FPGA IMPLEMENTATION OF OPTIMAL STEP SIZE NLMS ALGORITHM AND ITS PERFORMANCE ANALYSIS

L. Bharani[1], P. Radhika[2]

[1]PG Student, [2]Assistant Professor, Department of ECE, SRM University, Tamil Nadu, India
bharani023@gmail.com, gpradhika82@gmail.com

Abstract

The Normalized Least Mean Square error (NLMS) algorithm is most popular due to its simplicity. The conflicts of fast convergence and low excess mean square error associated with a fixed step size NLMS are solved by using an optimal step size NLMS algorithm. The main objective of this paper is to derive a new nonparametric algorithm to control the step size and also the theoretical performance analysis of the steady state behavior is presented in the paper. The simulation experiments are performed in Matlab. The simulation results show that the proposed algorithm as superior performance in Fast convergence rate, low error rate, and has superior performance in noise cancellation.

Index Terms: *Least Mean square algorithm (LMS), Normalized least mean square algorithm (NLMS)*

----------***----------

1. INTRODUCTION

The Least Mean Square (LMS) algorithm is introduced by Hoff in 1960.In diverse fields of engineering Least Mean Square algorithm is used because of its simplicity. It has been used in many fields such as adaptive noise cancellation, adaptive equalization, side lobe reduction in matched filters, system identification etc. By using simple architecture for the implementation of variant Block LMS algorithm in which weight Updation and error calculation are both calculated in block wise, Hardware outputs are verified with simulations from FPGA.

For the computation efficiency of the LMS algorithm some additional simplification are necessary in some application. There are many approaches to decrease the computational requirements of LMS algorithm that is block LMS algorithm [1-2]. In Block LMS algorithm technique involves calculation of a block of finite set of output values from block of input values. Efficient parallel processors can be used in block implementations of digital filters which results in speed gains [4]. LMS is one of the adaptive filtering algorithms derived from steepest descent algorithm used in wide variety of applications. Block LMS is one of the variants in which the weights are updated once per every block of data instead of updating on every clock cycle of input data.

1.1 Algorithm Formulation:

In [1], the adaptation step size is adjusted using the energy of the instantaneous error. The weight update recursion is given by

$$W(n+1) = W(n) + \mu(n)\, e(n)\, X(n) \quad (1)$$

And the step-size update expression is

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

The constant μ_{max} is normally selected near the point of instability of the conventional LMS to provide the maximum possible convergence speed. The value of is chosen as a compromise between the desired level of steady state misadjustment and the required tracking capabilities of the algorithm. The parameter controls the convergence time as well as the level of misadjustment of the algorithm.

The algorithm has preferable performance over the fixed step-size LMS [3]: At early stages of adaptation, the error is large, causing the step size to increase, thus providing faster convergence speed. When the error decreases, the step size decreases, thus yielding smaller misadjustment near the optimum [3]. However, using the instantaneous error energy as a measure to sense the state of the adaptation process does not perform as well as expected in the presence of measurement noise. This can be seen from (3). The output error of the identification system is

$$E(n) = d(n) - X^T(n)\, W(n) \quad (3)$$

Where the desired signal d (n) is given by,

$$D(n) = X^T(n)\, W^*(n) + \varepsilon(n) \quad (4)$$

1.2 Normalized Least Mean Square algorithm (NLMS)

To derive the NLMS algorithm we consider the standard LMS recursion, for which we select a variable step size parameter, μ (n). This parameter is selected so that the error value, e (n), will be minimized using the updated filter tap weights, w (n+1), and the current input vector, x (n).

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

$$E(n) = d(n) - wT(n+1)x(n) \qquad (5)$$

Next we minimize (e(n))2, with respect to μ(n). Using this we can then find a value for β (n) which forces e(n) to zero.

$$\mu(n) = 1/2xTx(n) \qquad (6)$$

This μ (n) is then substituted into the standard LMS recursion replacing x (n), resulting in the following.

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

$$W(n+1) = w(n) + 1/(xT(n)x(n))\, e(n)x(n) \qquad (7)$$

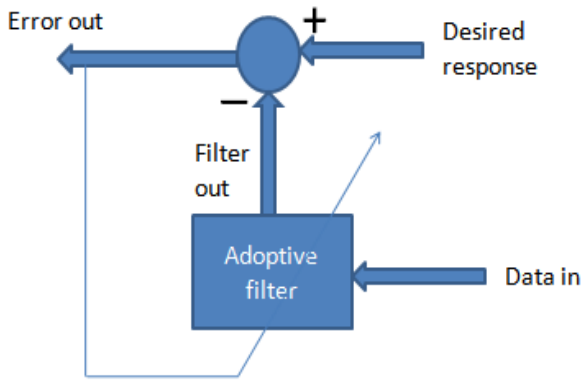

Fig -1: Adaptive filter structure

Often the NLMS algorithm is expressed as equation 5.20; this is a slight modification of the standard NLMS algorithm detailed above. Here the value of is a small positive constant in order to avoid division by zero when the values of the input vector are zero. The parameter μ is a constant step size value used to alter the convergence rate of the NLMS algorithm, it is within the range of 0<μ<2, usually being equal to 1.

1.3 FPGA Realization Issues:

To Field programmable gate arrays are ideally suited for the implementation of adaptive filters. However, there are several issues that need to be addressed [4-5]. When performing software simulation of adaptive filters, calculations are normally carried out with floating point precision. Unfortunately, the resources required of an FPGA to perform floating point arithmetic is normally too large to be justified, and measures must be taken to account for this. Another concern is the filter tap itself. Numerous techniques have been devised to efficiently calculate the convolution operation when the filters coefficients are fixed in advance. For an adaptive filter whose coefficients change over time, these methods will not work or need to be modified significantly.

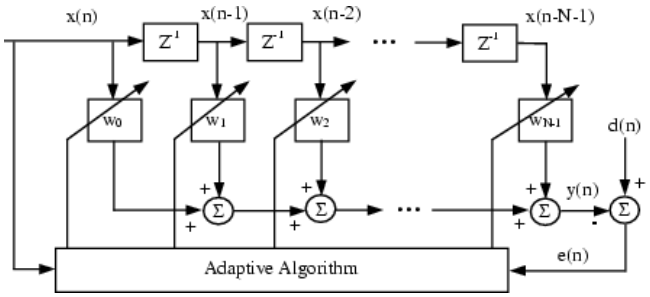

Fig -2: Design of Transversal Filters

2. IMPLEMENTATION OF ALGORITHMS

LMS algorithm mainly consists of two basic process
1) Filtering process
2) Adaptive process

Filtering process:
In filtering process FIR filter output is calculated by convolving inputs and tap weights. Estimation error is calculated by comparing the output with desired signal.

Adaptive process
In adaptive process tap weights are updated based on the estimation error.

Three Steps Involved
The Calculation of filter output, Estimation of error and tap weight Updation.

2.1 LMS adaptive filter: Basic Concepts:

In this algorithm filter weights are updated with each new sample as required to meet the desired output. The computation required for weights update is illustrated by equation (1). If the input values u(n),u(n - 1),u(n - 2)....u(n - N + 1) form the tap input vector u(n), where N denotes the filter length, and the weights w^0(n)……w^N-1(n) form the tap weight vector w(n) at iteration n, then the LMS algorithm is given by the following equations:

$$y(n) = w\hat{}h(n)*u(n)$$

$$e(n) = d(n)-y(n)$$

$$w\hat{}(n+1) = w\hat{}(n)+M*u(n)*e(n) \qquad (8)$$

Where y(n) denotes the filter output. D (n) denotes the desired output. E (n) denotes the filter error (the difference between the desired filter output and current filter output) which is used to update the tap weights. M denotes a learning rate, and W^(n+1) denotes the new weight vector that will be used by the next iteration.

2.2 Variable Step Size:

$$wi(n+1) = wi + 2\mu gx(n)$$

$$G(n) = e(n)x(n) \qquad (9)$$

Where g is a vector comprised of the gradient terms, gi(n)=e(n)x(n-i), i=0…N-1, the length corresponds to the order of the adaptive filter[6]. The values for μ (n) can be calculated in either of the methods expressed in equation 10. The choice is dependent on the application, if a digital signal processor is used then the second equation is preferred. However, if a custom chip is designed then the first equation is usually utilized. For both the Matlab and real time applications the first equation is being implemented. Here μ is a small positive constant optionally used to control the effect of the gradient terms on the update procedure, in the later implementations this is set to 1

$$\mu i\,(n) = \mu i(n-1) + psign\,(gi(n))sign(gi(n-1))$$

$$\mu i(n) = \mu i(n-1) + pgi(n)gi(n-1) \qquad (10)$$

In order to ensure the step size parameters do not become too large (resulting in instability), or too small (resulting in slow reaction to changes in the desired impulse response), the allowable values for each element in the step size are bounded by upper and lower values.

2.3 Resource usage in implementation

Here the architecture is designed to perform in real time implementation. In input buffering RAMS the continuous incoming data is stored that provide the calculations involved until for the weight updating [7]. From each of the input RAM blocks the data is read out alternatively and passed in to input data memory. In tap weight memory block tap weights are initially stored. Both weights and input data's are passed to the multiplier simultaneously for multiplication which multiplies both weight vector and data vector. This result is passed to adder which adds the output of the multiplier. Then the adder output is equivalent to the FIR filter output,(which ideally requires N multipliers depending on the number of taps which here is reduced to one) is then subtracted from another input sample to calculate the error.

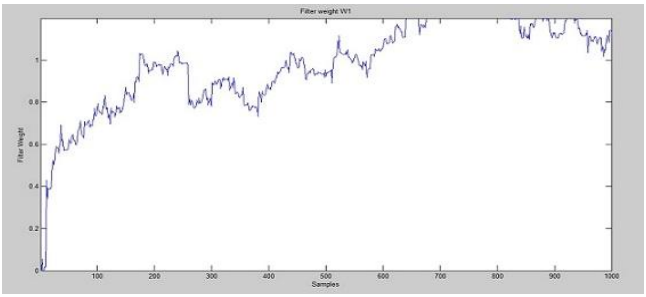

Fig -3: Filter weight updates using LMS

To calculate the weight updates the obtained error value is multiplied with data vector and step size to reduce the error from each calculation. Then updated weights are added to previous weights and written back to weight memory [8]. Then updated weights are ready for another data set. As mentioned earlier ideally FIR filter structure requires N multiplier depending on the number of weights considered for implementation. Here multiplier used is reduced to one so architecture presented consumes minimum hardware which is convenient for FPGA implementation. The updated weights are added to previous weights and written back to memory.

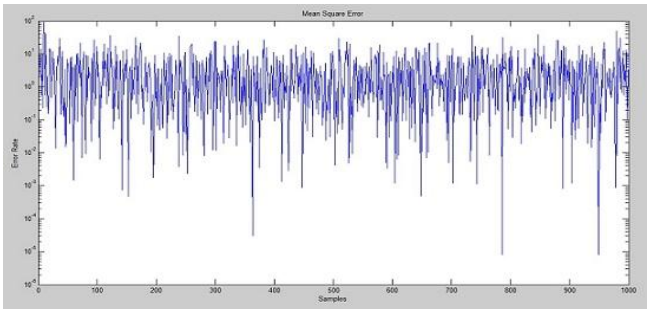

Fig -4: Mean Square error of LMS Algorithm

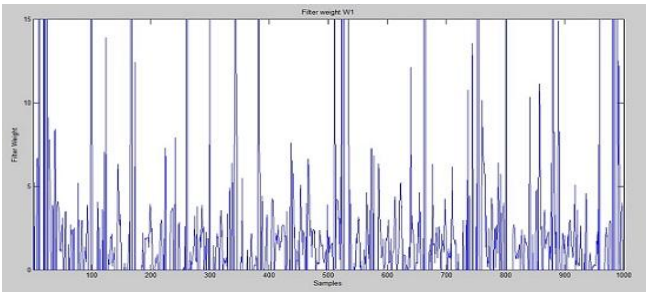

Fig -5: Filter weight updates using NLMS

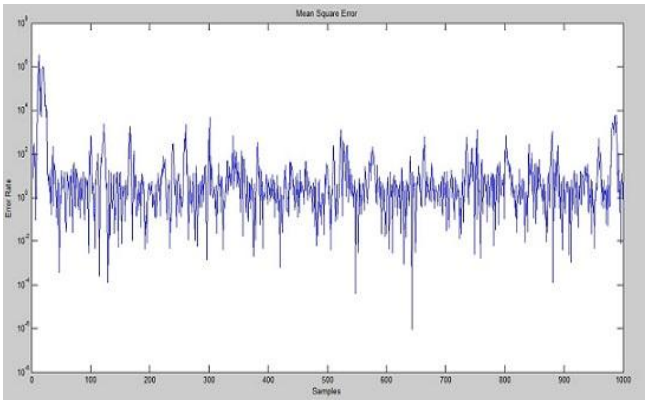

Fig -6: Mean square error graph OF NLMS

Table -1: Comparison of LMS, VSSLMS AND NLMS

Algorithms	Step Size	Error Rate
LMS	0.01-0.05	9.5009
VSS LMS	Variable(0-10)	6.4094
NLMS	1/x(n)*x(n)	2.4

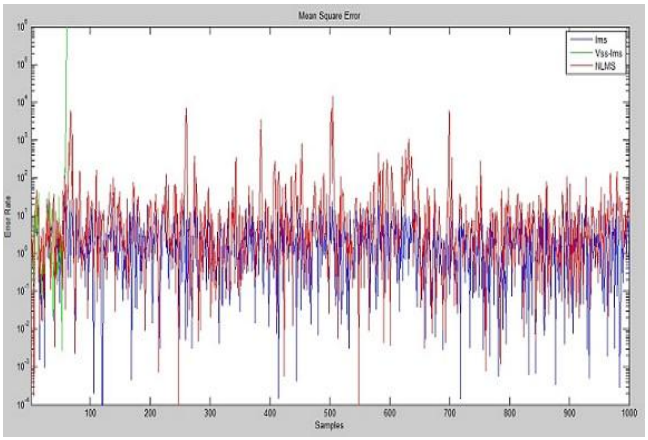

Fig -7: Comparison of LMS, VSSLMS AND NLMS

3. DISTORTION ANALYSIS

Normalized least mean square algorithm generates less mean square error rate when compared to the variable step size least mean square algorithm and least mean square algorithm. The NLMS algorithm, an equally simple, but more robust variant of the LMS algorithm, exhibits a better balance between simplicity and performance than the LMS algorithm. Finally we carried out hardware implementation of NLMS and the design was initially verified with Modelsim simulator tool and we successfully synthesize the Verilog HDL code with QUARTUS II EDA tool. Due to its good characteristics the NLMS has been largely used in real-time applications.

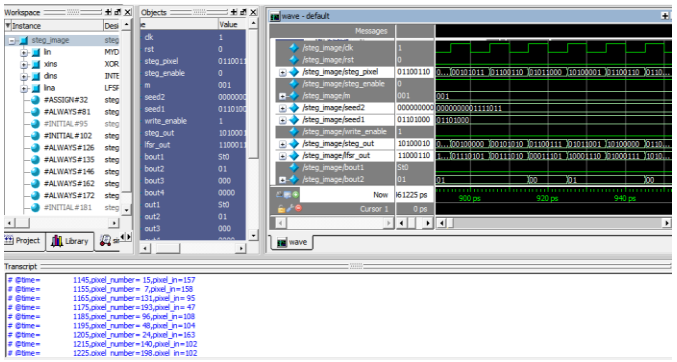

Fig -8: Simulated output

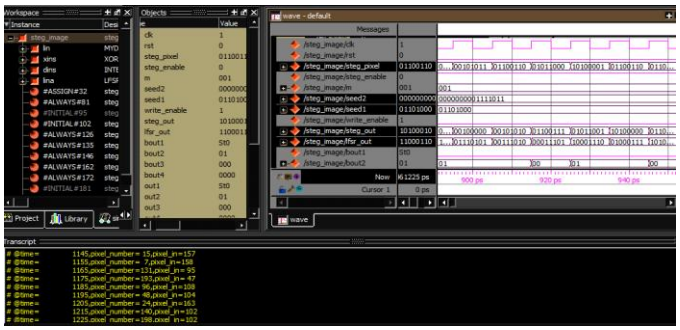

Fig -9: Area utilization Graph

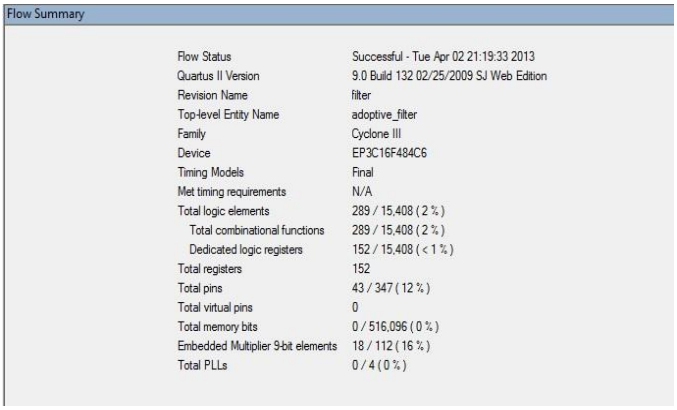

Fig -10: Area Utilization Report

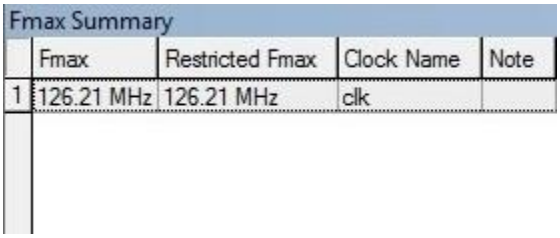

Fig-11: Frequency of Operation for Optimal step size NLMS algorithm

Table -2: Comparison of LMS and VSS- LMS with CYCLONE III family EP3C16F484C6

Filter type	Area utilization report (LE's used)	Frequency of operation (MHz)
LMS	496	72.16
Optimal NLMS	289	126.21

4. FPGA Implementation:

Implementation of Optimal step size NLMS algorithm is done in Altera Cyclone-IV target hardware. An audio signal, which is a real –time input is being processed along with the addition of noise and filtered output is retrieved back after Updation of weights. The multiplier used is reduced to one so the architecture presented consumes minimum hardware. This is convenient for this FPGA implementation. The audio signal which is retrieved back with low error rate is faster and the speed of operation increases more. The target hardware consumes less logic occupation which is considered as the most efficient design.

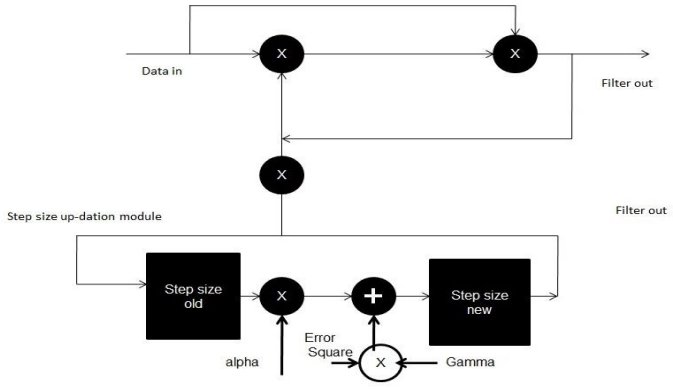

Fig-12: Block diagram of optimal step size NLMS Algorithm and its performance analysis

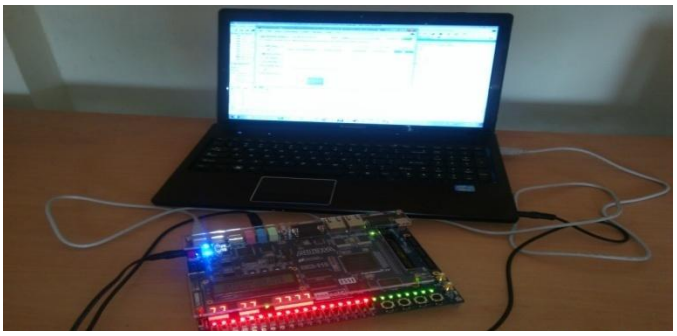

Fig-13: FPGA Implementation of audio signals using Cyclone 4 Hardware.

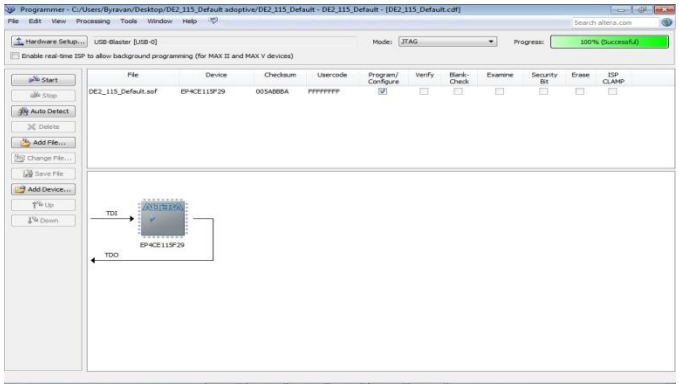

Fig-14: FPGA Optimal step size output

CONCLUSIONS

In comparison with existing LMS algorithms, the proposed algorithm has demonstrated consistently superior performance both in convergence and for final error level in application on both simulated data. The simulation results show that the proposed algorithm as superior performance in Fast convergence rate, low error rate, and has superior performance in noise cancellation. But in Implementation proposed system need additional units to update step size values. This will also lead some delay. So here we achieve better adoptive filtering by sacrificing power and area efficiency.

REFERENCES:

[1] Hsu-Chang Huang and Junghsi Lee, "A New Variable Step-Size Algorithm and its Performance analysis", IEEE Trans. Signal Process., vol. 60, no. 4, pp. 2055-2060, April 2012.

[2] R. H. Kwong and E. W. Johnston, "A Variable step size LMS algorithm", IEEE Trans. Signal Process., vol. 40, no. 7, pp. 1633-1642, Jul. 1992.

[3] G. A. Clark, S. K. Mitra and S. Parker ''Block implementation of Adaptive Digital Filters'', IEEE transaction on Acoustics, speech and signal processing, 1981.

[4] T. Yoshida, Y. Liguini, H.maeda ''Systolic array implementation of block LMS Algorithm'', Proc of IEEE Electronic letters 1998.

[5] Tian Lan, Jinlin Zhang, ''FPGA Implementation of Adaptive Noise Canceller'' Proc of 2008 International symposiums on information processing.

[6] C. Paleologu, J. Benesty, S. L. Grant and C. Osterwise, "Variable step-size NLMS algorithm designed for echo cancellation," in Proc. Asilomar, 2009, pp. 633-637

[7] Maurice G. Bellanger ''An FPGA Implementation of LMS adaptive filters for audio processing'', New York,Dekker,1987.

[8] Simon Haykin, Adaptive Filter theory. Pearson, 4th Edition

ACKNOWLEDGEMENTS

I sincerely acknowledge in all earnestness, the patronage provided by our Director Dr. C. Muthamizhchelvan, Engineering & Technology, to endeavour this project and I wish to express my deep sense of gratitude and sincere thanks to our Professor and Head of the Department Dr. S. Malarvizhi, for her encouragement, timely help and advice offered to me. I am very grateful to my guide Mrs. P. Radhika (Assistant Professor), who has guided with inspiring dedication, untiring efforts and tremendous enthusiasm in making this project successful and presentable and I extend my gratitude and heart full thanks to all the staff and non-teaching staff of Electronics and Communication Department and to my parents and friends, who extended their kind co-operation by means of valuable suggestions and timely help during the course of this project work.

BIOGRAPHIES:

L. Bharani was born in Chennai, India, in 1990. He received B.E from Sri Venkateswara college of College of Engineering, Sriperambadur, in 2011 and doing M.Tech in VLSI Design from SRM University, Chennai, India. His research interests, include Signal Processing, Digital Design.

P. Radhika received the B.E degree in electronics and communication engineering from Periyar Maniammai college of engineering, Thanjore in 2002, the M.Tech degree in VLSI Design from Sastra University, Thanjore in 2004.She is currently an Assistant Professor in the Department of ECE, SRM University, India. She is currently doing research in the area of Adaptive Filter Design. Her teaching and research interests include statistical signal processing, VLSI design.