Fast Block LMS Adaptive Filter Using DA Technique for High Performance in FGPA

Nagaraj Gowd H [1], K.Santha[2], I.V.Rameswar Reddy[3]
[1, 2, 3] Dept. Of ECE, AVR & SVR Engineering College, Kurnool, A.P, India

Abstract—This paper proposes a design and implementation of high throughput adaptive digital filter using Fast Block Least Mean Squares (FBLMS) adaptive algorithm. The filter structure is based on DA Technic, which is able to calculate the inner product by shifting, and accumulating of partial products and storing in look-up table, also the desired adaptive digital filter will be multiplier less. Thus a DA based implementation of adaptive filter is highly computational and area efficient. Furthermore, the fundamental building blocks in the DA architecture map well to the architecture of today's Field Programmable Gate Arrays (FPGA). FPGA implementation results conforms that the proposed DA based adaptive filter can implement with significantly smaller area usage, (about 45%) less than that of the existing FBLMS algorithm based adaptive filter.

Keywords— FBLMS, distributed arithmetic, FFT, cyclic convolution.

1. INTRODUCTION

Adaptive digital filters are widely used in the area of signal processing such as echo cancelation, noise cancelation, channel equalization for communications and networking systems [1], [2]. The necessity of hardware implementation requires various of performances such as high speed, low power dissipation and good convergence characteristics.
Fast block least mean square (FBLMS) algorithm proposed by Clark et al. [3] is one of the fastest and computationally efficient adaptive algorithm since here the process of filtering and adaption is done in frequency domain by using FFT algorithms. But there is still possibility to further enhanced the throughput of FBLMS algorithm based adaptive filters by applying the concept of Distributed arithmetic (DA) proposed by A. Peled and B. Liu [4]. Using bit level rearrangement of a multiply accumulate terms, DA can hide the complex hardware multipliers and therefore, the desired system becomes multiplier-less. DA is a powerful technique for reducing the size of a parallel hardware multiply-accumulate that is well suited for FPGA designs. Recently there has been a trend to implement DSP functions using FPGAs. While application specific integrated circuits (ASICs) are the traditional solution to high performance applications due to high development costs and time-to-market factor. The main reason behind the popularity of the FPGA is due to balance that FPGAs provide the designer in terms of flexibility, cost, and time-to-market.
The concept of DA has already applied to LMS based adaptive filters [5], [6], [7], [8] but, not to FBLMS algorithm based adaptive filters. This paper proposes a new hardware efficient implementation of FBLMS algorithm based adaptive filter using DA. The proposed architecture reduces the area requirement of original FBLMS by reducing the number of hardware multipliers using DA, result as adaptive filter with low power dissipation and high throughput.

2. DISTRIBUTED ARITHMETIC BACKGROUND

Consider the following inner product of two L dimensional vectors C and x, where c is a constant vector, x is the input sample vector, and y is the result.

$$y = \sum_{k=0}^{L-1} \mathbf{c_k x_k} \tag{1}$$

Using B-bit 2's complement binary representation scaled such that $|x_k| \leq 1$ produces

$$\mathbf{x_k} = -\mathrm{b}_{k0} + \sum_{n=1}^{B-1} \mathrm{b}_{kn} 2^{-n} \tag{2}$$

where b_{kn} are the bits (0 or 1) of $\mathbf{x_k}$, b_{k0} is the most significant bit, and $\mathrm{b}_{k(B-1)}$ is the least significant bit. Substituting (2) into (1) yields

$$y = \sum_{k=1}^{L-1} \mathbf{c}_k \left[-\mathrm{b}_{k0} + \sum_{n=0}^{B-1} \mathrm{b}_{kn} 2^{-n} \right] \tag{3}$$

$$= -\sum_{k=1}^{L-1} \mathbf{c}_k \mathrm{b}_{k0} + \sum_{n=1}^{B-1} \left[\sum_{k=1}^{L-1} \mathbf{c}_k \mathrm{b}_{kn} \right] 2^{-n} \tag{4}$$

The computation in distributed arithmetic is represented by (4). The values of bkn are either 0 or 1, resulting in bracketed term in (4) having only 2B possible values. Since c a constant vector, the bracketed term can be recomputed and stored in memory using either lookup table (LUT) or ROM. The lookup table is then addressed using the individual bits of input samples, Xk with the final result y computed after B cycles, regardless of lengths of vectors c and x. A comprehensive tutorial review of OA linear filters is given in [9].

3. THE EXISTING FAST BLOCK LMS (FBLMS) BASED ADPATIVE FILTER

Consider a BLMS based adaptive filter, that takes an input sequence x(n), which is partitioned into non-overlapping blocks of length P each by means of a serial-to-parallel converter, and the blocks of data so produced are applied to an FIR filter of length L, one block at a time. The tap weights of the filter are updated after the collection of each block of data samples, so that the adaptation of the filter proceeds on block-by-block basis rather than on a sample-by-sample basis as in conventional LMS algorithm [1].
With the j-th block, (j E Z) consisting of x (j P + r) , r E Zp = 0,1, ... , P -1, the filter coefficients are updated from block to block as,

$$\mathbf{w}(j+1)=\mathbf{w}(j)+\mu\sum_{r=0}^{P-1}x(jP+r)e(jP+r) \qquad (5)$$

where w(j) = [WO(j)Wl(j).....WL_l(j)]t is the tap weight vector corresponding to the j-th block, x (jP+r) = [x (jP+r) x(jP+r-1) ....x(jP+r - L+l)]t and e(jP+r) is the output error at n= jP+r , given by,

$$e(jP+r)=d(jP+r)-y(jP+r) \qquad (6)$$

The sequence d (j P + r) is the so-called desired response available during the initial training period and y (jP + r) is the filter output at n = j P + r , given as,

$$y(jP+r)=\mathbf{w}^t(j)\mathbf{x}(jP+r) \qquad (7)$$

The parameter μ, popularly called the step size parameter is to be chosen as 0≤μ≤ $\frac{2}{\|P\lambda_r\|}$ for convergence of the algorithm. For the lth sub-bloc within the i -th block, O ≤ l ≤K -1 i.e., forn = jP+r,r = 0,1, ... P-1,j = iK +l, the filter output y (n) = wt (j) X (n) is obtained by convolving the input data sequence x(n) with the filter coefficient vector wt (j) and thus can be realized efficiently by the overlap-save method via M = L + P -1 point FFT, where the first L -1 points come from the previous sub-block, for which the output is to be discarded. Similarly, the weight update term in (5) above, viz., be obtained by the usual circular correlation technique, by employing M point FFT and setting the last P -1 output terms as zero.

4. PROPOSED IMPLEMENTATION

The throughput of FBLMS based adaptive filter is limited by computational complexity lies in FFT (and IFFT) block. It is possible to enhance the throughput of system by implantation of that FFT (and IFFT) block with reduced hardware complexity. OA is one of the efficient techniques, in which, by means of a bit level rearrangement of a multiply accumulate terms; FFT can be implemented without multiplier. Since the main hardware complexity of the system is due to hardware multipliers and introduction of OA eliminates the need of that multipliers and resulting system will have high throughput and also have low power dissipation. There are many fast Fourier transform (FFT) algorithm like radix-2, Cooley-Tukey, Winograd, Good-Thomas, Rader etc. But using OA, FFT can be efficiently calculated by jointly employing the Good-Thomas and Rader algorithms. Good-Thomas algorithm re-expresses the discrete Fourier transform (OFT) of a size N = NIN2 as a two-dimensional NIN2 OFT, but only for the case where Nl and N2 are relatively prime. It is easily seen from the definition of the OFT that the transform of a length N real sequence x(n) has conjugate symmetry, i.e. X(N -K) = X*(K). This property facilitates to compute only half of the transform, as the remaining half is redundant and need not be calculated. Rader algorithm provides straightforward way to compute only half of the conjugate symmetric outputs without calculating the others, which is not possible with other algorithms like radix, Cooley-Thkey and Winograd. Algorithm presented here first decomposes the one dimensional OFT into a multidimensional OFT using the index map proposed by Good [10]. Next, a method which is based on the index permutation proposed by Rader [11] is used to convert the short OFTs into convolution. This method changes a prime length N OFT of real data into two convolutions of length (N- 1)/2. One convolution is cyclic and the other is cyclic or skew-cyclic. The index mapping suggest by Good and Thomas for n is

$$n=N_2n_1+N_1n_2 \bmod N\begin{cases}0\le n_1\le N_1-1\\0\le n_2\le N_2-1\end{cases} \qquad (8)$$

and as index mapping for k results

$$k=N_2\langle N_2^{-1}\rangle_{N_1}k_1+N_1\langle N_1^{-1}\rangle_{N_2}k_2 \bmod N\begin{cases}0\le k_1\le N_1-1\\0\le k_2\le N_2-1\end{cases} \qquad (9)$$

If we substitute the Good-Thomas index map in the equation for OFT matrix it follows

$$X[k_1,k_2]=\sum_{n_2=0}^{N_2-1}W_{N_2}^{n_2k_2}\left(\underbrace{\sum_{n_1=0}^{N_1-1}x[n_1,n_2]W_{N_1}^{n_1k_1}}_{N_1-\text{point, transform, } x[n_2,k_1]}\right) \qquad (10)$$

$$=\underbrace{\sum_{n_2=0}^{N_2-1}x[n_2,k_1]W_{N_2}^{n_2k_2}}_{N_2-\text{point, transform}} \qquad (11)$$

Steps for Good-Thomas FFT Algorithm
An N = N1N2-point Off can be computed according to following steps:
1) Index transform of input sequence, according to (8).
2) Computation of N2 OFfs of length Nl using Rader algorithm.

3) Computation of Nl OFfs of length N2 using Rader algorithm.
4) Index transform of input sequence, according to (9).

Consider the length N= 14, suppose we have Nl=7 and N2=2 then mapping for the input index according to n. 2nl+7n2 mod 14 and k = 4k l + 7k2 mod 14 for output index results and using these index transforms we can construct the signal flow graph as shown in Fig. 1.

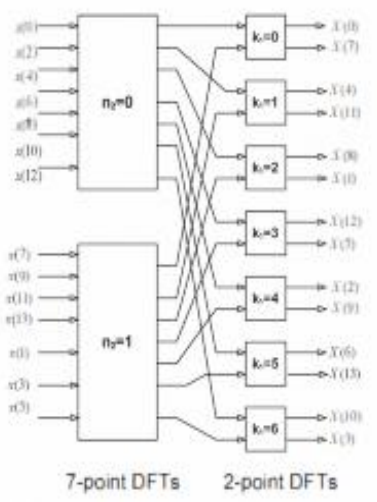

Fig. 1. Mapping using Good-Thomas algorithm.

From Fig. 1 we realize that first stage has 2 OFfs each having 7 -points and second stage has 7 OFfs each having of length 2. One of the interesting thing here is multiplication with twiddle factors between the stages is not required. Now consider Nl = 7 if the data are real we need to calculate only half of the transform. Also, as Rader showed the zero frequency term must be calculated separately.

In matrix t form, we write

$$\begin{bmatrix} X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 6 & 1 & 3 & 5 \\ 3 & 6 & 2 & 5 & 1 & 4 \end{bmatrix} \begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \end{bmatrix} + \begin{bmatrix} x(0) \\ x(0) \\ x(0) \end{bmatrix} \quad (12)$$

replacing W^k by $W^{(N-k)*}$

$$\begin{bmatrix} X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 3^* & 2^* & 1^* \\ 2 & 3^* & 1^* & 1 & 3 & 2^* \\ 3 & 1^* & 2 & 2^* & 1 & 3^* \end{bmatrix} \begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \end{bmatrix} + \begin{bmatrix} x(0) \\ x(0) \\ x(0) \end{bmatrix} \quad (13)$$

If real and imaginary parts of W matrix in (13) are separated, a simplification is possible. Consider first the real part using notation in matrix that k stands for cos(27rk/7). The real part of (13) becomes

$$\begin{bmatrix} X_R(1) \\ X_R(2) \\ X_R(3) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix} \begin{bmatrix} x(1)+x(6) \\ x(2)+x(5) \\ x(3)+x(4) \end{bmatrix} + \begin{bmatrix} x(0) \\ x(0) \\ x(0) \end{bmatrix} \quad (14)$$

Using the notation of k for sin(27rk/7) gives, for the imaginary part of (13).

$$\begin{bmatrix} X_I(1) \\ X_I(2) \\ -X_I(3) \end{bmatrix} = \begin{bmatrix} -1 & -2 & 3 \\ -2 & 3 & -1 \\ 3 & -1 & -2 \end{bmatrix} \begin{bmatrix} x(1)-x(6) \\ x(2)-x(5) \\ x(3)-x(4) \end{bmatrix} \quad (15)$$

The (14) and (15) are cyclic convolution relation. Since in our problem we always convolve with the same coefficients (In case of Off it is twiddle factor matrix), arithmetic efficiency can be improved by pre calculating some of the intermediate results. These are stored in table in memory and simply addressed as needed. Using distributed arithmetic this can be implemented efficiently and detail can be found in [12]. Here we will present only the structure Fig. 2 best suited to the Off calculated by cyclic convolution.
Initially Rl to R7 are cleared to zero and the Xi'S are loaded into registers Rl to R3 after addition. Then all Rl to R3 are shifted by one bit, the last bit of each register is in R4. The ROM output will be added to R5. Circular shift of R4 produces at ROM output are added to R5 to R7. when the first cycle is completed content of all Rl to R7 except R4 are right shifted by one bit and the second cycle starts. At Bth cycle function at AD D _SU B changed from adder to subtraction and after this Bth cycle the content of R5 to R7 gives final FFT coefficient and zero frequency component can be calculated separately applying accumulate and addition of Xi'

as shown in Fig. 2 and the FFT block so obtained has no multipliers at the expanse of increased adder requirement and memory requirement in order to store some pre calculated values.

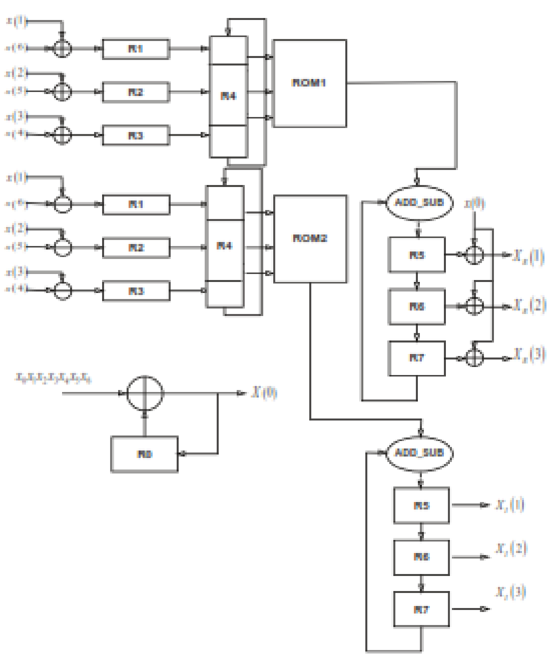

Fig. 2. Architracture for FFT using DA.

In the proposed architecture, using the DA based FFT (and IFFT) block that is described above, we recast the existing FBLMS based adaptive filter. Since, the DA based FFT block provides only half of the conjugate symmetric outputs without calculating others (reaming can calculated by conjugating them) and in DA based IFFT block we require to feed only half of the conjugate symmetric coefficients not all, which is not possible in existing FBLMS based adaptive filters since here radix-2 based FFT (and IFFT) blocks are employed and in radix-2 there is no such facility to calculate only half of the conjugate symmetric outputs hence, in this case half of the processed data are redundant. Since in our proposed FBLMS algorithm based adaptive filter, we require to calculate only half of the conjugate symmetric coefficients, under that condition the hardware requirements for our proposed system is approximately half of that of existing one. In our proposed architecture shown in Fig. 3 the computation off frequency domain outputs requires 8(If + 1) or 8(Mt + 1) number of multiplication and required number of addition is 16N + 2. 5(N1 + N2) + 2, here total number of multiplications are much less than that of required number of multiplications in the existing FBLMS algorithm based adaptive filter (can be compared from Table I and Table II) at the expanse of increased memory and adder requirement, which drastically reduces the hardware complexity for higher order filters as shown in Fig. 4 and it results with a adaptive filter which has high throughput and low power dissipation with reduced area requirement.

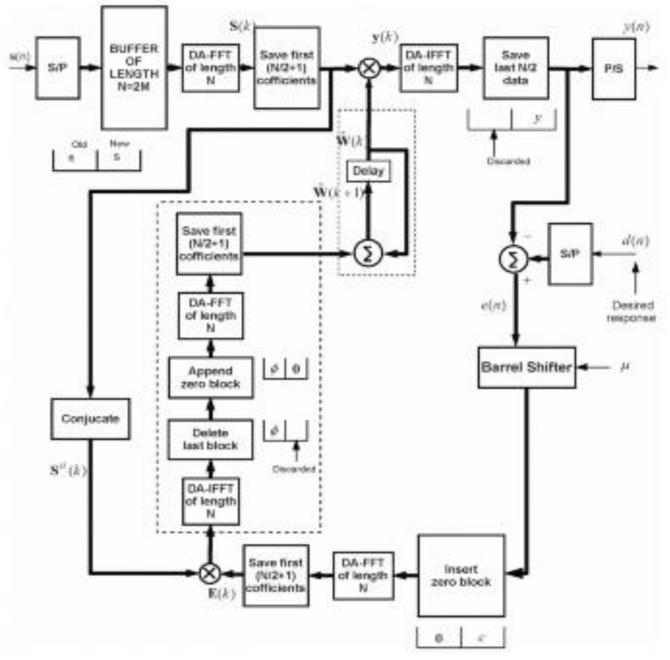

Fig. 3. Proposed DA based FBLMS after optimization.

Table I
COMPUTATIONAL COMPLEXITY IN EXISTING FBLMS ALGORITHM.

Multipliers	$10M\log_2 M + 26M$
Adders	$2N + 5N\log_2(N) + 1$

Table II
COMPUTATIONAL COMPLEXITY IN PROPOSED FBLMS ALGORITHM.

Multipliers	$8(\frac{N}{2}+1)$ or $8(M+1)$
Adders	$16N + 2.5(N_1 + N_2) + 2$

5. EXPERIMENTAL RESULTS

Proposed and existed architectures are implemented for filter length-7 and length-8 respectively. Verilog codes are written for both of these designs and synthesized using Xilnx 10.1 version. Family of device was Virtex II Pro and target device was 2vpl 00-ff1696-6.

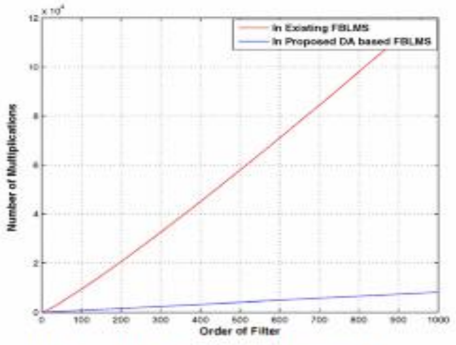

Fig. 4. Comparison of hardware complexity.

Fig. 5 shows the logic utilization of both the architectures. Although the filter length used in case of existed system is 1 more then that of our proposed one, but the FPGA resource utilization in case of existed system was approximately 45% greater than that of proposed one which can be clearly seen from Fig. 6. From Table III it is clear that our proposed architecture based adaptive filter is 30.2% faster than that of existed one.

Table III
TIMING COMPARISON OF PRESENTED ARCHITECTURES.

Design	Delay(ns)
Existing Architecture	9.61
Proposed Architecture	6.709

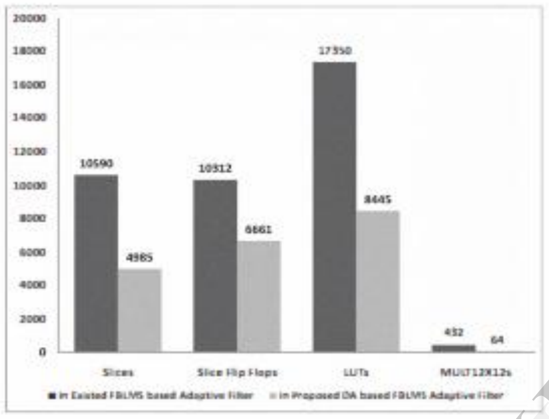

Fig. 5. Comparison of resources utilization in FPGA.

6. CONCLUSION

In this paper, we have proposed a new hardware-efficient adaptive filter structure for very high throughput FBLMS adaptive filters and its implementation details were presented.

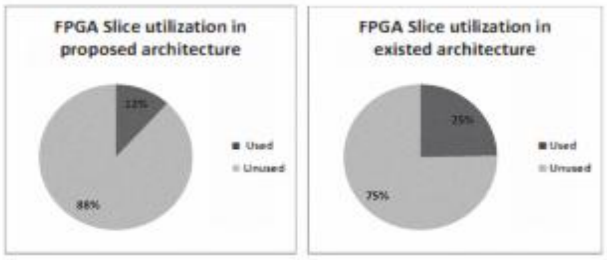

Fig. 6. Comparison of percentage slice utilization on targeted device.

The concept of DA involves for implementation of FFT block without any hardware multiplier using LUT and adders. Due to reduced hardware complexity the proposed DA based FBLMS adaptive filter is best suitable for implementation of higher order filters in FPGA efficiently with minimum area requirement, low power dissipation and high throughput.

REFERENCES

[I] S. Haykin, Adaptive Filter Theory, 4th ed., T. Kailath, Ed. Pearson Education, 2008.
[2] B. Farhang-Boroujeny, Adaptive Filters: Theory and Applications, Chichester, Ed. Wiley, 1998.
[3] S. K. M. Gregory A. Clark and S. R. Parker, "Block implementation of adaptive digital filters," IEEE Transactions on Circuits and Systems, vol. 28, pp. 584 - 592, 1981.
[4] A. Peled and B. Liu, "A new hardware realization of digital filters," IEEE Transactions On Acoustics, Speech, And Signal Processing, vol. 22, pp.45�6, December 1974.
[5] c. H. Wei and 1. 1. Lou, "Multi memory block structure for implementing a digital adaptive filter using distributed arithmetic," lEEE Proceedings, Electronic Circuits and Systems, vol. 133, February 1986.
[6] DJ Allred, W. Huang, Y.Krishnan, H. Yoo, D.V Anderson, "An FPGA implementation for a high throughput adaptive filter using distributed arithmetic." 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 324 - 325, 2004.
[7] Daniel 1. Allred, Walter Huang, Venkatesh Krishnan, Heejong Yoo, and David Y. Anderson, "LMS adaptive filters using distributed arithmetic for high throughput," IEEE Transactions on Circuits and Systems, vol. 52, pp. 1327 - 1337, July 2005.
[8] N. J. Sorawat Chivapreecha, Aungkana Jaruvarakul and K. Dejhan, "Adaptive equalization architecture using distributed arithmetic for partial response channels," IEEE Tenth International Symposium on Consumer Electronics, 2006.
[9] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," IEEE ASSP Magazine, July 1989.
[10] C. S. Burrus, "Index mappings for multidimensional formulation of the DFT and convolution," IEEE Transactions On Acoustics, Speech, And Signal Processing, vol. 25, pp. 239-242, June 1977.
[11] C. M. Rader, "Discrete fourier transforms when the number of datam samples is prime," IEEE Proceedings, vol. 56, pp. 1107-1108, June 1968.
[12] S. Chu and C. S. Burrus, "A prime factor FFT algorithm using distributed arithmetic," IEEE Transactions On Acoustics, Speech, And Signal Processing, vol. 30, April 1982.