Distributed Arithmetic Based FIR Adaptive Digital Filter Design Using LMS Algorithm: A Review

Mr. Wasim Maroofi
Dept. of Electronics & Tele-communication
Anjuman College of Engineering & Technology
Nagpur, India
e-mail: wasimmaroofi@gmail.com

Prof. Lalit Jain
Dept. of Electronics & Communication
NRI Institute of Information Science & Technology,
Bhopal, India
e-mail: lalitjain_engg@yahoo.co.in

Prof. Sanjay Ganar
Dept. of Electronics & Tele-communication
Anjuman College of Engineering & Technology
Nagpur, India
e-mail: sanjay_r_ganar@yahoo.com

Abstract—Adaptive filters have proven to be an instrumental system component in the modern signal processing applications. Innumerable systems that we presently depend on in our daily life would not exist without the use of adaptive filters. There are two conflicting algorithms that are commonly used for implementation of adaptive filters; namely the Recursive Least Square (RLS) and Least Mean Square (LMS) algorithm. RLS algorithm is a powerful technique giving greater convergence rates than LMS, but this increases the computational complexity. This has proved to be a hindrance towards its use in real time applications. On the other hand, the LMS algorithm is a simple yet robust way of yielding good performance. However, the convergence rates are slow in LMS, which can be improved by using Block LMS algorithm. In addition, a computationally efficient distributed arithmetic (DA) approach may be used to implement the Block Least Mean Square (BLMS) algorithm which may further reduce computational complexity. DA scheme employs bit-serial operations and look-up tables (LUTs) sharing and weight increment terms of BLMS algorithm to implement high throughput filters. This paper focuses to present a survey on various algorithms used to implement FIR adaptive filters using Distributed Arithmetic and Block LMS algorithm.

Keywords- *Distributed Arithmetic, LMS, BLMS, DLMS, DA, Adaptive Filter, FIR Filter*

*****

I. INTRODUCTION

Adaptive filters have become a mainstay in Digital Signal Processing. They have attracted the attention of many researchers during the last decades, due to their property of self-designing [1]. They are used in numerous applications that include acoustic echo cancellation, noise cancellation, channel equalization, wireless channel estimation, radar guidance systems and many other adaptive signal processing applications. The ability of an adaptive filter to operate satisfactorily in an unknown environment and track time variations of input statistics make the adaptive filter a powerful device for signal processing and control applications[1]. Adaptive filtering algorithms are broadly classified into two categories, namely the Least Mean Squares (LMS) and the Recursive Least Squares (RLS) algorithm. LMS algorithms represent the low complexity, simplicity of implementation and most easily applied adaptive algorithms, and hence it has found productive applications in many areas. On the other hand, the RLS algorithm demands increased circuit complexity, computational cost, and conformity. One of the main concerns in all practical situations is to provide algorithms which provide fast convergence of the adaptive filter coefficients and at the same time good filtering performance. LMS based Finite Impulse Response (FIR) adaptive filter is widespread because it uses simple, yet powerful adaptive algorithm and provides adequate convergence performance. The most prominent advantages of FIR filters can be listed as: FIR filters can be designed to be linear phase, simple and efficient to implement in hardware, suited to multi-rate applications, coefficients are easy to calculate, are always stable and design methods are usually linear. The adaptive algorithm for FIR filters have been successfully applied in diverse application areas such as communication, radar, sonar, seismology and biomedical engineering. It has been used in applications where reducing the computational requirements is necessary. LMS algorithm is a gradient descent algorithm which adjusts the adaptive filter taps modifying them by an amount proportional to the instantaneous estimate of the gradient of the error surface [2]. The LMS algorithm aims to reduce the mean squared error.

The RLS algorithm recursively finds the filter coefficients that minimize a weighted linear least squares cost function relating to the input signals. The input signals, in the beginning of RLS algorithm, are considered deterministic, while in LMS, the input signals are considered stochastic. As compared to other algorithms, RLS shows faster convergence performance which is achieved at the cost of high computational complexity. A distributed arithmetic (DA) approach can be employed for the application of block least mean square (BLMS) algorithm. DA is basically a bit-serial computational operation that forms an inner (dot) product of a pair of vectors in a single direct step. The key advantage of DA lies in its mechanization efficiency [3]. Distributed arithmetic (DA) is one way to implement multiplier-free convolution, where the MAC (multiply and accumulate) operations are replaced by a series of LUT (look-up tables) access and summations [4]. Design based on DA makes use of a unique LUT sharing scheme to compute the filter outputs and weight-increment terms of BLMS algorithm. In addition, it provides ample saving of adders which form the most important component of DA based structures [5]. DA based formulation of BLMS algorithm is used to reduce the area where both, (1)

convolution operation to compute filter output and (2) correlation operation to compute weight-increment term could be performed by using the same LUT. Hence, a DA based implementation of adaptive filter is highly computational and area efficient [6].

II. LMS Adaptive FIR Filter with Distributed Arithmetic Approach

A. Adaptive Filter Theory

Digital filtering is implemented in two ways which is either by using FIR filters or infinite impulse response (IIR) filters. The principal difference between FIR and IIR filtering is that in FIR filters, the output is dependent only on the inputs, while in IIR filters, the output is dependent on the inputs and the previous outputs. Moreover, FIR filters do not suffer from stability issues which is commonly observed in IIR filters.

An adaptive filter is a time-variant filter whose coefficients are adjusted in such a way so as to optimize a cost function or to satisfy some predetermined optimization criterion. They can automatically adapt (self-optimize/self-adjust) regardless of varying environments and changing system requirements. They can be suitably trained to perform specific type of filtering and decision-making tasks according to some updating equations which behave as training rules for the adaptive filter.

An adaptive filter consists mainly of two components, i.e. a linear filter and an adaptive algorithm. The Fig. 1 shows the diagram of an FIR adaptive filter.

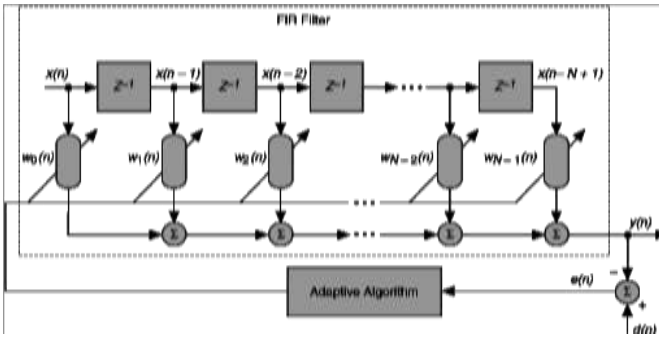

Figure 1. Adaptive FIR Filter with Adaptive Algorithm

where,
$x(n)$ is the input signal to a linear FIR filter at time n
$y(n)$ is the corresponding output signal
$d(n)$ is the desired signal to the adaptive filter (another input)
$e(n)$ is the error signal that denotes the difference between $d(n)$ and $y(n)$
z^{-1} is a unit delay
$w_i(n)$ is the multiplicative gain. This multiplicative gain also is known as the filter coefficient
i is an integer with a value range of [0, N–1]
The adaptive algorithm adjusts the filter coefficient $w_i(n)$ in an iterative manner to minimize the power of $e(n)$.

Various algorithms can be applied to the FIR adaptive filter to control the manner in which the filter adjusts its coefficients. The adaptive algorithms adjust the filter coefficients to minimize the following cost function $J(n)$:

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

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 $J(n)$, they can be categorized into the following two groups:

- Least Mean Squares (LMS) algorithms
- Recursive Least Squares (RLS) algorithms

In this paper, we will focus on the former algorithm.

B. The LMS Algorithm Overview

The LMS algorithm was developed by Windrow and Hoff [20] in 1959. In adaptive filtering, it is the most popular and widely used adaptive algorithm, appearing in numerous commercial and scientific applications. The algorithm uses a gradient descent to estimate a time varying signal. This method finds a minimum, if it exists, by taking steps in the direction negative of the gradient. It is done by adjusting the filter coefficients, so as to minimize the error. The gradient is the *del* operator (partial derivative) and is applied to find the divergence of a function, which is the error with respect to the nth coefficient in this case. The LMS algorithm reaches towards the minimum of a function to minimize error by taking the negative gradient of the function. A LMS algorithm can be implemented as shown in Fig. 2.

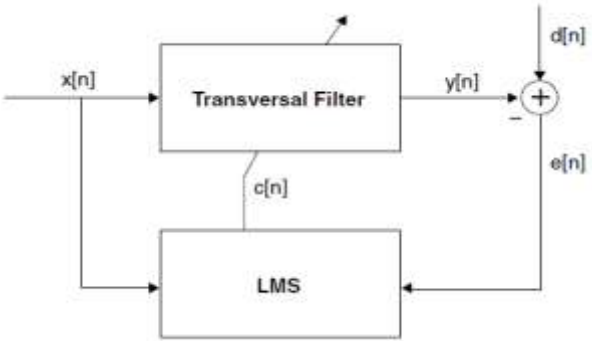

Figure 2. LMS Implementation Using FIR Filter

The desired signal d[n] is tracked by adjusting the filter coefficients c[n]. The input reference signal x[n] is a known signal that is fed to the FIR filter. The difference between d[n] and y[n] is the error e[n]. The error e[n] is then fed to the LMS algorithm to compute the filter coefficients c[n+1] to iteratively minimize the error. The following is the LMS equation to compute the FIR coefficients:

$$c[n+1] = c[n] + \mu * e[n] * [n]$$

The convergence time of the LMS algorithm depends on the step size μ. If μ is small, then it may take a long convergence time and this may defeat the purpose of using an LMS filter. But if μ is very large, the algorithm may never converge. Value of μ should be scientifically computed based on the effects the environment will have on d (n). The LMS adaptive filter is described by the equations:

$$W(n+1) = W(n) + \mu(n)\, e(n)\, X(n) \quad (1)$$
$$y(n) = W^T(n)X(n) \quad (2)$$
$$e(n) = d(n) - y(n) \quad (3)$$
$$e(n) = d(n) - W^T(n)X(n) \quad (4)$$

where,
$W(n) = [w_0(n)\ w_1(n)\ \ldots\ w_{L-1}(n)]^T$ is the coefficient (weight) vector,

X (n) = [x (n) x (n – 1) …. x (n – L + 1)] ^{T} is the input signal vector,
y (n) is the output response,
d (n) is the desired signal,
e (n) is the error signal and
μ (n) is the step size (convergence factor)

The three key reasons why the LMS adaptive filter is so popular are:
(i) It is rather easy to implement in software and hardware due to its computational simplicity and efficient memory usage.
(ii) It performs robustly in the presence of numerical errors caused by finite-precision arithmetic.
(iii) Its behavior has been analytically characterized to the point where a user can easily set up the system to obtain adequate performance with only limited knowledge about the input and desired response signals.

C. *Block LMS Algorithm*

The Block-based LMS (BLMS) algorithm is one of the many efficient adaptive filtering algorithms aimed at increasing convergence speed and reducing the computational complexity. The basic principle of the BLMS algorithm is that the filter coefficients remain unchanged during the processing of each data block and are updated only once per block [21]. To reduce the computational requirements of LMS algorithm, BLMS was introduced. Here the filter coefficients are held constant over each block of L samples, and the filter output y(n) and the error e(n) for each value of n within the block are calculated using the filter coefficients for that block. Then at the end of each block, the coefficients are updated using an average for the L gradients estimates over the block. The updating equation of the BLMS algorithm for linear adaptive filters is as follows:

$$w_{k+1} = w_k + \mu \sum_{m=0}^{L-1} x(kL+m)\, e\,(kL+m)$$

where k refers to the kth block of input data, k-th denotes the index of the incoming data, and L is the block length. It uses block processing technique in which block of output is calculated from a block of input samples during each iteration. The main difference between the BLMS and LMS is that the former updates all the filter coefficients once every L samples while the latter updates all the filter coefficients once every sample. The processing of the LMS adaptive digital filters can be increased L fold using BLMS algorithm, where L is the block size. Due to the fact that in the BLMS algorithm, the computation of the filter output and of the gradient itself are represented by linear convolution and correlation, they can be implemented efficiently using FFT. Based on the convolutional property of FFT and overlap – save method, the BLMS algorithm can be implemented in the frequency domain and the computational complexity can be reduced dramatically.

D. *Distributed Arithmetic (DA)*

Distributed arithmetic (DA) is an efficient multiplication-free technique for calculating inner products first introduced by Croisier, *et al.* [22] and Zohar [23] and further developed by Peled and Liu [24] more than three decades ago. The multiplication operation has been replaced by a mechanism that generates partial products and then sums the products together. The key difference between distributed arithmetic and standard multiplication is in the way the partial products are generated and added together. Since its introduction, distributed arithmetic has been widely adopted in many digital signal processing applications, including but not limited to digital filtering [25][26], discrete cosine transform [27][28], discrete Fourier transform [29].

The basic DA technique is bit–serial computational operation. Though it has the advantage of efficiency of mechanization [3][32], it suffers from slowness because of its inherent bit-serial nature. This disadvantage can be eliminated in two possible ways:

- It is not real if the time required to input eight 8-bit words one at a time in a parallel fashion is exactly the same as the time required to input (simultaneously on eight wires) all eight words serially.
- Speed can be increased by employing techniques such as bit pairing or partitioning the input words into the most significant half and least significant half, the least significant half of the most significant half, etc., thereby introducing parallelism in the computation.

It is mainly a bit level rearrangement of MAC operation. It efficiently implements multiply-accumulate using basic building blocks i.e. LUTs in FPGA. DA provides area saving of up to 50-80%.

As an example of direct DA [3][32] inner-product generation, consider the calculation of the following sum of products as:

$$y = \sum_{k=1}^{K} a_k x_k \tag{5}$$

The a_k are fixed coefficients, and the x_k are the input data words. If each x_k is a 2's-complement binary number scaled such that $|x_k|<1$ then each x_k can be presented as:

$$x_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} \tag{6}$$

where the b_{kn} are the bits, 0 or 1, b_{k0} is the sign bit, and $b_{k,n-1}$ is the least significant bit (LSB).
Combining (5) and (6) to express y in terms of the bits of x_k:

$$y = \sum_{k=1}^{K} a_k \left[-b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} \right] \tag{7}$$

Equation (7) is the conventional form of expressing the inner product. Direct mechanization of this equation defines a "lumped" arithmetic computation. Upon interchanging the order of the summations, we get:

$$y = \sum_{n=1}^{N-1} \left[\sum_{k=1}^{K} a_k b_{kn} \right] 2^{-n} + \sum_{k=1}^{K} a_k (-b_{k0}) \tag{8}$$

Here (8) defines a distributed arithmetic computation. Consider the bracketed term in (8):

$$y = \sum_{k=1}^{K} a_k b_{kn} \tag{9}$$

Because each b_{kn} may take on values of 0 and 1 only, expression in (9) will have only 2^K possible values. Instead of computing these values on line, we can pre-compute these values and store them in a ROM. The input data can be used to directly address the memory and the result, i.e. the expression (9), can be put into an accumulator. The memory contains the result y after N such cycles.

DA can be applied to BLMS and also to the other algorithms which may reduce computational complexity and

also the area usage. The DA based BLMS (DA-BLMS) structure in [5] contains one error bit-slice generator (EBSG), one DA-module and one weight-update cum bit-slice generator (WBSG). WBSG updates the filter weights and gives out the requisite bit-vectors which are in accordance with the DA-formulation. EBSG computes the error block according to (30) and generates its bit-vectors. The DA-module updates the LUTs and makes use of the bit-vectors generated by WBSG and EBSG to calculate the filter output and weight-increment terms according to (15) and (31). The structure for DA-based BLMS adaptive filter for N = 16 and L = 4 is shown in Fig. 3.

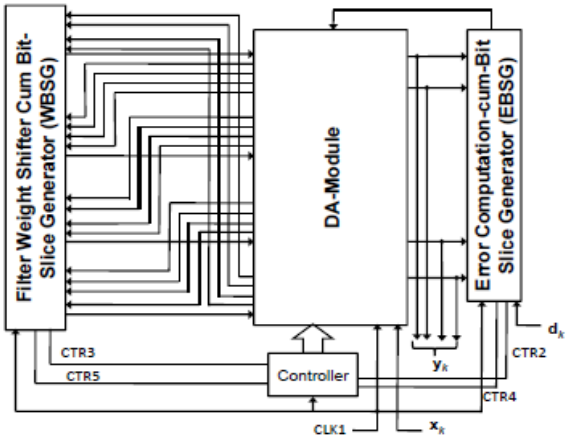

Figure 3. Structure for DA-BLMS adaptive filter for N=16 and L=4 [5]

The DA-module receives a block of L input samples [x_k = {x(kL), x(kL – 1), … , x(kL – L + 1}] of each B` bits wide in every iteration, and computes a block of filter output [y_k = {y(kL), y(kL – 1), … , y(kL – L + 1}] of B bits wide each. It also receives a block of errors e_k in each iteration, and calculates the weight-increment term for all the N components of the weight-vector w_k. By using this scheme, a common set of LUTs could be used for the computation of weight-increment terms and filter output. Now, because a block of input samples changes after each iteration, it becomes inevitable to update the LUTs in order to accommodate the new input-block in each iteration.

III. Literature Survey

Adaptive filters play a very crucial role in signal processing. LMS algorithm is a broadly used adaptive algorithm for its robustness and low hardware complexity [1]. Though in practical applications, many modified LMS algorithms have been proposed. In LMS algorithm, delay is encountered in the feedback error used to update the filter weights and hence cannot be used for pipeline implementation, when the sampling rate is high. A delayed LMS (DLMS) algorithm for pipelined use of LMS based adaptive filter has been proposed by [7]. In conventional LMS adaptive filter, the estimated signal is computed in each data interval and subtracted from the desired signal. The error is then used to update the filter coefficients prior to the arrival of next sample. In some practical applications of LMS algorithm, the adaptation step can be performed only after a fixed delay. In such cases DLMS algorithm is known to be used. In DLMS, the filter coefficients have to be updated with a delayed version of the error signal. The mere variation between the conventional LMS and Delayed LMS algorithm is that the correction term is computed from error corresponding to the previous iteration to update the filter weights of the present iteration. Several methods have been proposed to implement DLMS based adaptive filters efficiently in systolic VLSI architecture with minimum adaptation delay [7] [8] [9] [10]. For avoiding the adaptation delay a modified DLMS algorithm has been proposed by [11]. For some applications of adaptive FIR filtering, the adaptation algorithm can be realized only with a delay in the coefficient update. However, this has an opposing effect on the convergence behavior of the algorithm. It has been shown in this work, how the DLMS algorithm can be altered into the regular LMS algorithm by a marginal rise in computational overhead. A hardware-efficient pipelined architecture for the LMS adaptive FIR filter that produces the same output and error signals as would be produced by the standard LMS adaptive filter architecture without adaptation delays has been proposed by [12]. A new pipelined architecture for the DLMS algorithm has been proposed in [13]. In this work, good convergence characteristics, a short latency and high throughput characteristics with less than half an amount of calculation compared to the conventional architectures have been achieved simultaneously. Block LMS (BLMS) algorithm is suitable for fast and computationally-efficient implementation of adaptive digital filters. The working strategy of BLMS algorithm is that the filter block implements an adaptive LMS filter, where the adaptation of filter weights occurs once for every block of samples. If we consider the convergence performance, then the BLMS adaptive digital filters and LMS adaptive digital filters have same performance, except for throughput which is higher in BLMS adaptive filters. BLMS algorithms like time and frequency domain block filtered-X LMS (BFXLMS) have been proposed by [14] for specific applications. A BLMS algorithm with delayed weight adaptation for hardware implementation of FIR adaptive filters has been proposed by [15]. A delayed BLMS algorithm and a concurrent multiplier-based design for high throughput pipeline execution of BLMS adaptive digital filters have been proposed in [15]. A BLMS algorithm with delayed weight adaptation for hardware implementation of FIR adaptive filters has been proposed. The proposed algorithm has been referred to as delayed block least mean square (DBLMS) algorithm. Unlike the DLMS algorithm, the DBLMS algorithm takes a block of L input samples and returns a block of L output samples in every training cycle. The simulation result demonstrates that the DBLMS algorithm has convergence performance equivalent to that of the DLMS algorithm. The parallelism inherent in the DBLMS algorithm has been exploited to derive a greatly concurrent systolic architecture for FIR adaptive filters. The suggested architecture will be able to support L times higher sampling rate compared to the best of the existing designs involving pipelining and as a result would contain less number of samples of adaptation delays and would provide a more proficient implementation of LMS adaptive filters. An FPGA design and implementation of high throughput adaptive digital filter using Fast Block Least Mean Squares (FBLMS) adaptive algorithm has been suggested by [16] based on DA, followed by a low power and less complex implementation in [17]. The DA filter structure computes the inner (dot) product by first of all shifting and accumulating of partial products and finally storing them in LUT. In addition, the desired adaptive digital filter will be free from any multipliers which tend to eat up the chip area. So a DA based implementation of adaptive filter is

highly computational and area efficient. Moreover, the fundamental building blocks in the DA architecture map well to the architecture of today's Field Programmable Gate Arrays (FPGA). FPGA implementation results follow that the proposed DA based adaptive filter in [16] can be implemented with considerably smaller usage of area which can be up to 45% less compared to existing FBLMS algorithm based adaptive filter. The proposed implementation in [17] which is based on DA, substitutes multiply and accumulate (MAC) operations with a series of LUTs making this implementation highly power and area efficient. This is accomplished at the price of a modest increase in memory usage. This work further reduces the area usage of DA based adaptive filter implementation by about 52% which is less compared to that of the existing FBLMS algorithm based adaptive filter implementations. A variant of BLMS algorithm, where the filter weight updation and error calculation are both computed block wise is presented in [18]. The algorithm performs considerably well with a slight trade off in the learning curve time and maladjustment, both of which can be attuned by varying the step size depending on the requirement. The structure for BLMS adaptive filters supports a very low sampling rate because it has a single MAC cell for the computation of filter output and the weight increment term. The architecture can be further altered to accomplish the variants of LMS algorithm.

A new adaptive filter architecture for very high throughput using DA is presented in [19] where bit-serial operations and LUTs are used as a result of DA in order to implement high throughput filters that utilize only about one cycle per bit of resolution irrespective of filter length. Still, structuring these adaptive DA filters requires the LUTs to be recalculated for each adaptation which may cancel out any performance benefits of DA filtering. The use of an auxiliary LUT with special addressing, the efficiency and throughput of DA adaptive filters can be of the same order as fixed DA filters. The development of DA adaptive filters is described which also shows that practical implementations of DA adaptive filters have very high throughput compared to multiply and accumulate architectures. Also that DA adaptive filters have a potential area and power consumption improvement over digital signal processing microprocessor architectures has been presented.

IV. Conclusion

In this paper, we have tried to review the different implementations of distributed arithmetic based adaptive digital FIR filter design using various LMS algorithms. In section II, we first focus on the adaptive filter theory followed by the LMS algorithm and BLMS algorithm overview for use with adaptive FIR filter and also the use of distributed arithmetic is discussed in brief. In section III, we summarize the intensive work done by a plethora of researchers towards implementation of adaptive FIR filters.

Acknowledgment

The authors would sincerely wish to thank and express deep sense of gratitude to respected mentor and guide Prof. M. Nasiruddin, Associate Professor and Head of Department of Electronics and Telecommunication Engineering of Anjuman College of Engineering & Technology, Nagpur, for his technical guidance, support, encouragement, along with constructive reproach, which constantly motivated us to strive harder for excellence in our pursuit.

References

[1] Simon Haykin, Adaptive Filter Theory, 3rd ed., Prentice Hall, 2002, pp. 1–77.

[2] Komal R. Borisagar, Dr. G. R. Kulkarni, "Simulation and comparative analysis of LMS and RLS algorithms using real time speech input signal," Global Journal of Researches in Engineering, vol. 10, issue 5, pp. 44–47, October 2010.

[3] S.A.White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," IEEE ASSP Mag., vol. 6, pp. 4–19, Jul. 1989

[4] K. Krishna Reddy and Ch. K. Pratap Kumar M., "An LUT adaptive filter using DA," in International Journal of Engineering Development and Research, vol. 1, issue 3, pp. 364–368, December 2014.

[5] Mohanty, B.K.; Meher, P.K., "A High-Performance Energy-Efficient Architecture for FIR Adaptive Filter Based on New Distributed Arithmetic Formulation of Block LMS Algorithm," Signal Processing, IEEE Transactions on , vol.61, no.4, pp.921,932, Feb.15, 2013 doi: 10.1109/TSP.2012.2226453.

[6] M. Devipriya, V. Saravanan, N. Santhiyakumari, "Power efficient and high throughput of FIR filter using block least mean square algorithm in FPGA," International Journal of Research in Engineering and Technology, vol. 3, special issue 2, pp. 1–6, March 2014.

[7] R.Haimi-Cohen, H.Herzberg and Y.Beery, "Delayed adaptive LMS filtering: Current results," in Proc.IEEE Int. Conf. Acoust., Speech, Signal Process, Albuquerque, NM, pp. 1273–1276, Apr. 1990.

[8] L.D.Van and W.S.Feng, "Efficient systolic Architectures for 1-D and 2-D DLMS adaptive digital filters," in Proc. IEEE Asia Pacific Conf. Circuits Syst., Tianjin, China, pp. 399–402, Dec. 2000.

[9] L.D.Van and W.S. Feng, "An efficient architecture for the DLMS adaptive filters and its applications," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 48, no. 4, pp. 359–366, Apr. 2001.

[10] S.Ramanathan and V.Visvnathan, "A systolic architecture for LMS adaptive filtering with minimal adaptation delay," in 9th Int. Conf. VLSI Des., Bangalore, pp. 286–289, January 1996.

[11] R.D.Poltmann, "Conversion of the delayed LMS algorithm into the LMS algorithm," IEEE Signal Process. Lett., vol. 2, p. 223, Dec. 1995.

[12] S.C.Douglas, Q. Zhu, and K. F. Smith, "A pipelined LMS adaptive FIR filter architecture without adaptation delay," IEEE Trans. Signal Process., vol. 46, pp. 775–779, Mar. 1998.

[13] Kimijima T., Nishikawa K., and Kiya H., "A pipelined architecture for DLMS algorithm considering both hardware and complexity and output latency," EUSIPCO, Rhodes, GRECE, pp. 503-506, 1998

[14] Q.Shen and A.S.Spanias, "Time and frequency domain X block LMS algorithm for single channel active noise control," Control Eng. J., vol. 44, no. 6, pp. 281–293, 1996

[15] Mohanty, B.K.; Meher, P.K., "Delayed block LMS algorithm and concurrent architecture for high-speed implementation of adaptive FIR filters," TENCON 2008 - 2008 IEEE Region 10 Conference , vol., no., pp.1,5, 19-21 Nov. 2008.

[16] Baghel, S.; Shaik, R., "FPGA implementation of Fast Block LMS adaptive filter using Distributed Arithmetic for high throughput,", International Conference on Communications and Signal Processing 2011 (ICCSP), pp.443, 447, 10-12 Feb. 2011, doi:10.1109/ ICCSP.2011.5739356

[17] Baghel, S.; Shaik, R., "Low power and less complex implementation of fast block LMS adaptive filter using distributed arithmetic," Students' Technology Symposium (TechSym), 2011 IEEE , pp.214,219, 14-16 Jan. 2011, doi: 10.1109/TECHSYM.2011.5783848

[18] R. Jayashri, H. Chitra, S. Kusuma, A. V. Pavithra, V. Chandrakanth, "Memory based architecture to implement simplified block LMS algorithm on FPGA," International Conference on Communications and Signal Processing (ICCSP), 2011, vol., no., pp.179,183, 10-12 Feb. 2011, doi: 10.1109/ICCSP.2011.5739296

[19] Allred, D.J.; Yoo, Heejong; Krishnan, V.; Huang, W.; Anderson, D.V., "LMS adaptive filters using distributed arithmetic for high throughput," IEEE Transactions on Circuits and Systems I: Regular Papers, vol.52, no.7, pp.1327,1337, July 2005, doi:10.1109/TCSI.2005.851731

[20] Widrow, B. and Hoff, M.E., Adaptive switching circuits, IRE WESCON Conv. Rec., 4, 96–104, Aug. 1960

[21] Vijay K. Madisetti, Douglas B. Williams, The Digital Signal Processing Handbook, CRC Press, 1997, pp. 22-15.

[22] A. Croisier, D. Esteban, M. Levilion, and V. Rizo, "Digital filter for PCM encoded signals US Patent 3,777,130," 1973.

[23] S. Zohar, "New Hardware Realizations of Nonrecursive Digital Filters," IEEE Transactions on Computers, vol. C-22, no. 4, pp. 328–338, 1973.

[24] A. Peled and B. Liu, "A New Hardware Realization of Digital Filters," IEEE Transactions on ASSP, vol. 22, no. 6, pp. 456–462, 1974

[25] F. Taylor, "An analysis of the distributed arithmetic digital filter," IEEE Transactions on ASSP, vol. 34, no. 5, pp. 1165–1170, 1986.

[26] M. Martinez-Peiro, J. Valls, T. Sansaloni, A. Pascual, and E. Boemo, "A comparison between lattice, cascade and direct form FIR filter structures by using a FPGA bitserial distributed arithmetic implementation," in Proc. 6th IEEE International Conference on Electronics, Circuits and Systems ICECS '99, vol. 1, pp. 241–244 vol.1, 1999.

[27] Y.-H. Chan and W.-C. Siu, "On the realization of discrete cosine transform using the distributed arithmetic," IEEE Transactions on Circuits and Systems I, vol. 39, no. 9, pp. 705–712, 1992.

[28] H.-C. Chen, J.-I. Guo, T.-S. Chang, and C.-W. Jen, "A memory-efficient realization of cyclic convolution and its application to discrete cosine transform," IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 3, pp. 445–453, 2005.

[29] H.-C. Chen, J.-I. Guo, C.-W. Jen, and T.-S. Chang, "Distributed arithmetic realisation of cyclic convolution and its DFT application," IEEE Proceedings -Circuits, Devices and Systems, pp. 615–629, 2005.

[30] M. D. Meyer and D. P. Agrawal, "A Modular pipelined implementations of a delayed LMS transversal adaptive filter", in Proc. IEEE Int. Symp. Circuits Syst. (New Orleans, LA), May 1990, pp. 1943–1946.

[31] F. R. Guo and L. S. DeBrunner, "Two high-performance adaptive filter implementation schemes using distributed arithmetic",IEEE Transaction on Circuit and Systems-II: Express Briefs, vol. 58, no. 9, pp. 600–604, Sept. 2011.

[32] Xilinx Technical article on "The Role of Distributed Arithmetic in FPGA based Signal Processing", pp. 1-15, www.xilinx.com.