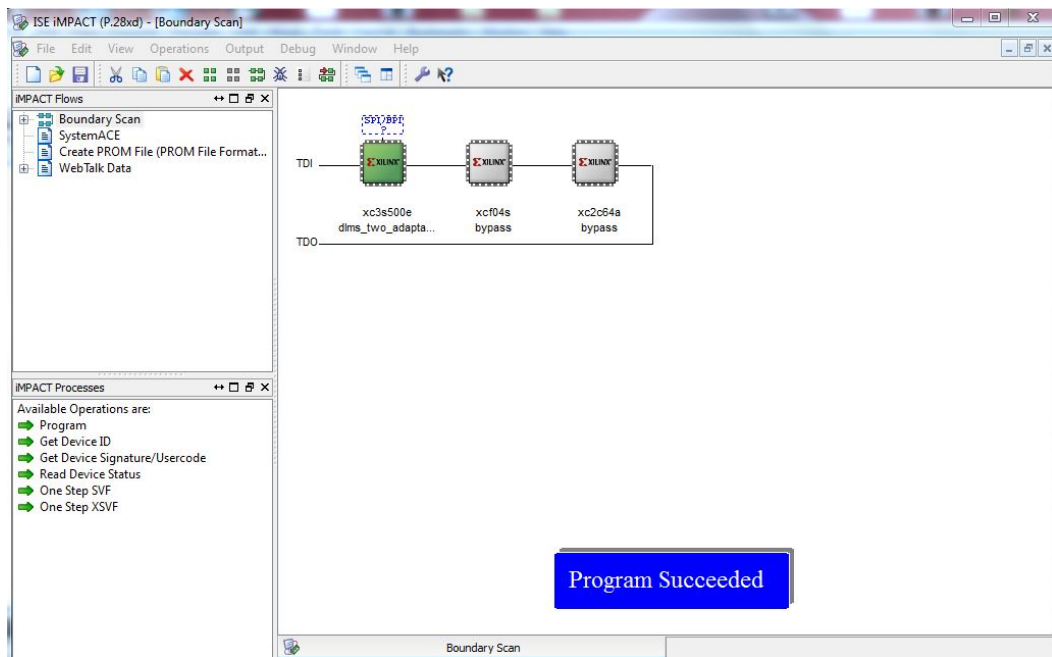


Figure 4.11: ISE iMPACT window while FPGA implementation on SPARTAN 3E Board.

Comparison Table:

Table 4.4: Comparison Table for three different adaptive filtering techniques.

sl No:	Method name	Area			Delay		
		Gate	Slice Flip-flop	LUTs	Max Delay	Gate Delay	Path Delay
1	Zero Adaptation Dealay	18,253	77	167	18.311ns	14.678ns	3.633ns
2	One Adaptation Dealay	33,441	33	110	20.99ns	17.846ns	3.144ns
3	Two Adaptation Dealay	34,017	88	99	15.496ns	13.281ns	2.215ns

Chapter 5

Conclusions

The proposed structures derives the low-complexity architecture for the LMS adaptive filter. Since the delayed LMS adaptive filters does the delayed weight adaptation approach, it provides the faster convergence speed and also the register complexity is less. Based on delayed LMS adaptive filters, three different architectures are proposed in this thesis categorized by number of adaptation delays. They are, (1) zero adaptation delay, (2) one adaptation delay, and (3) two adaptation delays. Besides this, an FPGA based design of adaptive interference cancellation for the EEG applications is done and this offers various advantages such as allowing to reduce the settling and design time of the canceller. Analyzed advantages and disadvantages of various techniques in this thesis work. Different types of artifacts that are present in the EEG signals are studied and analyzed. FPGA based implementation is done in Xilinx SPARTAN 3E board for all the three proposed adaptive filtering techniques and simulation are evaluated for the interference cancellation in EEG applications.

From the table (4.1) in chapter 4, it can be analyzed that the gate area is less in zero adaptation delay when compared to two adaptation and one adaptation delays. Besides that, the path delay and gate delays are less in two adaptation delay compared to one adaptation and zero adaptation delays. It can be inferred that zero adaptation delay structure has minimal gate area and two adaptation delay structure has minimal delay. However, the zero adaptation delay design is more preferable choice then the two adaptation delay design because it has significantly less area and also it offers satisfactory speed performance.

References

- [1] S. Haykin and B. Widrow, *Least-Mean-Square Adaptive Filters*. Hoboken, NJ: Wiley-Interscience, 2003.
- [2] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
- [3] P. K. Meher and S. Y. Park, "Critical-path analysis and low-complexity implementation of the lms adaptive algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 3, pp. 778–788, March 2014.
- [4] F. L. G. Long and J. G. Proakis, "The lms algorithm with delayed coefficient adaptation," *IEEE Trans. Acoust., Speech, Signal Process*, vol. 40, no. 1, pp. 230 – 232, Aug 2002.
- [5] M. D. Meyer and D. P. Agrawal, "A modular pipelined implementation of a delayed lms transversal adaptive filter," *IEEE International Symposium on Circuits and Systems*, pp. 429 – 436, May 1990.
- [6] D. Van and W. S. Feng, "An efficient systolic architecture for the dlms adaptive filter and its applications," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process*, vol. 48, no. 4, pp. 359 – 366, Apr 2001.
- [7] Y. Yi, R. Woods, L. Ting, and C. Cowan, "High speed fpga-based implementations of delayed-lms filters," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 39, no. 1, pp. 113–131, 2005. [Online]. Available: <http://dx.doi.org/10.1023/B:VLSI.0000047275.54691.be>
- [8] E. Mahfuz, C. Wang, and M. O. Ahmad, "A high-throughput dlms adaptive algorithm," in *2005 IEEE International Symposium on Circuits and Systems*, May 2005, pp. 3753–3756 Vol. 4.
- [9] P. K. Meher and M. Maheshwari, "A high-speed fir adaptive filter architecture using a modified delayed lms algorithm," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, May 2011, pp. 121–124.
- [10] M. Chakraborty and H. Sakai, "Convergence analysis of a complex lms algorithm with tonal reference signals," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 2, pp. 286–292, March 2005.
- [11] J. Vaňuš and V. Stýskala, "Application of optimal settings of the lms adaptive filter for speech signal processing," in *Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 International Multiconference on*, Oct 2010, pp. 767–774.
- [12] A. Mandal and R. Mishra, "Design and performance analysis of lms algorithm based adaptive filter embedded with cfar detector under non-homogeneous clutter scenarios," *International Journal of Adaptive Control and Signal Processing*, pp. n/a–n/a, 2015. [Online]. Available: <http://dx.doi.org/10.1002/acs.2646>

- [13] D. Sharma and R. Kaur, "Improvement in convergence speed and stability of least mean square and normalized least mean square algorithm," in *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on*, March 2015, pp. 1496–1500.
- [14] S. A. Ghauri and M. F. Sohail, "System identification using lms, nlms and rls," in *Research and Development (SCORED), 2013 IEEE Student Conference on*, Dec 2013, pp. 65–69.
- [15] J. M. Valin and I. B. Collings, "Interference-normalized least mean square algorithm," *IEEE Signal Processing Letters*, vol. 14, no. 12, pp. 988–991, Dec 2007.
- [16] T. Wang, Y. Cheng, B. Jiang, and R. Qi, "Fault detection based on finite impulse response adaptive filter for satellite attitude control systems," in *The 26th Chinese Control and Decision Conference (2014 CCDC)*, May 2014, pp. 209–213.
- [17] T. Pitchaiah, D. L. M, and P. V. S. Devi, "Fpga implementation of low area and delay efficient adaptive filter using distributed arithmetic," in *Advances in Engineering and Technology Research (ICAETR), 2014 International Conference on*, Aug 2014, pp. 1–5.
- [18] J. Chhikara and J. Singh, "Noise cancellation using adaptive algorithms," *International Journal of Modern Engineering Research (IJMER)*, vol. 2, no. 3, pp. 792 – 795, May 2012.
- [19] M. Bahoura and H. Ezzaidi, "Fpga-implementation of a sequential adaptive noise canceller using xilinx system generator," in *2009 International Conference on Microelectronics - ICM*, Dec 2009, pp. 213–216.
- [20] S. Thilagam and P. Karthigaikumar, "Implementation of adaptive noise canceller using fpga for real-time applications," in *Electronics and Communication Systems (ICECS), 2015 2nd International Conference on*, Feb 2015, pp. 1711–1714.
- [21] H. Li, N. Tong, N. Liu, and B. Tian, "A new variable-step-size lms adaptive filtering algorithm and its application in adaptive noise jamming cancellation system," in *Antennas, Propagation and EM Theory, 2008. ISAPE 2008. 8th International Symposium on*, Nov 2008, pp. 1346–1349.
- [22] W. Harrison, J. Lim, and E. Singer, "A new application of adaptive noise cancellation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 1, pp. 21–27, Feb 1986.
- [23] P. K. Meher and S. Y. Park, "Low adaptation-delay lms adaptive filter part-ii: An optimized architecture," in *2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug 2011, pp. 1–4.
- [24] —, "Area-delay-power efficient fixed-point lms adaptive filter with low adaptation-delay," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 362–371, Feb 2014.
- [25] M. K. Kim and S. P. Kim, "Artifact removal from eeg signals using the total variation method," in *Control Conference (ASCC), 2015 10th Asian*, May 2015, pp. 1–4.
- [26] M. Z. U. Rahman, R. A. Shaik, and D. V. R. K. Reddy, "Efficient and simplified adaptive noise cancelers for ecg sensor based remote health monitoring," *IEEE Sensors Journal*, vol. 12, no. 3, pp. 566–573, March 2012.
- [27] Y. Deng, W. Wolf, R. Schnell, and U. Appel, "New aspects to event-synchronous cancellation of ecg interference: an application of the method in diaphragmatic emg signals," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 9, pp. 1177–1184, Sept 2000.

- [28] W. Qi, “Eog artifacts removal in eeg measurements for affective interaction with brain computer interface,” in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2012 Eighth International Conference on*, July 2012, pp. 471–475.
- [29] R. Ramos, A. Manuel-Lazaro, J. D. Rio, 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, no. 6, pp. 2633–2640, Dec 2007.
- [30] M. Bahoura and H. Ezzaidi, “Fpga-implementation of wavelet-based denoising technique to remove power-line interference from ecg signal,” in *Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine*, Nov 2010, pp. 1–4.
- [31] A. R. Kasetwar and S. M. Gulhane, “A survey of fpga based interference cancellation architectures for biomedical signals,” in *Computer Communication and Informatics (ICCCI), 2013 International Conference on*, Jan 2013, pp. 1–7.