Journal of Electronics, Communication & Instrumentation Engineering Research (JECIER)
ISSN 2278-3598
Vol.2, Issue 3 Dec 2012 1-18

HARDWARE IMPLEMENTATION OF FX-T RECURSIVE LEAST MEAN SQUARE ALGORITHM FOR EFFECTIVE NOISE CANCELLATION USING TMS 320C5X PROCESSOR

[1]J. JEBASTINE & [2]B. SHEELA RANI

[1]Research Scholar, Sathyabama University, Tamil Nadu ,Chennai-600119

[2]Vice-Chancellor & Dean, PG Studies and Research, Sathyabama University, Tamil Nadu ,Chennai-600119

ABSTRACT

This paper proposes the model for Recursive Least Square (RLS) algorithm and Fast Transversal Recursive Least Square (FxT-RLS) Algorithm for effective noise cancellation in acoustics and speech signal processing. The designed model gives the results in a noise free signal as output for RLS and FxT-RLS Algorithm. The filter used here is adaptive filter and the algorithm used is Recursive Least Square algorithm and Fast Transversal Recursive Least Square Algorithm. A white Gaussian noise is given as a input to the block and the other given input is the original speech/voice signal. By varying the adaptive step size, signal to noise ratio are determined and compared for both the algorithms. Based on these results the optimum step size is found for noise free outputs and the best efficient algorithm is found. The FxT-RLS algorithm provides similar performance to the standard RLS algorithm while reducing the computational effort. This is accomplished by a combination of four transversal filters used in union. The FxT-RLS algorithm is found to be a suitable solution for adaptive filtering applications and hence chosen to implement in hardware. The C Code of FxT-RLS is written in Code Composer studio. The .out file from the Code composer studio is converted to .asc format and downloaded into the TMS320C5X processor. Thus hardware has been implemented for effective removal of noise in audio and speech processing and it can be widely used in Mobile and Radio communication.

KEYWORDS: RLS, FxT-RLS, Simulink Model, TMS320C5X Processor, Noise cancellation

INTRODUCTION

An adaptive filter is a filter that self-adjusts its transfer function according to an optimization algorithm driven by an error signal. The characteristics of adaptive filter are a) automatically adjust (adapt) in changing system , b) can perform specific filtering or decision making, c) have adaption algorithm for adjusting parameters. The general block diagram for adaptive filter is shown in Fig. (1).The adaptive algorithm determines filter characteristics by adjusting filter coefficients (or tap weights) according to the signal conditions and performance criteria (or quality assessment). A typical performance criterion is based on an error signal, which is the difference between the filter output signal and a given reference (or desired) signal. Adaptive filtering finds practical applications in many diverse fields Such as communications, radar, sonar, control, navigation, seismology, biomédical engineering and even in financial engineering. An adaptive filter is a self-designing and time-varying system that uses a recursive algorithm continuously to adjust its tap weights for operation in an unknown environment. The digital filter used here is FIR filter shown in Fig. (2) which can be realized using many different forms. The structure used here is transversal for designing M-Tap adaptive filter.

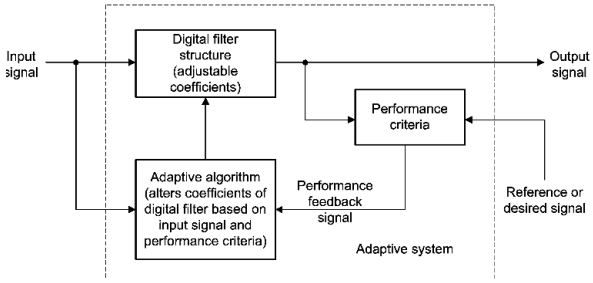


Fig. 1: Basic Functional Blocks of an Adaptive Filter

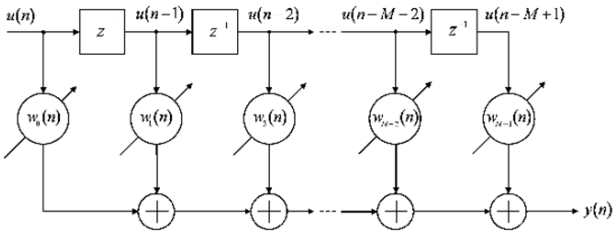


Fig. 2: An M-Tap Adaptive Transversal FIR Filter

The adjustable tap weights, wm (n), m = 0, 1,...,M − 1, indicated by circles with arrows through them, are the filter tap weights at time instance n and M is the filter length. These time-varying tap weights form an M × 1 weight vector expressed as

$$w(n) = [w_0(n), w_1(n), ..., w_{M-1}(n)]^T \quad (1)$$

where the superscript T denotes the transpose operation of the matrix. Similarly, the input signal samples, u(n − m), m = 0, 1,...,M − 1, form an M × 1 input vector

$$u(n) = [u(n), u(n-1), ..., u(n-M+1)]^T \quad (2)$$

With these vectors, the output signal y(n) of the adaptive FIR filter can be computed as the inner product of w(n) and u(n), expressed as

$$y(n) = \sum_{m=0}^{M-1} w_m(n)u(n-m) = w^T(n)u(n) \quad (3)$$

The error signal e(n) is the difference between the desired response d(n) and the filter response y(n), expressed as

$$e(n) = d(n) - w^T(n)u(n) \quad (4)$$

The weight vector w(n) is updated iteratively such that the error signal e(n) is minimized. A commonly used performance criterion (or cost function) is the minimization of the mean-square error (MSE), which is defined as the expectation of the squared error as

$$MSE = E\{e^2(n)\} \quad (5)$$

RLS ADAPTIVE ALGORITHM

The RLS adaptive filter is an algorithm which recursively finds the filter coefficients. It minimizes the weighted linear least squares cost function relating to the input signals. RLS algorithm reduces the mean square error in the

signal. In RLS, the input signals are considered deterministic.RLS algorithm is based on the fact that if the error has a) Zero mean, b) Statistically independent ,c) Gaussian distribution. The operation of RLS algorithm is explained in flow graph as shown in Fig.(3).

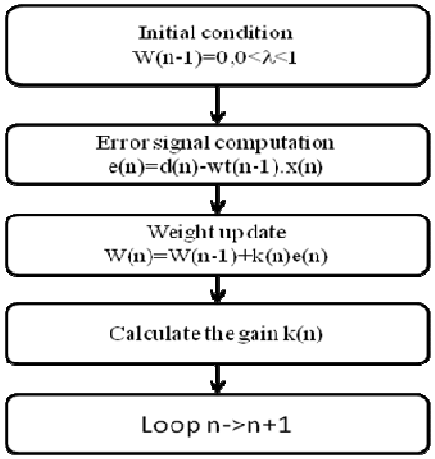


Fig. 3: Flow Graph of RLS Algorithm

FxT- RLS ADAPTIVE ALGORITHM

The Fast Transversal RLS (FxT-RLS) is engineered to deliver the solution to filtration issue with performance equivalent to the standard recursive least squares (RLS) method. Also, the FxT-RLS filter offers solution with reduced complexity that scales linearity with the filter order. This makes a very compelling solution for real-time noise reduction requests. Though FxT-RLS evolution is based on the derivation of the lattice-based least-squares filter, the structure of the four transverse filtering works together thereby computing update quantities reducing the computational complexity. The four transverse filters used for the formulation of updating the equations are:

a) Forward Prediction: The forward prediction transversal filter calculates the forward filter weights so as to reduce the prediction error (in the least mean square sense) for the next entry sample from the previous entry samples. This filter not only reduces the minimum weighted least squares error for forward prediction but also calculates the prediction error estimation of using both a priori and a posterior filter weights.

b) Backward Prediction: This filter not only calculates backward filter weights by minimizing the prediction error (in the least squares sense) of the u(n-m) sample using the vector input ub(n) but also computes the respective prediction error estimation using both a priori and a posterior filter weights, in addition to the minimum weighted least squares error for this backward prediction.

. c) Conversion Factor: The gain computation transversal filter is used to repeatedly calculate a gain vector used in updating the forward, backward and joint process estimation filter weights. This filter also provides recursive computation of the factors relating a priori and a posterior error quantities.

d) Joint-Process Estimation: The joint-process estimation transversal filter calculates filter weights such that the error between the estimated signal and the desired input signal d (n) is reduced. It is the joint process estimation weights that are equivalent to filter weights in other adaptive filtering algorithms. Figure (4) shows the block diagram of FT –RLS algorithm.

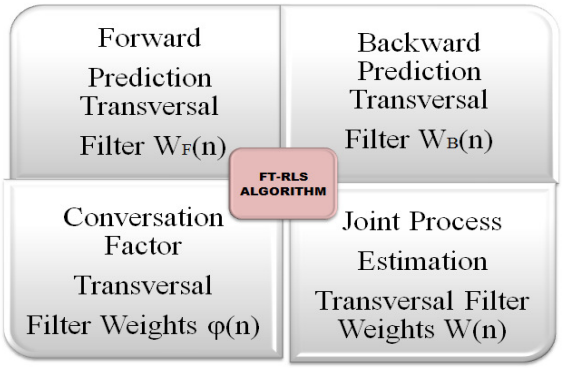


Fig. 4: Block Diagram of FxT-RLS Algorithm

However, the FxT-RLS algorithm brings numerical instability in a finite precision environment. The information about such instability is examined in detail. There were several ways proposed to establish a stable release of FxT-RLS. Commonly, these methods include expressing certain update equations in various ways. Though with high precision these quantities are same, they become unequal with finite precision. The variation in these values is commonly used as a measure of the numerical susceptibility. One workaround is to use a weighted average of the update equations. Using computation redundancy of some of those quantities provided numeric stability of the existence in finite precision. The result gives rise to a proposal for stabilization of FxT-RLS, that requires more computation per iteration of function, but is numerically robust compared to FxT-RLS and maintains the sequence. The recursive least-squares solutions need floating point operations for each iteration, where m is the size of the input vector ui.

$$u_j^T = [u_1(j)u_2(j),....,u_M(j)] \tag{6}$$

In reality, the inputs of uj are time-shifted versions of each other. More specifically, if we refer to the value of the first entry uj, then uj will have the form

$$u_j^T = [u_1(j)u_2(j-1),....,u_M(j-M-1)] \tag{7}$$

As shown in Fig.(5), the term z−1 represents a unit-time delay. The design block taking u(j) as an input and returns the inner product as an output is known as a transversal or FIR filter.

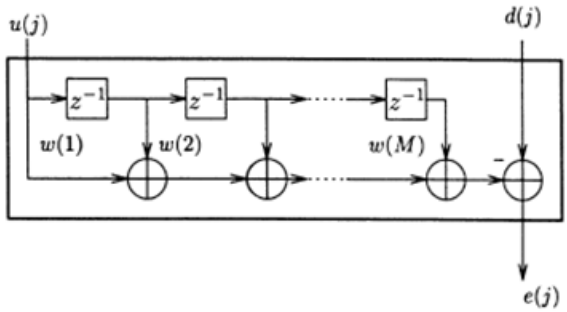


Fig.5: Pictorial Representation of Transversal FIR Filter

The shift structure of ui can be found in order to draw rapid variants to the RLS solution that would require O(M) operation for each iteration and not expected as O(M2). The M*M variables Pj needed in the RLS recursion exhibiting a certain matrix structure can be avoided by replacing the RLS recursions by an alternative set of recursions.

LOW-RANK PROPERTY

Let us assume π_0 in the special diagonal form

$$\pi_0 = \delta.diagonal[\lambda^2, \lambda^3, ...\lambda^{M+1}] \tag{8}$$

Where δ is a positive quantity (usually much larger than one, $(\delta << 1)$. In this case, we are led to a rank-two difference of the form

$$\begin{bmatrix} \lambda^{-1}\pi_0 & 0 \\ 0^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0^T \\ 0 & \pi_0 \end{bmatrix} = \delta.\lambda \begin{bmatrix} 1 & & \\ & 0 & \\ & & -\lambda^M \end{bmatrix} \tag{9}$$

This can be factored as

$$\begin{bmatrix} Q_0 & 0 \\ 0^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0^T \\ 0 & Q_{-1} \end{bmatrix} = \lambda M_0 P_0 M_0^T \tag{10}$$

Where $M_0 = (M+1) * 2$ and P_0 is a 2 * 2 signature matrix that are given by

$$M_0 = \sqrt{\delta} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & \lambda^{M/2} \end{bmatrix} and P_0 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{11}$$

FAST ARRAY ALGORITHM

As long as the three factors stated below is true, we can say by induction and time-shifting property of the input vectors ui , that if the low-rank property holds at a certain time instant i, that the following expression is provable.

$$\begin{bmatrix} Q_0 & 0 \\ 0^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0^T \\ 0 & Q_{-1} \end{bmatrix} = \lambda M_0 P_0 M_0^T \tag{12}$$

The three important facts hold:

The low-rank property also holds at time i+1, say

$$\begin{bmatrix} Q_0 & 0 \\ 0^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0^T \\ 0 & Q_{-1} \end{bmatrix} = \lambda M_{j+1} P_{j+1} M_{j+1}^T \tag{13}$$

There exists an array algorithm that updates Li to Li+1. Moreover, the algorithm also provides the gain vector gi that is needed to update the weight-vector estimate in the RLS solution. The signature matrices are equal. That is, all successive low-rank differences have the same signature matrix as the initial difference

$$P_j = P_0 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{14}$$

For M = 3, the pre array has the following generic form

$$A = \begin{bmatrix} x & x & x \\ 0 & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \tag{15}$$

Now let Θ_i be a matrix that satisfies

$$\Theta_j \begin{bmatrix} 1 & & \\ & 1 & \\ & & -1 \end{bmatrix} \Theta_j^T = \begin{bmatrix} 1 & & \\ & 1 & \\ & & -1 \end{bmatrix} = \begin{bmatrix} 1 & \\ & P_j \end{bmatrix} \tag{16}$$

and such that it transforms A into the form

$$B = \begin{bmatrix} x & x & x \\ 0 & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} = \begin{bmatrix} a & 0^T \\ b & c \end{bmatrix} \tag{17}$$

This means that Θ_i destroys two records in the top row of the pre-array. The problem is solved by first using a circular rotation that rotates using the left-most entry of the first row thereby destroying its second entry and then use a hyperbolic rotation that rotates again with the left-most entry and destroys the final entry of the top row. The unknown entries {a b C} can be found by using the same technique that we employed earlier during the derivation of the QR and inverse QR algorithms. By comparing entries on both sides of the equality

$$A \begin{bmatrix} 1 & \\ & P_j \end{bmatrix} A^T = \begin{bmatrix} a & 0^T \\ b & c \end{bmatrix} \begin{bmatrix} 1 & \\ & P_j \end{bmatrix} \begin{bmatrix} a & 0^T \\ b & c \end{bmatrix} \tag{18}$$

In short, we have obtained the validity of the array algorithm that effectively reduces the cost function in the pre windowed case and for the special choice of π0.It is also necessary to note that this fast procedure calculates the required gain vectors gi without explicitly evaluating the matrices Pi. Instead, the low-rank factors Li are propagated, which explains the lower computational requirements.

FxT- TRANSVERSAL FILTER

The fast algorithm of the previous section is an array version of fast RLS algorithms known as FTF (Fast Transversal Filter) and FAEST (Fast A posterior Error Sequential Technique). However, where the transformation that updates the data from time i to time i+1 is left implicit, the FTF and FAEST algorithms requires explicit sets of equations. The derivation of these explicit sets of equations can be stated as follows. Note that the factorization is highly not unique. The significance of this is that we have made S0 to be a signature matrix, i.e., a matrix with s on its diagonal. The fact that different factorization with an S0 is not restricted to be a signature matrix Different choices results in different sets of equations. More explicitly, assume we factor the difference matrix as

$$\begin{bmatrix} Q_0 & 0 \\ 0^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0^T \\ 0 & Q_{-1} \end{bmatrix} = \lambda M_0 P_0 M_0^T \tag{19}$$

where Li is an (M+1)*2 matrix and Mi is a 2*2 matrix that is not restricted to be a signature matrix. [We already know from the earlier array-based argument that this difference is always low-rank.] Given the factorization, it is easy to verify that two successive gain vectors satisfy the relation. This is identical to except that Si is replaced by Mi and Ni is replaced by Li. The fast array algorithm of the previous section provides one possibility for enforcing this relation and, hence, of updating gi to gi+1 via updates of Li . The FTF and FAEST algorithms follow by employing one such alternative factorization, where the two columns of the factor Li turn out to be related to the solution of two fundamental problems in adaptive filter theory: the so-called forward and backward prediction problems.

ORDERS-RECURSIVE FILTERS

The RLS algorithms discussed above are all fixed-order solutions that are capable of recursively evaluating successive weight estimates wi that correspond to a fixed order combiner of order. This way of calculating the minimum solution is not desirable from an order-recursive point of view. In other words, assume that we are facing a new optimization problem of the same form as before but where the vectors are now of order M+1 rather than M. The order-recursive filters of this section deal with this kind of minimization.Now let us suppose that our interest in solving is not to explicitly determine the weight estimate, but rather to determine estimates for the reference signals, say

$d_M(N) = u_{M,N}^T w_{M,N} = estimateofd(N)oforderM$

Likewise, for the higher-order problem,

$d_{M+1}(N) = u_{M+1,N}^T w_{M+1,N} = estimateofd(N)oforderM+1$ The resulting estimation errors will be denoted by

$$e_M(N) = d(N) - d_M(N) \quad (20)$$

$$e_{M+1}(N) = d(N) - d_{M+1}(N) \quad (21)$$

The consideration given below depends heavily on the orthogonality property of least-squares solutions and, therefore, serves as a good illustration of the power and significance of this property. It will further stimulate the introduction of the forward and backward prediction problems.

JOINT PROCESS ESTIMATION

Let as assume particular values for M and λ, say M = 3 and λ =1. These assumptions simplify the exposition without affecting the general conclusions. In particular, a no unity can always be incorporated into the discussion by properly normalizing the vectors involved in the derivation. The optimal solution is denoted by w3,N . The subscript N indicates that it is an estimate based on the data u(.) up to time N. Determining w3,N corresponds to determining the entries of a 3- dimensional weight vector so as to approximate the column vector dN by the linear combination A3,N,w3,N in the least-squares sense . We thus say that expression defines a third order estimator for the reference sequence. The resulting A posteriori estimation error vector is given by

$$e_{3,N} = d_N - A_{3,N} w_{3,N} \quad (22)$$

where, for example, the last entry of $e_{3,N}$ is given by

$$e_3(N) = d(N) - u_{3.N}^T w_{3,N} \quad (23)$$

It indicates the a posteriori estimation error in calculating d(N) from a linear combination of the three most recent inputs. According to the orthogonality property of least-squares solutions that the a posterior residual vector e3,N has to be orthogonal to the data matrix.

$$A_{3,N}^T e_{3,N} = 0 \quad (24)$$

We also know that the optimal solution $w_{3,N}$ provides an estimate vector that $A_{3,N}^T e_{3,N} = 0$ is the closest element in the column space of $A_{3,N}^T$ to the column vector d_N. Now assume that we wish to solve the next higher order problem, viz., of order M = 4: minimize over w4 the cost function

BACKWARD PREDICTION ERROR VECTORS

Let us continue to assume $\lambda = 1$ and M = 3. It was noticed that the desired decomposition is simply obtained by projecting the last column of A4,N onto the column space of its first three columns (i.e., onto the column space of A3,N) and keeping the residual vector as the desired vector m. This is same as Gram-Schmidt orthogonalization and it is equivalent to the following minimization problem. The special case replaces the sequence

$\{0, d(1),..., d(N)\}$ by the sequence

$$\{0,0,0,..., u(N-4), u(N-3)\} \tag{25}$$

It should be noted that the entries in every row of the data matrixA3,N are the three "future" values relevant to the entry in the last column of A4,N . Hence, the last element of the above linear combination serves as a backward prediction of u(N-3) in terms of {u(N),u(N-1,u(N-2))}. A similar observation holds for the other entries. The superscript b stands for backward. We thus say that a third-order backward prediction problem. The resulting a posteriori backward prediction error vector is denoted by

$$b_{3,N} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ . \\ . \\ . \\ u(N-4) \\ u(N-3) \end{bmatrix} - A_{3,N} w_{3,N}^b \tag{26}$$

The main reason for introducing the a posteriori backward residual vector b3,N was the major cause to solve the fourth-order problem thereby exploiting the solution of lower order, thus leading to an order-recursive algorithm. Let us assume now that the estimation error vectors e3,N and b3,N are available , which are both orthogonal to A3,N . Knowing that b3,N leads to an orthogonal decomposition of the column space of A4,N and then updating e3,N into a fourth-order a posteriori residual vector e4,N , which has to be orthogonal to A4,N , simply corresponds to projecting the vector e3,N onto the vector b3,N . This corresponds to finding a scalar coefficient k3 that solves the optimization problem. This is a standard least-squares problem and its optimal solution is denoted by

$$k_3(N) = \frac{1}{b_{3,N}^T b_{3,N}} b_{3,N}^T e_{3,N} \tag{27}$$

FORWARD PREDICTION ERROR VECTORS

Let us continue to assume $\lambda = 1$ and M = 3. The order-update of the backward residual vector helps us to introduce the forward prediction problem: minimize over w_3^f the cost function

$$\left\| \begin{bmatrix} u(1) \\ u(2) \\ u(3) \\ . \\ . \\ . \\ u(N+1) \end{bmatrix} - A_{3,N} \begin{bmatrix} w_3^f(1) \\ w_3^f(2) \\ w_3^f(3) \end{bmatrix} \right\|^2 \tag{28}$$

We denote the optimal solution by $\mathbf{W^f_{3,N+1}}$. The subscript indicates that it is an estimate based on the data u(.) up to time N + 1. Determining $\mathbf{W^f_{3,N+1}}$ corresponds to determining the entries of a 3-dimensional weight vector so as to approximate the column vector

$$\begin{bmatrix} u(1) \\ u(2) \\ u(3) \\ . \\ . \\ . \\ u(N+1) \end{bmatrix} \tag{29}$$

by a linear combination of the columns of A3,N . Since the entries of the successive rows of the data matrix A3,N are the past three inputs compared to the corresponding inputs of the column vector, this expression defines a third-order forward prediction problem. The resulting a posteriori forward prediction error vector is given by

$$f_{3,N} = \begin{bmatrix} u(1) \\ u(2) \\ u(3) \\ . \\ . \\ . \\ u(N+1) \end{bmatrix} - A_{3,N} w^f_{3,N+1} \tag{30}$$

Now assume that we wish to solve the next-higher order problem, viz., of order M = 4: minimize over w_4^f the cost function

$$\left\| \begin{bmatrix} u(1) \\ u(2) \\ u(3) \\ . \\ . \\ . \\ u(N+1) \end{bmatrix} - A_{4,N} \begin{bmatrix} w_4^f(1) \\ w_4^f(2) \\ w_4^f(3) \end{bmatrix} \right\|^2 \tag{31}$$

The change in the time index as we move from b3,N to b4,N+1 is because b4,N+1 is obtained by projecting b3,N onto f3,N+1, leading to the following expression for b4,N+1 ,

$$b_{4,N+1} = \begin{bmatrix} u(1) \\ u(2) \\ u(3) \\ . \\ . \\ . \\ u(N-4) \\ u(N-3) \end{bmatrix} - A_{4,N+1} \begin{bmatrix} w^b_{4,N+1}(1) \\ w^b_{4,N+1}(2) \\ w^b_{4,N+1}(3) \\ w^b_{4,N+1}(4) \end{bmatrix} \quad (32)$$

Finally, the joint process estimation problem involves a recursion of the form

$$e_{4,N} = e_{3,N} - k_3(N) b_{3,N} \quad (33)$$

Where

$$k_3(N) = \frac{b^T_{3,N} e_{3,N}}{b^T_{3,N} b_{3,N}} \quad (34)$$

SIMULINK MODELING

RLs Simulink Model for Speech Signal

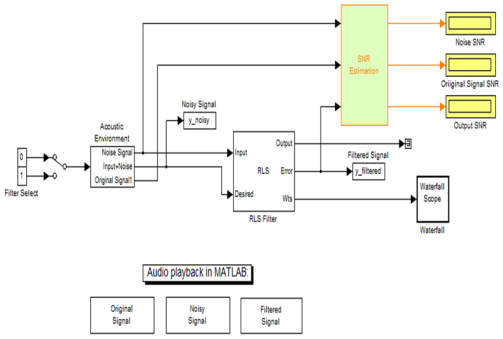


Fig. 6: RLS Simulink Model for Speech Signal

The Acoustic environment is develops the noise signal and the noise signal. The filter select feature allows choosing between the low pass and the band pass filter. The noise signal is given to the input of the RLS block. The noise added with the input signal is given to the desired block of the RLS filter. The RLS block removes the noise and gives the error free output. The SNR of the noise, original signal and output are found using SNR estimation.

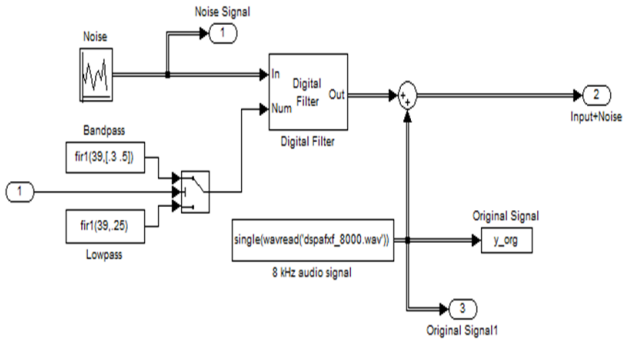


Fig.7: Acoustic Environment

The noise which is developed is uniform noise with a minimum value of -1 and a maximum value of 1.The initial seed is set to 23341. The samples are given discretely. The sample time is 1/8000 with 32 samples per frame. The band pass and low pass filter are given to the digital filter block at the num input. The noise signal is given to the In port of the digital filter. The original signal developed is 8 KHz signal at a sample time of 1/8000 and 32 samples per frame. This signal is added to the noise signal which comes out of the digital filter.

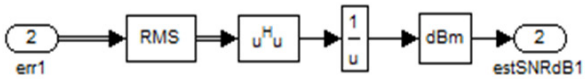


Fig.8: SNR Estimation for RLS Algorithm

The SNR is estimated by giving the desired signal to the RMS block. Then the matrix square value is found and the signal is given to the math block with reciprocal function. Then the power signal is given to the dB block which calculates the SNR of the desired signal.

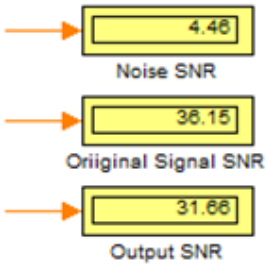


Fig. 9: SNR Output for RLS Algorithm

The SNR values of noise, original signal and the output are found. The output signal removes most of the noise in the signal.

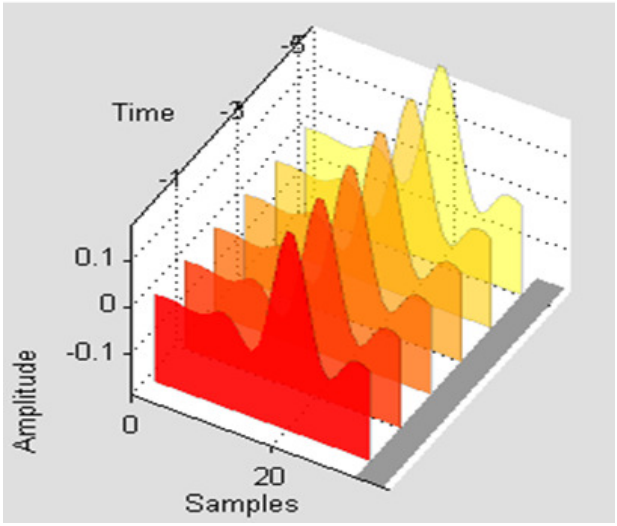


Fig.10: 3-D measurement of RLS Algorithm

The waterfall scope of the RLS is found which gives the three dimensional view. The axes are amplitude samples and time period. In the RLS Simulink model for speech signal, a sine wave is given as input. The noise given is Gaussian noise. Using an FDA tool, the noise is filtered and given to the RLS block. The noise and sine wave are added and given as input to the desired port. The scope of the noisy signal, original signal and the filtered signal are obtained. The frequency response is also found with FFT length of 256

FxT- RLS Model Using Audio Signal

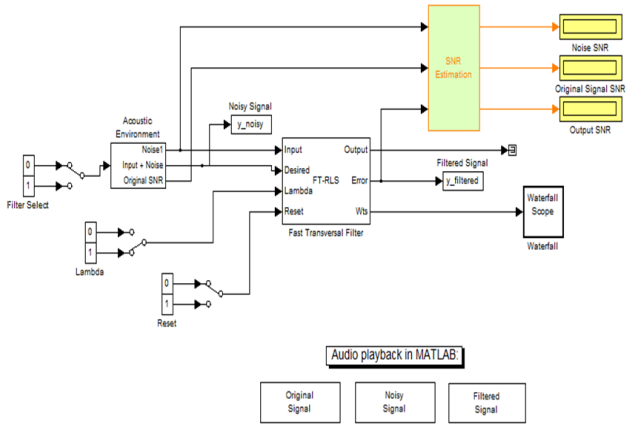


Fig. 11: FxT-RLS Simulink Model

In FxT-RLS the lambda value is set to one. The FxT-RLS filter removes the noise more efficiently than the RLS filter. The original signal, noisy signal and the filtered signal can be heard. The FxT-RLS filter removes the noise at a faster convergence rate than the RLS filter. The Fx- Transversal RLS provides SNR value higher than the RLS filter and is found to be more effective.

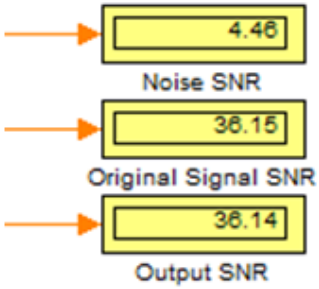


Fig. 12: FxT- RLS Output

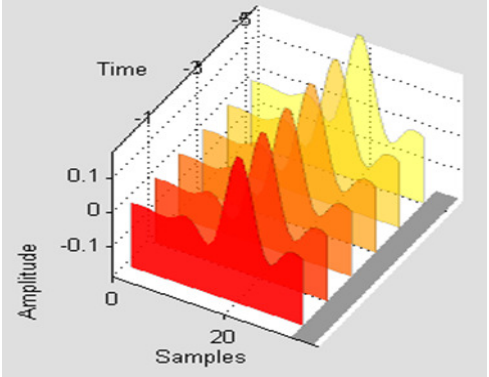


Fig. 13: 3-D Measurement of FxT-RLS Algorithm

The waterfall scope of FxT-RLS filter shows that it provides convergence at a faster rate than the RLS filter. In the Fast RLS the periodic interference and wideband signal are used. Here the added signal is given to the delay block and to the input port. The wideband signal is given as the input to the Err port. Using the flip, the filter taps and the user coefficients are given. The bit error is then calculated using the error calculation block.

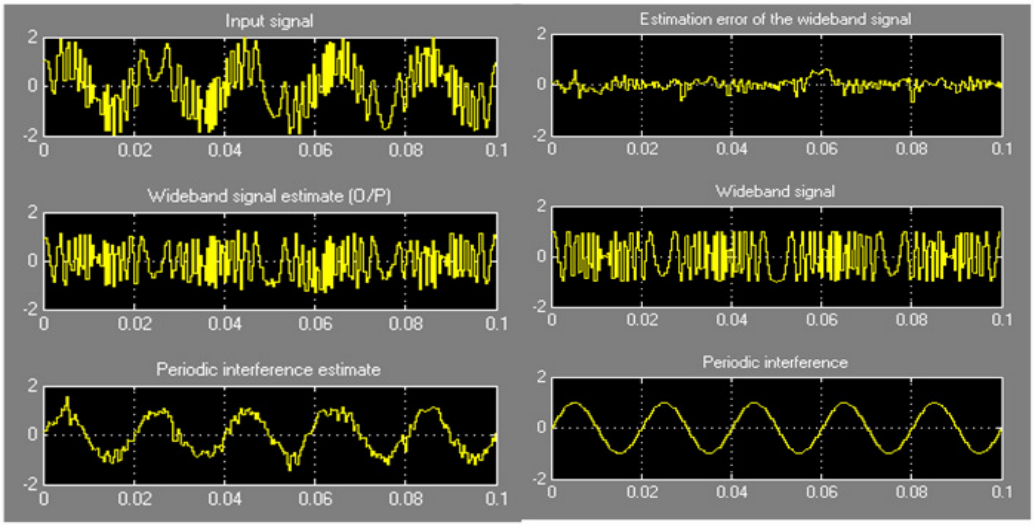


Figure 14: FxT- RLS Scope 1 and Scope 2

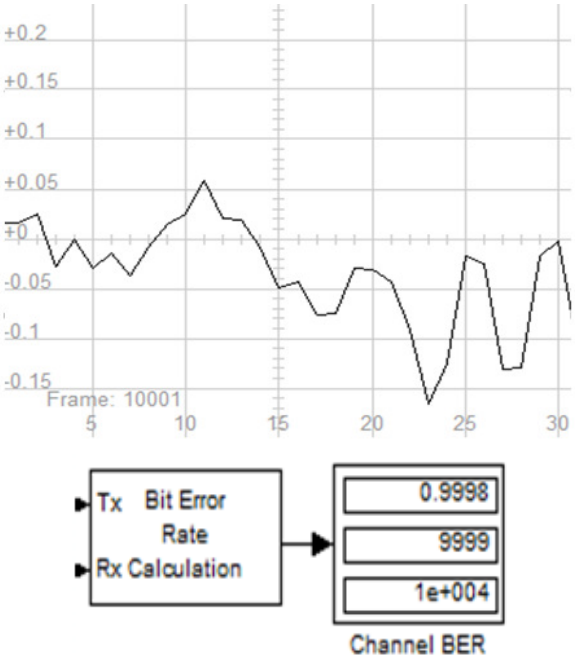


Figure 15: FxT- RLS Coefficients and Bit Error Rate

The scope of the Fast RLS filter has shown that the noise is removed from the periodic signal added. The coefficients for updating the algorithm are also found which has a maximum value less than 0.1. The bit error rate is calculated which removes the error efficiently.

COMPARISON OF RLS AND FxT-RLS ALGORITHM

The SNR output results of RLS and FxT-RLS shows that FxT-RLS is better than RLS. Also in the audio model, the noisy signal gets removed faster in FxT-RLS than RLS. This shows that FxT-RLS is a better choice than RLS in audio signal processing

Table 1: SNR Comparison of RLS and FxT-RLS Algorithm

PARAMETERS	LMS(SNR)	NLMS(SNR)	FAST-LMS(SNR)	RLS(SNR)	FT-RLS
White Gaussian Noise Signal	4.46dB	4.46dB	4.46dB	4.46dB	4.46dB
Original Speech Signal	36.15dB	36.15dB	36.15dB	36.15dB	36.15dB
Filtered Signal	30.15dB	31.66dB	36.12dB	31.66dB	36.14dB
Attenuation	-9.258dB	-13.18dB	-16.8dB	-14.234dB	-17.45dB
Mean Error Value	0.06258	0.0037	0.000052	0.00025	0.0000015
Time Delay Estimation	10.789mS	9.75mS	8.25mS	8.15mS	7.987mS

Table 2: Comparative Study various parameters of RLS and FxT-RLS Algorithm

PARAMETER	RLS	FT-RLS
ORDER	$O(M^2)$	O(M)
Computational Complexity	Higher computational burden	Reduced computational burden which scales linearly with filter order.
Updating coefficients	Slower	faster
Filter length	Small	large
Real time implementation	Difficult	Easier
Application	Wireless application such as beam forming, equalization and adaptive filtering	Noise and echo cancellation

From the above tabulation FxT-RLS is found to be a suitable algorithm to be used in noise cancellation application

HARDWARE IMPLEMENTATION OF FxT-RLS

The two signals that are generated in the processor are the sinusoidal wave input signal and the noise signal. The sinusoidal wave input signal is set at a frequency of 50Hz. The noise frequency is set at a frequency of 1000Hz. Here fixed point processing is used in the filter. The filter used is FIR filter which performs filtering by frequency sampling method. Each input sample is being sampled separately and the noise present at that particular instant is removed The FxT-RLS algorithm is generated by compiling the above files and the downloadable algorithm is developed using the TMS320C5X compiler. The FxT-RLS algorithm downloaded in the TMS320C5X processor using the top view debugger. The downloaded program is run and the output is observed in the CRO. Thus the noise in the total signal is removed and the

output obtained is 50Hz signal which is noise free. The sine wave output obtained from the DSO with 10ms. The hardware setup used for implementing the algorithm is shown below

The files needed to create the FxT-RLS downloadable file are

a) AC01 initialization

b)Begin program

c) C5x common initialization

d) DSK5A file

e)DSK5L file

f) DSK5D file

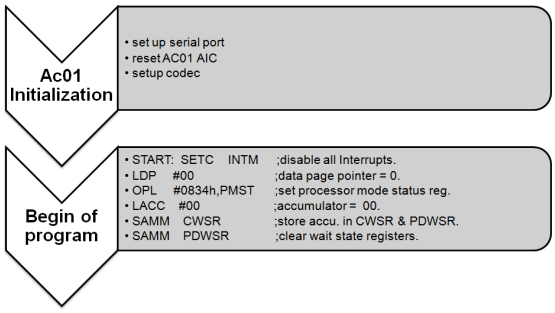
g) FT-RLS file

h) Sine table

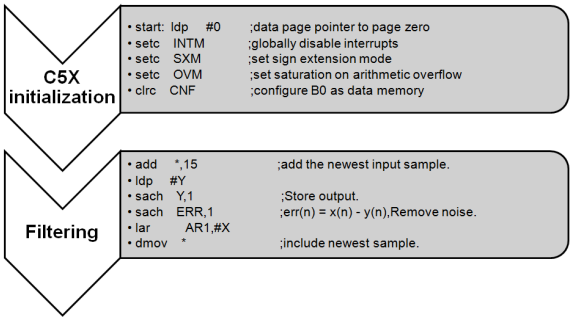
i) Wave generation

The AC01 contains the DAC and the ADC inside the chip itself and the program for converting the signals are written.The begin program contains the main program starting page from which it redirects to other files. The C5X initialization contains the initialization commands which are common for all the processors under the series. The DSK files contain the assembler; loader for creating the downloader files. The FxT-RLS file contains the programming for defining the filter and then updating the coefficients of the algorithm. The sine table contains the file for creating the table for fixed point in repeated manner. Steps to be followed for implementing the wave form using TMS 320C5X processor are as follows:

Step1:AC01 Initialization and Begin of program



Step2: C5X Initialization and Filtering



Step3: FxT-RLS Adaption of filter Coefficients

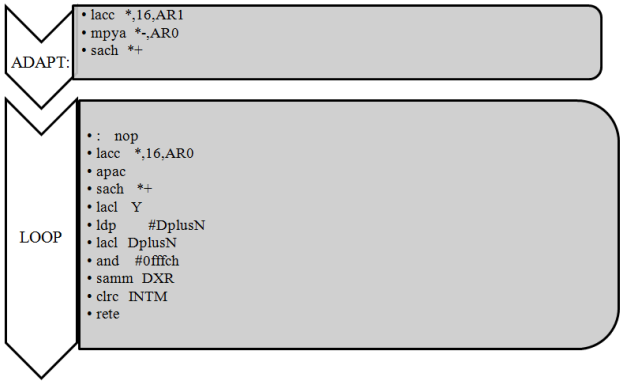


Figure 16: Flow Diagram for Implementing the FxT-RLS Algorithm in TMS 320c5X Processor

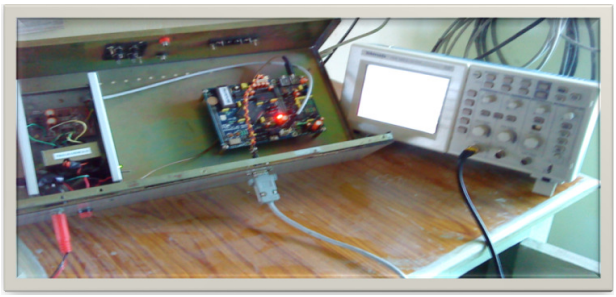


Fig.17: Hardware setup for FxT- RLS Algorithm

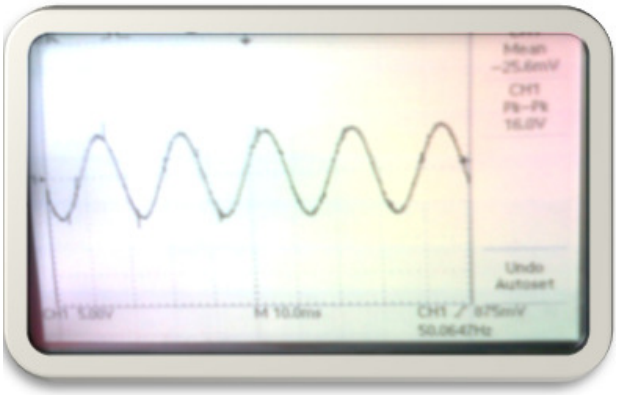


Fig. 18: Filtered output from hardware setup for FxT- RLS Algorithm

Table 3 Hardware Results for FxT-RLS Algorithm Using Speech Signal

S.No	Parameter	Frequency(Hz)	Time (msec)
1	Input signal (Sinusoidal)	50	20
2.	Noise Signal (White Gaussian)	1000	1
3.	Filtered output (original signal)	48.578	20

CONCLUSIONS

In this paper RLS and FxT-RLS algorithm are implemented in Matlab. The results from FxT-RLS and RLS have shown that FxT-RLS is better than RLS in SNR and removes the noise at a very fast convergence rate. The computational complexity of FxT-RLS is lower than the RLS .The noisy signal is removed faster in the FxT-RLS algorithm than the RLS algorithm as it updates the coefficient in a faster manner. Thus the FxT-RLS provides an improved output than the RLS algorithm by removing the noise more efficiently. So it is implemented in hardware of TMS320C5X processor and showed and it removed the noise efficiently. FxT-RLS algorithm is a highly suitable solution for adaptive filtering applications where a large filter order is required without sacrificing the performance offered by the standard RLS algorithm in the presence white Gaussian Noise. The Fast transversal algorithm can be made use in removing the noise that are developed in the telephone cables ,as they provide good noise removal at various frequencies. So real time implementation in telephone cables can be done.

REFERENCES

1. Dan J. Dechene (2007), 'Fast Transversal Recursive least-squares (FT- RLS) Algorithm', IEEE Transaction on signal processing
2. John M Cioffi, Fast (April 1984) 'Recursive – Least Squares Transversal Filters for Adaptive Filtering 'IEEE transaction on acoustics, speech and signal processing, vol. assp-32, no.2 .
3. Sanaullah khan, Arif and Majeed.T (2004) 'Comparison of LMS, RLS and Notch Based Adaptive Algorithms for Noise Cancellation of a Typical Industrial Workroom',8th International Mutitopic Conference, Page(s):169-173.
4. Slock D.T.M and Kailath.T (Jan 1991). "Numerically Stable Fast Transversal Filters for Recursive Least Squares Adaptive Filtering" IEEE Trans. on Signal Processing, vol. 39, pp. 92–113,
5. J.Jebastine, B.SheelaRani, July (2012),'Hardware Implementation of Fast Transversal LMS algorithm in Acoustics, Speech and Signal Processing (ASSP) using TMS 320C5X Processor", CiiT International Journal of Digital Signal Processing, Vol4, No7.
6. Wang Li Fang Chen Yi Ping (2009) "Implementation of Adaptive Filter Based on DSP Computer Simulation" , 26(9): 281-284.
7. Yaghoub Mollaei (Nov 2009) "Hardware Implementation of Adaptive Filter, Proceedings of 2009" IEEE Student Conference on Research and Development, UPM Serdang, Malaysia
8. B.Venkataramani & M. Bhaskar(2002), Digital Signal Processor Architecture, Programming and Application, TMH.
9. Avtar singh, S.Srinivasan (2003), DSP Implementation using DSP microprocessor with Examples from TMS32C54XX -Thamson / Brooks's cole Publishers.
10. D. T. M. Slock and T. Kailath, "Numerically Stable Fast Transversal Filters for Recursive Least Squares Adaptive Filtering," IEEE Trans. On Signal Processing, vol. 39, pp. 92–113, Jan 1991.

11. Mohammad Shukri Ahmad, Aykut Hocanın, Osman Kukrer,(2010), A Fast-Implemented Recursive Inverse Adaptive Filtering Algorithm, SIU2010 - IEEE 18.Sinyal isleme ve iletisim uygulamalari kurultayi Diyarbakir,Pg.No:945-948.

12. Jin Honghua, "Realization of adaptive filter based on TMS320VC5402", Journal of Northeast Agricultural University, , vol.37, pp.816 – 819..August 2007.

13. J.Jebastine ,B.SheelaRani ,June(2012) ,'Design and Implementation of Noise Free Audio Speech Signal Using Fast Block Least Mean Square Algorithm', 'Signal and Image Processing :International Journal(SIPIJ)'Volume 3, No.3.

14. Da-Zheng Feng and W. Xing, "Fast RLS Type Algorithm For Unbiased Equation Error Adaptive IIR Filtering Based On Approximate Inverse- Power Iteration", IEEE Trans. On Signal Processing, Vol. 53, No. 11, Nov. 2005.

15. Yuriy V. Zakharov, Member, IEEE, George P. White, and Jie Liu (July 2008), Low-Complexity RLS Algorithms Using Dichotomous Coordinate Descent Iterations, IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 56, NO. 7.

J.JEBASTINE graduated in Madras University, Chennai at 2002 in Electrical & Electronics discipline and pursues his Master's degree in Applied Electronics in Sathyabama University, Chennai and awarded at 2004. He is doing his research work in the field of Signal processing. Now he is an Associate Professor in the department of Electronics & Communication Engineering of Jeppiaar Engineering College, Chennai. He presented 10 papers in National and International Conferences.