Design and Implementation of Adaptive Filters for Low Power and High Speed Applications

Dissertation submitted in the partial fulfilment of requirement
for the award of the degree of

Master of Technology

In

VLSI Design

Submitted by

Aditya Bali

Roll. No. 601261003

Under the supervision of

Dr. Ravi Kumar

Assistant Professor, ECED

Department of Electronics & Communication Engineering

Thapar University, Patiala -147004 (Punjab)

(Established under the section 3 of UGC Act, 1956)

July, 2014

DECLARATION

I hereby declare that the work which is being presented in the dissertation entitled, **"Design and Implementation of Adaptive Filters for Low Power and High Speed Applications"** in partial fulfilment of the requirement for the award of degree of Master of Technology in VLSI Design submitted in Electronics and Communication Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of **Dr. Ravi Kumar, Assistant Professor, ECED** and refers other researcher's work which are duly listed in the reference section.

The matter presented in this dissertation has not been submitted in any other University / Institute for the award of any other degree.

Aditya Bali
Roll. No. 601261003

Date: 30/06/14

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

Dr. Ravi Kumar
Assistant Professor
Thapar University, Patiala

Date: 30/06/14

Countersigned By:

(Dr. Sanjay Sharma)
Professor and Head ECED
Thapar University, Patiala

(Dr.S.K.Mohapatra)
Dean of Academic Affairs
Thapar University Patiala

ACKNOWLEDGEMENT

First of all, I would like to express my gratitude to **Dr. Ravi Kumar, Assistant Professor,** Electronics and Communication Engineering Department, Thapar University, Patiala for his patient guidance and support throughout this report. I am truly very fortunate to have the opportunity to work with him. I found this guidance to be extremely valuable.

I am also thankful to **Head of the Department, Dr. Sanjay Sharma** as well as our P.G. co-ordinator **Dr. Kulbir Singh, Associate Professor.** Further, I would like to thank entire faculty member, staff of Electronics and Communication Engineering Department, and then friends who devoted their valuable time and helped me in all possible ways towards successful completion of this work. I thank all those who have contributed directly or indirectly to this work.

Lastly, I would like to thank my parents for their years of unyielding love and for constant support and encouragement. They have always wanted the best for me and I admire their determination and sacrifice.

Aditya Bali

ABSTRACT

The present work is an effort to design and implement adaptive filters that can be used as an adaptive noise canceller for de-noising of different signals. Two methods are used to design adaptive filters in SIMULINK. The first method involves the use of adaptive algorithm block present in SIMULINK library whereas second method involves the structural implementation of the weight update loop equation using basic blocks like multipliers, adders, delay elements etc. present in SIMULINK library. Five different 32 order adaptive filters (LMS, NLMS, SD, SE and SS) were designed using first method for de-noising of sinusoidal signal. Further optimum step size, range of step size and mean square error was calculated for each of these adaptive filters.

Two step sizes viz. 0.002 and 0.0005 experimentally optimized for above mentioned adaptive filters has been selected for the implementation of four different variable step size adaptive filters (LMS, Sign-Data, Sign-Error and Sign-Sign) designed using second method for de-noising of audio signals. Three different structures viz. direct-form structure, transposed-form structure and the proposed novel structure has been used for this purpose. Further, the filters were implemented on three different Field programmable gate arrays (FPGAs) viz. Spartan 6, Virtex 6 and Viretx 7 and their speed, power and area utilization are computed. It was observed that due to significant reduction in the critical path, the proposed structure uses less number of slice LUTs resulting in a reduction in silicon area without incurring any significant overhead in terms of power or delay.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DSPs	Digital Signal Processors
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
MSE	Mean Square Error
LMS	Least Mean Square
NLMS	Normalized Least Mean Square
DLMS	Delayed Least Mean Square
BLMS	Block Least Mean Square
SD	Sign-Data
SE	Sign-Error
SS	Sign-Sign
ATC	Air Traffic Control
ANC	Adaptive Noise Canceller
LTI	Linear Time Invariant
RLS	Recursive Least Square
SA	Sign Algorithm
FPGA	Field Programmable Gate Array
HPC	High Performance Computing
LEC	Line Echo Canceller
PSO	Particle Swarm Optimization Technique
MAC	Multiply and Accumulate
VHDL	Very High Speed Integrated Circuit Hardware Description Language
XSG	Xilinx System Generator
HLS	High Level Synthesis
SCAF	Self Correcting Adaptive Filter
ECG	Electrocardiogram
SPT	Sum Power of Two
FSM	Finite State Machine

ASIC	Application Specific Integrated Circuits
DRC	Design Rule Check
LVS	Layout versus Schematic
LUT	Look up Table
FDATOOL	Filter Design and Analysis Tool
HDL	Hardware Description Language
RTL	Register Transfer Logic
ROM	Read only Memory
RAM	Random Access Memory
LE	Logic Element
CLB	Configurable Logic Block
PLD	Programmable Logic Device
SRAM	Static Random Access Memory
EPROM	Erasable Programmable Read only Memory
ICON	Integrated Controller core
ILA	Integrated Logic Analyzer core
VIO	Virtual Input Output Core
IBERT	Integrated Bit Error Ratio core
ATC2	Agilent Trace Core 2

Chapter 1

Introduction and Literature Review

1.1 Adaptive Filters

Adaptation as the name suggests is the ability to change according to the stimuli to produce the favourable results. In physical systems adaptation is used in adaptive filters [1] and neural networks [2]. Over the last three decades, adaptive filtering has become a necessity owing to the great advances that have been made in the area of digital signal processing [3]. Both speed and complexity of digital signal processors (DSPs) have increased manifolds accompanied by significant reduction in power consumption. Adaptive filters work as self-learning machines whose coefficients adjust themselves to the changing environment [4]. In other words, an adaptive filter changes its coefficients according to an adaptation algorithm to minimize the error and to estimate the signal statistics iteratively. This error signal, also referred to as the cost function [5], [6], is a distance measurement between the reference or desired signal and the output of the adaptive filter. The importance of adaptive filters lies because of the fact they give high performance and are robust to any sort of noise environment [7].

Fig. 1.1 depicts the basic structure of adaptive filter. Here x(n) is the input signal, y(n) is the output signal, d(n) is desired signal, e(n) is the error signal and w_n is the filter weights or coefficients. The first block is a digital filter which can be an FIR or IIR filter. However, in general we use FIR filters because they are less complicated and highly stable. The second block is a weight adaptation block which updates the filter weights or coefficients according to the adaptive algorithm to reduce the error signal. The coefficients of the adaptive filter are determined during a training sequence where a known data pattern is transmitted. The adaptive algorithm adjusts the filter coefficients to force the received data to match the training sequence data. After the training sequence is completed, the switches are put in the other position, and the actual data is transmitted. During this time, the error signal is generated by subtracting the output signal from the reference signal. The input signal is filtered to produce an output that is typically passed on for subsequent processing. This, signal is then passed on to a circuit to modify the parameters of the filter until the quality of the filter output is as good as possible.

The input signal is the sum of a desired signal d(n) and interfering noise v(n) i.e.

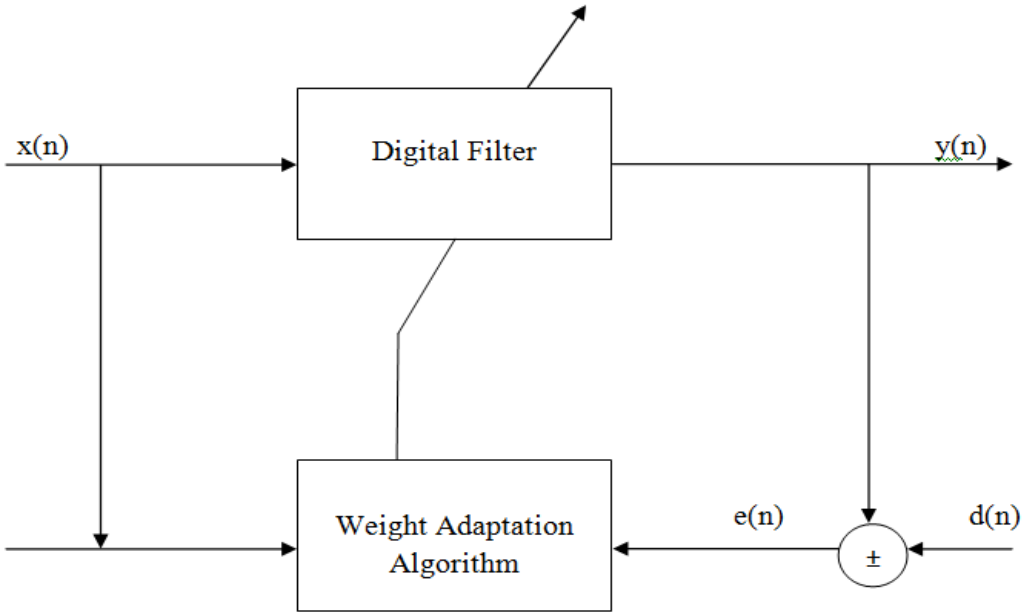

Fig.1.1 Basic Structure of Adaptive Filter [1].

$$x(n) = d(n) + v(n) \tag{1.1}$$

The coefficients for a filter of m^{th} order filter are defined as

$$w_n = [w_n(0), w_n(1), \ldots., w_n(p)]^T \tag{1.2}$$

The error signal or cost function is the difference between the desired and the estimated signal

$$e(n) = d(n) - y(n) \tag{1.3}$$

The variable filter estimates the desired signal by convolving the input signal with the impulse response. In vector notation this is expressed as

$$y(n) = w_n \times x(n) \tag{1.4}$$

where,

$$x(n) = [x(n), x(n-1), \ldots\ldots x(n-p)]^T \tag{1.5}$$

is an input signal vector. Moreover, the variable filter updates the filter coefficients at every time instant

$$w_{n+1} = w_n + \Delta w_n \tag{1.6}$$

where Δw_n is a correction factor for the filter coefficients [1].

The performance [8] of the adaptive filter depends upon the step size (δ), type of adaptive algorithm, filter length and the structure or architecture used for the implementation of the adaptive filter. Out of all these attributes step size (δ) is the most important one as it affects performance of adaptive filter by a highest degree even if all other attributes are set to their optimum values. It determines the amount of correction applied as the filter adapts from one iteration to the next. Choosing the appropriate step

size (δ) [5] is not always easy, usually requiring experience in adaptive filter design. The step size (δ) of the adaptive filter should be selected such that it is neither too small nor too large, as it affects the performance of the filtering algorithm by changing the convergence rate, satiability and mean square error (MSE) [9]. If the step size (δ) is large then coefficients of the adaptive filter will diverge instead of converging and hence the filter will not be able to remove noise and decrease error. This becomes an issue with stability, as the resulting filter might not be stable. On the other hand if the step size (δ) is too small then it will increase the convergence time of the adaptive filter coefficients and hence the adaptive filter will remove noise or decrease error with a slower speed. This becomes an issue of speed and accuracy [5]. As a rule of thumb, smaller step sizes improve the accuracy of the convergence of the filter to match the characteristics of the unknown system at the expense of the time it takes to adapt. The problem of choosing a step size (δ) can be solved by using a variable step size (δ). In case of a variable step size adaptive filter (δ) we can choose two or more than two step sizes. A larger step size (δ) can be used when a signal is changing slowly and a smaller step size (δ) can be used when the signal is changing at faster rate [10].

There are large numbers of adaptive algorithms such as least mean square (LMS), normalized least mean square (NLMS), delayed least mean square algorithm (DLMS), block least mean square algorithm (BLMS), sign-data algorithm (SD), sign-error algorithm (SE) and sign-sign algorithm (SS) etc [4], [5]. Proper choice of adaptive algorithm is important as different adaptive algorithms have different filtering capability. For example, NLMS algorithm is the most complex algorithm where as Sign-Sign (SS) algorithm is the simplest one. However, the performance of the NLMS algorithm is far better as compared to SS algorithm. The LMS algorithm is a compromise between the above two extreme algorithms. It is neither too complex like NLMS algorithm nor its performance is too poor like SS algorithm. The other variants of Sign algorithms viz. Sign-Data (SD) algorithm and Sign-Error (SE) algorithms are more complex than SS algorithm but less complex than LMS algorithm. The performance of SD algorithm is at par with LMS algorithm but the performance of SE algorithm is poor and it is at par with SS algorithm. The BLMS algorithm [11] is an enhanced version of the LMS algorithm which is used to increase the convergence rate without affecting performance capability by performing filtering operations of different block simultaneously whereas DLMS [12-13] is a pipelined version of the LMS algorithm which is used to reduce the power consumption of adaptive filter.

Length of adaptive filter is also important as it changes the number of iterations required to minimize error signal. Larger the length lesser will be the number of iterations to minimize error and vice-versa [1]. Since, the adaptive filters are generally used in real time scenarios where the input and desired signal are not co-related and hence number of iterations to minimize the error is on higher side. Thus, in order to decrease the number of iterations and increase the convergence speed, the length of the adaptive filter is on higher side.

Furthermore, the structure of adaptive filter plays an important role in filtering like basic (Direct-Form) adaptive filter have better filtering performance over transposed form adaptive filter whereas transposed form adaptive filter have smaller delay as compared to direct form adaptive filter.

1.2 Why Adaptive Filters

Adaptive filters are required when some parameters of the desired processing operation are not known in advance. In other words it can be said that adaptive filters are used in the situations where the statistics of the signal are unknown or changes with time. This happens mostly in the case of random or non-deterministic signals. Thus, it can be said that adaptive filters are used mostly for de-noising of random or non-deterministic signals [1], [5]. As most of the random or non-deterministic signals occur in real time situations such as in case of an airplane cruise where the noise characteristics continuously changes with planes speed, height and environmental conditions. In such a case it is imperative to remove noise from the pilot's mic, so that there is a proper communication between the pilot and air traffic control (ATC) tower. Adaptive filters come handy in these situations. Thus, an adaptive filter may be understood as a self-modifying digital filter that adjusts its coefficients in order to minimize an error function.

1.3 Applications of Adaptive Filters

There are four main applications of adaptive filters [14] which are explained below:

System Identification [15]: The adaptive system identification is primarily responsible for determining a discrete estimation of the transfer function for an unknown digital or analog system. The same input x(n) is applied to both the adaptive filter and the unknown system from which the outputs are compared Fig. 1.2. The output of the adaptive filter

y(n) is subtracted from the output of the unknown system resulting in a desired signal d(n). The resulting difference is an error signal e(n) used to manipulate the filter coefficients of the adaptive system trending towards an error signal of zero. After a number of iterations of this process are performed, and if the system is designed correctly, the adaptive filter's transfer function will converge to, the unknown system's transfer function. For this configuration, the error signal does not have to go to zero, although convergence to zero is the ideal situation, to closely approximate the given system. There will, however, be a difference between adaptive filter transfer function and the unknown system transfer function if the error is nonzero and the magnitude of that difference will be directly related to the magnitude of the error signal.

Adaptive Noise Cancellation [16]: In this configuration as shown in Fig. 1.3 the input x(n), a noise source $N_1(n)$, is compared with a desired signal d(n), which consists of a signal s(n) corrupted by another noise $N_0(n)$. The adaptive filter coefficients adapt to cause the error signal to be a noiseless version of the signal s(n). Both of the noise signals for this configuration need to be uncorrelated to the signal s(n). In addition, the noise sources must be correlated to each other in some way, preferably equal, to get the best results. Due the nature of the error signal the error signal will never become zero. The error signal should converge to the signal s(n), but not converge to the exact signal. The only option is to minimize the difference between those two signals.

Adaptive Linear Prediction [17]: This is a configuration of adaptive filter that is used to predict future values of a signal from its past values. It is shown in Fig. 1.4. To achieve this goal the same input signal is applied to both the input port and reference or desired port i.e. d(n) = x(n). Also, the input to the adaptive filter is the delayed version of the system input. The adaptive filter coefficients are being trained to minimize the error and hence predict the next input signal from the statistics of the input signal x(n).

Adaptive Inverse System Configuration [18]: This configuration as shown in Fig. 1.5 is used to model the inverse of the unknown system u (n). Here, the input x(n) is passed through the unknown filter u(n). The output of this unknown filter is passed through the adaptive filter resulting in an output y(n). The input signal is also passed through a specified delay to get its delayed version d(n). The error signal e(n) is the difference between d(n) and y(n). As the error signal starts converging towards zero, the adaptive filter coefficients w(n) also gets converged to the inverse of the unknown system u(n). In

this configuration, the error can theoretically go to zero. This will only be true, if the unknown system consists only of a finite number of poles or the adaptive filter is an IIR filter. If neither of these conditions is true, the system error will converge only to some constant due to the limited number of zeroes available in an FIR system. This configuration is particularly useful in adaptive equalization where the goal of the filter is to eliminate spectral changes that are caused by a prior system or transmission line.

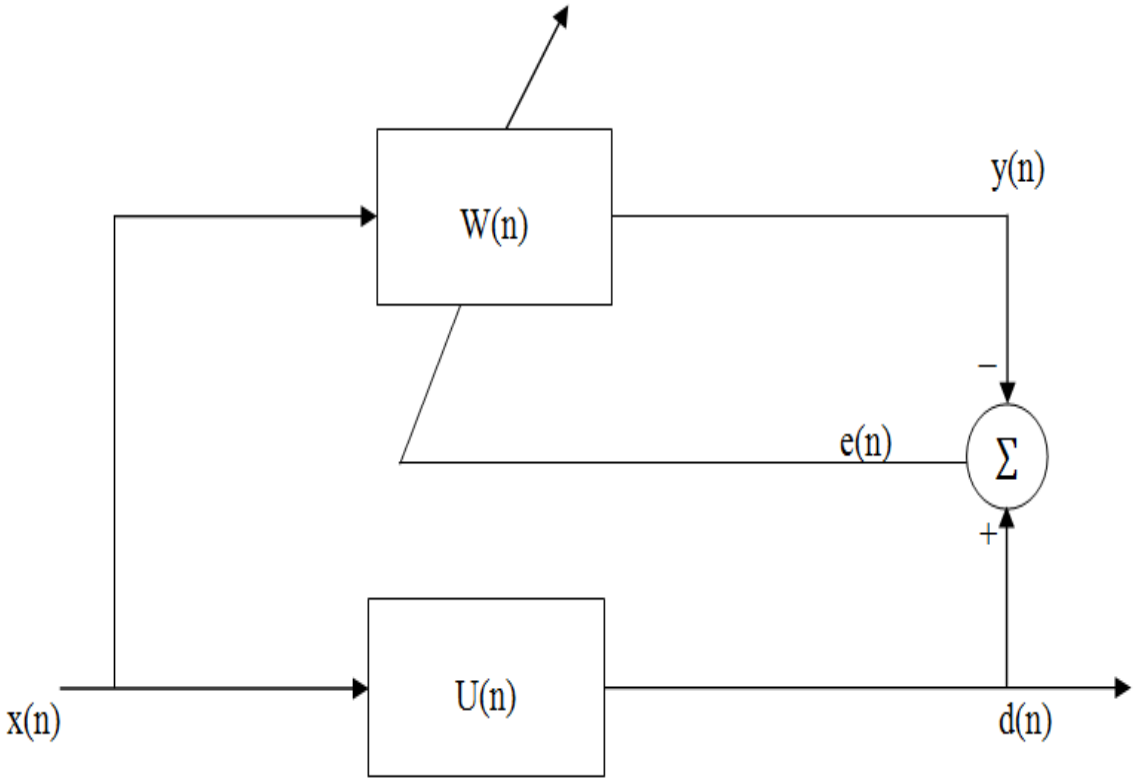

Fig. 1.2 System identification configuration [14], [15].

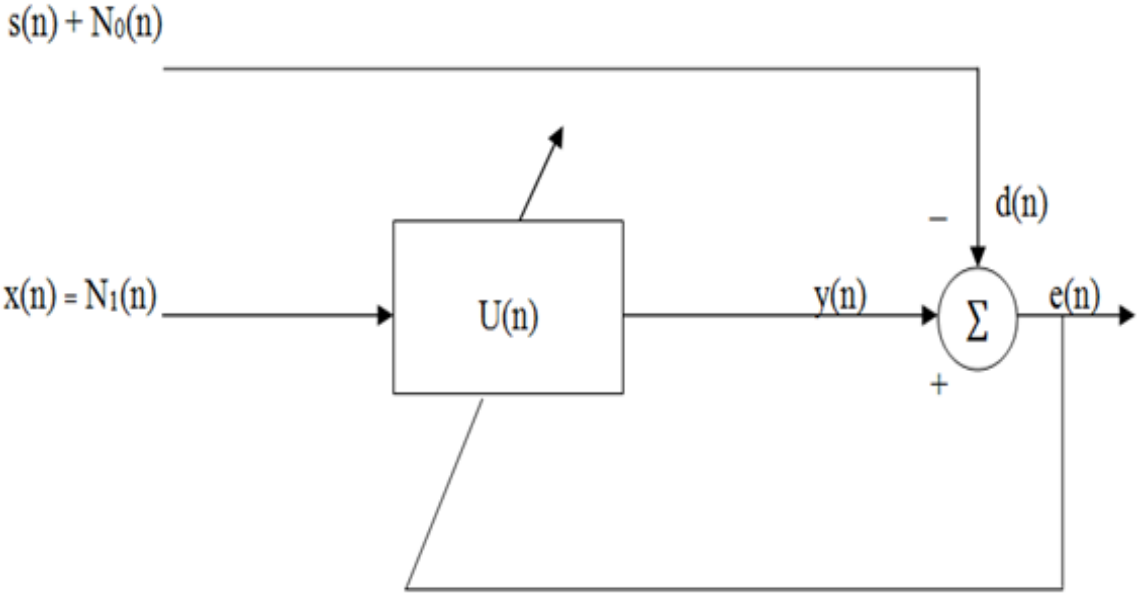

Fig. 1.3 Adaptive Noise Cancellation System [14], [16].

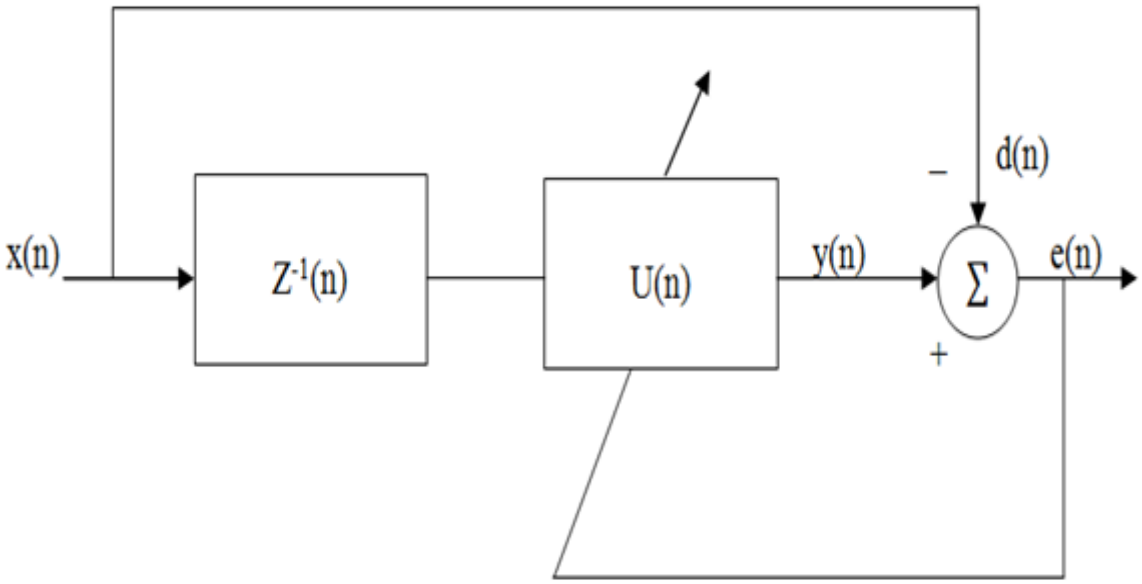

Fig. 1.4 Adaptive Linear Prediction [14], [17].

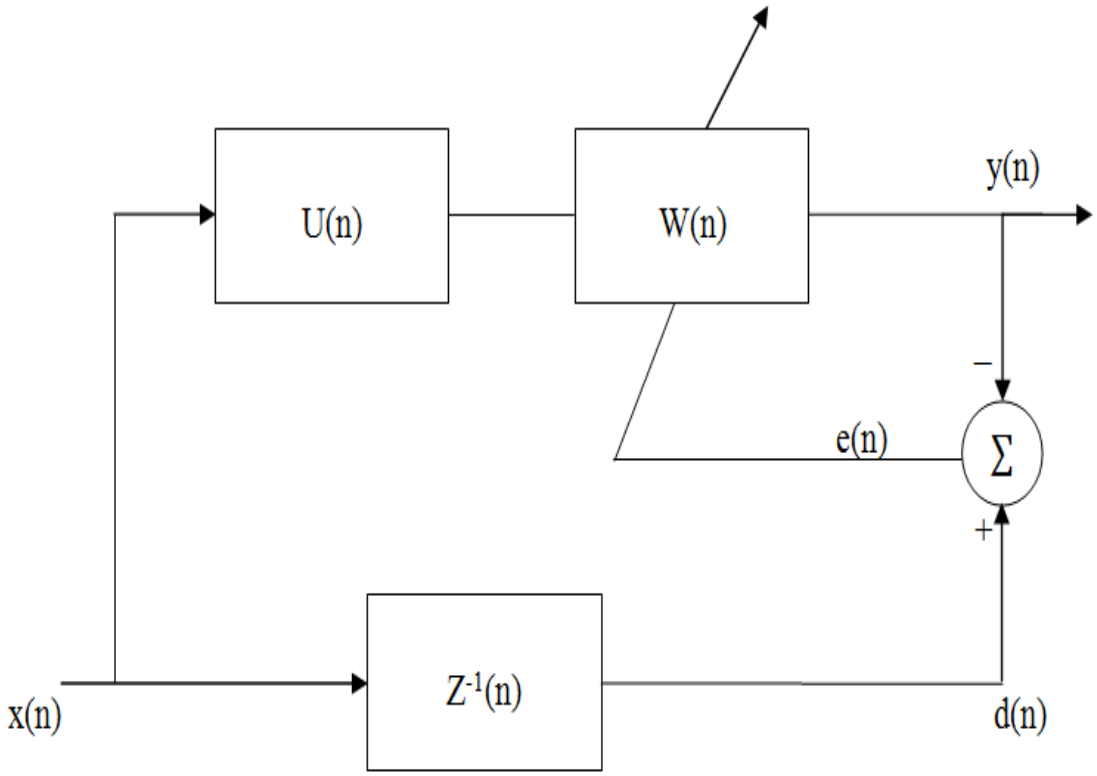

Fig. 1.5 Adaptive inverse system [14], [18].

1.4 Literature Review

B. Widrow *et. al.* (1975), [19], explains the concept of adaptive noise canceller (ANC). It consist of two signals first one is noise corrupted signal which is a primary input and second one is reference input which is a noise correlated version of primary noise. The reference input is adaptively filtered iteratively and is subtracted from the primary input

to get the desired signal estimate. They further showed that in case of periodic disturbances the ANC acts as a notch filter having narrow bandwidth, infinite null and ability of tracking exact frequency of interference whereas in case of random signals the ANC acts as linear time-invariant (LTI) system and adaptive filter converges on dynamic solution rather than static one.

B. Widrow *et. al.* (1976), [20], presented the performance and learning characteristics of LMS adaptive filter. They showed that when the inputs are stochastic the mean square error (MSE) which is the difference between the input signal and the desired or reference signal is a quadratic function of filter coefficients or weights and when the inputs are fixed, the MSE is a paraboloid function of filter weights. Also, for non stationary inputs, a misadjustment (deviation from optimal Wiener performance) arises which is due to ‘‘lag’’ of the adaptive process in tracking the moving minimum point and for stationary inputs LMS algorithm, approaches the theoretical limit of misadjustment and speed of adaptation.

D. L. Jones *et. al.* (1992), [21], had implemented a transposed-form LMS adaptive filter and compared its performance and convergence behaviour with direct-form adaptive filters. They proposed a method that can be used for determining the maximum convergence factor. It was found that the transposed-form LMS adaptive filter convergence behaviour was similar to the Delayed LMS adaptive filter. It lies between that of the conventional LMS adaptive filter and the delayed LMS adaptive filter with a delay equal to the maximum delay in the transpose-form LMS adaptive filter.

D. T. M. Slock (1993), [22], had compared the convergence rate of LMS algorithm with that of normalized LMS (NLMS) algorithm. He proposed a simple model for the input signal vector which simplified analysis of the convergence behaviour of above mentioned algorithms and concluded that the NLMS algorithm have faster convergence rate than the corresponding LMS algorithm, when the adaptive filter design is based on limited knowledge of input signal statistics. He also, deduced that the convergence speed of NLMS algorithm based adaptive filter can further increased by using a variable step size adaptive filter.

E. Eveda (1994), [23], had compared the performance of three different adaptive filters viz. least mean square (LMS), recursive least square (RLS) and sign algorithm (SA). The

parameters of comparison are steady-state excess mean-square estimation error ζ and the steady-state mean-square weight deviation, η. He showed that for LMS and SA algorithm, η does not depends upon the spread of the eigen values of the input covariance matrix, R, whereas it does depends for RLS algorithm and η is a monotonic increasing function of eigen spread for all the three algorithms. Also, for the optimum value of adaptation parameter LMS and RLS algorithm converges exponentially whereas SA converges linearly.

N. R. Shanbhag *et. al.* (1997), [24], had presented low power and high speed adaptive filter architectures. These architectures are obtained by the application of algebraic and algorithm transformations. In order to reduce the power dissipation strength reduction transformation was applied at algorithmic level which was previously applied at algorithmic level. The result was a decrease in the power dissipation by 21% as compared to conventional structures. The proposed structure was then pipelined by the means of relaxed lookahead transformation. This results an increase in the operating speed with a small hardware ovherhead. Also, the pipelined structure further reduces the power dissipation. Thus, by combining both of these techniques a power reduction in the range of 60-90% can be achieved.

S. J. Visser *et. al.* (2002), [25], had presented the use of Field Programmable Gate Array (FPGA) in on-board satellites to remove noise and interference from data signals as they are flexible, scalable and gives high performance. He firstly, implemented adaptive filters (prototype) on FPGA and after getting some useful results he implemented a finite impulse response (FIR) LMS adaptive filter on HPC-I (High Performance Computing) which was a payload in the Australian satellite FeDSat. The purpose was to find the performance of FPGA in digital signal processing for space applications.

A. Elhossini *et. al.* (2006), [26], had proposed three different architectures for implementing 16 bit fixed point LMS adaptive filter. These architectures are designed using Xilinx multimedia board having an on-board AC97 audio codec which can be used as audio processing system (audio capture/playback) and are using implemented Virtex II FPGA. Further, these designed architectures with different filter lengths were compared for area and performance. The results obtained showed that there is a 90% reduction in the critical path when a hardware accelerator is used which results an increase in the speed 3.86 times as compared to conventional architectures. However, when a pure

hardware implementation is used, then the performance is further increased i.e. speed increases 82.6 times as compared to conventional structures with slightly decrease in performance.

M. Vella *et. al.* (2006), [27], had proposed a high speed FIR LMS adaptive filter that is robust and stable to noise (echo tail generated), increasing data rates and have more processing power. The proposed adaptive filter was then used as a Line Echo Canceller (LEC) and further was implemented on a Xilinx Spartan 3 FPGA. The implemented adaptive filter results showed that the area and speed are optimized as compared to conventional filters.

Z. Gao *et. al.* (2008), [28], had presented the field programmable gate array (FPGA) implementation of an infinite impulse response (IIR) adaptive filter combined with the particle swarm optimization (PSO) technique. The PSO technique conducts an organised random search of unknown parameters by guiding a population of parameter estimates to converge on a optimum solution. The advantage of this technique is that it is independent of the structure of the adaptive filter and is capable of converging on the comprehensive solution, which makes it useful for optimizing non-linear and IIR adaptive filters.

R. Mustafa *et. al.* (2009), [29], had designed a 64-tap 9-bit signed integer coefficients, LMS adaptive FIR filter that can be used as an active noise controller (ANC). This designed ANC was used to remove a 24 KHZ uniform random noise signal. The architecture of the adaptive filter is based on the multiply-adder structure which is used to perform multiply and accumulate (MAC) operation for the FIR adaptive filter. The designed ANC was synthesized by using Altera Quartus II and later implemented on Altera Cyclone II FPGA. Performance of the implemented adaptive filter was compared for different step sizes and it was found that at $1/2^{10}$ step size fastest convergence speed of 1.46 ms was achieved.

Y. Mollaei (2009), [30], had designed a normalized least mean square (NLMS) adaptive filter that can be used to remove noise from the corrupted audio signals. The designed NLMS adaptive filter was then implemented Texas Instruments TMS320C6711 digital signal processor having an embedded target for TI C6000 SIMULINK toolbox along with a real-time workshop to perform hardware adaptive noise cancellation.

A. R. Muñoz *et. al.* (2011), [31], has proposed and adaptive algorithm that is robust toimpulsive noise and implemented it on Xilinx FPGA using two approaches. The first FPGA implementation was based on VHDL. In this approach a finite state machine (FSM) model of the adaptive filter was designed which was then used to write the VHDL code. The written VHDL code was then synthesized using Xilinx ISE and implemented on FPGA. The second FPGA implementation uses Simulink, Xilinx System Generator (XSG) high level synthesis (HLS) tool and Xilixn ISE. In this approach the adaptive filter was implemented using XSG block sets and later by using Xilixnx ISE it was implemented on FPGA. The performance of the implemented adaptive filter was then compared in terms of accuracy, performance and logic operation.

F. Nekouei *et. al.* (2012), [32], has presented the hardware implementation conventional LMS adaptive filter having fixed step size and self correcting adaptive filter (SCAF) on a spartan3 XC3S400 Field Programmable Gate Arrays (FPGA). They compared the performance of both the structures in terms of convergence rate, hardware utilization and maximum operating frequency, silicon area consumed and power dissipation and found that the proposed structure performs better in all the above mentioned aspects except power dissipation.

I. Tudosa *et. al.* (2012), [33], had proposed a twelve coefficients LMS adaptive filter structure and implemented in on the Cyclone II FPGA. The proposed adaptive filter was then used to remove the power line interference from electrocardiogram (ECG) signal in real-time. Further, it was observed that the proposed structure performs better than the conventional structure to process the ECG signal in real time.

S. Choudhary *et. al.* (2012), [34], has proposed a sign-based adaptive filter which is free from multiplier in the filtering part of the adaptive filter. For this they represented the coefficients of adaptive filters using sum power of two (SPT) which reduces the complexities of multipliers to just a few shifts and add operations.

A. B. Diggikar *et. al.* (2012), [35], had designed an LMS based adaptive noise canceller (ANC) using feed forward system that can be used to reduce unwanted noise from a single source speech signal. A feed forward system is specified by two inputs per channel. First input is the reference signal which contains the sound (noise) signal that needs to be removed, and second input is the error signal which is used for the compensation of the

sound (noise corrupted) signal. A finite state machine (FSM) model of the ANC system was designed which was then used to write the corresponding VHDL code. This VHDL code of the ANC system was then used to implement the system on FPGA.

M. Z. U. Rahman *et. al.* (2012), [36], has implemented sign and error non-linear adaptive filters which are free from multipliers in the weight updation loop. The implemented adaptive filters can be used to remove noise from the electrocardiogram (ECG) signal. Because of the absence of multipliers in the weight updation loop the speed of the implemented adaptive filters are higher as compared to conventional LMS based adaptive filter. The proposed adaptive filters are best suited for applications for which lower computational complexities are required. Further, it was observed that the proposed adaptive filter have higher operating speed and better signal to noise ratio than the conventional adaptive filters.

G. Cai *et. al.* (2013), [37], has proposed an adaptive filter and implemented it on Quartus II FPGA. The proposed design was divided into four modules viz. control module, error calculation module, weights calculation module and storage module. The control module is the central module of the system, which is divided into two sub modules: first one is amicro-code control unit and the second one is a control unit for standard hardware implementation. The function of micro-code control unit is to generate a control signal for process control and program storage whereas the function of control unit is to generates control codes to achieve control of the system. The function of error calculation module is to compute the output error which can be used to adjust the filter weights. It mainly consists of adders, multipliers, multiplexers and registers. The function of weight calculation module is to calculate and update weights after each iteration. It mainly consists of multipliers, adders and registers. The function of storage module is to store and update the signal and weights. It mainly consists of dual port RAM.

V. Ramakrishna *et. al.* (2013), [38], has reported low power VLSI implementation of least mean square (LMS) adaptive filters that was used as a adaptive noise canceller (ANC). The ANC model was coded in verilog, simulated in ModelSim to check the for functional and timing correctness and finally synthesized using Xilinx ISE. The synthesized model was then implemented on Xilinx XC3s200 field programmable gate array (FPGA). It was found that the proposed ANC model consumes low power (0.156 W) which is complete simulation power) as compared to the conventional ANC.

1.5 Motivation and Objectives

It can be inferred from the literature survey presented above that there is still ample scope left for analyzing LMS variants for different types of inputs. Furthermore, the variants performing satisfactorily for a given filtering tasks can be evaluated for hardware implementability. This had served as the primary motivation for the author to undertake an extensive study (both in terms of computational performance and in terms of hardware implementability) of the popular LMS variants. Out of the several derivatives of the LMS algorithm proposed in the literature survey so far, the author was motivated to choose the following four viz. Normalized LMS, sign-data (SD), sign-error (SE) and sign-sign (SS) because of following reasons:

1) The pure LMS algorithm is sensitive to the scaling of its input signal x(n). This makes the choice of an optimum learning rate a tough one which is not the case in NLMS algorithm.

2) The sign algorithm and its variants viz. SD, SE and SS are preferred over conventional LMS algorithm because of their simplicity in implementation.

In a nut shell this dissertation attempts to achieve the following **objectives**:

1) To undertake performance analysis and comparison of LMS algorithm variants for specific de-noising tasks.

2) To design an adaptive filter (ANC) on SIMULINK and simulate it to check its functionality in de-noising of sine wave and acoustic signal.

3) To describe the complete functionality of the adaptive filter by using a hardware description language (VHDL) and verify it by simulating on ISim Simulator.

4) To synthesize the designed adaptive filter using Xilinx ISE (FPGA synthesizer) and Leonardo Spectrum (ASIC synthesizer) so that it can further be implemented on FPGA or as an ASIC.

5) To compare implementation results on three different FPGA device families.

6) To perform timing, resource utilization and power analysis by using Xilinx ISE 14.5 and Xpower Analyzer /Estimator.

7) To draw the schematic and layout of the designed system on 250 nm technology, check its DRC and LVS, and perform simulations.

1.6 Novel Aspects of this Dissertation

The work presented in this dissertation claims novelty on several accounts some of which are enlisted as follows:

1) Performance analysis in both computational and hardware domains. To the best of author's knowledge this study has been undertaken for the first time in both the domains.

2) Design of a novel filter structure for critical path reduction and speed improvement.

3) Reduction in resource utilization without incurring any significant speed or power penalty.

1.7 Organization of Dissertation

As explained in the previous sections, the aim of the work is to design and implement a simple, efficient, low-power and high-speed adaptive filter that can be used as adaptive noise canceller (ANC) to remove noise from sinusoidal and audio signals. In order to achieve the same, the dissertation has been divided into the following six chapters:

Chapter 2 gives an overview of different adaptive algorithms that can be used in designing an adaptive filter.

Chapter 3 describes the simulation platforms and the tools used in the due process of design and implementation of the adaptive filters.

Chapter 4 outlines different structures including the proposed structure for implementing different variants of adaptive algorithms that are used in designing adaptive filter.

Chapter 5 describes the results and discussion. The results are divided into three parts: Simulation results of LMS variants, FPGA synthesis results, and ASIC synthesis results. These results illustrate the functionality, device utilization summaries and the power consumption analysis of the different variants of LMS algorithm implemented using all the three structures.

Finally, Chapter 6 sums up the conclusions of the work and suggests some ideas for the future work.

Chapter 2
Introduction to Adaptive Algorithms

Adaptive algorithms form the backbone of ubiquitous computing and are increasingly being applied to a plethora of fields ranging from fault diagnosis to noise removal [4], [39]. Adaptive algorithms are based on the minimization of a certain cost function (optimization problem). The most widely used cost function is the quadratic error. This cost function is optimum when errors follow a Gaussian distribution. There are a large number of adaptive algorithms such as LMS, NLMS, DLMS, BLMS, RLS, Leaky LMS, Sign-Data (SD), Sign-Error (SE) and Sign-Sign (SS) etc. The selection of an adaptive algorithm depends upon various factors such as filtering performance, application, hardware utilization, delay, clock speed and power. This chapter concentrates mainly on four algorithms viz. LMS, SD, SE and SS.

2.1 LMS Algorithm

The Least Mean Square (LMS) algorithm was proposed by Widrow and Hoff in 1960 [1] which is based on steepest decent method [4], [6]. LMS algorithm works by estimating the gradient of the input vector. It uses an iterative procedure that updates the coefficients or weight vector in the direction of negative of the gradient vector which eventually leads to decrease in the mean square error (MSE). This algorithm is more popular as compared to other adaptive algorithms due to its inherent simplicity, stability and satisfactory convergence performance. For each input sample, the LMS algorithm calculates an output, finds the error which is the difference between the calculated output and the desired response, and finally uses that error to update the weights or coefficients in every cycle.

Consider an N order LMS adaptive filter having input x(n) and weights or coefficients w(n). At the end of the n^{th} iteration the input x(n) and weights w(n) have the values

$$x(n) = [x(n), x(n-1), \ldots\ldots x(n-N+1)]^T \quad (2.1)$$

$$w_n = [w_n(0), w_n(1), \ldots., w_n(N-1)]^T \quad (2.2)$$

By using the property of convolution the output y(n) of the adaptive filter at some discrete time n which is given by following equation

$$y(n) = \sum_{k=0}^{N-1} w(n)\, x(n-k) \quad (2.3)$$

The output of the adaptive filter as represented by equation (9) can also be represented in vector form as

$$y(n) = w^t(n).x(n) \quad (2.4)$$

The error signal e(n) which is the difference between the desired or reference signal d(n) and filter output is given as

$$e(n) = d(n) - y(n) \quad (2.6)$$

On squaring error signal e(n) given by equation (11) we get the following equation

$$e^2(n) = d^2(n) - 2w^t(n).x(n) + w^t(n).x(n)\, x^t(n).w(n) \quad (2.7)$$

In order to optimize the adaptive filter, the mean square error (MSE) must be minimized. For this the MSE must be represented in terms of cost function J.

$$J = E[e(n).e^*(n)] = E[|e(n)^2|] \quad (2.8)$$

Taking gradient ∇ of the cost function, a gradient vector ∇J is obtained as

$$\nabla J(n) = -2P + 2Rw(n)$$

$$\text{or,} \quad \nabla J(n) = -2x(n)d(n) + 2x(n)\, x^t(n).w(n) \quad (2.10)$$

where, R is the autocorrelation matrix of x(n) and P is the cross correlation matrix of d(n) and x(n).

Consider a cost function J(w) i.e. a continuously differentiable function of some unknown weight vector w. To find an optimal solution suppose a initial weight w(0) that satisfies the condition

$$J(w(0)) \leq J(w) \quad \text{for all } w \quad (2.11)$$

Equation (2.11) is a mathematical statement of unconstrained optimization. As, w(0) is the initial weight (assumed) and after each iteration new weights w(1), w(2)…….w(n-1) are generated in such a way that the cost function J(w) is reduced at each iteration. Thus,

$$J(w(n+1)) \leq J(w(n)) \quad (2.12)$$

where, w(n) is the value of weight at previous iteration and w(n+1) is the present value of weight.

Substituting equation (2.10) in steepest decent algorithm equation a new recursive equation is obtained for updating weights which is given below

$$w_{n+1} = w_n + \mu.e(n).x(n) \quad (2.13)$$

Equation (2.13) is the governing equation for the LMS algorithm which is used for updating weights [35]. Here a new parameter μ which is known as step size is introduced

in the equation. The function of this parameter is to control the step width of the iteration and thus controls the stability and convergence speed of the algorithm [4], [5].

In case of LMS algorithm MSE is a quadratic function of filter weights which means that there is only one extrema (optimum weight) that minimises MSE to optimum value. It approaches towards these optimum weights by ascending/descending down the MSE vs. filter weight curve [5].

2.2 Normalized LMS (NLMS) Algorithm

In standard form of LMS algorithm, the correction μ.e(n).x(n) applied to tap vector w(n) at iteration n+1 is directly proportional to the tap input vector x(n). Therefore when x(n) is large, the LMS algorithm experiences a gradient noise amplification problem. To overcome this difficulty, NLMS algorithm is used. In this, the correction μ.e(n).x(n) applied to tap vector w(n) at iteration n+1 is "normalized" with respect to the squared Euclidean norm of the tap input vector x(n) at iteration n, hence the term "normalized". The NLMS algorithm updates the coefficients of an adaptive filter using the following equation

$$w_{n+1} = w_n + \frac{\mu}{\|x(n)\|^2}.e(n).x(n) \tag{2.14}$$

where, ‖x(n)‖ Euclidean norm of the tap input vector x(n). The NLMS updates the tap-weight vector in such a way that the value w(n+1) computed at time n+1 exhibits the minimum change (in a Euclidean norm sense) with respect to the known value w(n) at time n [30].

2.3 Sign LMS Algorithm

The least-mean-square (LMS) adaptive filtering algorithm is very popular because of its simplicity. However, there are many applications for which even simpler approaches are required in order to implement the adaptive algorithm in real-time. Consequently, the sign algorithm and its variants have been actively studied in recent years. Sign based LMS [34] algorithm are truncated version of LMS algorithm in which the signum function is applied either at input signal vector or at error signal vector or at both the signal vector. This reduces the computing time and dynamic range requirements by turning multiplications into bit shifts.

The signum function, defined by equation (2.15), can be used to simplify the standard LMS algorithm.

$$\text{sign}(x) = \begin{cases} 1; & x > 0 \\ 0; & x = 0 \\ -1; & x < 0 \end{cases} \quad (2.15)$$

Applying the sign function to the standard LMS algorithm returns the following three types of sign LMS algorithms namely, the sign-data, the sign-error and the sign-sign algorithm.

Sign-Data (SD) Algorithm:- In this algorithm the sign function is applied to input signal vector x(n). This algorithm updates the coefficients of an adaptive filter using the following equation:

$$w(n+1) = w(n) + \mu \text{sign}(x(n))e(n) \quad (2.16)$$

From equation (2.16) it can be observed that when input signal vector x(n) is zero, then this algorithm does not involve any multiplication and addition operation and when input signal vector x(n) is non-zero then this algorithm involves one multiplication and one addition/subtraction operation.

Sign-Error (SE) Algorithm:- In this algorithm the sign function is applied to the error signal vector e(n). This algorithm updates the coefficients of an adaptive filter using the following equation:

$$w(n+1) = w(n) + \mu x(n)\text{sign}\big(e(n)\big) \quad (2.17)$$

From equation (2.17) it can be found that when error signal vector e(n) is zero then this algorithm does not involve multiplication and addition operation whereas when the error signal vector e(n) is not zero, this algorithm involves one multiplication and one addition/subtraction operation.

Sign-Sign (SS) Algorithm: - In this algorithm the sign function is applied to both the input signal vector x(n) and error signal vector e(n) This algorithm updates the coefficients of an adaptive filter using the following equation

$$w(n+1) = w(n) + \mu \text{sign}(x(n))\text{sign}\big(e(n)\big) \quad (2.18)$$

From equation (2.18) it can be seen that when either error signal vector e(n) or input signal vector x(n) or both signal vectors are zero then this algorithm does not involve multiplication and addition operation whereas when neither error signal vector e(n) or input signal x(n) are zero, then this algorithm involves only one addition/subtraction operation.

2.4 Performance Measures in Adaptive Algorithms

There are seven important parameters of adaptive algorithms which can affect their performance [8], [20-22], [33]. These are step size, convergence rate, minimum mean square error, computational complexity, stability, robustness, and filter length. Each of these parameters are discussed below in breif:

Step Size: It is one of the most important parameter of the adaptive algorithm and can affect the performance of the adaptive filter significantly. It gives us an idea about the rate by which the input signal and mean square error gets changed after each iteration. Proper selection of the step size is important as it affects other filter parameters too.

Convergence Rate: The convergence rate of an adaptive algorithm is the rate at which the adaptive algorithm reaches its its final value. However, it is not independent of all other performance parameters. There is a trade-off between the convergence rate and other performance parameters. For example, increasing the convergence rate will increase the step size and decrease the stability of the adaptive algorithm and decreasing it would decrease the step size and increase the stability of the adaptive algorithm.

Mean Square Error (MSE): It is a parameter that indicates how a system can adapt to a given solution. A small MSE indicates that the adaptive filter has been precisely modelled, predicted, adapted and/or converged to the optimum solution for the given adaptive system whereas a large MSE usually indicates that the adaptive filter cannot be accurately modelled for the given adaptive system or the initial state of the adaptive filter is a insufficient or sparse starting point that cause the adaptive filter to diverge converge instead of converging to a optimum solution. MSE strongly depends upon the step size of the adaptive filter. If the step size is not optimum then the MSE would be high as the weights of the coefficients of the adaptive filter will not converge to a optimum value or diverge from the optimum value whereas for an optimum step size MSE would be small.

Computational Complexity: The most fundamental measure of computational complexity of an adaptive algorithm is the number of iterations required to minimize the MSE. It is particularly dependent upon the choice of the adaptive algorithm and the type of the input signal. For example, of all the adaptive algorithms explained in the previous sections NLMS algorithm computational complexity is highest whereas SS algorithm has

the lowest computational complexity. It is particularly important in real time adaptive filter applications where there are hardware limitations that may affect the performance of the system.

Stability: Stability is perhaps the most important performance criteria for an adaptive algorithm. Normally, there are only a few adaptive systems that are completely asymptotically stable systems. However, in general the adaptive systems that are implemented are marginally stable, with the stability determined by the initial conditions, transfer function of the system and the step size. The stability of an adaptive algorithm has an inverse relationship with the step size and the convergence rate.

Robustness: The robustness of an adaptive algorithm means whether a given adaptive algorithm works under given set of conditions or not. In other words, it is the ability of the adaptive algorithm to withstand both input and quantization noise. Robustness of the adaptive system is directly proportional to the stability of the adaptive system and is one of the important factor that can influence stability of a system.

Filter Length: The length of the adaptive filter is a measure of precision of the adaptive filter. It is one of the important parameter that not only can affect the adaptive algorithm performance but can also affect other performance parameters of the adaptive algorithm as it is inherently related to many of them. It can affect the convergence rate, by increasing or decreasing computation time, stability of the system, and minimum MSE. For example, if the filter length is increased then the number of computations will increase which In turn decrease the maximum convergence rate and vice-versa. In term of stability, an increase in adaptive filter length will add additional poles or zeroes that may be smaller than those that already exist. This will increase the stability.

Chapter 3
Simulation Platform and Implementation Tools

In this chapter various tools and platforms used for designing the adaptive filter have been discussed. The adaptive filter was first designed in SIMULINK using FDATOOL and various other blocksets that are available in the SIMULINK library. The designed adaptive filter was then simulated to find out whether it works in a proper way or not. After, the adaptive filter was designed in SIMULINK, it needs to be implemented on hardware (FPGA/ASIC). For, this VHDL code of the adaptive filter was generated using SIMULINK HDL coder using fixed point arithmetic. This VHDL code of the adaptive filter was then simulated on ISIM to check whether the code describes the complete functionality of the adaptive filter or not. After that the code was synthesized on Xilinx ISE, a programming file (bit file) was generated which was then downloaded on FPGA so that the design gets implemented on FPGA. The functionality of the implemented design on FPGA was then verified by ChipScope Pro. Also, the VHDL code of the adaptive filter was synthesized on Leonardo Spectrum which is an ASIC synthesizer using 250 nm technology. A synthesized file (.v file) was generated which was then imported on Mentor Design Architect (part of Mentor Pyxis Package) and was used for creating schematic of adaptive filter at 250 nm technology. Transient analysis was performed on this schematic to check out its behaviour. Further, its layout was designed using Mentor IC Station (part of Mentor Pyxis Package). DRC and LVS was performed on this schematic using Mentor calibre (part of Mentor Pyxis Package). At last post layout simulation was performed and parasitic were extracted. Subsequent sections would give a brief idea about the tools and platforms that were used for designing an adaptive filter.

3.1 Filter Design and Analysis Tool (FDATOOL)

It is a graphical user interface (GUI) that is used to design and analyze digital filters quickly. The digital filter in turn can further be used to design the adaptive filter. By using this tool a low pass FIR filter having filter length 31 and cut off frequency 0.5 radian per second was designed using hamming window method and fixed point arithmetic. Further, the designed filter was analyzed in terms of magnitude response, phase response, pole-zero plot, stability and cost of implementation by using various tools

available in it. Also, the VHDL code of the designed filter was generated using filter design HDL coder. The designed filter was then exported into SIMULINK where it can be used as an independent block in different systems.

3.2 Matlab and SIMULINK

Matlab platform was used to study the convergence rate and mean square error variations whereas SIMULINK was used to design the adaptive noise removal system, simulate the designed system to verify its functionality and to generate the VHDL code of the designed system. Besides this the effect of parameter variations like variations in step size, variation filter length was also observed during designing process on SIMULINK.

3.3 ISim Simulator

It is a verification and simulation tool for hardware description language (HDL) like VHDL, Verilog etc. In other words it can be said that it was used to verify and simulate HDL codes prepared by the programmer. The verification is done by performing timing simulations to indicate the results. Fig. 3.1 shows the basic steps for simulating a design using ISim simulator.

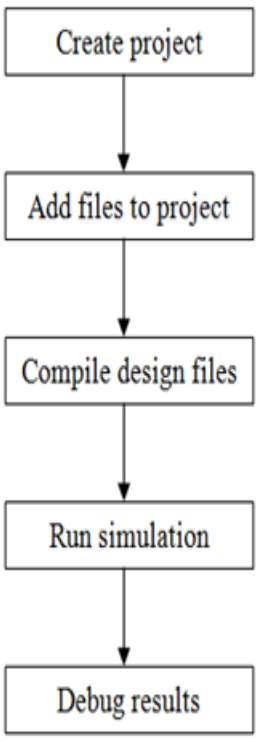

Fig. 3.1 Design Flow of ISIM Simulator.

3.4 Xilinx ISE

Xilinx ISE is a tool that was used to implement the design (HDL code) on the FPGA. In other words this tool interfaces and verifies the HDL codes for hardware implementation compatibility on FPGA devices. Fig. 3.2 shows the basic steps for the implementation of design on FPGA using Xilinx ISE 14.5. The first step in the design flow of this tool is creating a new project and specifying the target device. After that a source file (HDL) is added to the created project. Next the syntax of the added source file (design) is checked and after that it is synthesized. Here, it may be noted that some of the HDL syntax or constructs may not be synthesized by the software even though source file syntax is checked and its functionality is verified by the simulator.

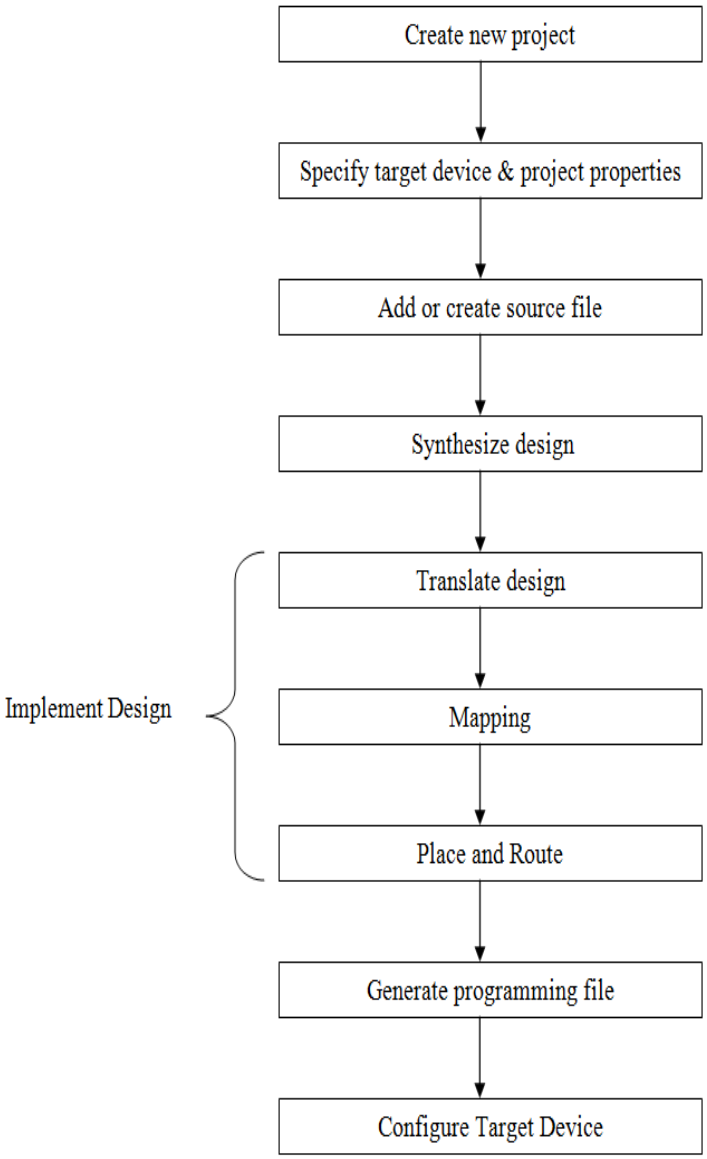

Fig. 3.2 Design Flow of Xilinx ISE for FPGA Implementation.

After the successful completion of the synthesis process, the RTL schematic and technology schematic of the synthesized code can be generated. Next the synthesized design code is implemented which involves three steps first one is translation, second one is mapping and third one is routing. After that a programming file (bit file) is generated and the target device is configured. This file contains the information about hardware that needs to be realized on the target device. The programming file is then downloaded on the FPGA and the functionality of the implemented design of FPGA is checked.

3.5 Field Programmable Gate Array (Xilinx ®)

FPGA's are the programmable logic devices whose logic characteristics can be changed by electrically programming the device. In other words FPGA is an integrated circuit that can be configured by the programmer after manufacturing, hence it is known as "field-programmable". The fundamental block of the FPGA that is used for the implementation of logic is look-up tables (LUTs).These LUTs can either act as function generator or can be configured as ROM or RAM. Different manufactures give different names to their basic unit block; however the fundamental unit is same for all of them which is LUTs. For example, Altera calls them logic element (LE) while Xilinx names them as configurable logic block (CLBs). These basic unit blocks are further connected to each other through programmable interconnects. The main advantage of FPGA is that it combines user control and time to market of PLDs with densities and cost benefits of ASICS.

There are three different technologies that are used for programming generic FPGAs. These are Antifuse, SRAM and EPROM or floating gate. Antifuse technology is generally used by Actel FPGAs. Antifuse is a two terminal device which offers high resistance path between its terminals when not programmed and low resistance path when programmed by applying high voltages between its terminals. The main advantage of this technology is small size whereas its drawback is extra circuitry is required for generating high voltage for programming antifuse. SRAM technology is used by Xilinx FPGAs. In this technology state of SRAM controls pass transistor or multiplexer outputs which are responsible for making connection between two wires segments. If a one is stored in the SRAM cell the pass transistor offers a low resistance path by acting as short circuit and eventually makes connection between two wires. If a zero is stored then transistor acts as open circuit and offers a high resistance path between two wires. SRAM based FPGAs

are volatile and hence must be configured each time when power is switched on. The main advantage of SRAM based FPGAs is its fast re-programmability whereas its drawback is its large area. EPROM based FPGAs uses floating gate transistor that must be turned off by injecting a charge on to the floating for making contacts between two wires. This injected charge increases the threshold voltage and the transistor is turned off. For disconnecting the two wires the transistor must be turned on. For this it is exposed to UV light so that its threshold voltage is reduced. This programming technology combines the advantage of both the previous technologies i.e. fast re-programmability and small size. Also, floating gate based FPGAs are non-volatile in nature [40]. Since the present work is based on Xilinx FPGAs its architecture (general) as shown in Fig. 3.3 has been discussed in brief in the next paragraph.

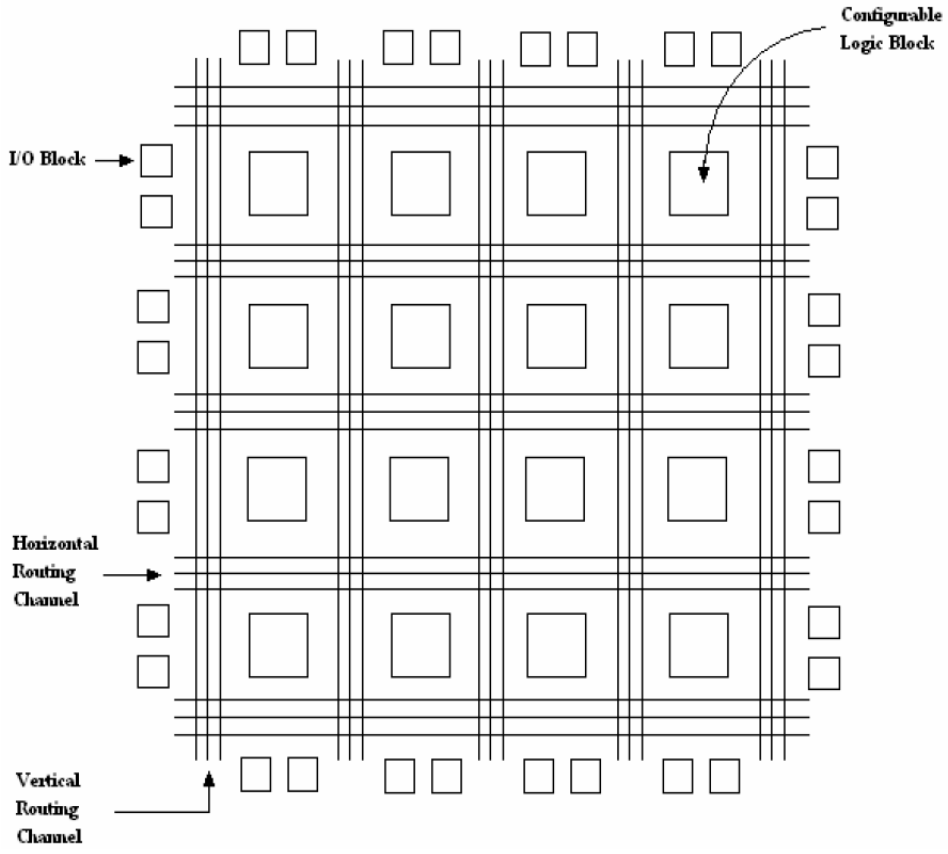

Fig. 3.3 Xilinx FPGA Architecture [40].

The Xilinx FPGAs are SRAM based devices that contain configurable logic blocks (CLBS) for the implementation of logic. The CLBS are formed by the combination of LUTs, flip-flops, multiplexers and gates. Each CLB can be configured (programmed) to implement any boolean function of its input variables. Typically CLBs have between 4-6 input variables. Functions of larger number of variables are

implemented using more than one CLB. These CLBs are generally placed in an island style arrangement and connected to each other through vertical and horizontal routing lines. The input and output pins of the CLBs are connected to vertical and horizontal routing lines through programmable connection boxes. Also the intersection of vertical and horizontal routing lines is called as programmable switch box. The present work is based on the implementation of the design on three different Xilinx FPGA devices (Spartan6, Virtex6 and Virtex7) has been used.

3.6 ChipScope Pro

As the density of FPGA devices is increasing, it is becoming very difficult or impractical to test these devices by attaching test equipment probes. ChipScope Pro comes handy in such situations. Thus, it is a tool that is used to check functionality of the logic implemented on the FPGA when the density of logic implemented is high or when there are large numbers of IOBs in a logic design. It works by integrating key logic analyzer hardware components with the logic design implemented on the device. This tool has five cores viz. Integrated Controller core (ICON), Integrated Logic Analyzer core (ILA), Virtual Input/Output core (VIO), Integrated Bit Error Ratio core (IBERT) and Agilent Trace Core 2 (ATC2).Out of the five cores mentioned above the two primary cores viz. ICON and VIO are responsible for successful operation of the tool. The VIO core is used to generate virtual inputs and outputs for the FPGA whereas ICON core is a controlling core that controls the operation of the VIO core.

3.7 Xpower Analyzer/Estimator

It is a tool that is used to calculate the total simulation power (static power and dynamic power) of the design that is implemented on the FPGA. This tool first considers the design's toggle rates, resource utilization, I/O loading etc. and then combines these factors with the target device models to calculate the power. The device models are extracted from measurements, simulation performed. The accuracy of the power calculated using this tool is dependent upon the input parameters such as clock, enable, toggle rates, device utilization etc. It also gives the variation of power with supply voltage and junction temperature, variation of different current components with corresponding voltages and also specifies different power components of the implemented design.

3.7 Leonardo Spectrum

It is a tool that is used to synthesize both ASICs and FPGAs but in the work it is used as an ASIC synthesizer. The first step for using this tool is to run the command spectrum in the command prompt. This will open the tool in the command prompt. Next run the command *set exclude_gates {PadOut PadInC}*. This command excludes the gates *PadInC* and *PadOut* from being imported when library is loaded for the synthesis of design. After that load the library (on which the design is synthesized 250 nm in this case) and then load the design files (VHDLfiles) which needs to be synthesized. Next optimize the synthesized design by typing *optimize* command. After that save the synthesized optimize netlist by *write name.v* where name specifies name of output file. At last area and delay report can be checked by typing commands *report_area* and *report_delay*. The complete flow of synthesizing an HDL code using Leonardo spectrum is shown in Fig. 3.4.

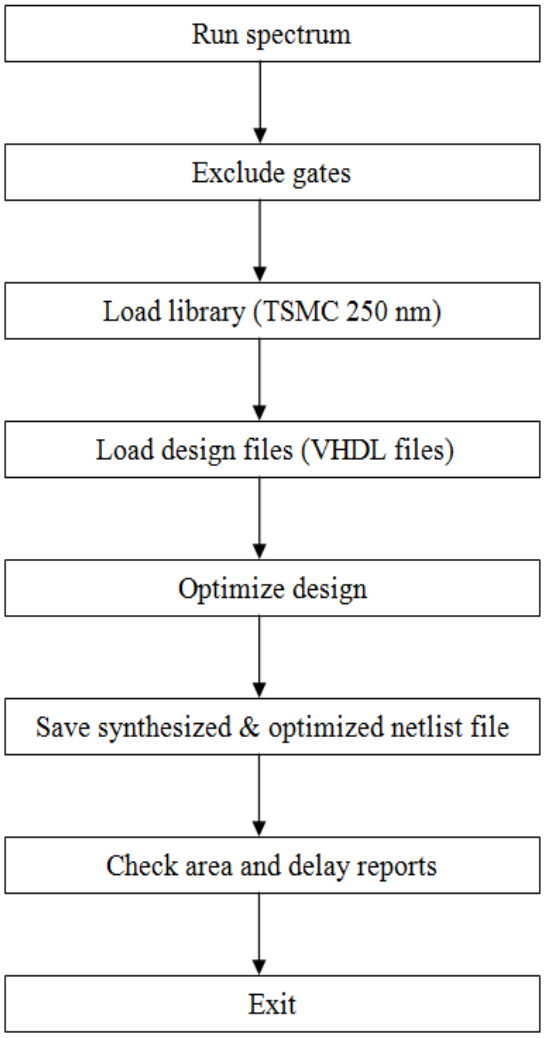

Fig. 3.4 Design Flow of Leonardo Spectrum.

3.8 Mentor Design Architect

It is a tool that was used to create the schematic of the design and perform simulations. After, the design was synthesized on Leonardo Spectrum a synthesized and optimized netlist file (.v file) was created which was then imported on this tool and was further used to create the schematic of the design. In this work the schematic was created on 250 nm technology and transient analysis was performed to check the functionality of design implemented.

3.9 Mentor IC Design

This tool was used to create the layout of the design (schematic) that was implemented on Design Architect.

3.10 Mentor calibre

This is a tool that was used to perform DRC and LVS of the design. Fig. 3.5 shows the steps for ASIC implementation.

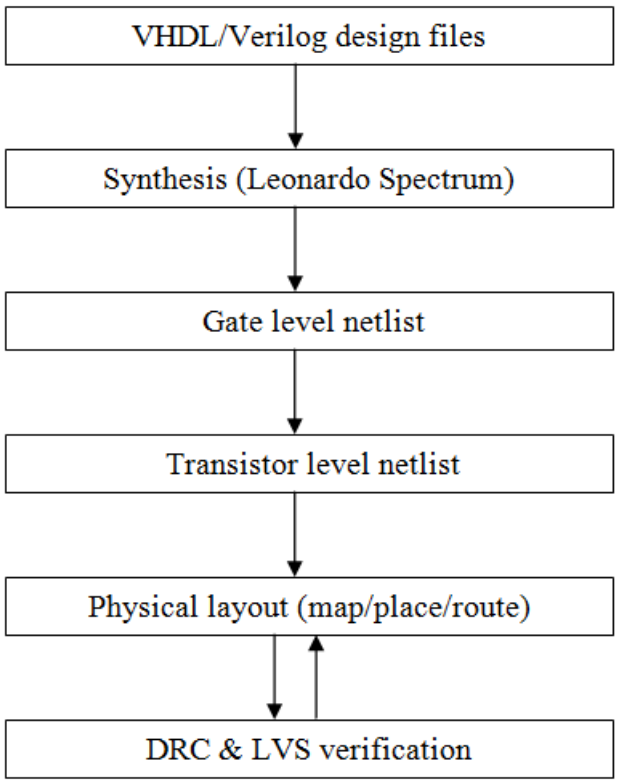

Fig. 3.5 Design Flow of ASIC Implementation.

Chapter 4

Adaptive Filter Structures and SIMULINK Models

There are two primary ways to design an adaptive filter in SIMULINK. First, one involves the use of digital filter to design adaptive filter whereas second one is based on the implementation of the weight update loop equation of adaptive algorithm using different structures. Both these design techniques are explained below.

4.1 Design and Implementation Using Digital Filters

This is the simplest technique used for designing an adaptive filter. It involves designing of a digital filter that can be further used to design an adaptive filter. The designed digital filter is cascaded with coefficients or weights updating (adaptive algorithm) block so that it could work as an adaptive filter. Any changes made in the digital filter eventually have its effect on the adaptive filter. In this present work, the digital filter was designed using three structures viz. direct form, transposed form and symmetric form and five updating (adaptive) algorithms viz. LMS, NLMS, SD, SE and SS were used with each of these three structures. The designed adaptive filter (acting as adaptive noise canceller ANC) was then used for de-noising of sinusoidal signal. A generalized structure of the implemented model is shown in Fig. 4.1 whereas Fig. 4.2 shows its SIMULINK implemented version.

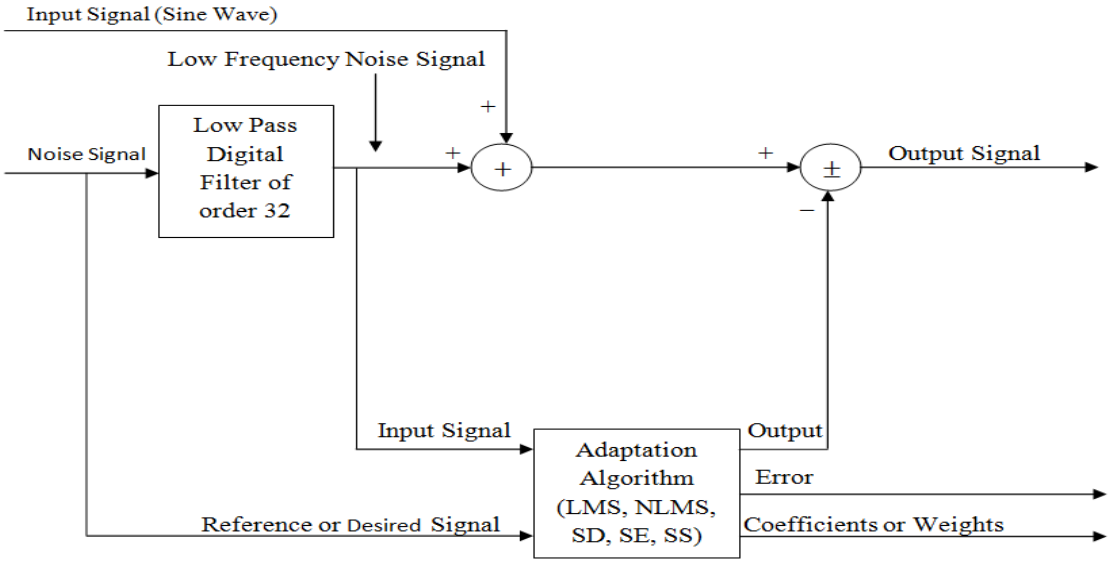

Fig. 4.1 Implementation of Adaptive Filter by using Digital Filter.

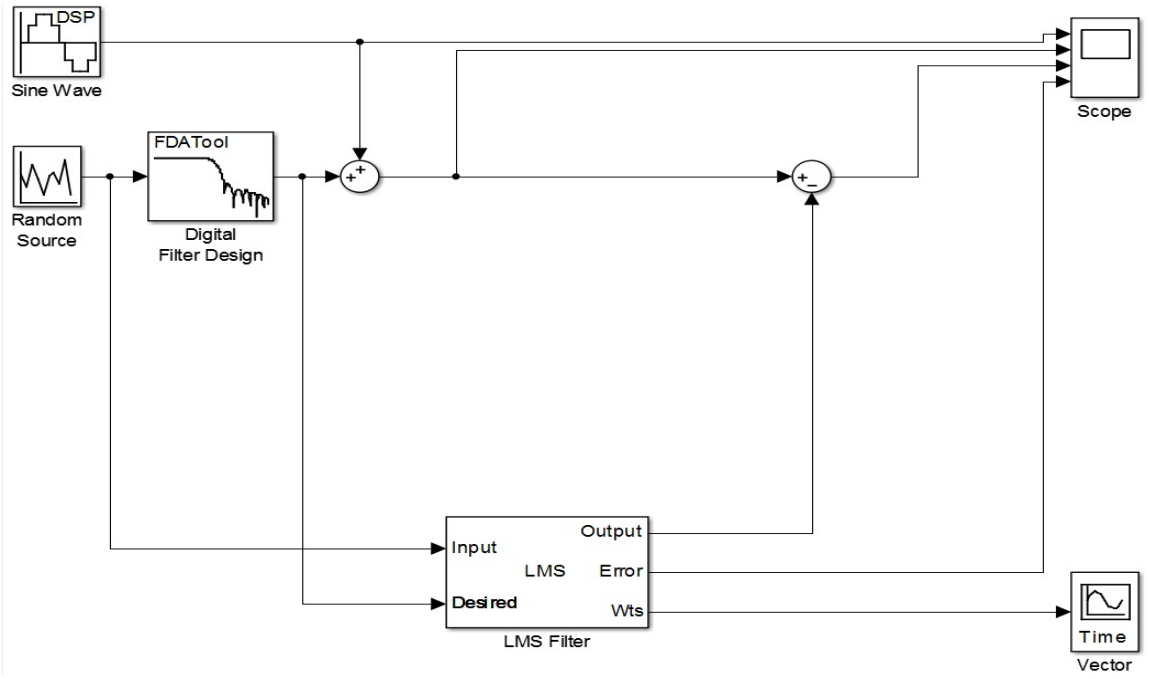

Fig. 4.2 SIMULINK Model of a 32 Order Adaptive Filter Implemented using Digital Filter.

The performance of the designed system implemented using different digital filter structures and updating algorithms was then compared in terms of de-noising performance and hardware implementation. The advantage of this design method as compared to other methods it is very simple and takes less time to design and adaptive system whereas the drawback of this design method is that this method cannot be used for de-noising of complex input signals like acoustic signals, audio signals etc. since the digital filter having a particular cut off frequency does not respond to signals outside a pre-specified band.

4.2 Design and Implementation Using Weight Update Loop Equation

This method of designing the adaptive filter is more complex than the method which was described in previous section. In this method to design an adaptive filter, the weight update loop equation of the adaptive algorithm used is implemented using basic implementation blocks like multipliers, adders, multiplexers, registers and comparators. This represents a unit order adaptive filter. Thus, in order to implement an m^{th} order adaptive filter this weight update loop equation must be implemented m times. Different structures (direct form, transposed form, distributed arithmetic etc.) can be used to implement the weight update loop equation. In this work three different structures were used to implement the weight update loop equation of four different adaptive algorithms. The structures used are direct form structure, transposed form structure and a novel

structure proposed by author whereas algorithms used are LMS, SD, SE and SS. Figs. 4.3, 4.4, 4.5 shows the implementation of m^{th} order LMS adaptive filter using above mentioned three structures. All the four adaptive algorithms mentioned above were implemented using each of these three structures. Each structure has its own advantage as compared to other structure. For example, the direct-form structure has the advantage of low power dissipation at the cost of lower clock speed whereas the transposed form structure has the advantage of higher clock speed and better hardware utilization efficiency at the cost of high power dissipation. Thus it can be said that direct-form structures can be used for low power applications whereas transposed-form structures can be used for high speed applications.

A novel structure has been proposed by author that combines the advantage of both the previous structures. It had better silicon area utilization, hardware utilization efficiency and clock speed as compared to direct form adaptive filters and its power dissipation is lower than transposed form adaptive filter. The new structure was designed by modifying direct form structure; by replacing the chain of cascaded unit delays by a single delay block of specific value. The value of this single delay is dependent upon the length of the filter. This proposed structure delay block delays the input by the specified number of sample periods and outputs all the delayed versions simultaneously. In other words, it is used to discretize a signal in time or resample a signal at a different rate.

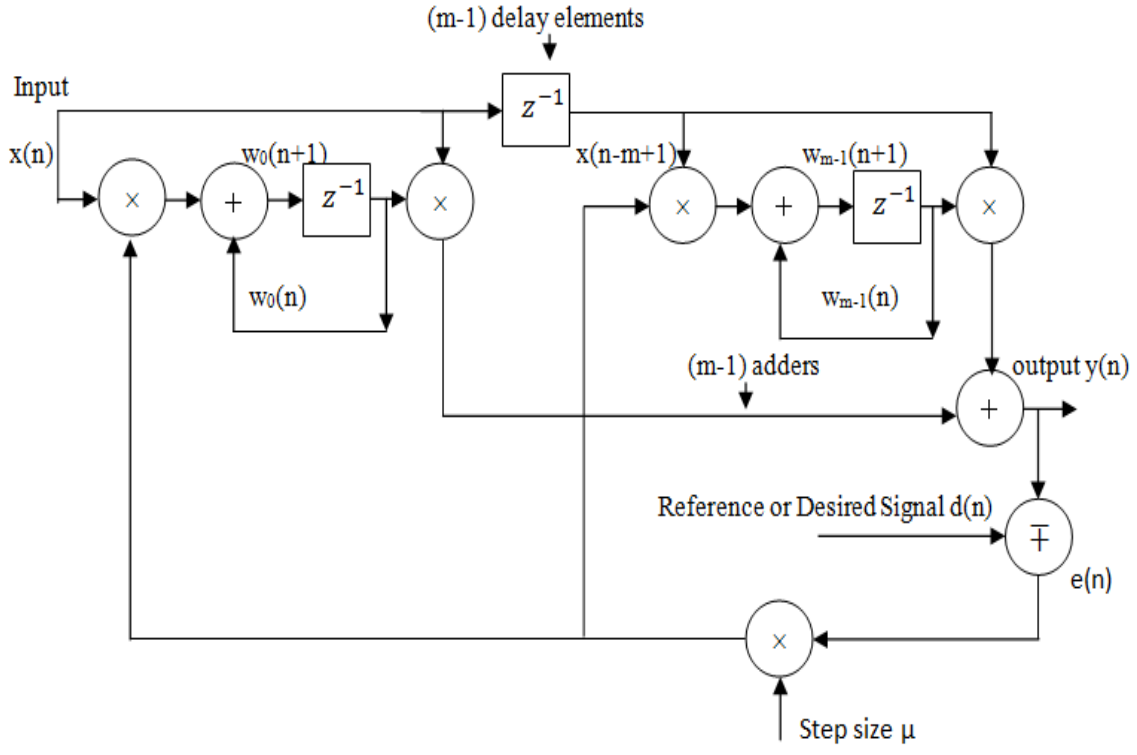

Fig. 4.3 An m^{th} Order LMS Adaptive Filter Implemented using Direct-Form structure.

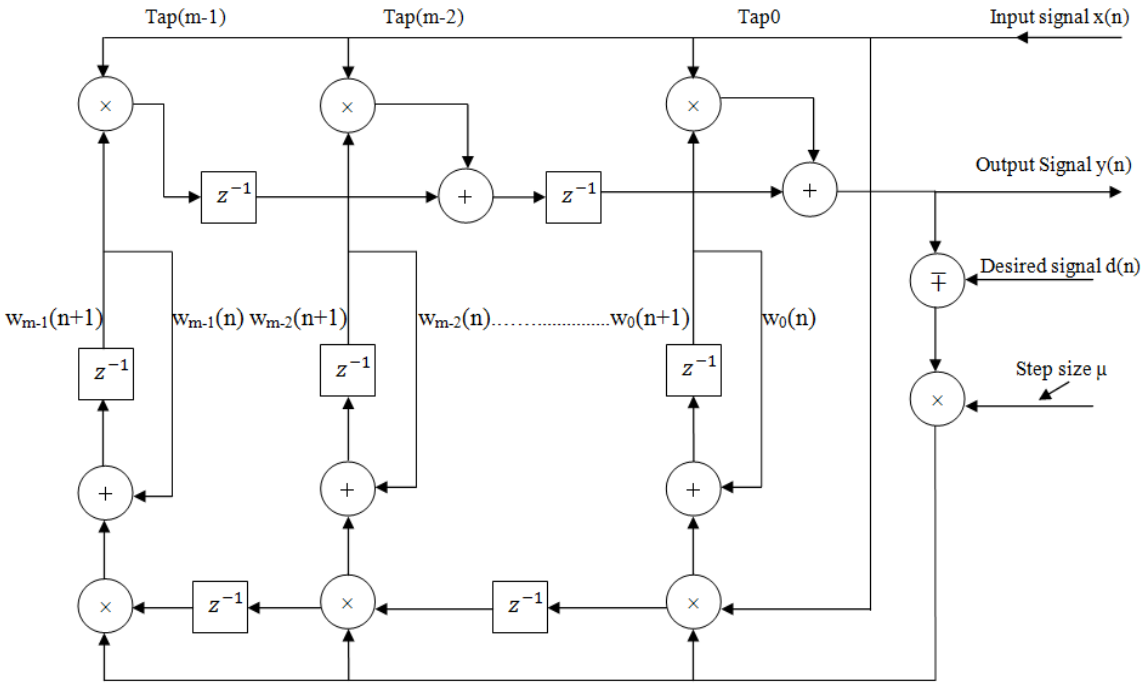

Fig. 4.4 An m[th] Order LMS Adaptive Filter Implemented using Transposed-Form structure.

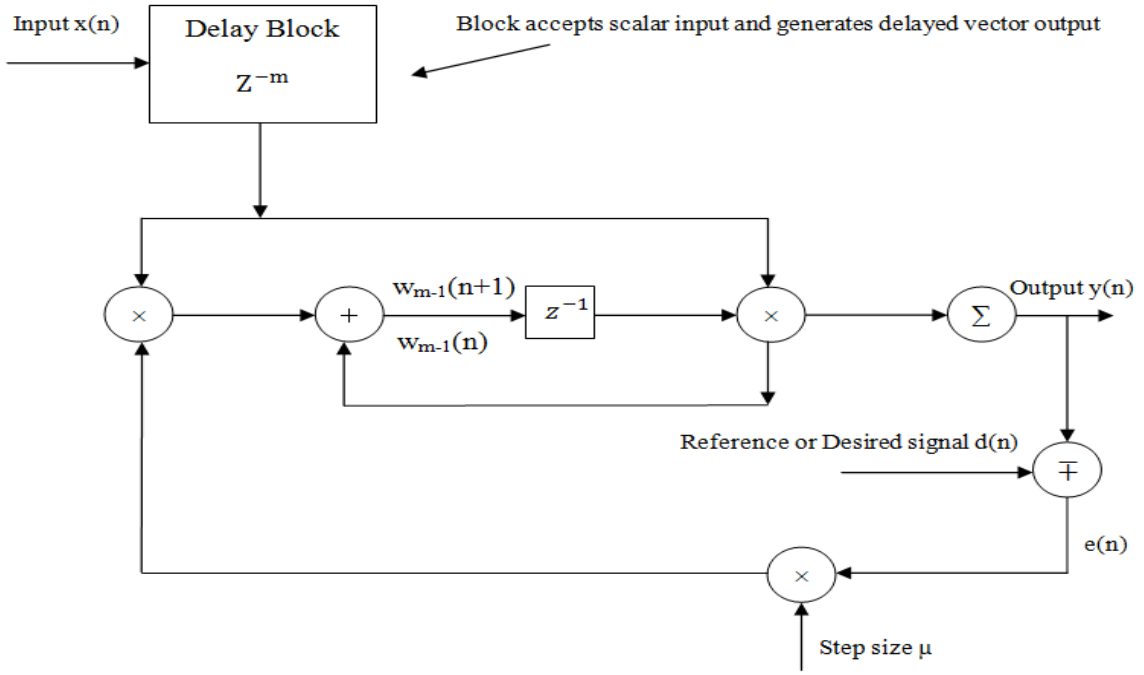

Fig. 4.5 An m[th] Order LMS Adaptive Filter Implemented using Proposed Structure.

Similarly, adaptive filters based on other adaptive algorithms viz. sign-data (SD), sign-error (SE) and sign-sign (SS) was also implemented using above mentioned structures. For, sign-data (SD) algorithm the signum function is applied to the input signal vector x(n), for sign error (SE) algorithm the signum function is applied to the error signal vector e(n) and for sign-sign (SS) algorithm the signum function is applied to both input signal vector x(n) and error signal vector e(n). The advantage of sign-based adaptive filter

as compared to traditional LMS adaptive filter is reduced power dissipation and reduced silicon area utilization at the cost of a minor decrease in delay. The SIMULINK model of a 40th order LMS adaptive filter using direct-form and transposed-form structure mentioned above is shown in Figs. 4.6-4.10. The four blocks namely LMSx10_1, LMSx10_2, LMSx10_3 and LMSx10_4 shown in Fig. 4.6 represents a 10th order adaptive filter. The internal structure of these four blocks for both the above mentioned structures is shown in Figs. 4.7-4.8. The 10 blocks shown in Figs. 4.7-4.8 namely LMS_Tap1 to LMS_Tap10 represents a unit order adaptive filter. This unit order adaptive filter is nothing but the implementation of the weight update loop equation of the adaptive algorithm used in designing adaptive filter. The unit order LMS adaptive filter implemented using direct-form and transposed-form structure is shown in Figs. 4.9-4.10.

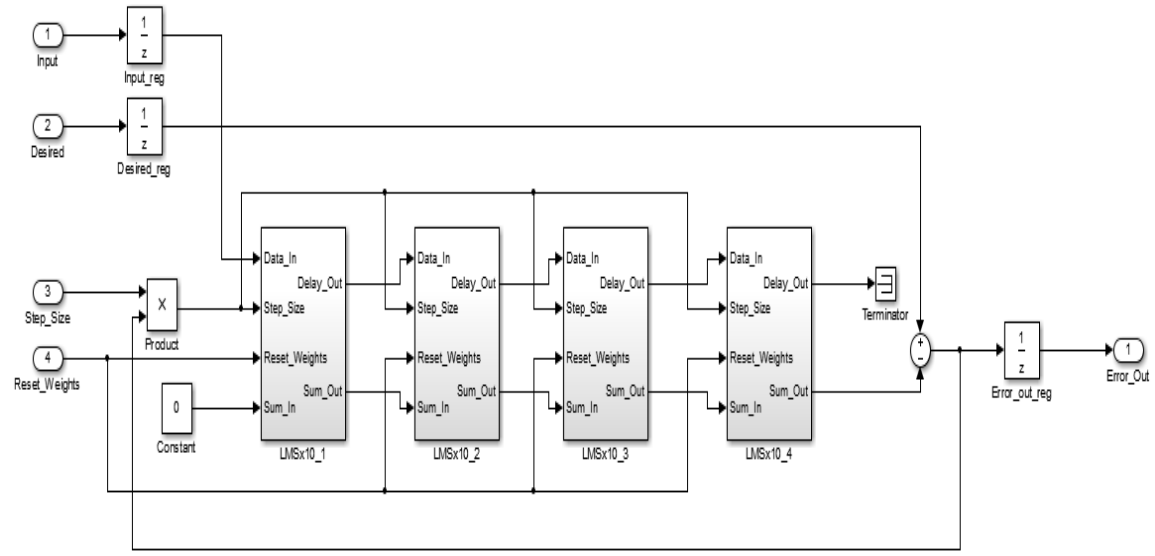

Fig. 4.6 SIMULINK Model of a 40th Order LMS Adaptive Filter Implemented using Direct-Form and Transposed- Form Structure (Outer structure).

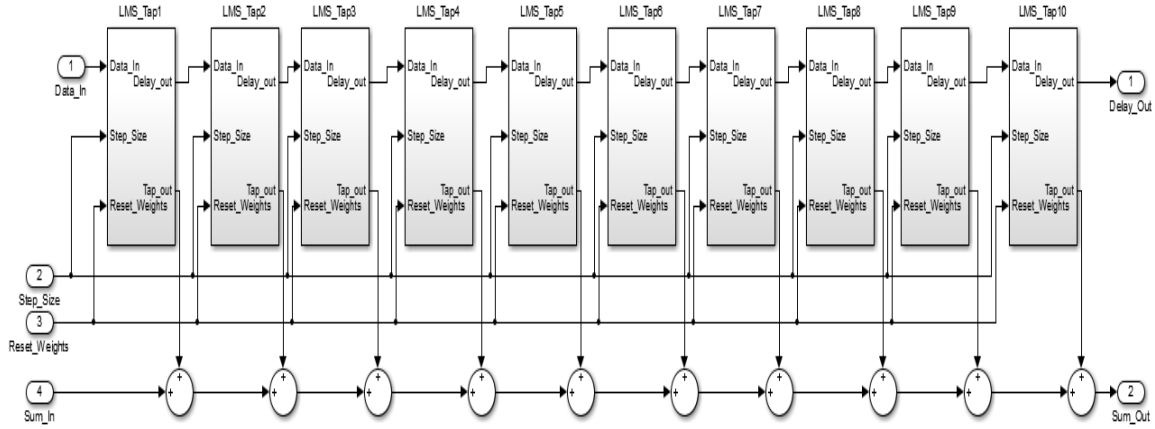

Fig. 4.7 SIMULINK Model of a 10th Order LMS Adaptive Filter Implemented using Direct-Form Structure (Inner structure for each of four blocks as shown in Fig. 4.6).

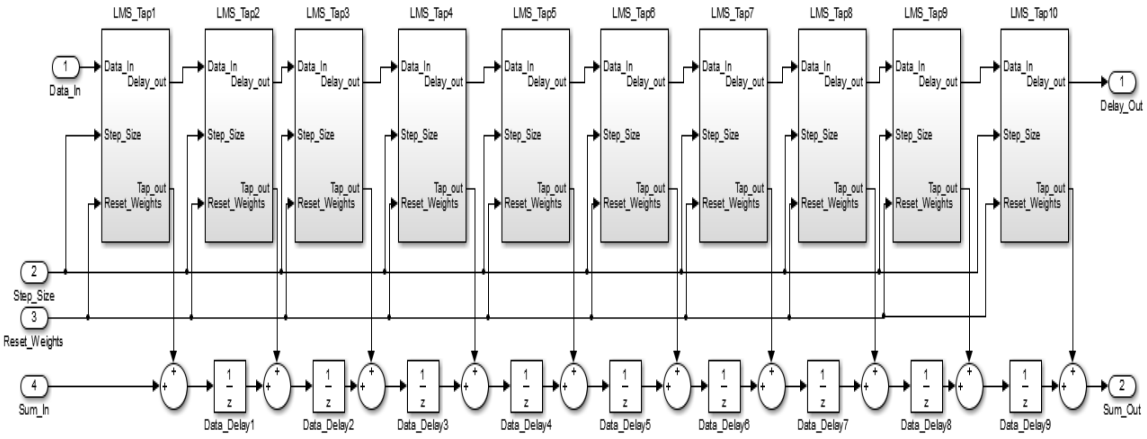

Fig. 4.8 SIMULINK Model of a 10th Order LMS Adaptive Filter Implemented using Transposed-Form Structure (Inner structure for each of four blocks as shown in Fig. 4.6).

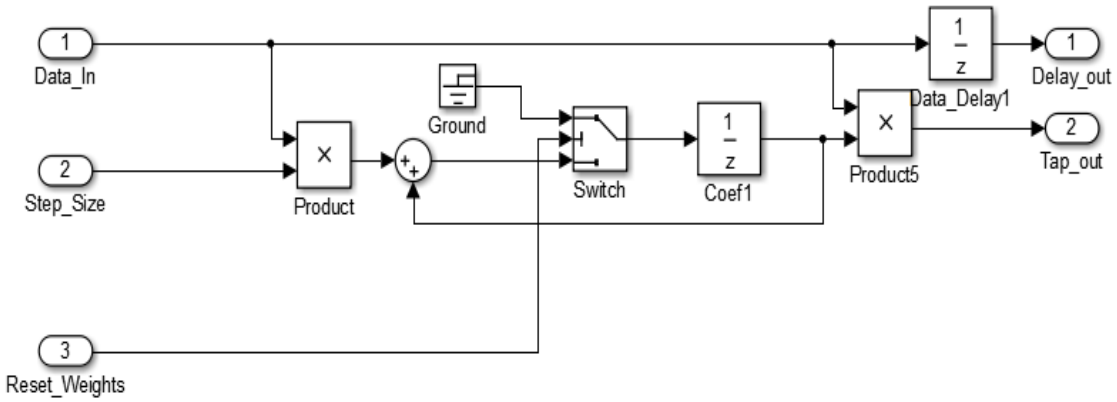

Fig. 4.9 SIMULINK Model of a Unit Order LMS Adaptive Filter Implemented using Direct-Form Structure (Inner structure for each of ten blocks as shown in Fig. 4.7).

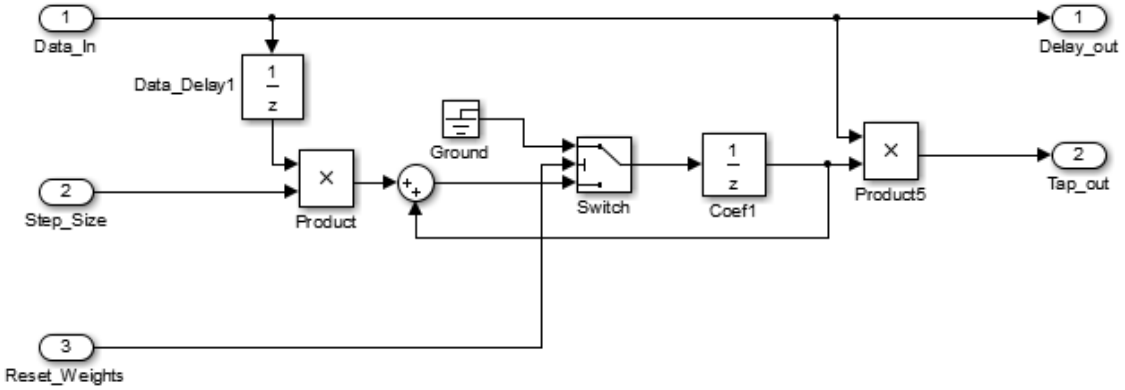

Fig. 4.10 SIMULINK Model of a Unit order LMS Adaptive Filter Implemented using Transposed-Form Structure (Inner structure for each of ten blocks as shown in Fig. 4.8).

The SIMULINK model of the 40th order LMS adaptive filter using the proposed structure is shown in Fig 4.11. The proposed structure uses a delay block that accepts scalar input and generates vector delayed outputs. As, a result of this the delay and the silicon area utilization of the implemented adaptive filter is reduced with a slight increase in the power dissipation.

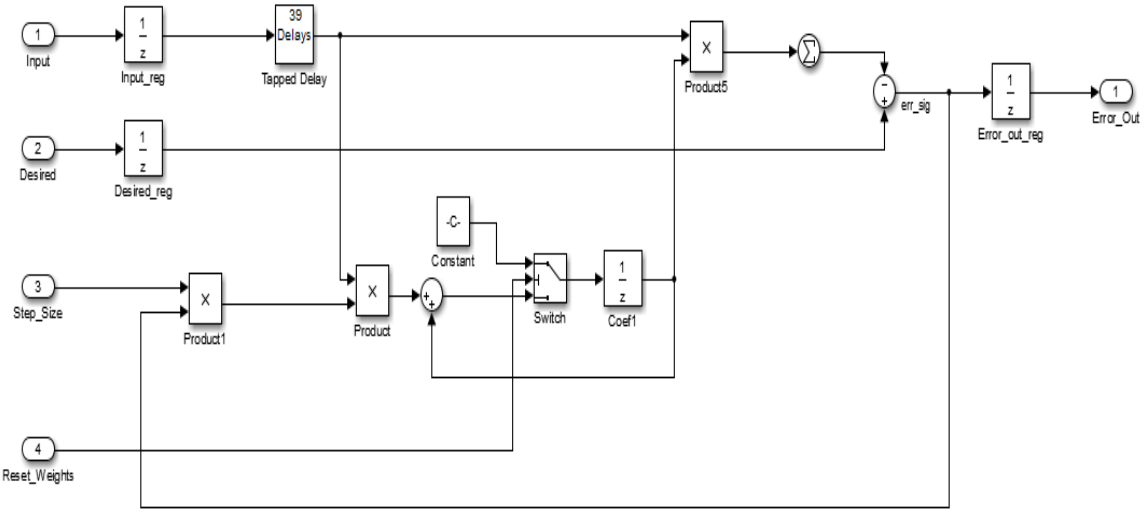

Fig. 4.11 SIMULINK Model of a 40th Order LMS Adaptive Filter Implemented using Proposed Structure.

The designed adaptive filter (having filter length 40) using all the structures and algorithms mentioned above was then used to implement an adaptive noise canceller (ANC). The general structure of the ANC is shown in the Fig. 4.12. It was used to remove the noise from the audio signal. For this an acoustic environment was created in SIMULINK which is shown in Fig. 4.13. The noise source used in the acoustic environment is Gaussian noise which is then converted into discrete form and is sent as input signal to the adaptive filter through the exterior mic port present in the acoustic environment. Further, this noise is passed through the digital filter so that it can be converted into low frequency (coloured) noise and is added into the discrete audio signal having sampling frequency 8 KHz and small gain of 2^{-15} to generate noise corrupted signal. This noise corrupted signal is sent as desired signal to the adaptive filter through the pilot's mic port present in the acoustic environment. The noise signal is passed through the digital filter inspite of directly adding to the audio signal to reduce the number of iterations required for removing noise by the adaptive filter as only low frequency noise is present. The digital filter can act both as the band pass and low pass filter depending upon the output of the switch block. The switch block passes first and the third input depending upon the value of the second input. First and third inputs are called data inputs whereas second input is called control input as it controls the switch behaviour. The control input is a binary input having value either 0 or 1. When the control input is 0 then the low pass filter transfer function is passed and the digital filter acts as low pass filter whereas when the control input is 1 the high pass filter transfer function is passed and the digital filter acts as high pass filter.

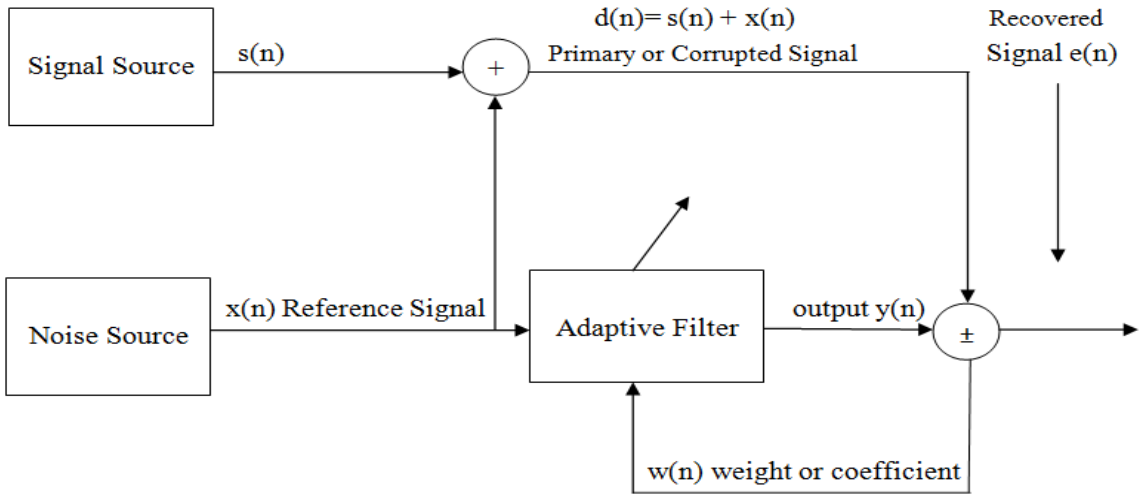

Fig. 4.12 Designed Acoustic Noise Canceller.

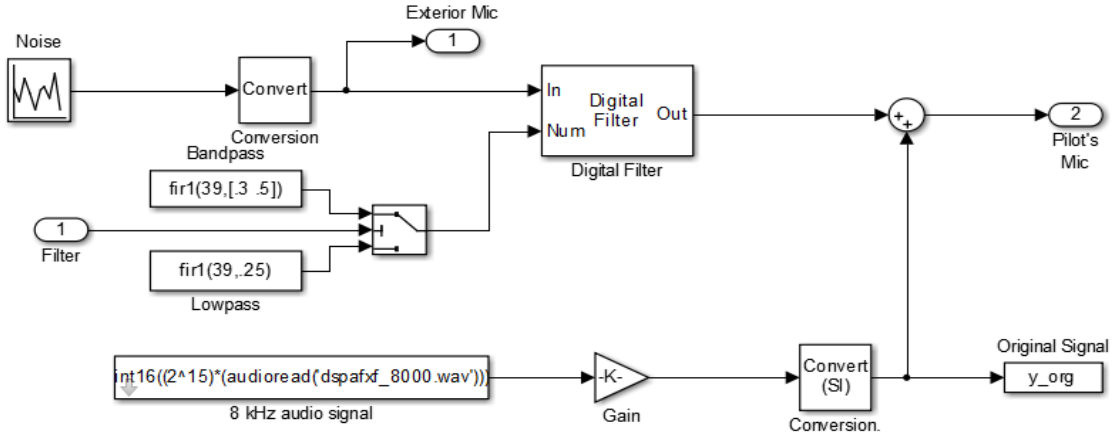

Fig. 4.13 SIMULINK Model of an Acoustic Environment.

The adaptive filter used in the adaptive noise canceller is a variable step size one. The advantage of using variable step size adaptive filter is that it improves convergence rate and stability without compromising performance. Two different step sizes viz. 0.002 and 0.0005 optimized experimentally have been used for implementing the adaptive filter. When the signal is changing at the faster rate or when there is lot of variations in the signal then the smaller step size is used whereas when the signal is changing frequently then a larger step size is used. The SIMULINK model of the designed ANC is shown in Fig. 4.14.

After the ANC was designed and simulated in the SIMULINK it needs to be implemented on FPGA/ASIC. For this the VHDL code of the designed system was generated using SIMULINK design HDL coder. This HDL code was then simulated, synthesized using Xilinx ISE 14.5. The synthesized code was then implemented on three Xilinx FPGAs namely spatran6, vertex6 and virtex7. The functionality of the

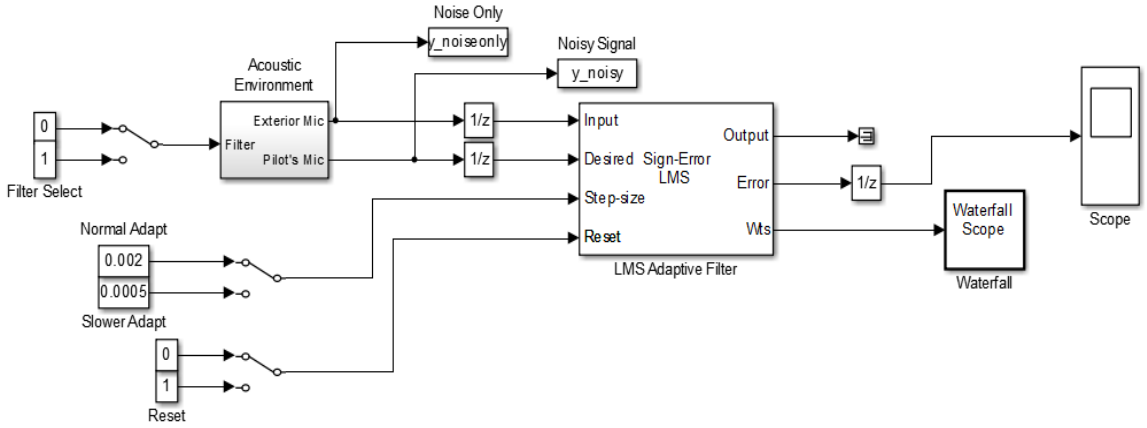

Fig. 4.14 SIMULINK Model of the Designed ANC.

implemented code on FPGA was then verified using ChipScopePro. The HDL code is further synthesized using Leonardo Spectrum which is an ASIC synthesizer. The synthesized code was then used in designing circuit level implementation (schematic and layout) using 250 nm technology. The hardware implementation results pertaining to parameters such as power, area, delay, gate count etc. are described in the subsequent section.

Chapter 5

Results and Discussions

This chapter is divided into three parts: Simulation results of LMS variants, FPGA synthesis results, and ASIC synthesis results. The first part summarizes simulation results of the designed ANC model with SIMULINK and ISIM. Next section is on FPGA synthesis results of different adaptive filters using Xilinx ISE and ChipScope Pro. The last section contributes the synthesis results of ASIC system.

5.1 Simulation Results of LMS variants

Simulation results of ANC designed using 32 order adaptive filter for the de-noising of sinusoidal signal is shown in Figs. 5.1(a)-5.1(g) whereas Figs. 5.2(a)-5.2(f) represents the variations of mean square error (MSE) with time for different adaptive algorithms used for designing adaptive filter. All of the three structures mentioned in chapter 4 were simulated with four variants of LMS algorithm. However, for the sake of convenience results pertaining to the proposed structures only are being reproduced here.

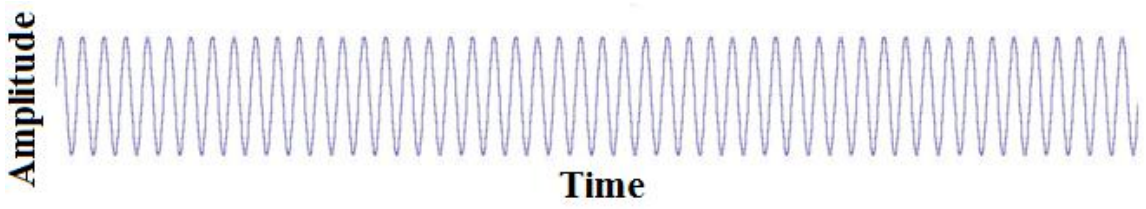

Fig. 5.1(a) Input Signal (Sine Wave).

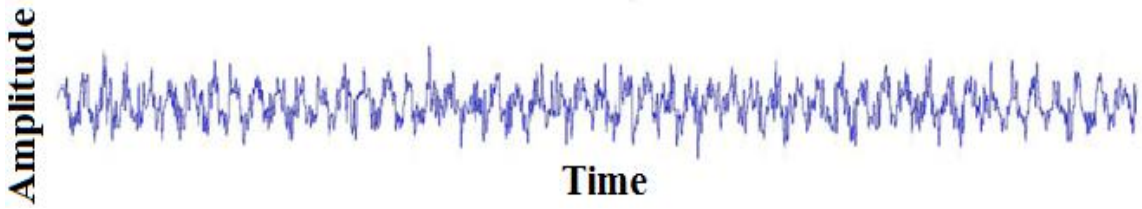

Fig. 5.1(b) Signal with Low Frequency Noise.

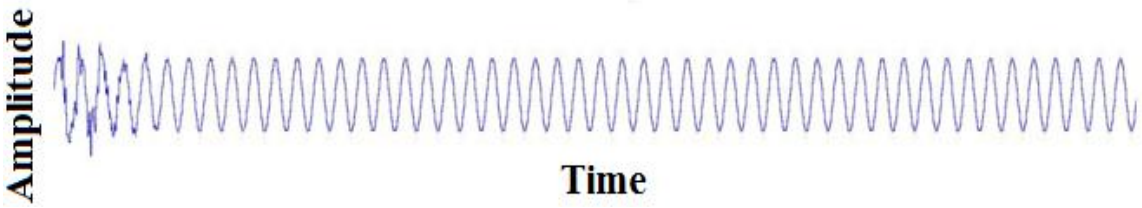

Fig. 5.1(c) LMS output for optimum step size 0.030.

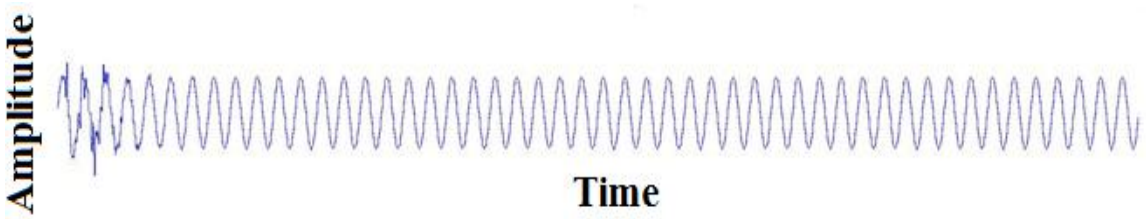

Fig. 5.1(d) NLMS output for optimum step size 0.9.

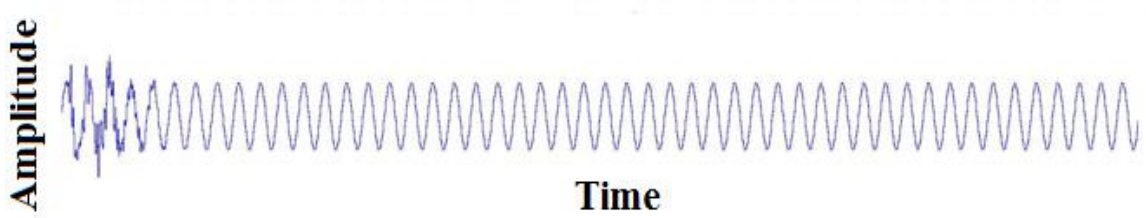

Fig. 5.1(e) SD output for optimum step size 0.029.

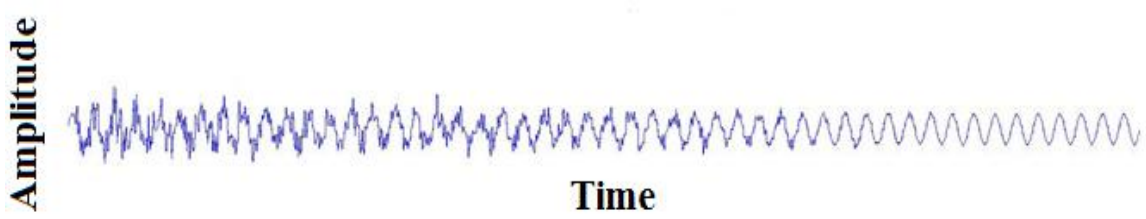

Fig. 5.1(f) SE output for optimum step size 0.0005.

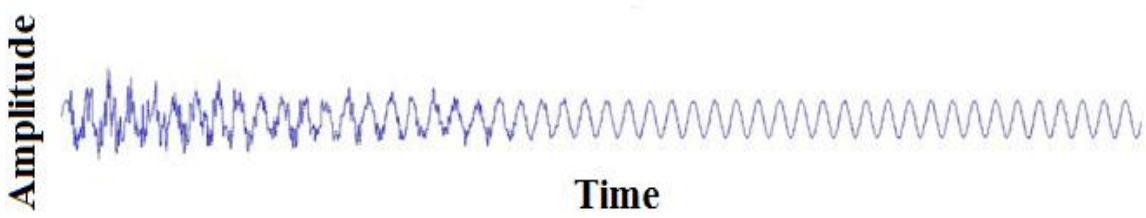

Fig. 5.1(g) SS output for optimum step size 0.0010.

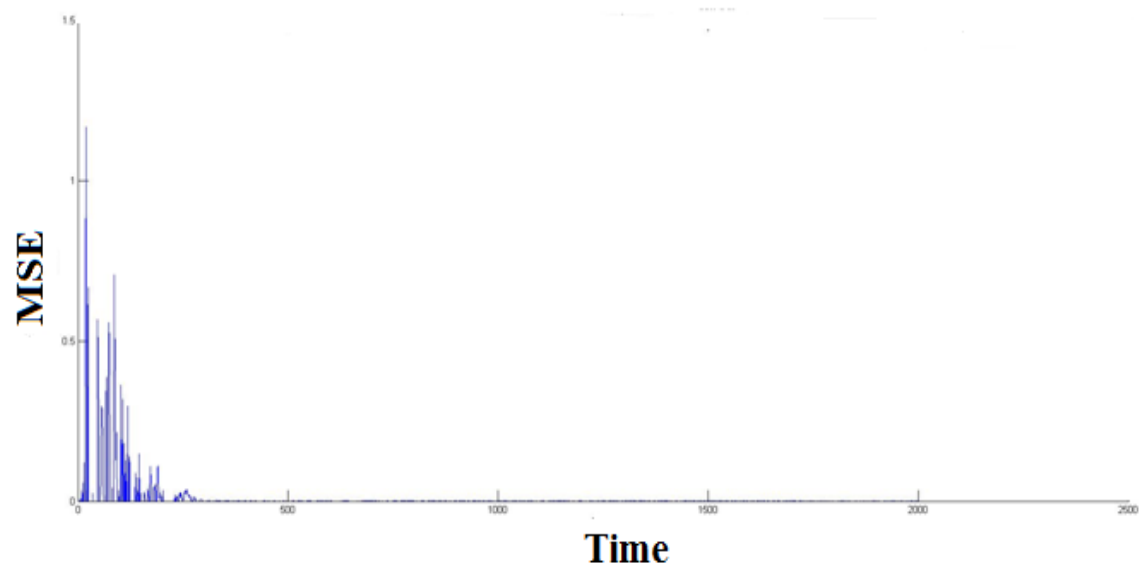

Fig. 5.2(a) LMS MSE for optimum step size 0.030.

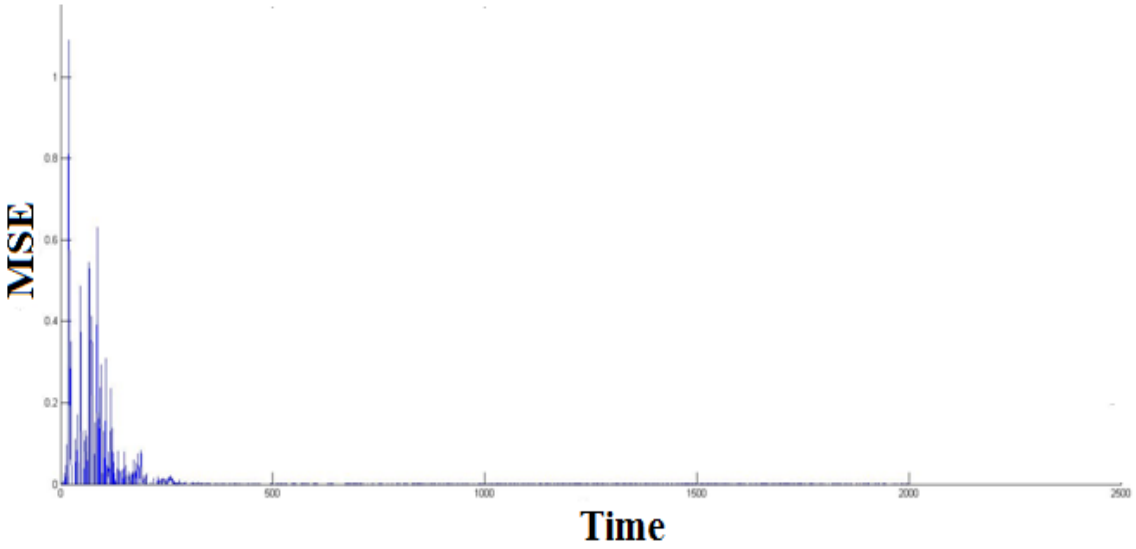

Fig. 5.2(b) NLMS MSE for optimum step size 0.9.

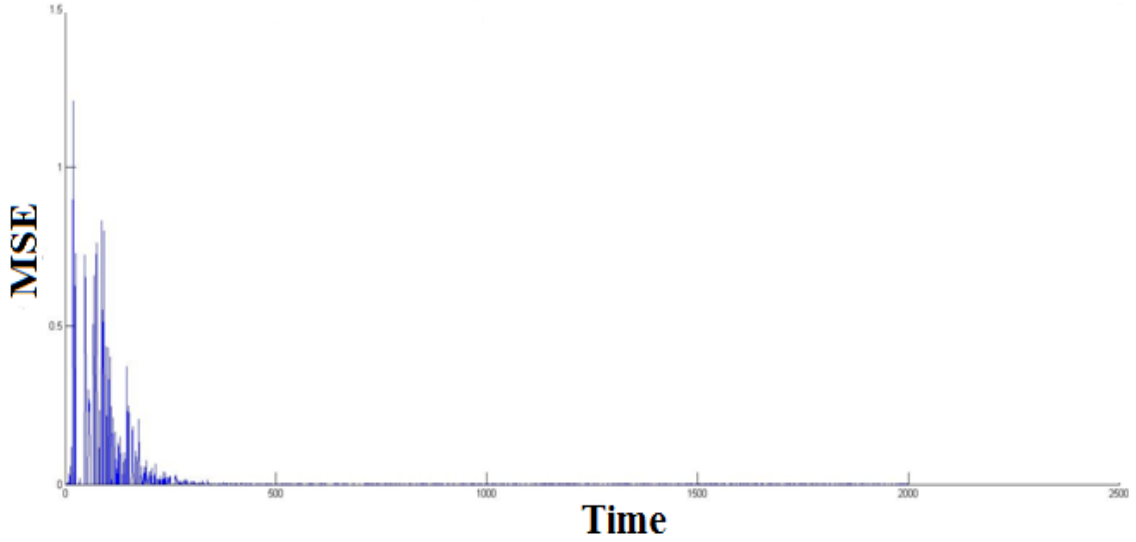

Fig. 5.2(c) SD MSE for optimum step size 0.029.

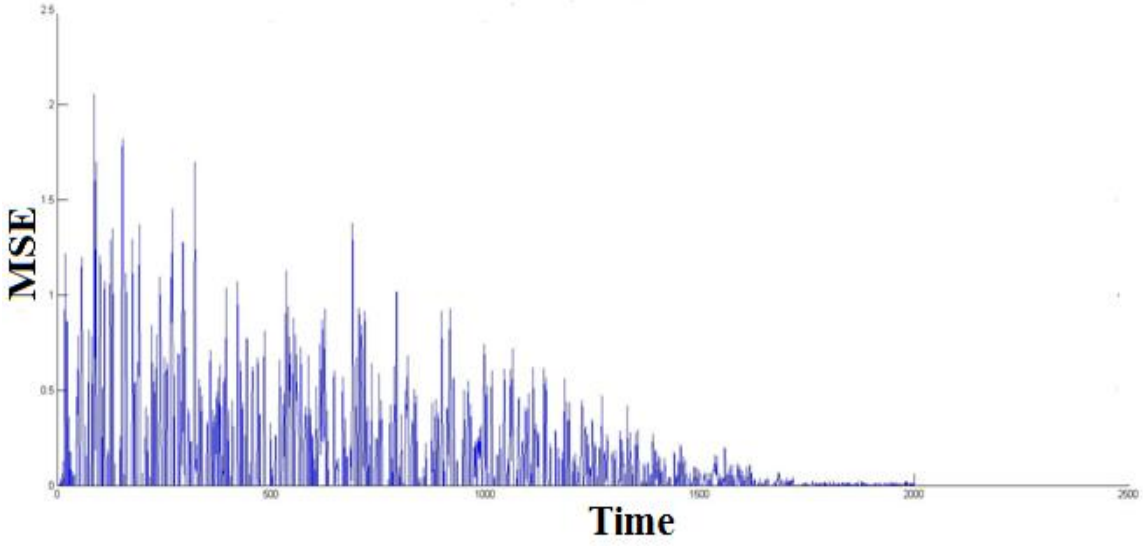

Fig. 5.2(d) SE MSE for optimum step size 0.0005.

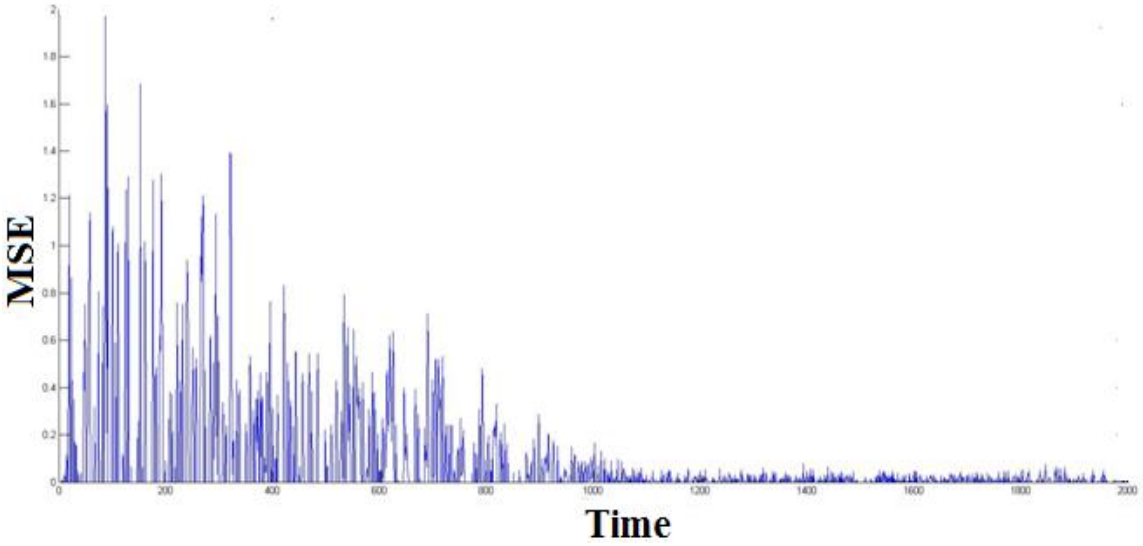

Fig. 5.2(e) SS MSE for optimum step size 0.0010.

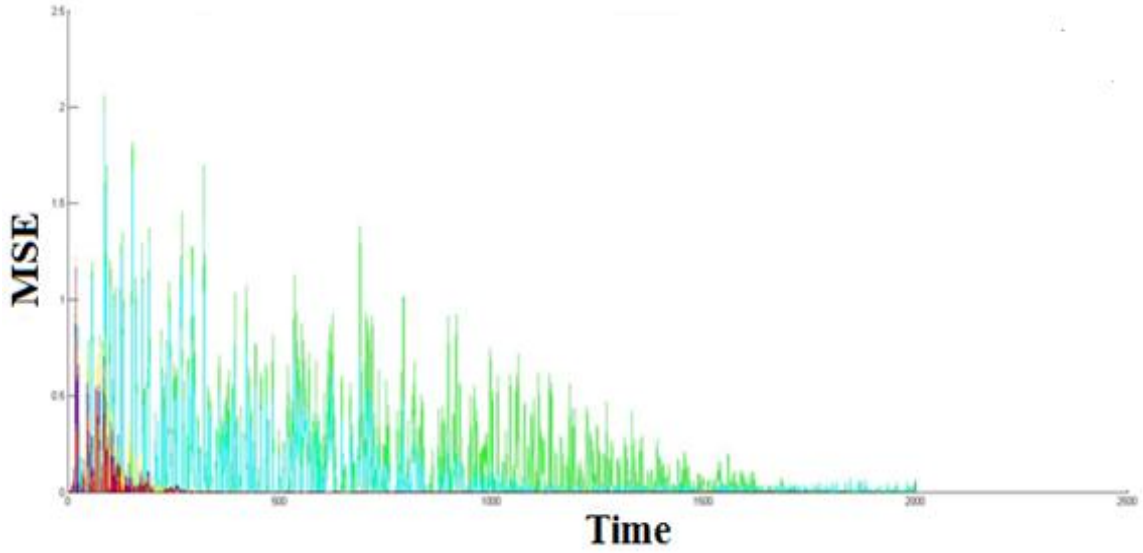

Fig. 5.2(f) Comparison of MSE for all the algorithms.

In case of LMS algorithm if the step size is increased beyond 0.055 then the algorithm fails to converge and remove noise. If the step size is decreased from 0.055 to 0.029 the convergence rate and noise removing capability gets increased whereas error gets decreased. Further, if the step size is reduced beyond 0.029 then convergence rate and noise removing capability decreases. This algorithm works efficiently up to step size 0.002.If the step size is further reduced from 0.002 then this algorithm fails to converge and remove noise in 2001 frames. Also, it is found that at step size 0.030 and 0.029 the MSE comes out to be 0 but the optimum step size is 0.030 because its convergence rate is better than the convergence rate of step size 0.029 .

The NLMS algorithm works efficiently if the step size lies in the range of 1.9 to 0.04. In other words, if the step size is increased above 1.9 or decreased below 0.04 then

NLMS algorithm will diverge (do not converge) and hence it will not be able to remove noise. However, the convergence speed changes with changing step size. The maximum step size in case of NLMS algorithm is 2 (theoretically) but practically works up to 1.9. The convergence rate and noise removing capability is increased when step size is reduced from1.9 to 0.8. At step size 0.9 and step size 0.8 the mean square error is 0 but step size 0.9 had been considered as optimum as its convergence rate is better than step size 0.8. On further decreasing the step size beyond 0.8 to 0.04 the convergence rate and noise removing capability decreases gradually. If the step size is decreased beyond 0.04 then NLMS algorithm fails to remove noise in 2001 frames as the weights do not converge. Beyond 0.05 step size the convergence rate decreases drastically and hence the time required for processing of the signal increases by large amount.

The Sign-Data algorithm fails to work if the step size is increased beyond 0.048 as weights do not converge. At 0.048 step size its convergence rate and noise removing capability is very poor. Also, a considerable amount of noise is present at the output at this (0.048) step size and the value of error at this step size is also high which is equal to 0.06547. If the step size is reduced from 0.048 to 0.029 it is found that the convergence rate and the noise removing capability get increased. Further, decreasing the step size beyond 0.029 to 0.002 it is found that the convergence rate and the noise removing capability get decreased gradually while the error get increased. Again, on decreasing the step size beyond 0.002, it is found out that this (SD) algorithm fails to remove noise in 2001 frame as weights do not converge. The optimum step size comes out for this algorithm is 0.029 and the corresponding value or mean square error (MSE) is 3.852e-14. The range of step size where this algorithm works efficiently is from 0.048 to 0.002.

The Sign-Error algorithm fails to remove noise if the step size is increased beyond 0.0040. At step size 0.0035 it starts to remove noise but does not remove it completely. However, the convergence rate of weights is high in this case (step size 0.0035). This means that the amount of noise which it removes is very fast and then it saturates to some value of noise which can't be removed by using this step size. As, the step size is reduced gradually from 0.0035, it had been found that the noise removing capability increases but the convergence rate decreases. This is up to the case where step size is 0.0005. If the step size is reduced beyond this point (0.0005 step size) then both the convergence rate and the error removing capability decreases. Further if we decrease the step size beyond 0.0002 then this algorithm fails to remove noise (in 2001 frames). The optimum step size comes out in this case is 0.0005, which is a compromise between the error removing capability

and the convergence rate. The main drawback of this algorithm is its slower convergence rate. The value of MSE for optimum step size is 2.888e-5.

The Sign-Sign algorithm fails to remove considerable amount of noise if the step size is increased beyond 0.0030. Also, if the step size is increased beyond 0.0040 this algorithm fails totally to remove a small amount of noise. At step size 0.0030 it starts to remove noise but does not remove it completely. As the step size is reduced gradually from 0.0030, it has been found that the noise removing capability increases but the convergence rate of weights decreases. This is up to the case where step size is 0.0005. If the step size is reduced beyond this point (0.0005 step size) then both the convergence rate of weights and the error removing capability decreases. The optimum step size comes out in this case is 0.0010, which is a compromise between the error removing capability and the convergence rate of weights. Unlike all other previous algorithm, this algorithm does not remove noise in a considerable amount i.e. the error in this case does not reduce to a value close to 0 even in the case of optimum step size. The value of MSE for optimum step size is 0.001805.

The above described results are summarized below in Table 5.1. These reslts are used for slecting two representative step sizes one for slower adaptation and one for faster adaptation that can be further used in desiginig a 40 order variable step size adaptive filter which can be used for de-noising of audio signals. The simulation results for the ANC designed by using 40 order variable step size adaptive filter for the de-noising of audio signals are shown in Figs. 5.3(a)-5.5(d).

Algorithm Name	Optimum Step Size (δ)	Range of Step Size (δ)	MSE
LMS	0.030	$0.002 \leq \delta \leq 0.055$	0
NLMS	0.9	$0.002 \leq \delta \leq 1.9$	0
Sign-Data (SD)	0.029	$0.002 \leq \delta \leq 0.048$	3.852e-14
Sign-Error (SE)	0.0005	$0.0002 \leq \delta \leq 0.0040$	2.888e-5
Sign-Sign (SS)	0.0010	$0.0005 \leq \delta \leq 0.0040$	0.001805

Table 5.1 Optimum Step Size, Range Of Step Size and MSE for Different Variants Of LMS Algorithms

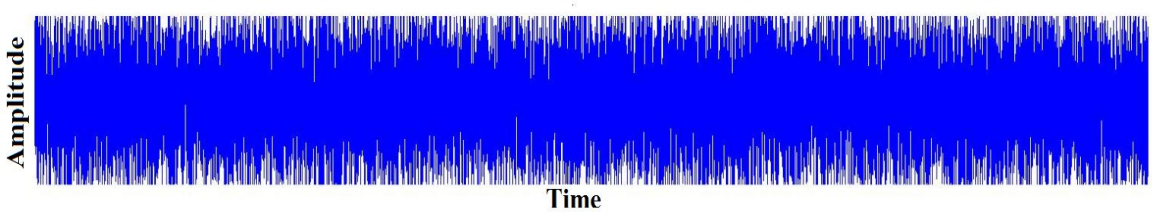

Fig. 5.3(a) Noise Signal.

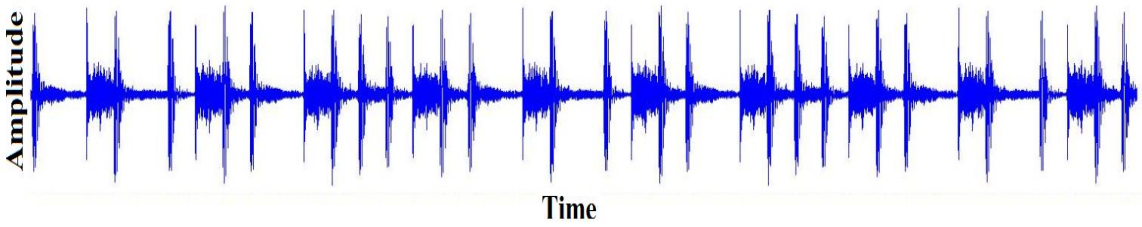

Fig. 5.3(b) Input Signal (Audio Signal).

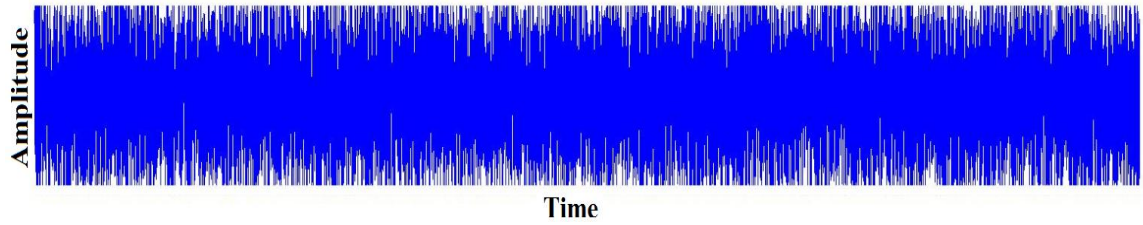

Fig. 5.3(c) Signal with Noise.

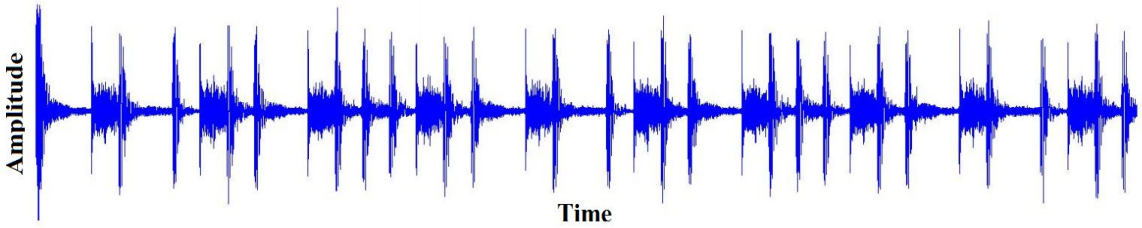

Fig. 5.3(d) LMS Output.

Fig. 5.3(e) SD Output.

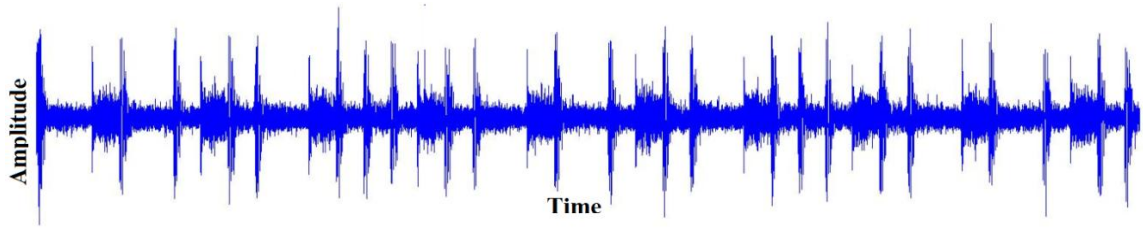

Fig. 5.3(f) SE Output.

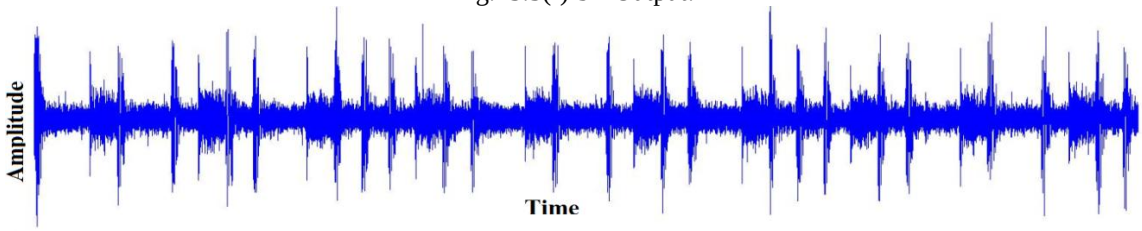

Fig. 5.3(g) SS Output.

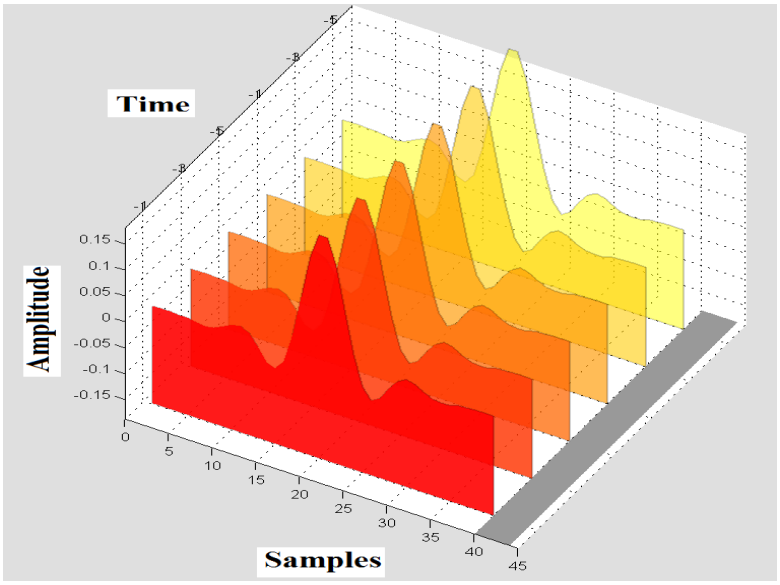

Fig. 5.4 Variations of Amplitude, Samples and Time for the Output Signal.

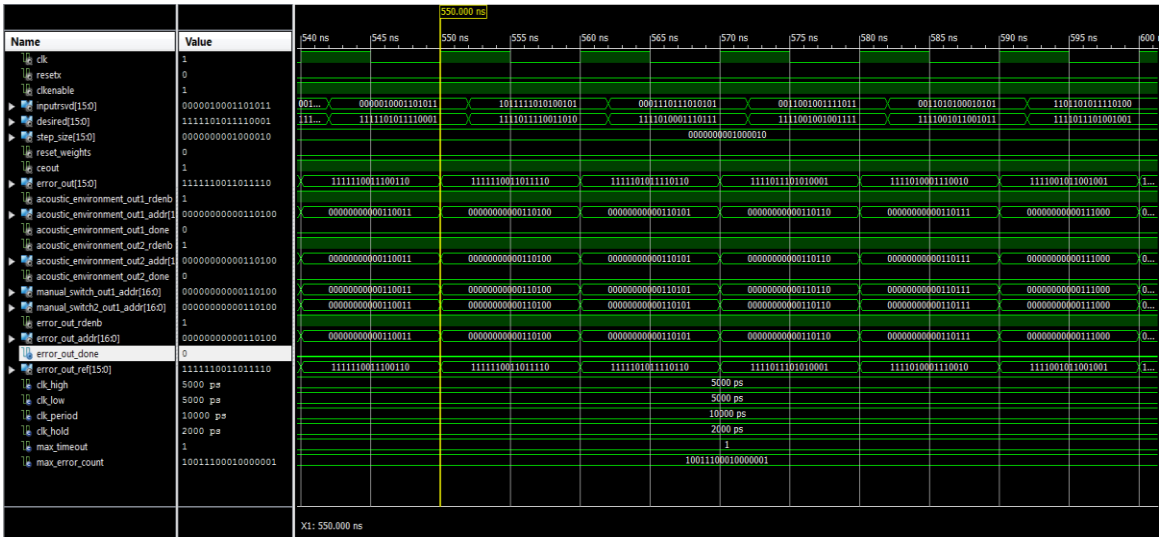

Fig. 5.5(a) Simulation Result of a 40th Order LMS Adaptive Filter.

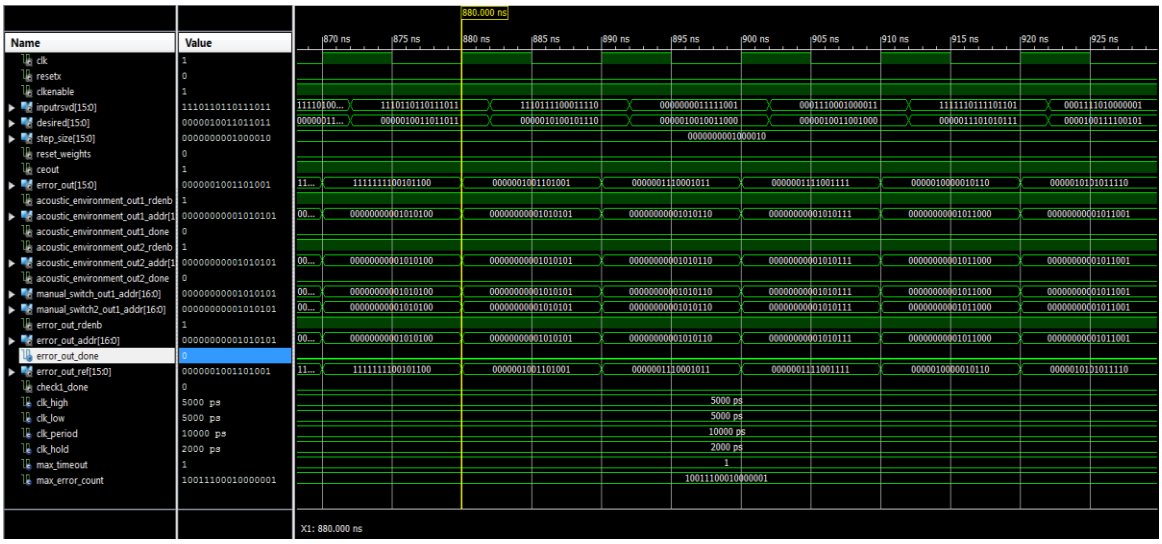

Fig. 5.5(b) Simulation Result of a 40th Order SD Adaptive Filter.

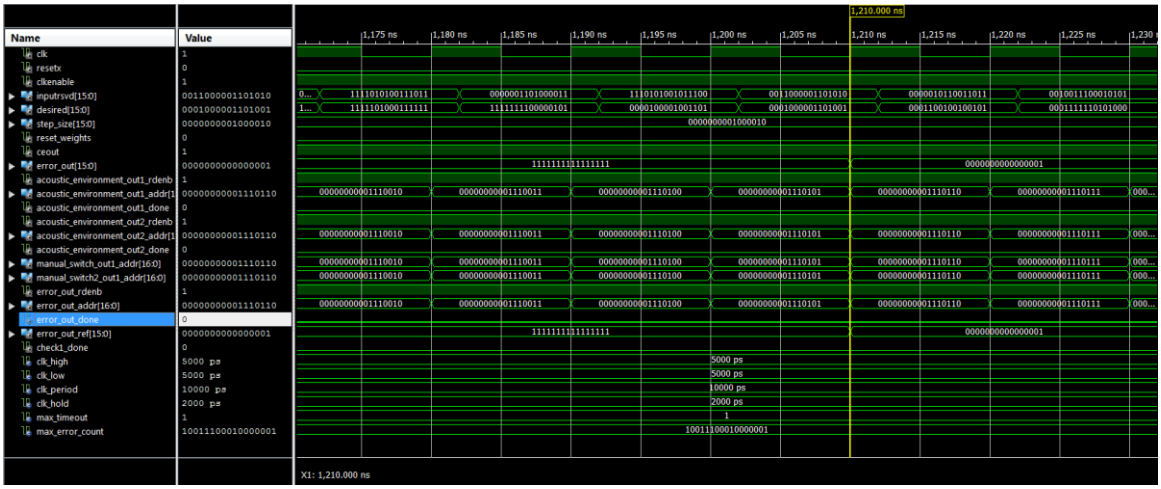

Fig. 5.5(c) Simulation Result of a 40th Order SE Adaptive Filter.

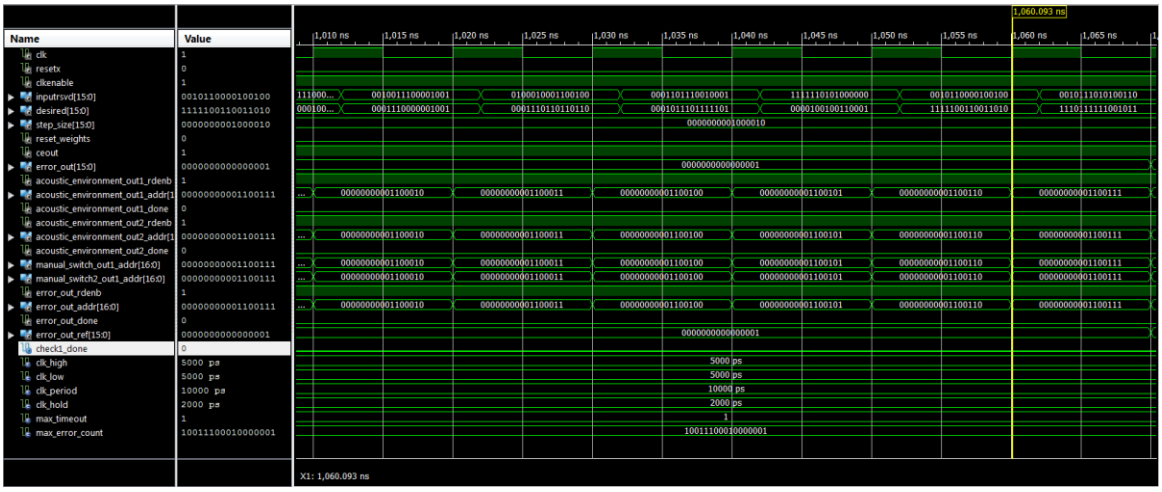

Fig. 5.5(d) Simulation Result of a 40th Order SS Adaptive Filter.

5.2 FPGA Synthesis Results

Different variable step size 40 order adaptive filters used for de-noising are synthesized on Xilinx ISE 14.5 so that it can be implemented on FPGA devices. A detailed analysis of the implementation results reported in this section in the form of tables is now being presented. These results describe the performance of the adaptive filter with respect to the algorithm chosen, the filter structure, and the implementation technology. In all the three cases, algorithm convergence rate, MSE, clock speed, delay, hardware utilization efficiency, area and power have been the common performance matrices.

Performance Analysis in Terms of Algorithm Chosen: It is evident from Tables 5.2-5.5 that delay, number of multiplexers and comparators in case of LMS algorithm is lowest whereas SS algorithm has the highest value of all these parameters. Further, LMS algorithm has the highest number of adders and highest power consumption whereas SS

algorithm has the least power consumption and lowest number of adders. Also, it has been found that number of adders in SD algorithm is same as in case of SS algorithm.

Performance Analysis in Terms of Filter Structure: It is well known that popular filter structures yield better performance according to one metric or the other. Thus, the choice of an appropriate structure might give the required performance in terms of a parameter of interest. Inspection of Table 5.2 and Table 5.4 reveals that among the above mentioned structures the direct form one have resulted the least power consumption and a lower clock speed making it suitable for low power application with an increased delay and a larger silicon area. Furthermore, due to reduction in critical path the clock speed off the transposed structure is significantly increased making it suitable for high speed applications. However, the increase in the clock speed is at the cost of increased power dissipation. The author believe that structure proposed in Fig. 4.5 combines the advantages of several popular structures thus striking a balance between different requirements of an optimum adaptive filter. For example, the simultaneous generation of delayed vector output does away with the need of multiple delays as in the case of other structures mentioned here. This results in enhanced clock speed as compared to direct form structure, reduced power consumption as compared to transposed structure, reduction in delay as compared to direct form structure and least silicon area consumption among all the structures. Though the dedicated hardware support for basic components such as adders, multipliers etc. are by default area efficient for FPGAs, variations in the number of components utilized can introduce significant area overhead. It was found that unlike the direct form and transposed form structures the proposed structure uses higher bit adders such as 38 bit and 37 bit ones. As a result of this it is expected that the proposed structure results in faster addition as compared to others two structures. Also, it had been observed that the number of multiplexers and registers are reduced by a significant amount as compared to other two structures. This is due to the fact that in the proposed structure there is a provision of simultaneous generation of delayed vector outputs.

Performance Analysis in Terms of Device Family: The proposed structure along with the other two was implemented on three different device families viz. Spartan 6, Virtex 6, and Virtex 7. All the three above mentioned device families have something to offer either in terms of low power and low cost implementation or in terms of increased system

Algo Name	Structures	Multiplier		Adder								Subtractor	Muxes		Registers				Comparators
		16 x 16 bit	32 x 16 bit	16 bit	32 bit	33 bit	34 bit	35 bit	36 bit	37 bit	38 bit	17 bit	16 bit 2:1	32 bit 2:1	16 bit	32 bit	624 bit	640 bit	
LMS	Direct	41	40	1	161	39	0	0	0	0	0	1	124	158	83	0	0	0	0
	Transposed	41	40	1	240	0	0	0	0	0	0	1	200	80	120	1	0	0	0
	Proposed	41	40	1	161	20	10	5	2	1	1	1	84	82	3	0	1	1	0
SD	Direct	41	40	1	81	39	0	0	0	0	0	1	125	238	83	0	0	0	2
	Transposed	41	40	1	160	0	0	0	0	0	0	1	201	160	120	1	0	0	2
	Proposed	41	40	1	81	20	10	5	2	1	1	1	85	162	3	0	1	1	2
SE	Direct	41	40	1	160	39	0	0	0	0	0	1	125	160	83	0	0	0	2
	Transposed	41	40	1	239	0	0	0	0	0	0	1	201	82	120	1	0	0	2
	Proposed	41	40	1	160	20	10	5	2	1	1	1	85	84	3	0	1	1	2
SS	Direct	41	40	1	80	39	0	0	0	0	0	1	126	240	83	0	0	0	4
	Transposed	41	40	1	159	0	0	0	0	0	0	1	202	162	120	1	0	0	4
	Proposed	41	40	1	80	20	10	5	2	1	1	1	86	164	3	0	1	1	4

Table 5.2 Hardware Utilization Summary.

Algo. Name	Structure	Clk. Freq (Max.) (MHz)			Delay (min.) (ns)			Min. I/P arrival time before clk. (ns)			Max. O/P required time after clock (ns)			Max. Combinational Path Delay (ns)		
		Spartan6	Virtex6	Virtex7	Spartan6	Virtex6	Virtex7	Spartan6	Virtex6	Virtex7	Spartan6	Virtex6	Virtex7	Spartan6	Virtex6	Virtex7
LMS	Direct	8.334	13.798	14.245	119.996	72.475	70.198	22.496	12.056	10.904	3.495	0.562	0.511	5.672	0.489	0.483
	Transposed	31.027	51.431	55.943	32.230	19.450	17.875	21.967	12.247	11.048	3.495	0.562	0.511	5.800	0.508	0.502
	Proposed	24.939	40.265	42.617	40.097	24.835	23.465	22.496	12.056	10.904	3.495	0.562	0.511	5.672	0.489	0.483
SD	Direct	8.252	13.766	14.236	121.176	72.641	70.247	22.362	11.931	10.791	3.495	0.562	0.511	5.562	0.472	0.466
	Transposed	28.664	49.029	53.596	34.887	20.396	18.658	21.833	12.123	10.934	3.495	0.562	0.511	5.690	0.492	0.486
	Proposed	23.971	39.994	42.760	41.718	25.004	23.368	22.362	11.931	10.791	3.495	0.562	0.511	5.567	0.472	0.466
SE	Direct	8.198	13.687	14.163	121.988	73.060	70.607	23.691	12.189	10.939	3.826	0.627	0.576	5.669	0.488	0.482
	Transposed	29.466	49.966	54.663	33.938	20.014	18.294	23.196	12.381	11.083	3.495	0.562	0.511	5.800	0.508	0.502
	Proposed	23.933	39.395	41.892	41.783	25.384	23.871	23.752	12.189	10.939	3.826	0.627	0576	5.669	0.488	0.482
SS	Direct	8.101	13.618	14.110	123.448	73.434	70.870	23.557	12.065	10.826	3.826	0.627	0.576	5.562	0.472	0.466
	Transposed	27.326	47.712	54.420	36.595	20.959	19.077	23.062	12.256	10.969	3.495	0.562	0.511	5.690	0.492	0.486
	Proposed	23.009	39.112	42.008	43.461	25.568	23.805	23.591	12.065	10.826	3.826	0.627	0.576	5.564	0.471	0.465

Table 5.3 Timing Analysis Summary.

performance and accelerated design productivity. This has motivated the author to investigate speed, power and device utilization parameters of the filters implemented on individual device families. Being the latest among the three Virtex 7 combines very high level of system integration (28 million logic cells) with an unprecedented performance bandwidth (up to 2.8 Tb/sec of total serial bandwidth). It was also verified from the map and place-route reports that Virtex 7 implementation gives a higher clock speed and less

FPGA Chip	Structures	Spartan6 6slx150tfgg900-4				Virtex6 6slx150tfgg900-4				Virtex7 xc7vx415t-3ffg1927			
		LMS	SD	SE	SS	LMS	SD	SE	SS	LMS	SD	SE	SS
Used Slice Registers	Direct	1312 (1%)	7649 (1%)	1328 (1%)	740 (1%)	1342 (1%)	752 (1%)	1328 (1%)	738 (1%)	1342 (1%)	742 (1%)	1328 (1%)	738 (1%)
	Transposed	1957 (1%)	1434 (1%)	1986 (1%)	1434 (1%)	1986 (1%)	1434 (1%)	1986 (1%)	1434 (1%)	1986 (1%)	1434 (1%)	1986 (1%)	1434 (1%)
	Proposed	1942 (1%)	1410 (1%)	1928 (1%)	1397 (1%)	1942 (1%)	1382 (1%)	1928 (1%)	1368 (1%)	1942 (1%)	1382 (1%)	1928 (1%)	1368 (1%)
Used Slice LUTs	Direct	8450 (9%)	7289 (7%)	8545 (9%)	7323 (7%)	8326 (5%)	7131 (4%)	8336 (5%)	7132 (4%)	8415 (3%)	7183 (2%)	8424 (3%)	7184 (2%)
	Transposed	8490 (9%)	7669 (8%)	8556 (9%)	7698 (8%)	8290 (5%)	7547 (4%)	8293 (5%)	7550 (4%)	8381 (3%)	7642 (2%)	8378 (3%)	7643 (2%)
	Proposed	8112 (8%)	6849 (7%)	8139 (8%)	6871 (7%)	7930 (5%)	6703 (4%)	7918 (5%)	6703 (4%)	8016 (3%)	6754 (2%)	8019 (3%)	6746 (2%)
Logic Used	Direct	8354 (9%)	7249 (7%)	8390 (9%)	7289 (7%)	8194 (5%)	7130 (4%)	8193 (5%)	7132 (4%)	8291 (3%)	7182 (2%)	8289 (3%)	7184 (2%)
	Transposed	8299 (9%)	7663 (8%)	8332 (9%)	7696 (8%)	8139 (5%)	7543 (4%)	8140 (5%)	7544 (4%)	8235 (3%)	7639 (2%)	8236 (3%)	7640 (2%)
	Proposed	7938 (8%)	6822 (7%)	7963 (8%)	6847 (7%)	7781 (4%)	6702 (4%)	7774 (4%)	6703 (4%)	7879 (3%)	6753 (2%)	7872 (3%)	6746 (2%)
Occupied Slices	Direct	2851 (12%)	2677 (11%)	3075 (13%)	2682 (11%)	2916 (7%)	2902 (7%)	2850 (7%)	2443 (6%)	2829 (4%)	2747 (4%)	2844 (4%)	2724 (4%)
	Transposed	2920 (12%)	2812 (12%)	3014 (13%)	2849 (12%)	2919 (7%)	2981 (7%)	2868 (7%)	2848 (7%)	2886 (4%)	3180 (4%)	2876 (4%)	3224 (5%)
	Proposed	2720 (11%)	2444 (10%)	2762 (11%)	2471 (10%)	2713 (6%)	2671 (6%)	2610 (6%)	2500 (6%)	2716 (4%)	2338 (3%)	2720 (4%)	2340 (2%)

Table 5.4 Device Utilization Summary.

Algo Name	Structures	Spartan6 xc6slx150t-4fgg900			Virtex6 xc6vhx250t-3ff1154			Virtex7 xc7vx415t-3ffg1927		
		Static Power (W)	Dynamic Power (W)	Total Power (W)	Static Power (W)	Dynamic Power (W)	Total Power (W)	Static Power (W)	Dynamic Power (W)	Total Power (W)
LMS	Direct	0.114	0.025	0.139	0.319	0.036	0.355	0.247	0.025	0.272
	Transposed	0.116	0.094	0.210	0.322	0.140	0.462	0.248	0.099	0.347
	Proposed	0.115	0.073	0.188	0.321	0.106	0.427	0.249	0.073	0.322
SD	Direct	0.114	0.024	0.138	0.318	0.036	0.354	0.247	0.024	0.271
	Transposed	0.116	0.084	0.200	0.322	0.129	0.451	0.248	0.092	0.340
	Proposed	0.115	0.067	0.182	0.321	0.101	0.422	0.249	0.069	0.318
SE	Direct	0.114	0.024	0.138	0.319	0.036	0.355	0.247	0.025	0.272
	Transposed	0.116	0.089	0.205	0.322	0.136	0.458	0.248	0.097	0.345
	Proposed	0.115	0.070	0.185	0.321	0.103	0.424	0.249	0.072	0.321
SS	Direct	0.114	0.023	0.137	0.318	0.035	0.353	0.247	0.024	0.271
	Transposed	0.116	0.080	0.196	0.322	0.126	0.448	0.248	0.093	0.341
	Proposed	0.115	0.065	0.180	0.321	0.099	0.420	0.249	0.068	0.317

Table 5.5 Power Analysis.

delay than others. Therefore, we have used Virtex 7 implementation results as a representative for comparing the performance of individual structures across the four algorithms and the same has been depicted in the form of bar plots shown in Figs. 5.6-5.8 The proposed structure uses the least number of slice LUTs as evident from Fig. 5.8. Furthermore, the maximum delay of proposed structure is significantly lower than that of direct form one. The results of Fig. 5.7 are significant since the reduction in delay for the

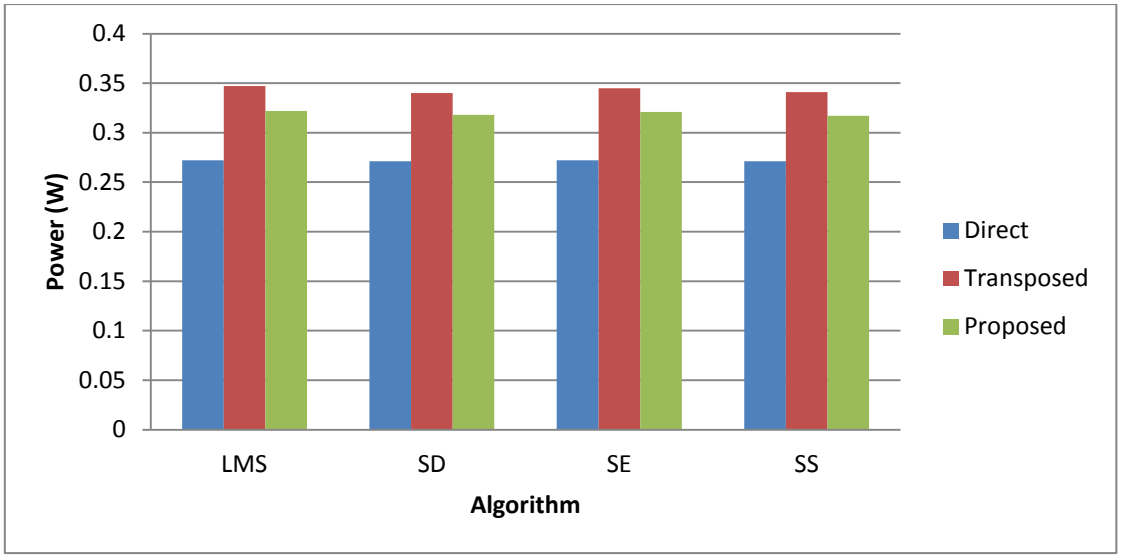

Fig. 5.6 Total Power Consumption in Virtex7.

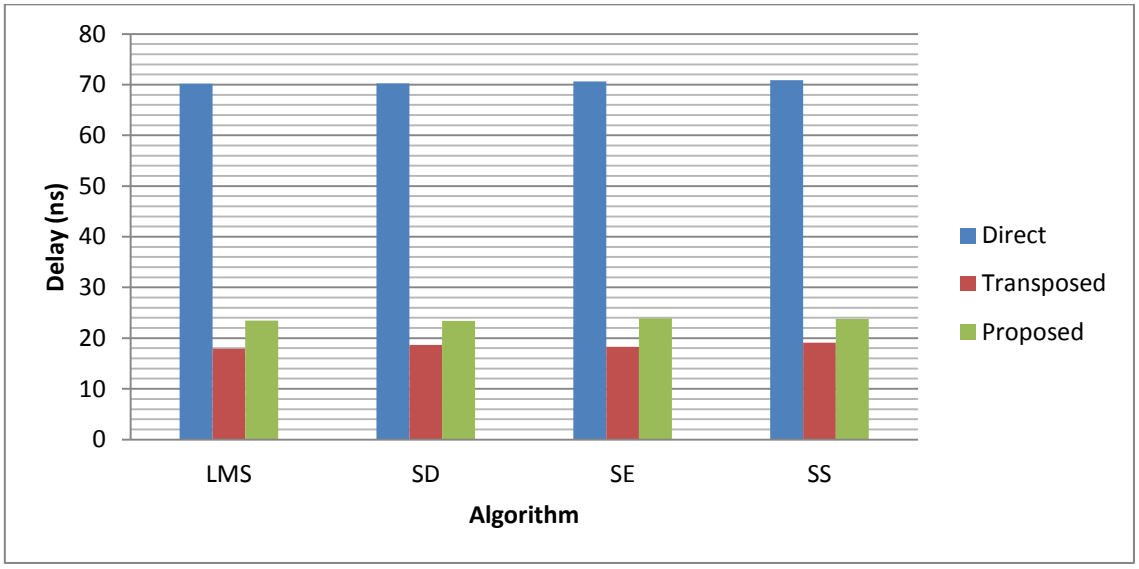

Fig. 5.7 Delay in Virtex7.

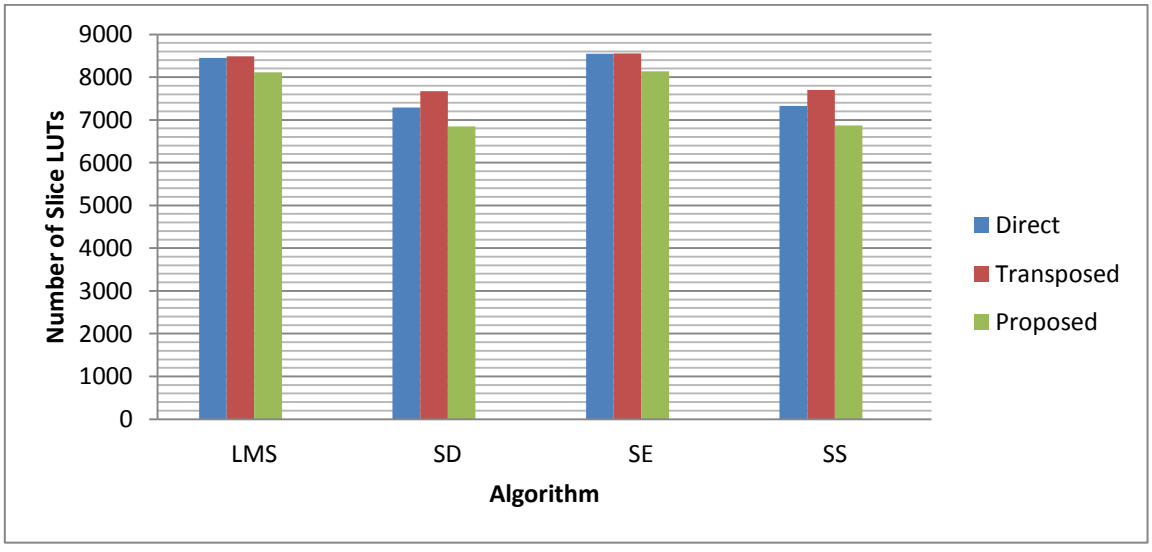

Fig. 5.8 Used Slice LUTs in Virtex7.

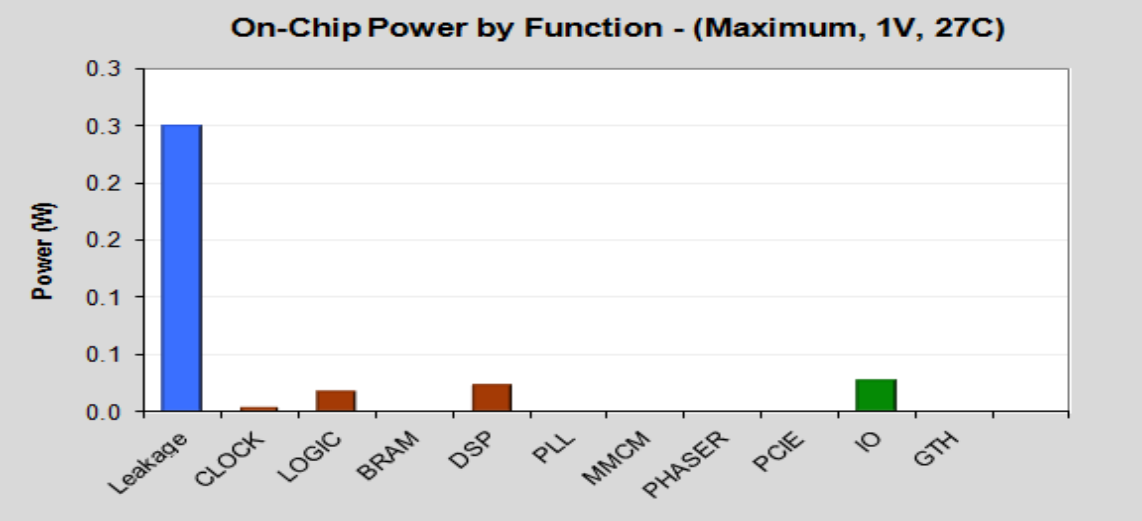

Fig. 5.9 On-Chip Power by Function for LMS Algorithm in Virtex7.

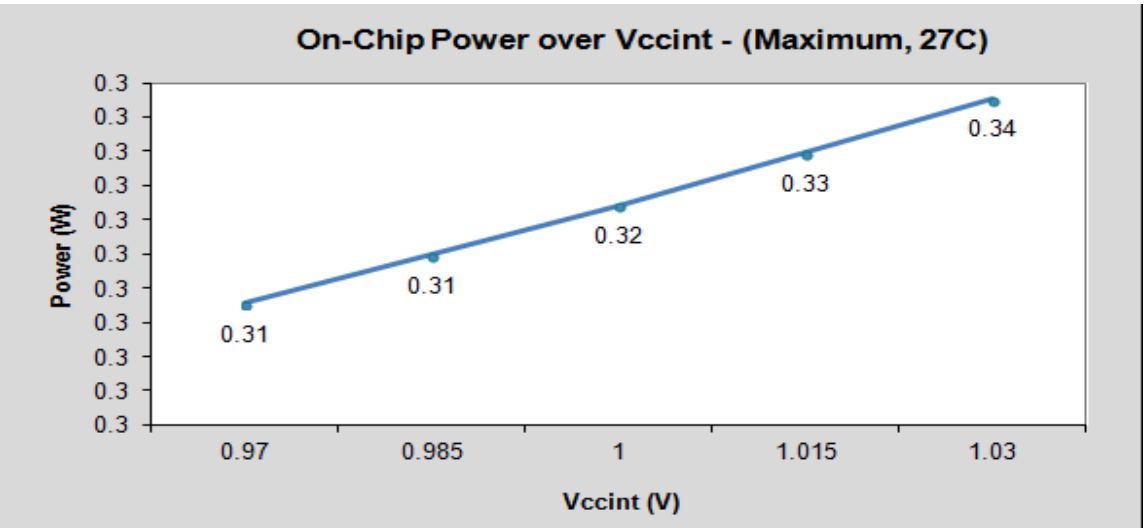

Fig. 5.10 Variation of Power with Supply Voltage at 27C.

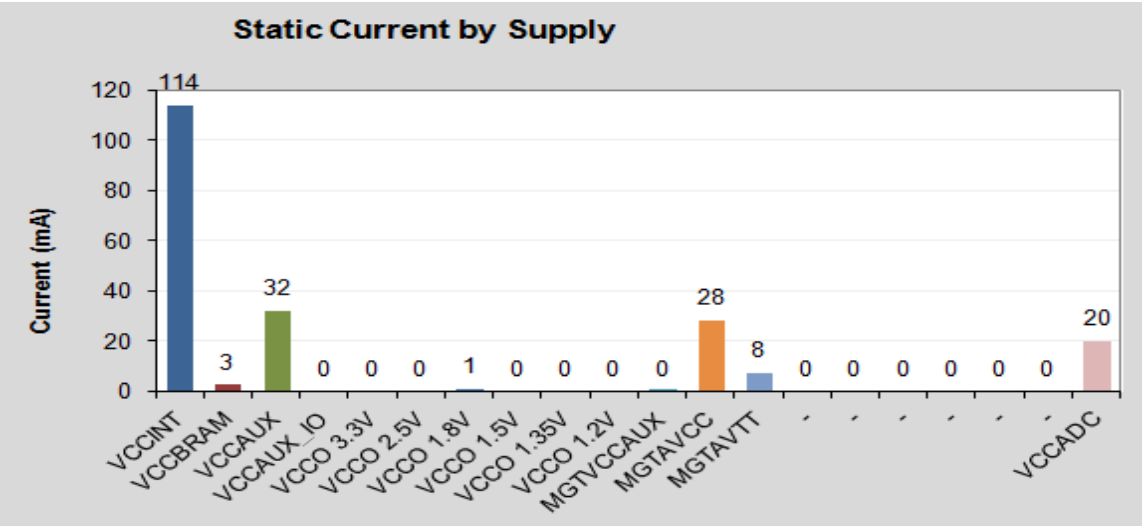

Fig. 5.11 Variation of Static Current with Different Voltages.

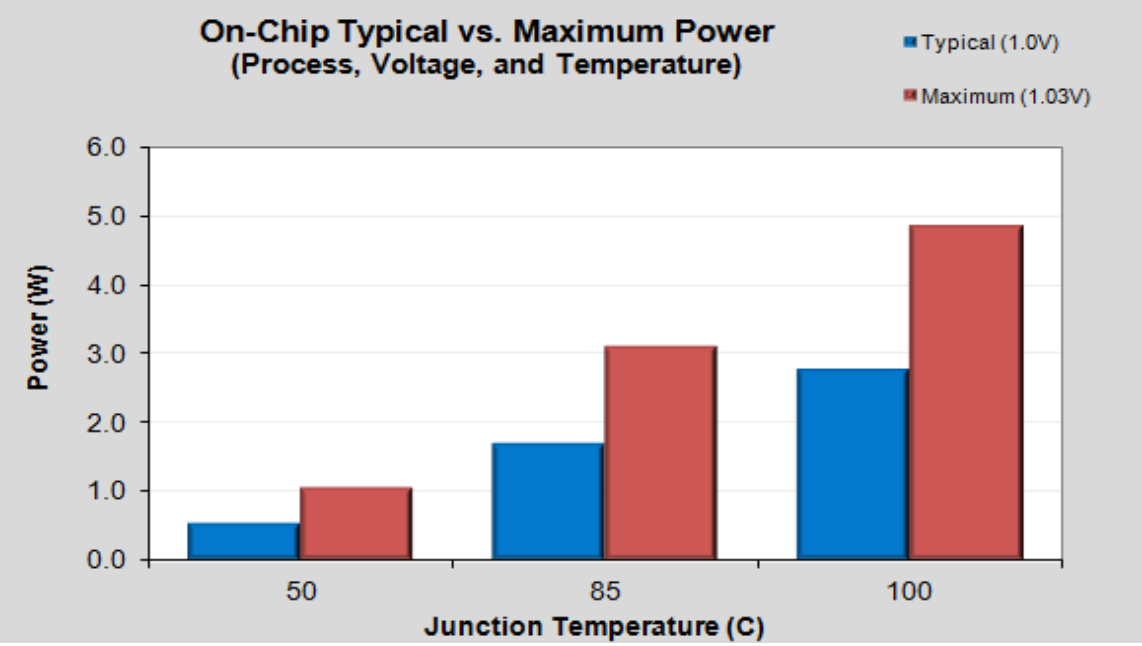

Fig. 5.12 Variation of On-Chip Typical and Maximum Power with Junction Temperature.

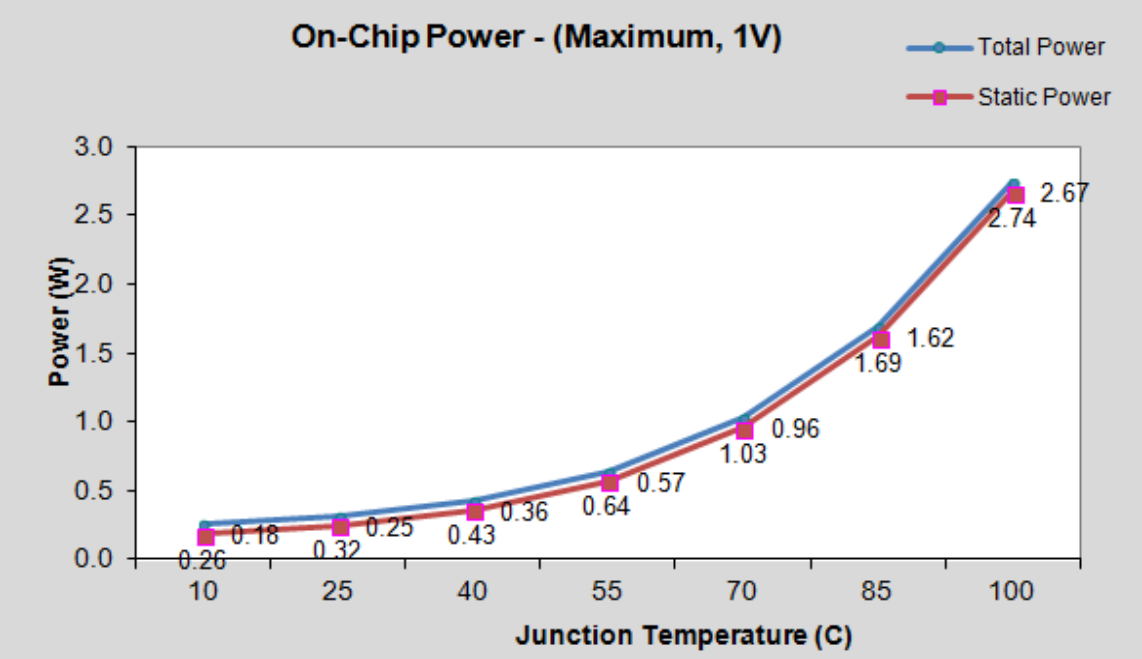

Fig. 5.13 Variation of Total Power and Static Power with Junction Temperature.

proposed structure is due to reduction in critical path. In the power front, the performance of proposed structure is better than the transposed one as depicted in Fig. 5.6. Similar results have been obtained for Spartan 6 and Virtex 6 families and can be read from Tables 5.2-5.5. Also, the functionality of the implemented designs on FPGA was checked using ChipScope Pro. However, for the sake of convenience the result of 40 order variable step size LMS adaptive filter implemented using proposed structure on Spartan 6 are shown in obtained are shown in Fig. 5.14 and Table 5.6.

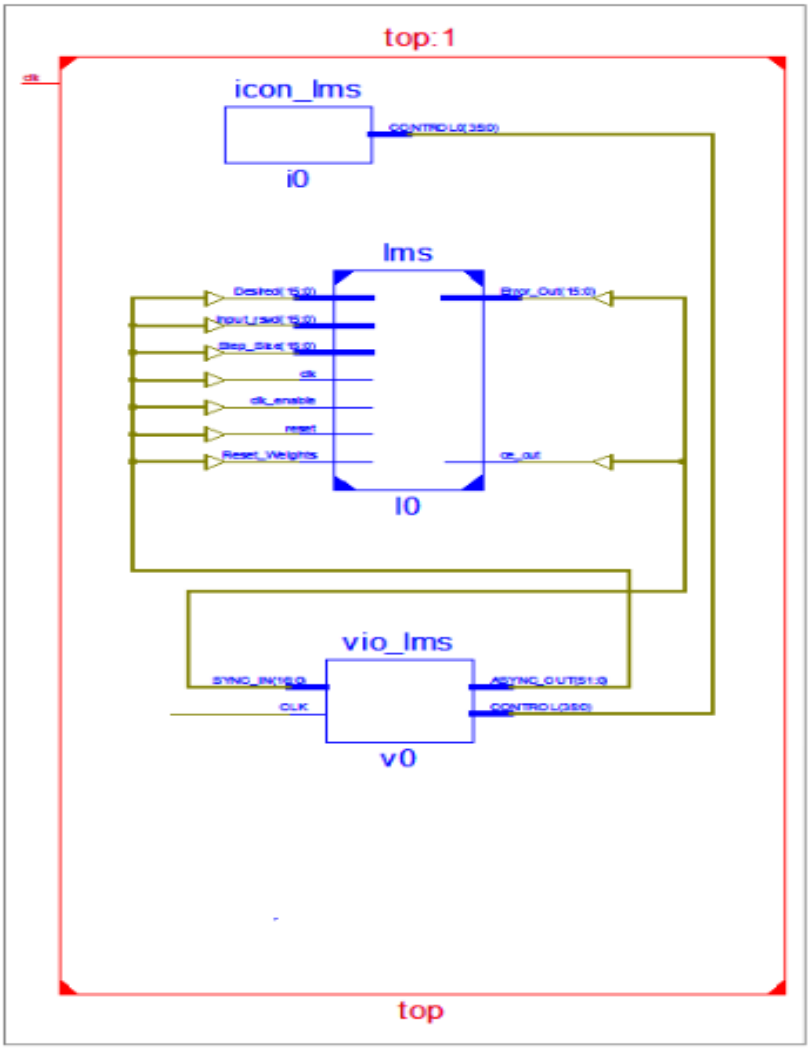

Fig. 5.14 RTL Schematic of ChipScope Pro.

Bus/Signal	Value
AsyncIn[0]	0
AsyncIn	10001011
AsyncIn_1	0001011110001111010101
AsyncOut[0]	0
AsyncOut	10000101
AsyncOut_1	11100000100000000000000
AsyncOut[32]	0
AsyncOut_2	10000101
AsyncOut_3	00000001110000010000000

Table 5.6 ChipScope Pro VIO Core Input and Output.

5.3 ASIC Synthesis Results

The designed adaptive filter (ANC) VHDL code was also synthesized by Leonardo spectrum ASIC synthesizer. Some of the synthesis results that are not interpreted from the synthesis results of Xilinx ISE 14.5 (like gate counts, transistor counts etc.) are listed in Table 5.6. The purpose of synthesis using Leonardo Spectrum was to generate the synthesized verilog netlist (.v file) that can be further used to create the schematic (Transistor level circuit) of the designed ANC at 250 nm technology using “Mentor IC

Station". Transient analysis was performed on the created schematic to check out the functionality of the implemented design. The results of which are shown in Figs. 5.15(a)-5.15(d). Also the layout for all the adaptive filters was created on 250 nm technology and it's DRC and LVS was checked. The layout of the unit cell of four different variants of LMS adaptive filters are shown in Figs. 5.16(a)-5.19(b) while DRC and LVS reports for the unit cell of LMS adaptive filter is shown in Tables 5.8-5.10.

Algorithm	Structures	Nets	Total Accumulated Area		Critical Paths	Delay (ns)
			Accumulated Instances	Number of Gates		
LMS	Direct	48640	1207200	1446720	54942	248.07
	Transposed	30515	1227320	1472864	56541	160.35
	Proposed	29418	1193920	1432064	54686	162.73
SD	Direct	48640	1033480	1278880	56756	256.75
	Transposed	25785	1096840	1313376	58538	177.15
	Proposed	25464	1028560	1262368	60167	178.25
SE	Direct	33024	1201040	1423616	58762	266.10
	Transposed	30747	1218280	1447264	60841	190.35
	Proposed	29750	1181520	1408928	46651	195.25
SS	Direct	30432	1008640	1294176	48730	275.25
	Transposed	25299	1037200	1262240	46875	196.40
	Proposed	24471	993920	1247264	48954	197.55

Table 5.7 Leonardo Spectrum Synthesis.

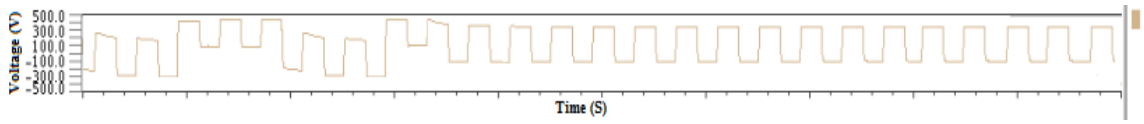

Fig. 5.15(a) Simulation Result of LMS Adaptive Filter.

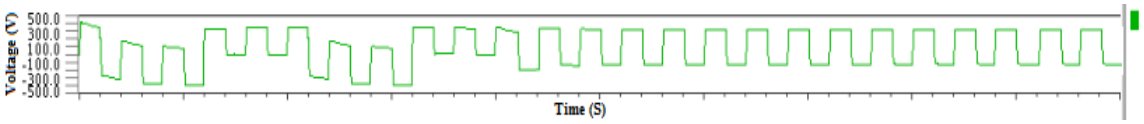

Fig. 5.15(b) Simulation Result of SD Adaptive Filter.

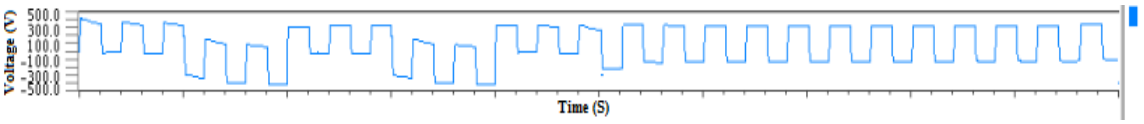

Fig. 5.15(c) Simulation Result of SE adaptive Filter.

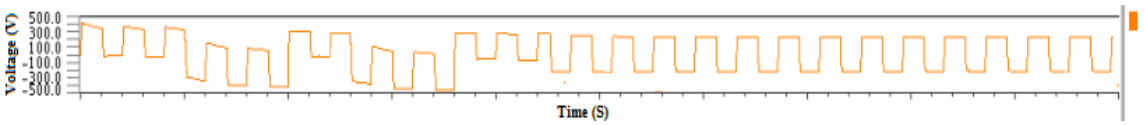

Fig. 5.15(d) Simulation Result of SS Adaptive Filter.

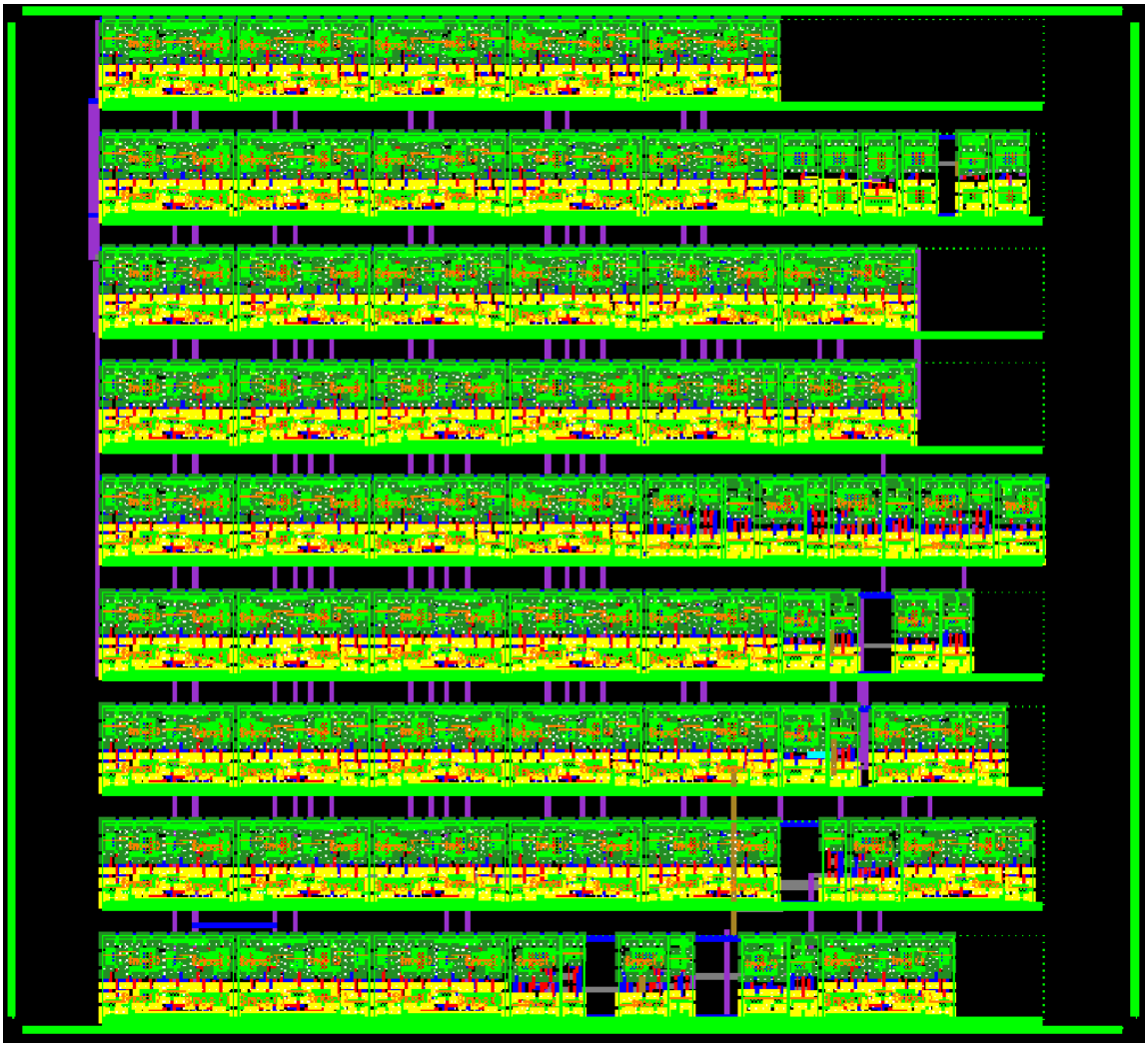

Fig. 5.16(a) Layout of a Basic Block (Unit Cell) of LMS Adaptive Filter.

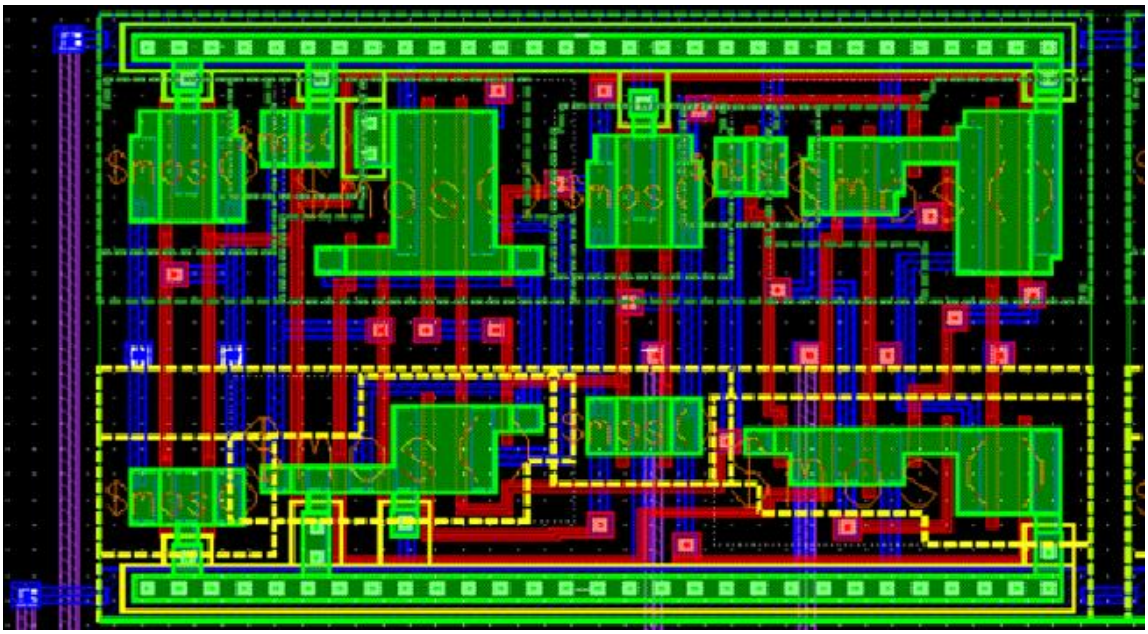

Fig. 5.16(b) Zoomed View of a Small Sub-Section.

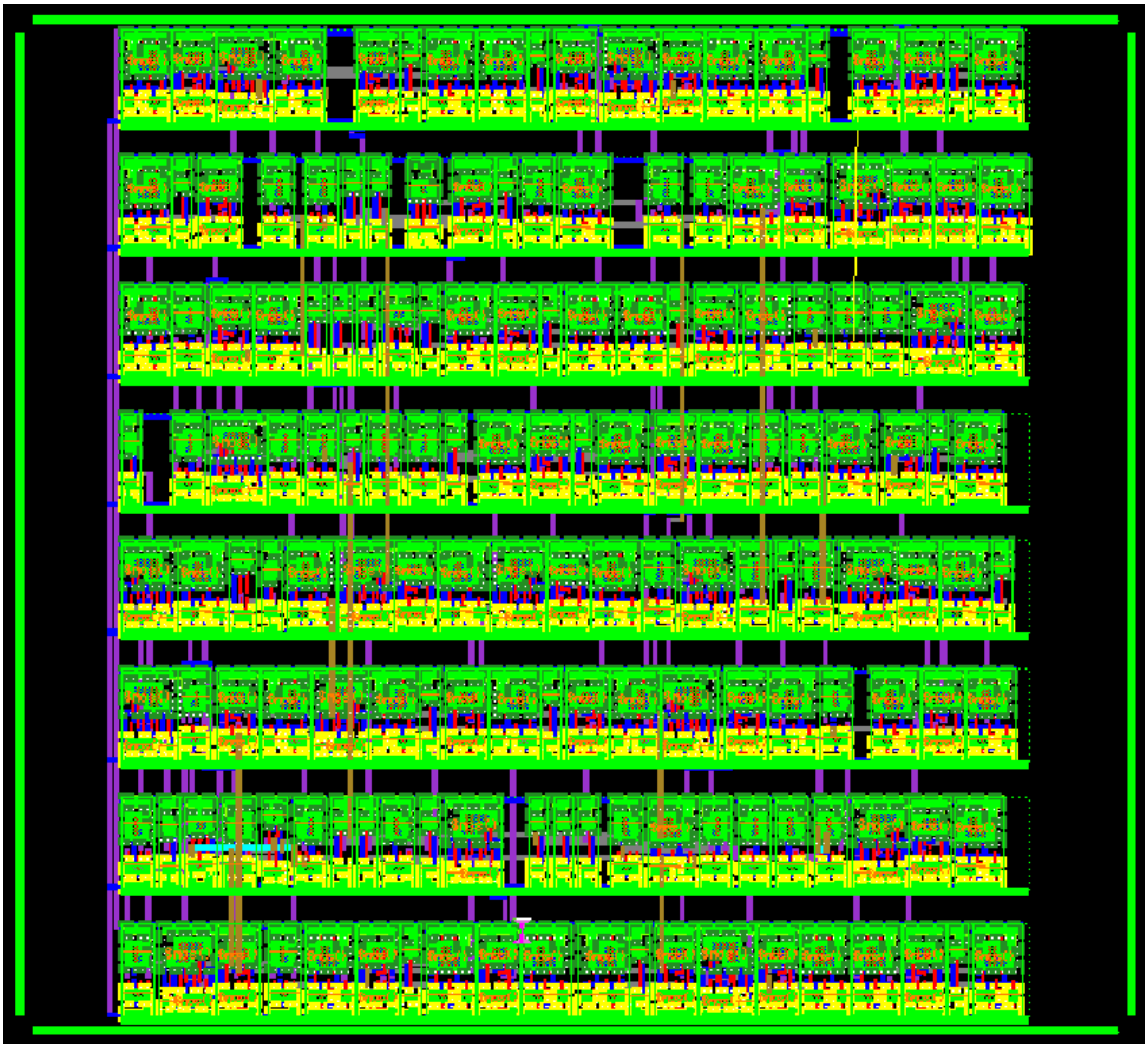

Fig. 5.17(a) Layout of a Basic Block (Unit Cell) of SD Adaptive Filter.

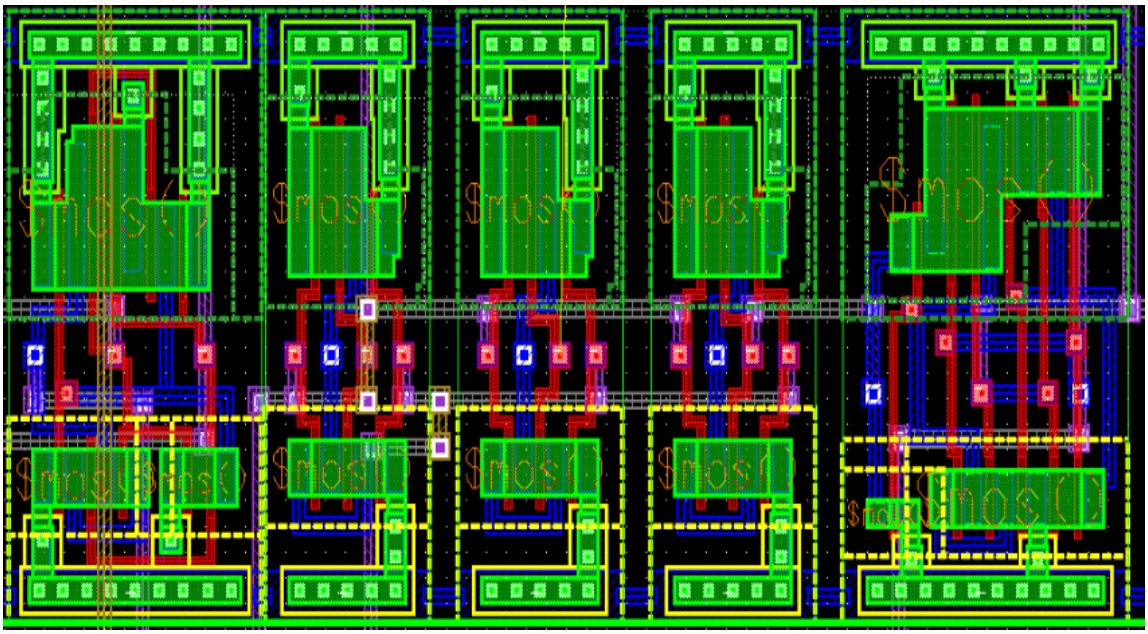

Fig. 5.17(b) Zoomed View of a Small-Subsection.

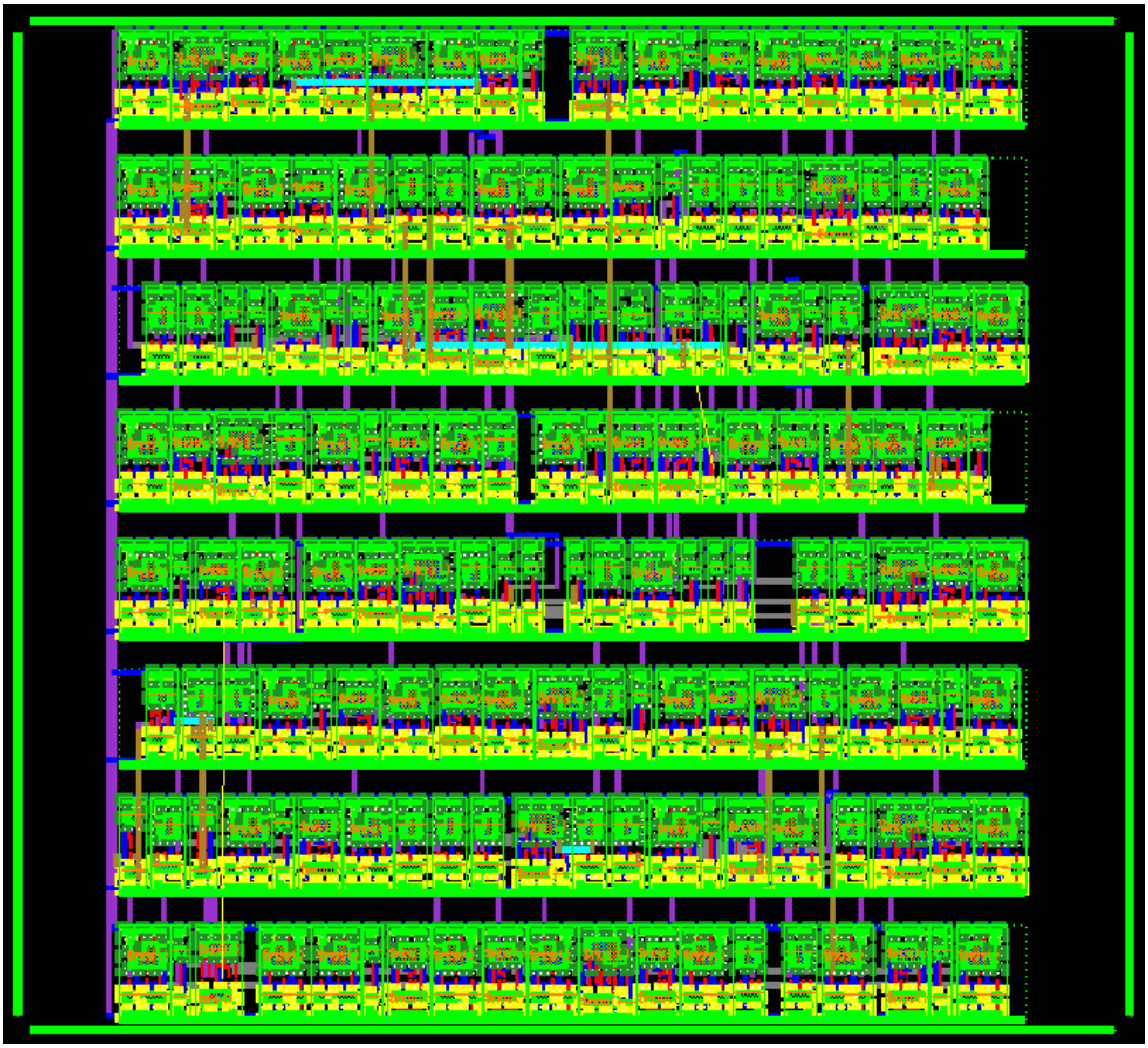

Fig. 5.18(a) Layout of a Basic Block (Unit Cell) of SE Adaptive Filter.

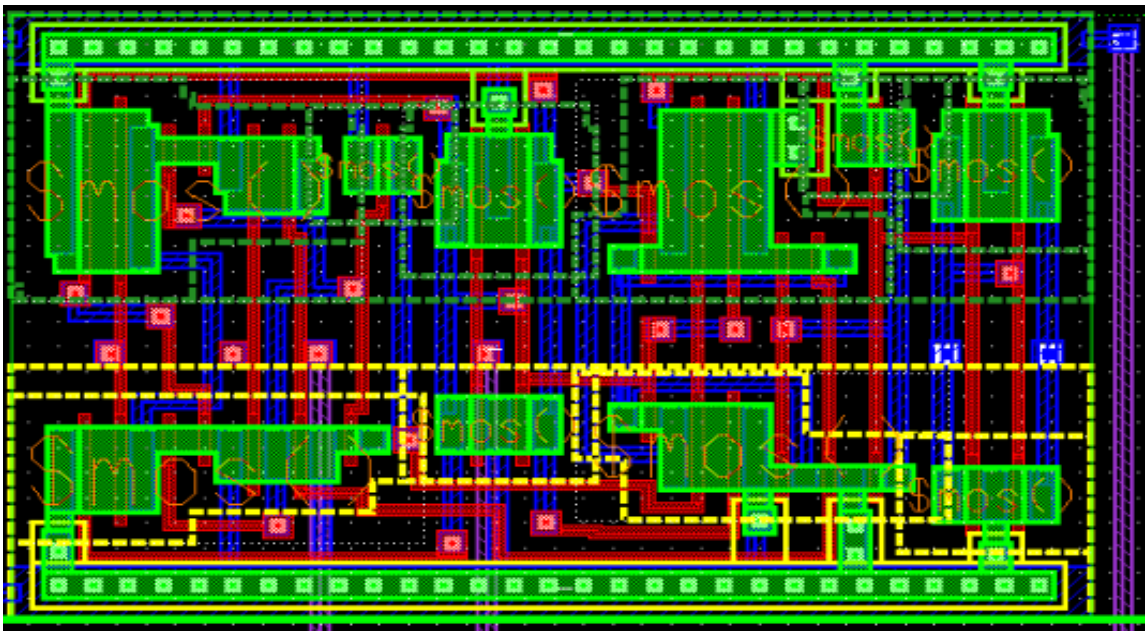

Fig. 5.18(b) Zoomed View of a Small-Subsection.

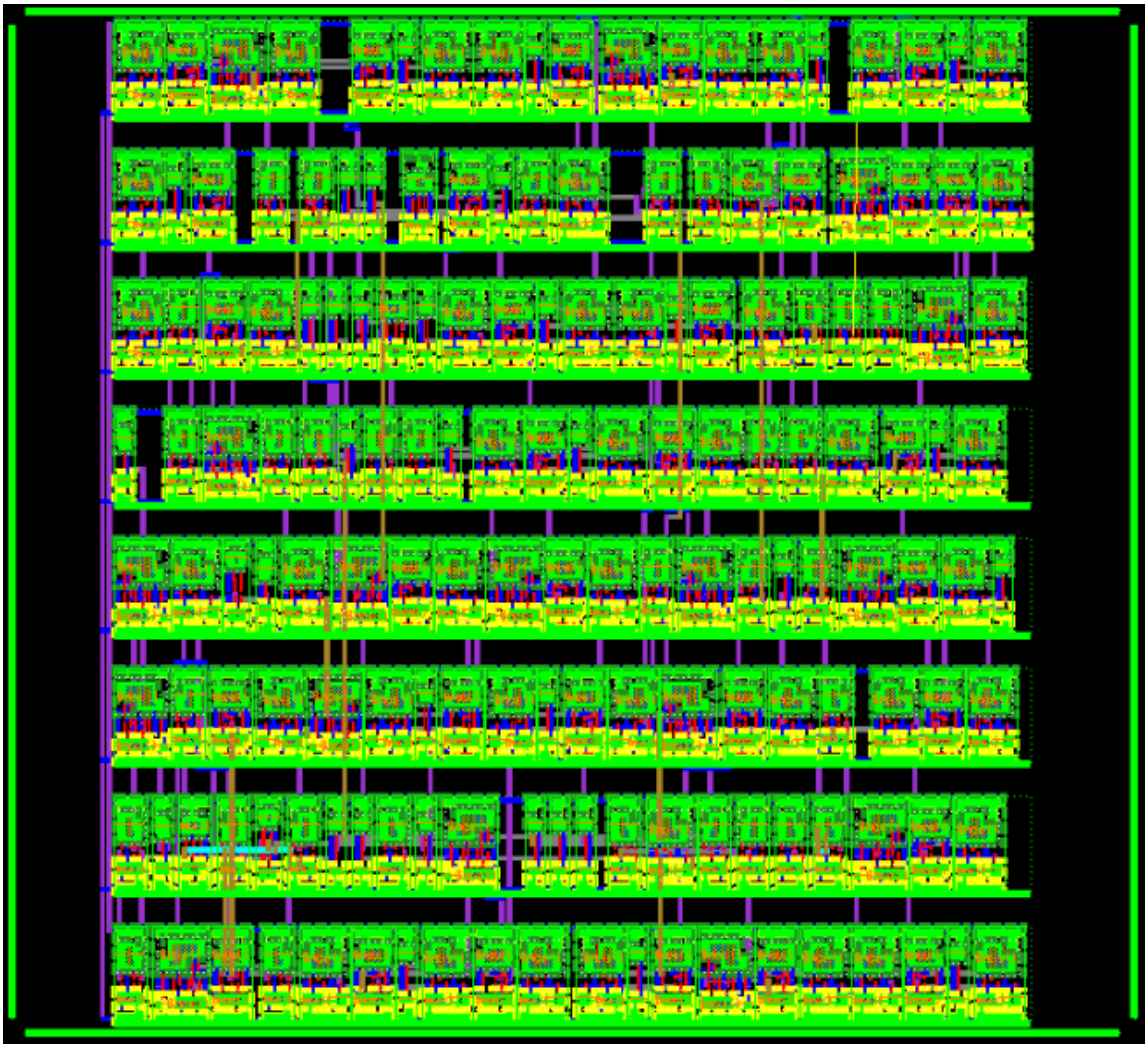

Fig. 5.19(a) Layout of a Basic Block (Unit Cell) of SS Adaptive Filter.

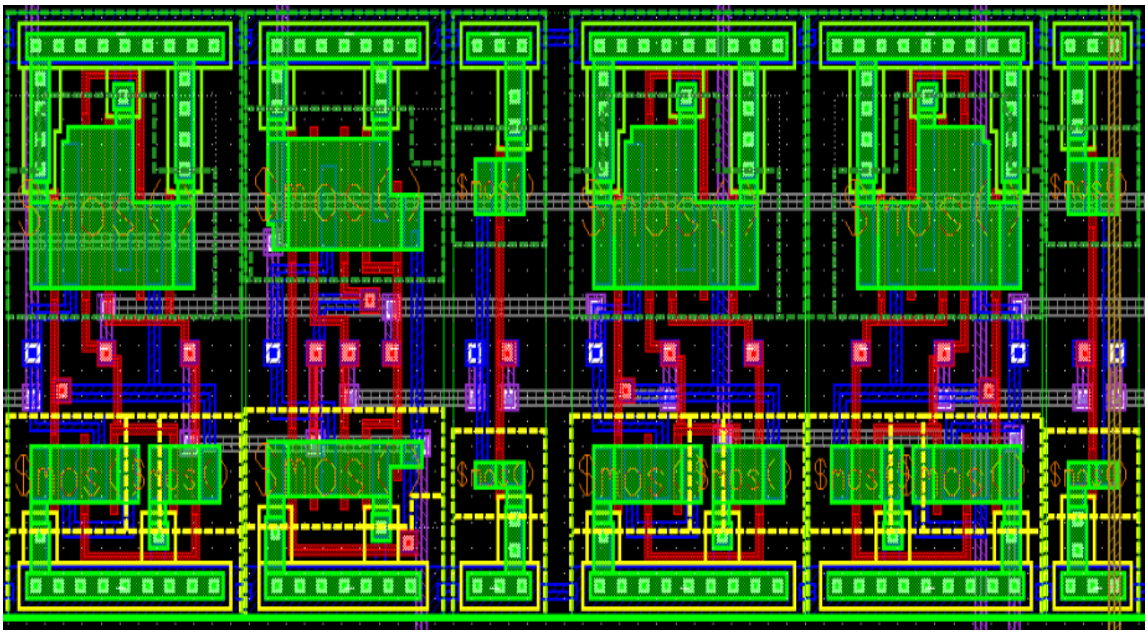

Fig. 5.19(b) Zoomed View of a Small-Subsection.

```
================================================================
                  CALIBRE::DRC-H SUMMARY REPORT

Calibre Version:            v2012.2_17.11    Thu May 10 12:53:48
Maximum Results/RuleCheck: 1000
Maximum Result Vertices:   4096
----------------------------------------------------------------
_
ORIGINAL LAYER STATISTICS

LAYER N_WELL ..............  TOTAL Original Geometry Count = 7
(553)
LAYER POLY ................  TOTAL Original Geometry Count = 152
(12008)
LAYER ACTIVE ..............  TOTAL Original Geometry Count = 49
(3871)
LAYER P_PLUS_SELECT .......  TOTAL Original Geometry Count = 11
(869)
LAYER N_PLUS_SELECT .......  TOTAL Original Geometry Count = 10
(790)
LAYER CONTACT_TO_ACTIVE ...  TOTAL Original Geometry Count = 156
(12324)
LAYER CONTACT_TO_POLY .....  TOTAL Original Geometry Count = 21
(1659)
LAYER METAL1 ..............  TOTAL Original Geometry Count = 274
(9352)
LAYER VIA .................  TOTAL Original Geometry Count = 1
(421)
LAYER METAL2 ..............  TOTAL Original Geometry Count = 180
(1026)
LAYER VIA2 ................  TOTAL Original Geometry Count = 1
(37)
LAYER METAL3 ..............  TOTAL Original Geometry Count = 22
(58)
----------------------------------------------------------------
_
RULECHECK RESULTS STATISTICS (BY CELL)

TOTAL CPU Time:                   7 Sec
TOTAL REAL Time:                  7 Sec
TOTAL Original Layer Geometries: 884 (42968)
TOTAL DRC RuleChecks Executed:    84
TOTAL DRC Results Generated:      0 (0)
----------------------------------------------------------------
```

Table 5.8 DRC Summary Report.

```
                    OVERALL COMPARISON RESULTS


          #      ###################        _   _
        #        #                 #       *   *
    #   #        #     CORRECT     #         |
     # #         #                 #        \___/
      #          ###################
```

```
  Warning:  Ambiguity points were found and resolved arbitrarily.

LAYOUT CELL NAME:         lms
SOURCE CELL NAME:         lms
```

Table 5.9 LVS successful completion.

```
                CELL COMPARISON RESULTS ( TOP LEVEL )

                  #         ####################          __   __
                 #          #                  #         *   *
            #    #          #      CORRECT     #           |
             # #             #                  #          \___/
              #              ####################

Warning:  Ambiguity points were found and resolved arbitrarily.

LAYOUT CELL NAME:           lms
SOURCE CELL NAME:           lms

------------------------------------------------------------------------

INITIAL NUMBERS OF OBJECTS
--------------------------

                  Layout     Source           Component Type
                  ------     ------           --------------
 Ports:              69         69
 Nets:            120355     120355     *
 Instances:       115699     115699     *     MN (4 pins)
                  115699     115699     *     MP (4 pins)
                  ------     ------
 Total Inst:      231398     231398

NUMBERS OF OBJECTS AFTER TRANSFORMATION
---------------------------------------

                  Layout     Source           Component Type
                  ------     ------           --------------
 Ports:              69         69     *
 Nets:             53901      53901     *
 Instances:          265        265     *     MN (4 pins)
                   12156      12156     *     MP (4 pins)
                   11964      11964     *     SPMN_2_1 (5 pins)
                     763        763     *     SPMN_2_2 (6 pins)
                       1          1     *     SPMN_3_2 (7 pins)
                       4          4     *     SPMN_3_3 (8 pins)
                      62         62     *     SPMP_2_1 (5 pins)
                     896        896     *     SPMP_2_2 (6 pins)
                      11         11     *     SPMP_2_2_1 (7 pins)
                      43         43     *     SPMP_3_2 (7 pins)
                       1          1     *     SPMP_3_2_2 (9 pins)
                      11         11     *     SPMP_3_3 (8 pins)
                   17902      17902     *     _invv (4 pins)
                     150        150     *     _invx2v (4 pins)
                      41         41     *     _invx3v (4 pins)
                   12680      12680     *     _nand2v (5 pins)
                      90         90     *     _nand3v (6 pins)
                      82         82     *     _nand4v (7 pins)
                    3979       3979     *     _nor2v (5 pins)
                     104        104     *     _nor3v (6 pins)
                     270        270     *     _nor4v (7 pins)
                   11315      11315     *     _smn2v (4 pins)
                      66         66     *     _smn3v (5 pins)
                   22885      22885     *     _smp2v (4 pins)
                       9          9     *     _smp3v (5 pins)
                  ------     ------
 Total Inst:        95750      95750
```

Table 5.10 Top Cell LVS Comparison Report.

Chapter 6
Conclusion and Future Scope

6.1 Conclusion

The adaptive noise canceller used for de-noising of different signals has been designed and implemented successfully. A comprehensive analysis of the present work reveals a number of facts related to performance of adaptive filters in de-noising tasks. By varying step size for all the algorithms used, the optimum step size and range the of step size where each algorithm works efficiently was found. Among all the algorithms that have been studied it was found that NLMS is best algorithm both in terms of convergence rate and the noise removing capability. This is because of the fact that the optimum step size of NLMS algorithm is largest among the optimum step sizes of other algorithms. Also, NLMS algorithm works for wide range of the step sizes as compared to other all other algorithms. So, its region of operation is large as compared to all other algorithms. The next algorithm which comes after NLMS algorithm in terms of performance is LMS algorithm. This algorithm has the second highest optimum step size and second widest step size operating range. After LMS algorithm the next algorithm which comes in terms of performance and noise removing capability is Sign-Data algorithm. The advantage of this algorithm is that it is simpler than NLMS and LMS algorithm, its error removing capability is at par with LMS algorithm and it consumes less silicon area as compared to other two. This simplicity is because of the presence of signum function at the input that truncates the input signal and consequently reduces the number of addition operations. However, the convergence rate and operating range of step size for which this algorithm removes noise is small as compared to LMS and NLMS algorithm. The Sign-Error algorithm and Sign-Sign algorithm are similar in terms of performance as both algorithms have slow convergence rate and both algorithms works on very small step size. As, a result of smaller step size and slower convergence rate these algorithms are incapable in removing considerable amount of noise from the signal. Also, the sign-sign algorithm is the simplest algorithm as compared to all other algorithms because of the presence of signum function both at input and error port which truncates both the input and error signal and hence reduces the number of addition operations. Further, the sign-error

algorithm is more complex than sign-data algorithm because the presence of signum function at error signal has less impact as compared to presence of signum function at input signal. Thus the number of addition operations is more in sign-error algorithm as compared to the sign-data algorithm.

In terms of the structures that are used for implementing adaptive filters it has been observed that the direct form structures are suitable for low power applications whereas transposed form structures are suitable for high speed applications. A novel structure has been proposed that combines the advantage of both the direct form and transposed form structures. The new structure was designed by modifying direct form structure by replacing the chain of cascaded unit delays by a single delay block of specific value. The value of this single delay is dependent upon the length of the filter. The novel structure gives an optimum tradeoff in terms of used silicon area irrespective of all the four variants of LMS algorithm employed. This reduction in area owes itself to reduction in the number of multiplexers and registers which in turns is due to the generation of vectorized delay. Also, the proposed structure performed better than the direct form structure in terms of in terms of speed where a three-fold increase in the speed has been witnessed. However, when compared to the transformed form structure (one with the minimum delay) the reduction in speed was not more than 15%. Despite a three-fold increase in the speed compared to the direct form structure the proposed structure exhibits a significantly lower penalty in terms of power which makes it a good alternative for more efficient ASIC and FPGA implementations.

6.2 Future Scope

The future Scope of the work includes the following:

1. The designed adaptive noise canceller can further be implemented using more advanced technologies (less than 250 nm) to increase the operating speed.

2. The proposed adaptive filter structure can be implemented by using pipelining technique so that its power dissipation can be reduced further.

3. The proposed adaptive filter structure can be implemented using Parallel processing technique so that ultra high speed adaptive filters can be designed.

4. Pipelining and Parallelism techniques can be implemented simultaneously to optimize the performance of the proposed structure.

5. The proposed structure can also be used to design multiplier free adaptive filters by using radix 4 or sum power of two or distributed arithmetic implementation techniques.

References

[1] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice- Hall, 1986.

[2] Jih-Gau Juang, Li-Hsiang Chien, F. Lin, "Automatic Landing Control System Design Using Adaptive Neural Network and its Hardware Realization," *IEEE Systems Journal*, vol. 5, no. 2, pp. 266-277, June 2011.

[3] Monson Hayes H, *Statistical Digital Signal Processing and Modelling*. John Wiley & Sons Inc:Kundli, 2002.

[4] B. Widrow and S. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[5] A. Sayed, *Fundamentals of Adaptive Filtering*. New York: Wiley, 2003.

[6] E. Soria, J. D. Martín, A. J. Serrano, J. Calpe, and J. Chambers, "Steady-state and tracking analysis of a robust adaptive filter with low computational cost," *Signal Process.*, vol. 87, no. 1, pp. 210–215, Jan. 2007.

[7] K. J. Lee, J.P.Lee, D. Shin, D. W. Yoo, H. J. Kim, "A Novel Grid Synchronization PLL Method Based on Adaptive Low-Pass Notch Filter for Grid-Connected PCS," *IEEE Trans. Ind. Electron.*, vol. 61, no. 1, pp. 292–301, Jan. 2014.

[8] Jimaa S, Al-Ali S, "Convergence Performances of various Adaptive Filter Algorithms with Application to System Identification," *Proceedings of IEEE GCG conference and Exhibition*, Dubai, 2011, pp. 19-22.

[9] B. F. Boroujeny, *Adaptive Filters: Theory and Applications*. New York: Wiley, 1998.

[10] R. H. Kwong and E. W. Johnsto, "A variable step size LMS algorithm," *IEEE Trans. Signal Process.*, vol. 40, no. 7, pp. 1633-1642, 1992.

[11] B. K. Mohanty and P. K. Meher, "A High-Performance Energy-Efficient Architecture for FIR Adaptive Filter Based on New Distributed Arithmetic Formulation of Block LMS Algorithm," *IEEE Transactions on Signal Processing*, Vol. 61, No. 4, pp. 921-932, Feburary 2013.

[12] M. D. Meyer and D. P. Agrawal, "An Arbitrarily High Sampling Rate DLMS Adaptive Filter," in *Proc. IEEE International Symposium on Circuits and Systems*, 1992, pp. 2176-2179.

[13] K. K. Parhi, *VLSI Digital Signal Processing Systems—Design and Implementation*, New York: Wiley, 1999.

[14] U. M. Baese, *Digital Signal Processing With Field Programmable Gate Arrays*, 3rd ed. Berlin, Germany:Springer-Verlag, 2007.

[15] E. S. Nejevenko and A. A. Sotnikov, "Adaptive modeling for hydroacoustic signal processing," *Pattern Recognit. Image Anal.*, vol. 16, no. 1,pp. 5–8, Jan. 2006.

[16] S. Ikeda and A. Sugiyama, "An adaptive noise canceller with low signal distortion for speech codecs," *IEEE Transactions on Signal Processing*, Vol. 47, No. 3, pp. 665-674, March 1999.

[17] P. Prandoni and M. Vetterli, "An FIR Cascade Structure for Adaptive Linear Prediction," *IEEE Transactions on Signal Processing*, Vol. 46, No. 9, pp. 2566-2571, September 1998.

[18] P. M. S. Burt, "Inverse Identification Adaptive IIR Filtering: Convergence Speed Analysis and Successive Approximations Algorithm," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2007, pp. 1309 - 1312.

[19] B. Widrow, J. R. Glover Jr., J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. Dong Jr., R.C. Goodlin, "Adaptive noise cancelling: Principles and applications," *Proceedings of the IEEE*, Vol. 63, No. 12, pp. 1692 - 1716, Dec. 1975.

[20] B. Widrow, J. M. McCool, M. G. Larimore, C. R. Johnson Jr., "Stationary and nonstationary learning characteristics of the LMS Adaptive Filters," *Proceedings of the IEEE*, Vol. 64, No. 8, pp.1151-1162, August 1976.

[21] D. L. Jones, "Learning Characteristics of Transpose-Form LMS Adaptive Filters," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 39, No.10, pp. 745-749, October 1992.

[22] D. T. M. Slock, “On the Convergence Behavior of the LMS and the Normalized LMS Algorithms,” *IEEE Transactions on Signal Processing* Vol. 41, No. 9, pp. 2811-2825, September 1993.

[23] E. Eweda, “Comparison of RLS, LMS, and Sign Algorithms for Tracking Randomly Time-Varying Channels,” *IEEE Transactions on Signal Processing*, vol. 42. No. II, pp. 2937 – 2944, November 1994.

[24] N. R. Shanbhag and M. Goel, “Low-Power Adaptive Filter Architectures and their Application to 51.84 Mb/s ATM-LAN,” IEEE Transactions On Signal Processing, Vol. 45, No. 5, pp. 1276 – 1289, MAY 1997.

[25] S. J. Visser, A. S. Dawood, and J. A. Williams, “FPGA Based Real-time Adaptive Filtering for Space Applications,” in *Proc. IEEE International Conference on Field-Programmable Technology*, 2002, pp. 322-326.

[26] A. Elhossini, S. Areibi, and R. Dony, "An FPGA implementation of the LMS adaptive filter for audio processing," in *Proc. IEEE International Conference on Reconfigurable Computing and FPGA's*, September 2006, pp. 1-8.

[27] M. Vella, and C.J. Debono, "The implementation of a high speed adaptive FIR filter on a field programmable gate array," in *Proc. IEEE Mediterranean Electrotechnical Conference (MELECON)*,Benalmadena, Spain, May 2006, pp. 113-116.

[28] Z. Gao, X. Zeng, J. Wang, J. Liu, "FPGA implementation of adaptive IIR filters with particle swarm optimization algorithm," in *Proc. 11th IEEE International Conference on Communication Systems (ICCS)*, Singapore, 2008, pp. 1364-1367.

[29] R. Mustafa, M. A. Mohd Ali, C. Umat, and D. A. Al-Asady, "Design and implementation of least mean square adaptive filter on altera cyclone II field programmable gate array for active noise control," in *Proc. IEEE Symposium on Industrial Electronics and Applications (ISIEA)*, Kuala Lumpur, Malaysia, October 2009, pp. 479-484.

[30] Y. Mollaei, "Hardware implementation of adaptive filters," in *Proc. IEEE Student Conference on Research and Development (SCOReD)*, Malaysia, November 2009, pp. 45-48.

[31] Rosado-Muñoz A, Bataller-Mompeán M, Soria-Olivas E, Scarante C, Guerrero-Martínez, JF, "FPGA Implementation of an Adaptive Filter Robust to Impulsive Noise: Two Approaches," *IEEE Transactions on Industrial Electronics*, Vol. 58, No. 3, pp. 860-870, March 2011.

[32] Nekouei.F, Talebi.N.Z, Kavian.Y.S, Mahani.A, "FPGA Implementation of LMS Self Correcting Adaptive Filter (SCAF) and Hardware Analysis," in *Proc. 8th IEEE, IET International Symposium on Communication Systems, Networks and Digital Signal Processing*, 2012, pp. 1-5.

[33] Tudosa I, Adochiei N, "LMS Algorithm Derivatives used in Real-Time Filtering of ECG Signals: A Study Case on Performance Evaluation," *Proceedings of IEEE International Conference and Exposition on Electrical and Power Engineering*, Romania, 2012, pp.565-570.

[34] S. Choudhary, P. Mukherjee, M. Chakraborty, S.S. Rath, "A SPT Treatment to the Realization of the Sign-LMS Based Adaptive Filters," in *Proc. IEEE transactions on circuits and systems—I: regular papers*, vol. 59, no. 9, pp. 2025 – 2033, September 2012.

[35] A. B. Diggikar and S. S. Ardhapurkar, "Design and Implementation of Adaptive filtering algorithm for Noise Cancellation in speech signal on FPGA," in *Proc. International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, 2012, pp. 766-771.

[36] M. Z. U. Rahman, R. A. Shaik, and D. V. R. K. Reddy, "A Non-Linearities based Noise Canceler for Cardiac Signal Enhancement in Wireless Health Care Monitoring," in *Proc. IEEE Global Humanitarian Technology Conference*, 2012, pp. 288-292.

[37] G. Cai, C. Liang, J. Yang, and H. Li, "Design and Implementation of LMS Adaptive Filter Algorithm Based on FPGA," in *Proc. 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA)*, 2013, pp. 383-385.

[38] V. Ramakrishna and T.A. Kumar, "Low Power VLSI Implementation of Adaptive Noise Canceller Based on Least Mean Square Algorithm," in *Proc. 4th IEEE*

International Conference on Intelligent Systems, Modelling and Simulation, 2013, pp. 276-279.

[39] S. Kelkar and R. Kamal, "Adaptive Fault Diagnosis Algorithm for Controller Area Network," *IEEE Trans. Ind. Electron*., vol. 61, no. 10, pp. 5527–5537, Oct. 2014.

[40] J. Rose, A. E. Gamal and A. S. Vincentelli, "Architecture of Field Programmable Gate Arrays," *Proceedings of the IEEE*, Vol. 81, No. 7, pp. 1013 - 1029, July. 1993.

Publications

1. Aditya Bali and Ravi Kumar, "Performance of LMS Algorithm Derivatives in De-Noising: A Comparative Study," in *Proc. International Conference on Electrical, Electronics and Computer Science Engineering (IEECSE),* 2014, pp. 6-12. (Published).

2. Aditya Bali and Ravi Kumar, "FPGA Implementation and Hardware Analysis of LMS Algorithm Derivatives: A Case Study on Performance Evaluation," *International Journal of Electronics Letters*. (Communicated).

3. Aditya Bali and Ravi Kumar, "FPGA Implementation of Variable Step Size Adaptive Filters for Signal De-Noising," *IEEE Transactions on Industrial Electronics.* (Communicated).