International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682
Vol.4, No.5, May 2015 DOI:10.15693/ijaist/2015.v4i5.13-19

Asic Implementation Of Alms Adaptive Filter

M.Karthika[1], P.Elayaraja[2], K Deepa[3]

[1 & 3]PG Scholar, Kongunadu College of Engineering and Technology, Trichy, Tamil Nadu
[2]Assistant Professor, Kongunadu College of Engineering and Technology, Trichy, Tamil Nadu

Abstract:

Novel architecture to implement the pipeline fixed point adjoint LMS adaptive filter to reduce the delay and noise in a desired output signal. In this work we are implement in a real time input signal such us audio, cardiac etc., a input signal affected from external and internal noise sources, the unwanted noise signal can often be improved by an adaptive filter with adjoint least mean squares (ALMS) algorithm. The existing architecture has not achieved the sufficient output and reduces the performance for the continuous process and many more applications, when the desired signal exhibits large power fluctuations and delay. The adjoint least mean square algorithms provide the secondary paths estimates, secondary paths coefficients and states to achieve the minimum adaptation delay. The adaptive filter with adjoint least mean square (ALMS) algorithm implement in a real time input signal to improve the signal – to – noise ratio (SNR), Mean square error (MSE) and the response time in desired output signal. The proposed architecture implemented in both MATLAB and Cadence environment and obtain the minimum delay time 0.08239, Improve the SNR 12.6803 and MSE 9.9974e-004 in MATLAB and occupied cell area 0.284µm, total power consumed 160.77 mW and the minimum delay time is 0.17ns.

Index Terms— Adaptive filters, adaptive noise cancellation, noise reduction, fixed-point arithmetic, LMS, ALMS.

I. INTRODUCTION

Adaptive Noise Cancellation

Noise Cancellation is a variation of optimal filtering that involves producing an estimate of the noise by filtering the reference input and then subtracting this noise estimate from the primary input containing both signal and noise. It makes use of an auxiliary or reference input which contains a correlated estimate of the noise to be cancelled. As shown in the Fig. 1, an Adaptive Noise Canceller (ANC) has two inputs – primary and reference. The primary input receives a signal s from the signal source that is corrupted by the presence of noise n uncorrelated with the signal. The primary input receives a signal s from the signal source that is corrupted by the presence of noise n uncorrelated with the signal.

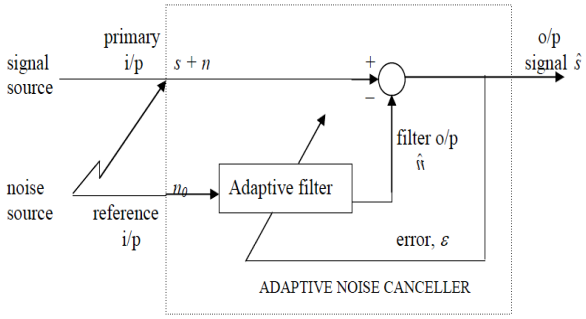

Fig.1 Adaptive Noise Canceller

The reference input receives a noise n0 uncorrelated with the signal but correlated in some way with the noise n. The noise no passes through a filter to produce an output nˆ that is a close estimate of primary input noise. This noise estimate is subtracted from the corrupted signal to produce an estimate of the signal at sˆ, the ANC system output.

Adaptive Filter Algorithms

Apply different algorithms to the FIR adaptive filter to control how the filter adjusts the coefficients. The adaptive algorithms adjust the filter coefficients to minimize the following cost function J(n):

$$J(n) = E[e^2(n)] \qquad (2)$$

where $E[e^2(n)]$ is the expectation of $e^2(n)$, and $e^2(n)$ is the square of the error signal at time n. Depending on how the adaptive filter algorithms calculate the cost function J(n), the Adaptive Filter Toolkit categorizes those algorithms into the following two groups:

- Least Mean Squares (LMS) algorithms.

LMS algorithms calculate J(n) by using the following equation:

$$J(n) = e2(n) \qquad (3)$$

This equation shows that the LMS algorithms use the instantaneous value of e2(n) at time n as the estimation of *E[e2(n)]*.

- Recursive Least Squares (RLS) algorithms

RLS algorithms calculate J(n) by using the following equation:

$$J(n) = \frac{1}{N}\sum_{i=0}^{N-1} \lambda^i e^2(n-i) \qquad (4)$$

Where, N is the filter length and λ is the forgetting factor. This algorithm calculates not only the instantaneous value $e^2(n)$ but also the past values, such as $e^2(n–1)$, $e^2(n–2)$, ..., $e2(n–N+1)$. The value range of the forgetting factor is (0, 1). In this paper Section 2 addressed the existing system delayed LMS algorithm and its pipelined architecture. Section 3 proposed adjoint LMS filter with architecture. Section 4 contains the results and shows the entire architecture implement in cadence design environment and finally Section 5 concludes this work.

II. DELAYED LMS ALGORITHM

The Least Mean Square (LMS) adaptive filter is the most popular and most widely used adaptive filter, not only because of its simplicity but also because of its satisfactory convergence performance. Since the conventional LMS algorithm does not support pipelined implementation because of its recursive behavior, it is modified to a form called the delayed LMS (DLMS) algorithm, which allows pipelined implementation of the filter.

The delayed LMS (DLMS) algorithm is described by the following equations:

$$y(n) = XT(n)W(n-1-D_1)$$

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

$$W_{n+1} = W_n + \mu.e_n.X_n \qquad (6)$$

The weights of LMS adaptive filter during the n th iteration are updated according to the following equations [2] :

$$W_{n+1} = W_n + \mu.e_n.X_n \qquad (7)$$

dn is the desired response, *yn* is the filter output, and en denotes the error computed during the n th iteration. μ is the step-size, and N is the number of weights used in the LMS adaptive filter. In the case of pipelined designs with m pipeline stages, the error en becomes available after m cycles, where m is called the "adaptation delay." The DLMS algorithm therefore uses the delayed error en −m, i.e., the error corresponding to (n − m) th iteration for updating the current weight instead of the recent-most error.

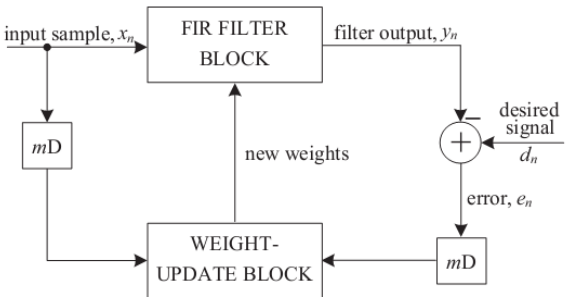

Fig. 2 Structure of the conventional delayed LMS adaptive filter

The weight-update equation of DLMS adaptive filter is given by

$$W_{n+1} = W_n + \mu.e_{n-m}.X_{n-m}. \qquad (10)$$

The block diagram of the DLMS adaptive filter is shown in Fig. 2, where the adaptation delay of m cycles amounts to the delay introduced by the whole of adaptive filter structure consisting of finite impulse response (FIR) filtering and the weight-update process. The delay introduced by the pipeline stages in FIR filtering, and the other part is due to the delay involved in pipelining the weight-update process.

$$W_{n+1} = W_n + \mu.e_{n-n1}.X_{n-n1}.$$
$$e_{n-n1} = d_{n-n1} - Y_{n-n1}$$
$$Y_n = W^T{}_{n-n2}.X_n \qquad (11)$$

We notice that, during the weight update, the error with n1 delays is used, while the filtering unit uses the weights delayed by n2 cycles. The modified DLMS algorithm decouples computations of the error-computation block and the weight-update block and allows us to perform optimal pipelining by feed-forward cut-set retiming of both these sections separately to minimize the number of pipeline stages and adaptation delay.

Pipelined Structure of the Error-Computation Block

The proposed structure for error-computation unit of an *N*-tap DLMS adaptive filter is shown in Fig. 4. It consists of *N* number of 2-b partial product generators (PPG) corresponding to *N* multipliers and a cluster of

L/2 binary adder trees, followed by a single shift–add tree. Each subblock is described in detail.

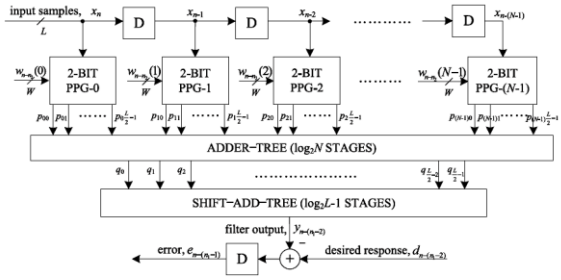

Fig.3 Error computation block

All the *L*/2 partial products generated by each of the *N* PPGs are thus added by *(L/2)* binary adder trees. The outputs of the *L*/2 adder trees are then added by a shift-add tree according to their place values. Each of the binary adder trees require log2 *N* stages of adders to add *N* partial product, and the shift–add tree requires log2 *L* − 1 stages of adders to add *L*/2 output of *L*/2 binary adder trees.2 The addition scheme for the error-computation block for a four-tap filter.

III. PROPOSED SYSTEM

- Filters the input vector $\chi(n)$ through the adaptive filter coefficients vector $\omega(n-1)$ to produce the filter output vectory $y(n)$
- Filter $y(n)$ through the secondary path filter s to produce the secondary actuator response at the sensor $ys(n)$
- Evaluates the current error sampling $e(n)=d(n)+ys(n)$.note the error here is formed by adding the signal rather than subtracting them to be compatible with real world sensors such as microphones and accelerometers

Adjoint LMS with Adaptive Filter

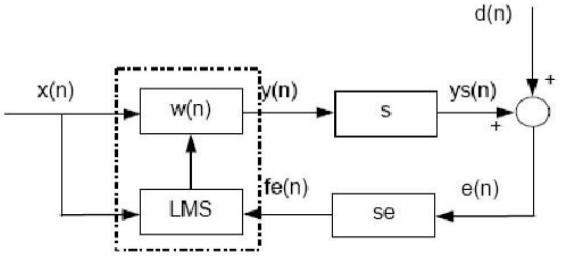

Fig. 4 Block Diagram of Adjoint LMS

- Filters the mirrored error vector $e(n)$ through the estimate of the secondary path *se* to produce the filter-error signal $fe(n)$
- Uses $\chi(n)$ and $fe(n)$ to calculate the normalized gradient vector and uses this to update the adaptive filter coefficients $\omega(n)$
- Supports both real and complex signals

The wiener solution to the above problem is given by $w(\omega)= s(\omega)^{-1} p(\omega)$ where $w(\omega)$ is the controller response at frequency ω. $s(\omega)$ is the response of the secondary path and $p(\omega)$ is the response of the primary of the secondary path at the same frequency the adaptive controller will asymptotically approach this wiener solutions provided that $s(\omega)$ is a minimum phase functions (does not have zeros outside the unit circle) and the controller length is large enough to accommodate the above convolution if $s(\omega)$ is not minimum phase functions the adaptive controller will approach the causal part of the solutions if the controller is too short the solutions will be truncated

Adjoint LMS Algorithm:

A new algorithm termed *adjoint LMS* which provides a simple alternative to the previously mentioned algorithms. In adjoint LMS, the error (rather than the input) is filtered through an adjoint filter of the error channel.Performance regarding convergence and mis adjustment are equivalent. However, linearity is not assumed in the derivation of the algorithm. Furthermore, equations for single-input-single-output (SISO) and multiple-input-multiple-output (MIMO) are identical and both remain order *N*.

Algorithm Specifications

An adaptive filter is specified as

$$y(k)=\sum_{n=0}^{M1} w_n(k)x(k-n)=W^T(k)X(k) \quad (13)$$

where *k* is the time index, *y* is the filter output, *x* the filter input, and w_n the filter coefficients. The vectors are,

$$W(k)=[w0(k), w1(k),.....w_{M1}(k)]^T \quad (14)$$

and

$$X(k)=[x(k), x(k-1),.....x(k-M1)]^T \quad (15)$$

provide for compact notation. It is also often convenient to write the filter operation by $y(k)=W(q^{-1},k)x(k)$ (16)

where

$$W(q^{-1},k)=\sum_{n=0}^{M1} w_n(k)q^{-1} \quad (17)$$

with q^{-n} representing a time delay operator *(i.e., $q^{-n}x(k) = x(k-n)$).*

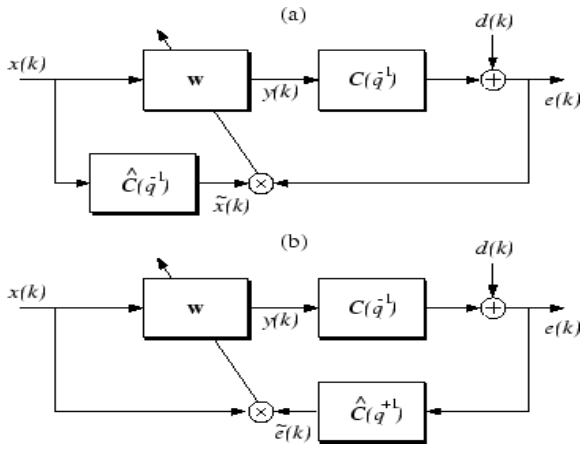

Fig. 5 ***(a) Filtered-x LMS, (b) Adjoint LMS***

The standard filtered x-LMS is illustrated in Fig. 6.2a where there exists a physical channel represented by $C(q^{-1},k)$ between the output of the filter and the available desired response. The output error is defined as

$$e(k) = d(k)-C(q^{-1},k)y(k) \quad (18)$$

and the filtered x-LMS algorithm expressed as

$$w(k+1) = w(k)+\mu e(k)X'(k)$$

$$x'(k) = C(q^{-1},k)x(k) \quad (19)$$

where, *x'*corresponds to the inputs filtered through a model *C'*of the error channel (μ controls the learning rate). This algorithm can be derived from the standard LMS algorithm assuming linearity by simply commuting the order of the filter and the channel. Thus the original $x$input become filtered by the channel (channel model) before entering the filter and the error appears directly at the output of the adaptive filter

We now present an alternative algorithm called adjoint LMS. The equations are

$$W(k+1)=W(k)+\mu\tilde{e}(k-M2)X(k-M2) \quad (20)$$

$$\tilde{e}(k)=\hat{C}(q^{+1},k)e(k) \quad (21)$$

These equations differ from Equations 20 and 21 in that the error rather than the input is now filtered by the channel model as illustrated in Fig. 6.2b ($M2$ is the order of the FIR channel model). Furthermore, the filtering is through the adjoint channel model (q^{-1} is replaced with q^{+1}). Graphically, an adjoint system is found for any filter realization by reversing the flow direction and swapping branching points with summing junctions and unit delays with unit advances. This is illustrated in Fig. 6.3 for a FIR tapped delay line. However, the method applies to all filter realizations including IIR and lattice structures. The consequence of the non causal adjoint filter is that a delay (equal to the channel model delay) must be incorporated into the weight update in Equation 20 to implement an on-line adaptation..

The adjoint system is found by reversing flow direction, swapping summing junctions with branching points and delays with advances. Adjoint LMS is clearly a simple modification of filtered-x LMS. For SISO systems the computational complexity of adjoint LMS and filtered x-LMS are identical. The real advantage comes when dealing with MIMO systems. In this case the adaptive filters are represented by an *L x P* matrix of transfer functions $W(q^{-1}.k)$ and the channel by a *P x Q* transfer function matrix $C(q^{-1},k)$. Filtered x-LMS does not generalize directly since matrices do not commute and it makes no sense to filter the input *X* by ***C*** since dimensions may not even match.

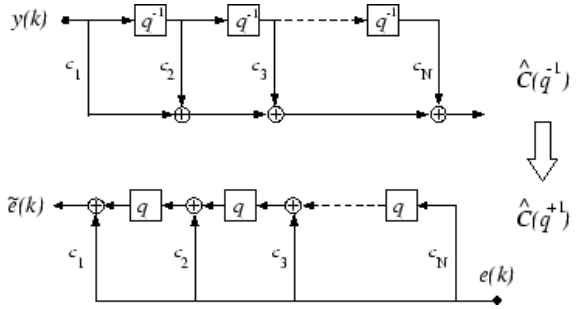

Fig. 6 FIR model of the channel and corresponding adjoint model.

$$W_{lp}(k+1)=W_{lp}(k)+\mu e^{T}(k)\tilde{X}_{lp}(k) \quad (22)$$

For $1 < l < L$ and $1 < p < P$, and there is now a filtered *matrix* of inputs for each filter W_{ip} formed as:

$$\tilde{X}^{T}_{lp}(k)=[\tilde{X}_{lp1}(k)\tilde{X}l_{p2}(k).....\tilde{X}_{lpQ}(k)] \quad (23)$$

with each row in the matrix found by filtering the input through the corresponding secondary path:

$$\tilde{X}_{lp1}(k)=C_{pq}(q^{-1},k)x_l(k) \quad (24)$$

The implementation of Multiple Error LMS results in a total of *L x P x Q* filter operations. In the cases of adjoint LMS, however, we encounter no such problem. Equations generalize directly:

$$Wlp(k+1) = Wlp(k) + \mu \tilde{e}(k-M2)Xl(k-M2) \quad (25)$$

$$\tilde{e}(k) = \tilde{C}(q^{+1}, k)e(k) \quad (26)$$

The output error $\mathbf{e}$ is dimension Q (number of channel outputs) whereas the error $\tilde{\mathbf{e}}$ after filtering through the adjoint MIMO channel model is order P(number of primary filter outputs) as desired. The clear advantage of this form is that operations remain order N, where N is the total number of filter parameters (compare the weight update matrix operation in Equation 7 to the vector operation in Equation 10). Table 6.1 gives a comparison of multiplications for some specific parameter values.

Multiplications	**Adjoint LMS**
$\tilde{\mathbf{e}}(k)$	$P \times Q \times M2 = 192$
weight updates	$L \times P \times M1 \times 2 = 384$
Total	567
Multiplications	Multiple Error LMS
filtered inputs	$L \times P \times Q \times M2 = 1,536$
weight updates	$L \times P \times M1 \times (Q+1) = 1,728$
Total	3264

Table 1: Comparison of computational complexity

The above table consider the Reference inputs, L=16. Adaptive filter outputs, P=16, Adaptive filter taps, M1=8, Channel outputs, Q=32, Channel model taps, M2 = 16.

Fixed-Point Implementation and Optimization

The most widely used format for floating-point arithmetic is the IEEE 754 standard. This standard details four floating-point formats - basic and extended each in single and double precision bit widths. Representation of every numeric value, in any number system, is composed of an Integer and a fractional part. The boundary that delimits them is called the radix point. The fixed-point format for representing numeric values derives its name from the fact that in this format, the radix point is fixed in a certain position. For integers this position is immediately to the right of the least significant digit.

Every floating-point number can be divided into three fields, sign S, exponent E, and fraction F. Using the binary number system, it is possible to represent any floating-point number as: $(-1)^{8}*1.f*2^{e\text{-}BIAS}$.

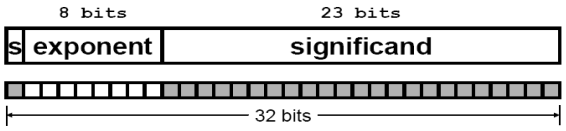

Fig.7 IEEE 754 Single precision format

IV. RESULTS

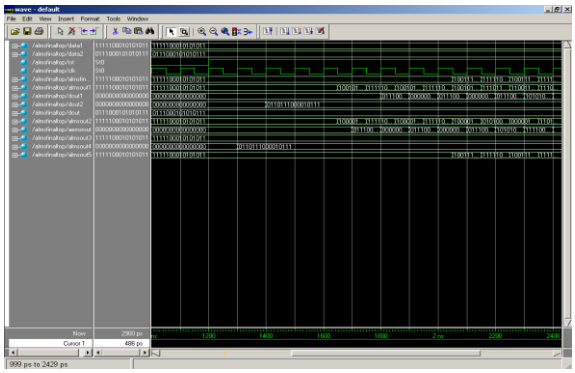

Fig. 8 Simulated output wave form for Adjoint LMS algorithm.

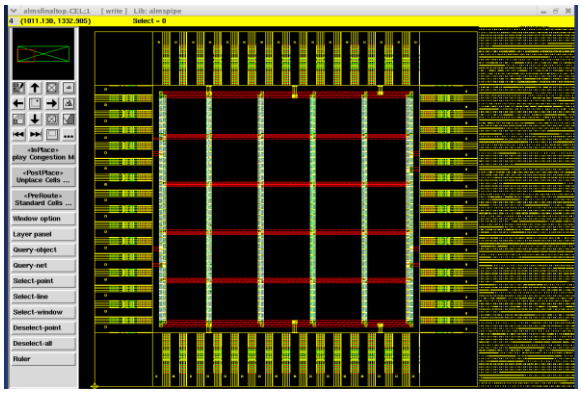

Fig. 9 After floor plan

Fig.8 shows the simulated and noise eliminated output waveform. It consider the two input signal such us real time input 16-bit data signal in data1 and the reference input signal 16-bit data signal in data2 and the

final output for the ALMS with adaptive filter output 16-bit in alms final output.

Fig. 9 to 12 shows the ASIC implementation of floorplan, placement, routing and final chip design for the proposed ALE adjoint-LMS architecture.

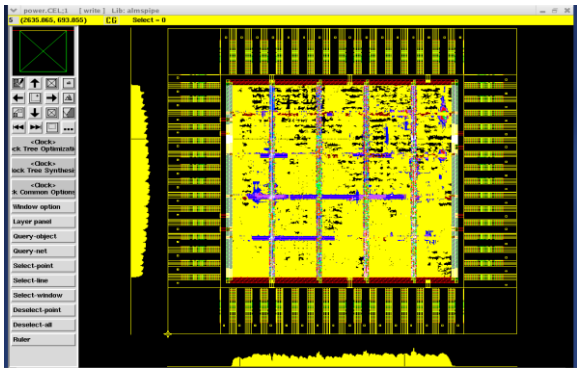

Fig. 10 Placement.

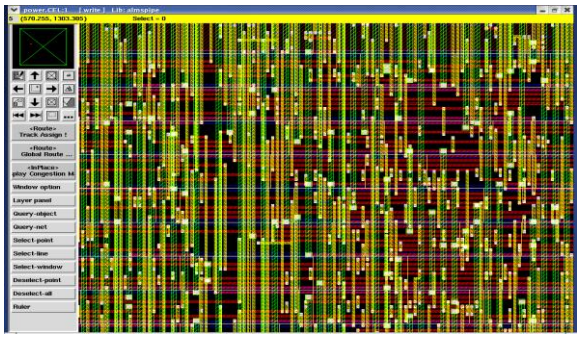

Fig.11 Routing

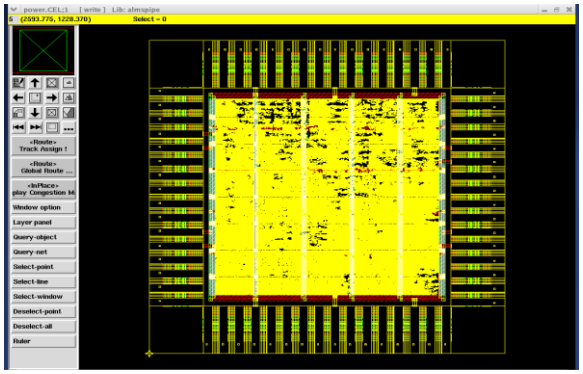

Fig.12 Final Chip Design

V. CONCLUSION

The proposed an efficient architecture for the design of a modified adjoint LMS adaptive filter. By using a Partial Product Generator (PPG), the combinational b locks can achieve efficient area-delay product and energy - delay product. The proposed design gives the large efficient output comprises the existing output with large complexities. The architecture implemented in cadence encounter and it's obtain the total area is 2.8nm, the total dynamic power consumption is 160.77mw, the leakage power is 548.96mw and the total time consumption is 11.15ps and 0.17 ps for input to the output response time. In feature the architecture implemented in Recursive Least Squares (RLS) algorithms to achieve the better performance compare to the least mean square algorithm.

REFERENCES

[1] Pramod Kumar Meher "Area-Delay-Power Efficient Fixed-Point LMS Adaptive Filter With Low Adaptation-Delay", VOL. 22, NO. 2, FEBRUARY 2014

[2] Muthulakshmi.G, Revathi.S, "VLSI Implementation of Delayed LMS Adaptive Filter with Efficient Area-Power-Delay", (IJISME), Volume-2, Issue-2, January 2014

[3] Rubiya C. H. Muralidharan V. Varatharaj M. ijird, "Area and Power Efficient Fixed Point LMS Adaptive Filter using Ripple Carry Adder", Vol 3 Issue 5, May, 2014.

[4] A.S.Sneka Priyaa, Dr.C.Santhi , "Power Optimized Architecture of LMS Adaptive Filter with Low Adaptation Delay", *(IJARTET), 7TH FEB 2015.*

[5] Farheen Fathima, P.Anuradha, "VLSI Implementation of Efficient Fixed-Point LMS Adaptive Filter with Low Adaptation Delay", ICRACVES-2014.

[6] Tadaaki Kimijima, Kiyoshi Nishikawa, "An Efficient architecture of pipelined LMS adaptive filters", IEICE Trans. Fundamentals Vol.E82-A,N0 8, Aug 1999.

[7] Julie E. Greenberg,"Modified LMS Algorithms for Speech Processing with an Adaptive Noise Canceller", IEEE Transactions On Speech And Audio Processing, VOL. 6, NO. 4, JULY 1998.

[8] T.J.Milna, S.K Mythili , "Survey on DLMS Adaptive Filter With Low Delay", IJRCCT, Vol 3, Issue 9, September – 2014.

[9] Ahmed Elhossini, Shawki Areibi, Robert Dony. "An FPGA Implementation of the LMS Adaptive Filter for Audio Processing", (April 2004).

[10] Sundar G. Sankaran and A. A. (Louis) Beex, "Normalized LMS Algorithm with Orthogonal Correction Factors", (1998 IEEE)

[11] Rohita P. Patil, Prajakta P. Bharadkar, "Survey on Different Architectures of DLMS Adaptive Filter", IJR, Volume 2, Issue 3, March 2015

AUTHOR PROFILE

M.Karthika received her B.E degree in electronics and communication engineering, from sasurie academy of engineering in Tamilnadu, India in 2013. She is currently pursuing her M.E in Applied Electronics from Kongunadu college of engineering and Technology, Tamilnadu, India in 2015. Her research interests include CMOS 0.18µm standard cell design and Lowpower VLSI design.

P.Elayaraja received his B.E degree in electronics and communication engineering, from Jayaram College of engineering and technology Tamilnadu, India in 2003. He received his ME degree in communication system, from jayaram college of engineering and Technology, Tamilnadu, India in 2007. He is currently pursuing his Ph.d in image processing his areas of research interest is image processing.

K Deepa received her B.E degree in electronics and communication engineering, from KSR college of engineering and Tamilnadu, India in 2013. She is currently pursuing her M.E in Applied Electronics from Kongunadu college of engineering and Technology, Tamilnadu, India. Her areas of research interest are low power VLSI and CMOS based mixed signal and analog design.