Architecture design for Adaptive Noise Cancellation

M.RADHIKA, O.UMA MAHESHWARI, Dr.J.RAJA PAUL PERINBAM
Department of Electronics and Communication Engineering
Anna University
College of Engineering, Guindy, Chennai 600 025
INDIA

Abstract:- Adaptive noise cancellation is a technique of estimating the signals corrupted by additive noise. In this paper the Convergence of filter tap coefficients, Mean Squared Error and SNR of different signals are compared for NLMS and AFA algorithms using MATLAB. Architecture design for AFA algorithm is described and VHDL simulation is carried out. The inputs from the MATLAB are converted into 18 bit floating point representation. Floating point multiplier and adder are the components used in VHDL simulation and the results are verified by plotting the output using MATLAB.

Keywords: - AFA, NLMS, ANC, RLS, VHDL.

1 Introduction

Separating a desired signal from background noise is an important and common problem in signal processing. The change in signal characteristics is generally fast. This requires the utilization of adaptive algorithms, which converge rapidly. It is well known that two of most frequently applied algorithms for noise cancellation are Normalized Least Mean Squares (NLMS) and Recursive Least Squares (RLS) algorithms. Considering the two algorithms, it is obvious that NLMS algorithm has the advantage of low computational complexity. On the contrary, the high computational complexity is the weakest point of RLS algorithm but it provides a fast adaptation rate. Thus, it is clear that there is always a tradeoff between computational complexity and fast convergence. RLS has high computational complexity and stability problems so Adaptive Filtering with Averaging (AFA) algorithm is used for noise cancellation which has high convergence rate comparable to that of the RLS algorithm and at the same time low computational complexity. Simulation carried out with different noise and signal reveal its robustness maintaining fast convergence and at the same time keeping the computational complexity at a low level.

Section 2 deals with the Adaptive Noise Cancellation (ANC) and the adaptive algorithms. Section 3 deals with the MATLAB simulation section 4 deals VHDL simulation, Architecture design for AFA algorithm, floating point multiplier and adder. Section 5 deals with conclusion.

2 Adaptive Noise Cancellation

Adaptive Noise Cancellation removes the background noise from useful signals. Usually the background noise does not keep steady and it will change from time to time, hence adaptive filter is chosen which will adjust itself according to the changing environment. Noise cancellation can be applied to remove the noise in the speech signals especially in the jet or to cancel out the interference present in the signals, etc. The basic idea of the Adaptive Noise Cancellation is to pass the corrupted signal through a filter that tends to suppress the noise while leaving the signal unchanged it does not require prior knowledge of the signal

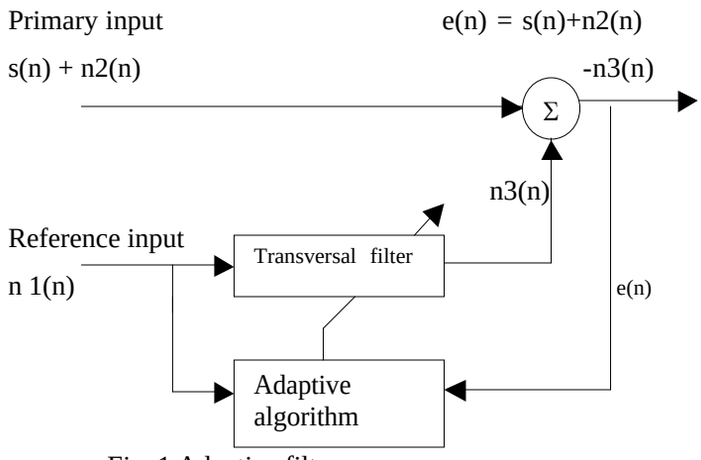


Fig. 1 Adaptive filter

Fig. 1 shows the classical scheme for adaptive noise cancellation using digital filter with finite impulse response (FIR). The primary input consists of signal s(n) and noise n2(n) while the reference input consists of noise n1(n) alone. The two noises n1(n)

and n2(n) are correlated to each other. The system tries to reduce the impact of the noise in the primary input by exploring the correlation between the two noise signals. This is equivalent to the minimization of the mean-square error $E[e^2(n)]$ where

$$e(n) = s(n) + n2(n) - n3(n)$$

Having in mind that by assumption, s(n) is correlated neither with n2(n) nor with n1(n) we have $E[e^2(n)] = E[s2(n)] + E[n2(n) - n3(n)]$. In other words the minimization of $E[e^2(n)]$ is equivalent to the minimization of the difference between n2(n) and n3(n). Obviously $E[e^2(n)]$ will be minimal when $n3(n) \approx n2(n)$ i.e. when the impulse response of the adaptive filter closely mimics the impulse response of the noise path. The minimization of $E[e^2(n)]$ can be achieved by updating the filter taps wi(n).

The weight adjustment is directly proportional to the tap input vector therefore, when the input is large, the LMS filter suffers from a gradient noise amplification problem. To overcome this difficulty we use the normalized LMS filter. The adjustment applied to the tap weight vector at iteration n+1 is normalized with respect to the squared Euclidean norm of the tap input vector at iteration n – hence the term normalized. Wherever fast convergence is vital, NLMS algorithm is not applicable, the more complex RLS algorithm maintains a good rate of adaptation but the prize to be paid is an order of magnitude increase in complexity, so in this section a proposed adaptive algorithm applied for noise cancellation based on AFA is discussed.

To recursively adjust the filter coefficients, so that the error is minimized, the algorithm for approximating the vector of filter coefficient is obtained by taking the averages of filter tap coefficients W. In order to increase the stability, the weights are updated not only through the approximation sequence but also through the observed signals N1 and e(n)

3 MATLAB Simulation

In this section we compare the performance of the NLMS and AFA algorithms. The algorithms are implemented according to the steps. Convergence of weights, SNR and mean square error are compared. We use sine chirps and a voice recording for our input signals. The results shows that the noise present in the signal is eliminated and error reduces gradually as the signal converges.

Algorithm for NLMS and AFA algorithm is given in the following section.

Primary input : $s(n)+ n2(n)$
Reference input : $n1(n)$

Noise estimation : $n3(n) = \sum_{i=0}^{N-1} w_i(n) * n1(n-i)$

Error estimation : $e(n)= s(n) + n2(n) - n3(n)$

Coefficient update:

NLMS:

$$w_i(n+1) = w_i(n) + \mu \frac{e(n)*n1(n-i)}{\sum_{i=0}^{N-1} n1^2(n-i)}$$

AFA:

$$W(n) = \frac{1}{n} \sum_{k=1}^{n} W(k)$$

$$n1e_i(n) = \sum_{k=1}^{n} n1(k-i) * e(k)$$

$$W_i(n+1) = W_i(n) + \frac{1}{n} \sum_{k=1}^{n} n1e_i(n)$$

N-filter order, n-number of samples.

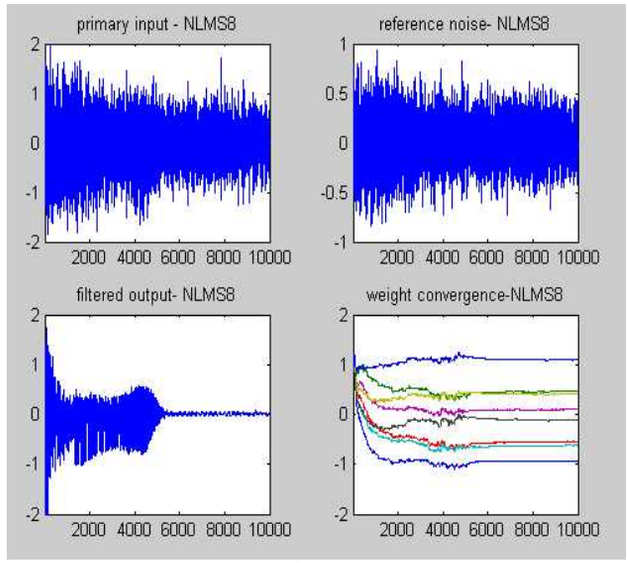


Fig. 2 NLMS order 8 - μ =0.02

Figure 2 shows that, the noise present in the word "return" is eliminated using NLMS algorithm of order 8. The convergence of weight vectors shows that the convergence rate of NLMS algorithm is slow and the mean square error is high initially and it reduces to zero.

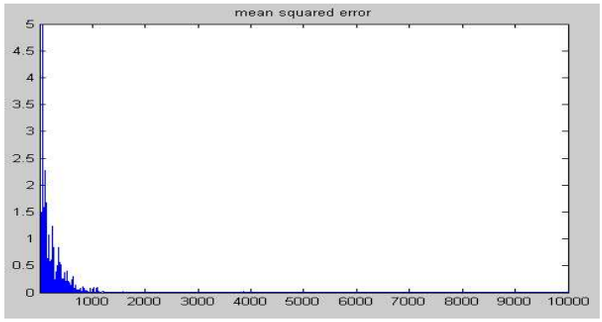


Fig. 3 Mean Squared Error.

We will use chirp signals of shorter duration than our voice signal to ease in the processing requirements

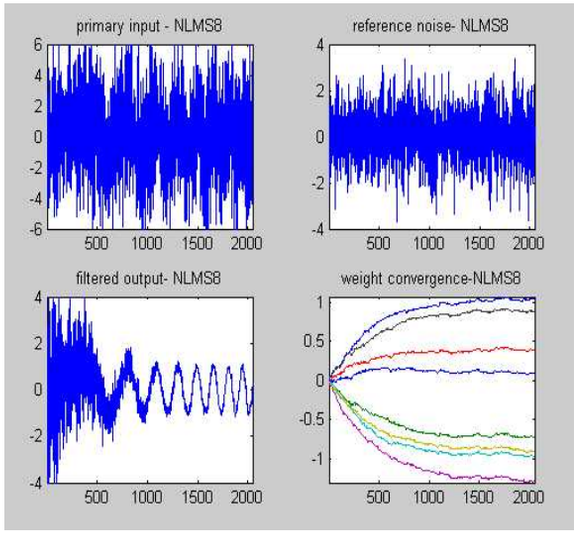


Fig. 3 NLMS – order 8 μ =0.02

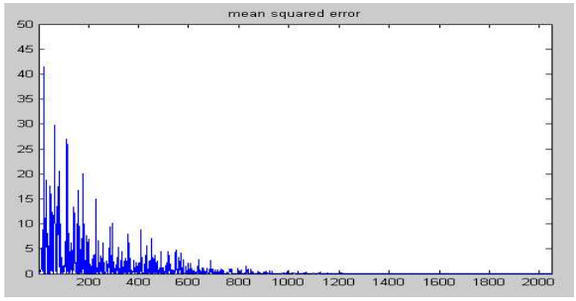


Fig. 4 Mean Squared Error

As the number of iterations reduces, the algorithm takes more time to converge and hence MSE is more before converging.

3.2 AFA Algorithm:

Figure 5 shows the simulation of AFA algorithm for order8. The results shows that the convergence of the filter tap coefficients is fast and the mean squared error reduces to a least value with in less number of iterations

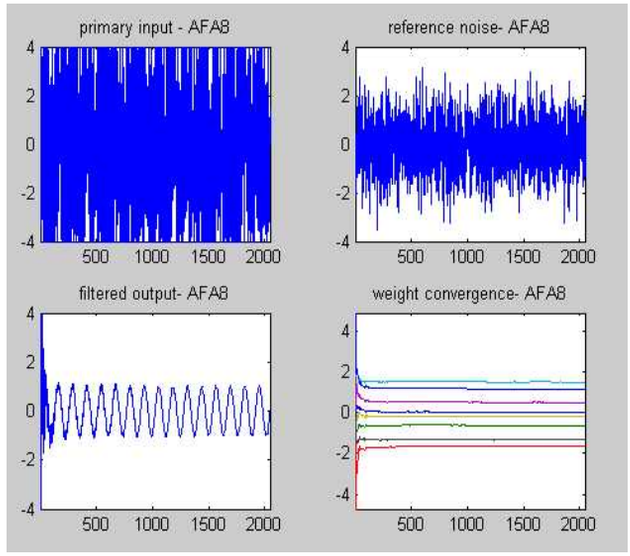


Fig.5 AFA algorithm order 8

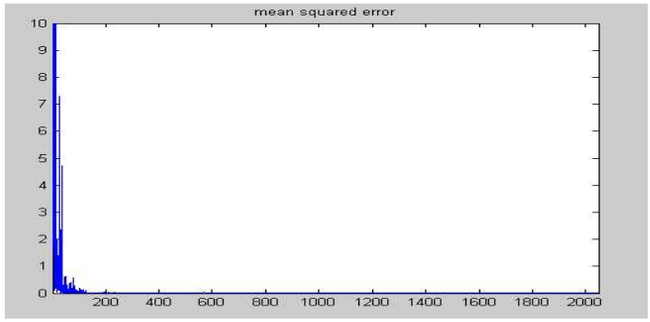


Fig.6 Mean Squared Error. Order8

Comparison of Signal to Noise Ratio is carried out and the MATLAB results are tabulated

Algorithm	Initial SNR	Final SNR
NLMS order2	-4.15	4.7
NLMS order8	-3.39	7.07
AFA order 2	- 4.5	7.1
AFA order8	- 6.22	11.89

Table : 1 Signal to Noise Ratio

There is a better improvement in signal to noise ratio as the number of taps increases.

4 VHDL Simulation

To carryout the simulation in VHDL the equations or the operations of AFA algorithm are taken as a state and the execution of some state will be enabled after the execution of the prior states.

4.1 Architecture for AFA Algorithm.

Figure 7 shows the architecture design for AFA algorithm. The operations performed are multiplication, addition and division. To have a good dynamic range, floating point arithmetic's are used.

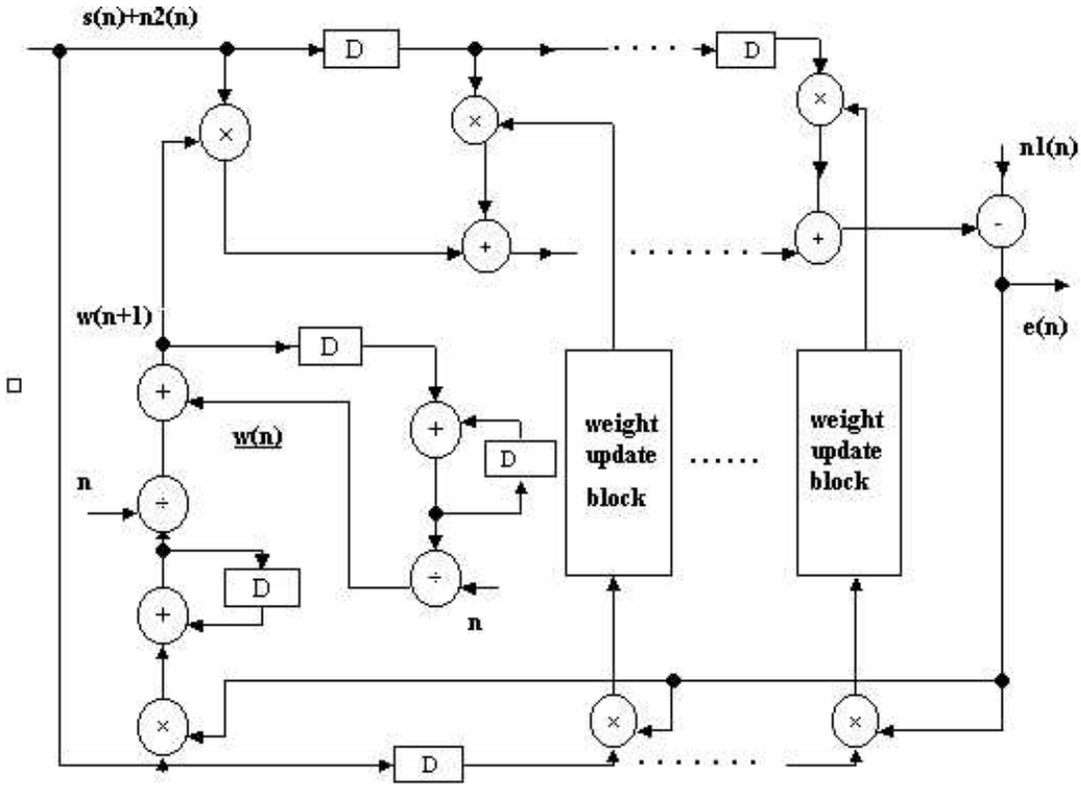
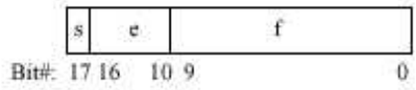


Figure 7: Architecture for Adaptive Filtering with Averaging Algorithm

Among the available 16,18,32[2] bit floating point representations 18 bit representation is used

18 bit IEEE 754 format shown in Figure



the 18 bit floating point value(v) is computed by

$$v = -1^{s}2^{(e-63)}(1.f)$$

the sign field, s, is bit 17 and is used to specify the sign of the number. Bits 16 down to 10 are the exponent field. This 7 bit quantity is a signed number represented by using a bias of 63. Bits 9 down to 0 are used to store the binary representation of the floating point number. The range of real numbers that this format can represent is

$\pm$ 3.6875 x 10^{19} to $\pm$ 1.626 x 10^{-19}

4.2 Floating Point Addition:

Floating point addition is performed in the following steps

1. If the absolute value of v1 is less than the absolute value of v2 then swap v1 and v2.The absolute value is checked by comparing the exponent and mantissa of each value.

2 Subtract e2 from e1 in order to calculate the number of positions to shift f2 to the right so that the decimal points are aligned before addition or subtraction in the next part.

3. Shift 1.f2 to the right (e2 - e1) places calculated in the previous stage. Add 1.f1 to 1.f2 if s1 equals s2, else subtract 1.f2 from 1.f1. Set the sign and the exponent of the final result, v3, to the sign and the exponent of the greater value v1.
4. Normalization of f3 is done by shifting it to the left until the high order bit is a one. Adjusting exponent of the result, e3 , is done by subtracting it by the number of positions that f3 was shifted left.

4.3 Floating Point Multiplication:

For floating point multiplication,

1. The exponents, e1 and e2 are added and the result along with the carry bit is stored in an 8-bit register. If the addition of two negative exponents results in a value smaller than the minimum exponent that can be represented, i.e. -63, underflow occurs. In this case the floating point number is set to zero. If overflow occurs, the result is set to the maximum number the format can represent.
2. If the floating point number is not zero, the implied one is concatenated to the left side of the f1 and f2 terms. The sign bits are only registered in this stage.

3. Integer multiplication of the two 11-bit quantities, 1.f2 and 1.f1, is performed. The top 12 bits of the 22-bit result is stored in a register. The exponent is adjusted depending on the high order bit of the multiplication result. The sign bits of the two operands are compared. If they are equal to each other the result is assigned a positive sign, and if they differ the result is negative.
4. Normalization of the resulting mantissa is performed. The resulting sign, exponent and mantissa fields placed into an 18-bit floating point word.

For division, reciprocal is got from the file and it is being multiplied. So the floating point multiplier is used to perform division. VHDL simulation for AFA algorithm of order 2 for 500 samples is carried out. The primary and reference input from the MATLAB is converted to 18 bit floating point representation using MATALB code and stored in separate files and is given as input for VHDL, the output file which has floating point representations is converted to real values using MATLAB and the output is verified by plotting.

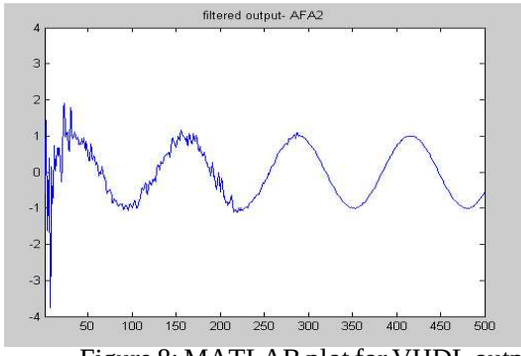


Figure 8: MATLAB plot for VHDL output

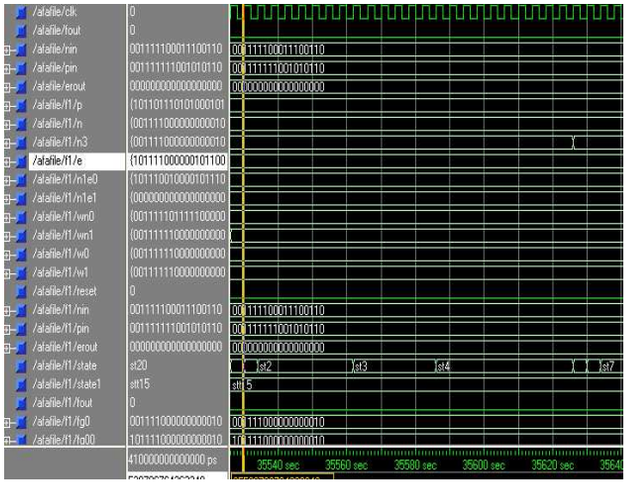


Figure : 9 Modelsim timing diagram

5 Conclusion

In this paper the convergence of filter tap coefficients, mean squared error and signal to noise ratio of NLMS and AFA algorithms are compared. From the experimental results it is very promising that AFA algorithm has high adaptation rate and low computational complexity compared to NLMS algorithm. Architecture for adaptive noise cancellation using AFA algorithm is designed and the VHDL simulation is carried out using 18 bit floating point multiplier and adder. The VHDL results are verified by plotting the output using MATLAB.

References:

[1] Georgi Iliev and Nikola Kasabov,"Adaptive Filtering with Averaging in Noise Cancellation for Voice and Speech Recognition," IEEE Trans. Acoust., Speech, Signal Processing.
Jan. 2002.

[2] Nabeel Shirazi, Al Walters, and Peter Athanas, "Quantitative Analysis of Floating Point Arithmetic on FPGA Based Custom Computing Machines," Presented at the IEEE Symposium on FPGAs for Custom Computing Machines, Napa Valley, California, April 1995.

[3] W. Harrison, J. Lim, E. Singer, "A new application of adaptive noise cancellation," IEEE Trans. Acoust., Speech, Signal Processing, vol. 34, pp. 21-27, Jan. 1986.

[4] S. Ikeda, A. Sugiyama, "An adaptive noise canceller with low signal distortion for speech codecs," IEEE Trans. Signal Processing, vol. 47, pp. 665-674, Mar. 1999.

[5] S. Haykin, Adaptive Filter Theory. Englewood Cliffs, NJ: Prentice-Hall, 1996.