Jisha M. Vijayan, Arathy Iyer / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 1, January -February 2013, pp.1671-1675

Area Efficient Fast Block LMS Adaptive Filter Using Distributed Arithmetic

Jisha M. Vijayan, Arathy Iyer
M.Tech VLSI (IV Sem) S.N.G.C.E., Kadayiruppu
Ernakulam, India
Asst. Professor, Dept. of ECE S.N.G.C.E., Kadayiruppu
Ernakulum, India

Abstract—

In FBLMS algorithm, the filter weights are adapted in the frequency domain by using the FFT algorithm. The main hardware complexity of the system is due to hardware multipliers in FFT/IFFT block. Introduction of Distributed Algorithm eliminates the need of that multipliers by a mechanism that generates partial products and then sums the products together and resulting system will have high area efficiency. Using DA, FFT can be efficiently calculated by jointly employing the Good-Thomas and Rader algorithms. In the proposed FBLMS algorithm using DA, it is required to calculate only half of the conjugate symmetric coefficients, under that condition the hardware requirements for the proposed system is approximately half of that of existing one.

***Keywords*— FBLMS, signed DA, Unsigned DA, Good-Thomas, Rader algorithm.**

I. INTRODUCTION

The term estimator or filter is commonly used to refer to a system that is designed to extract information about a prescribed quantity of interest from noisy data. At the core of DSP applications is the digital filter. Digital filters are generally used for: Separation of signals that have been combined, Restoration of signals that have been distorted, Transform operations. In fast block LMS algorithm the filter parameters are adapted in the frequency domain by using the FFT algorithm.

In this paper, a new DA based FBLMS is proposed to reduce area and power consumption. Distributed arithmetic (DA) is an efficient multiplication-free technique for calculating inner products. The multiplication operation is 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. In the proposed architecture, using the DA based FFT (and IFFT) block FFT can be efficiently calculated by jointly employing the Good-Thomas and Rader algorithms. DA based FFT block provides only half of the conjugate symmetric outputs without calculating others 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 in the 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.

II. DISTRIBUTED ARITHMETIC

Distributed arithmetic (DA) is first introduced by Croisier, and Zohar and further developed by Peled and Liu more than three decades ago. The DA is a direct method for sum of products operations, partial products can pre-compute by difference equation and storing in look-up-table (LUT) contained in memory, input signals are used for addressing.

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

$$y = \langle c, x \rangle = \sum_{n=0}^{N-1} c[n] \times x[n] \qquad (1)$$

$$= c[0]x[0] + c[1]x[1] + \ldots + c[N-1]x[N-1] \qquad (2)$$

An unsigned DA system assumes that the variable $x[n]$ is given by

$$x[n] = \sum_{b=0}^{B-1} x_b[n] \times 2^b \quad with\ x_b[n] \in [0,1] \qquad (3)$$

where $x_b[n]$ denotes the b^{th} bit of $x[n]$, i.e., the n^{th} sample of $\boldsymbol{x}$. The inner product y can, therefore, be represented as:

$$y = \sum_{n=0}^{N-1} c[n] \times \sum_{n=0}^{B-1} x_b[n] \times 2^b \qquad (4)$$

in more compact form

$$y = \sum_{b=0}^{B-1} 2^b \times \sum_{n=0}^{N-1} \underbrace{c[n] \times x_b[n]}_{f(c[n], x_b[n])} \quad (5)$$

$$y = \sum_{b=0}^{B-1} 2^b \times \sum_{n=0}^{N-1} f(c[n], x_b[n]) \quad (6)$$

In case of signed DA, we use modified equation in order to process a signed two's complement number. We use, therefore, the following $(B + 1)$-bit representation

$$x[n] = -2^B \times x_B[n] + \sum_{b=0}^{B-1} x_b[n] \times 2^b \quad (7)$$

the outcome y is defined by

$$y = -2^B \times f(c[n], x_B[n]) + \sum_{b=0}^{B-1} 2^b \times \sum_{n=0}^{N-1} f(c[n], x_b[n]) \quad (8)$$

III. PROPOSED SYSTEM

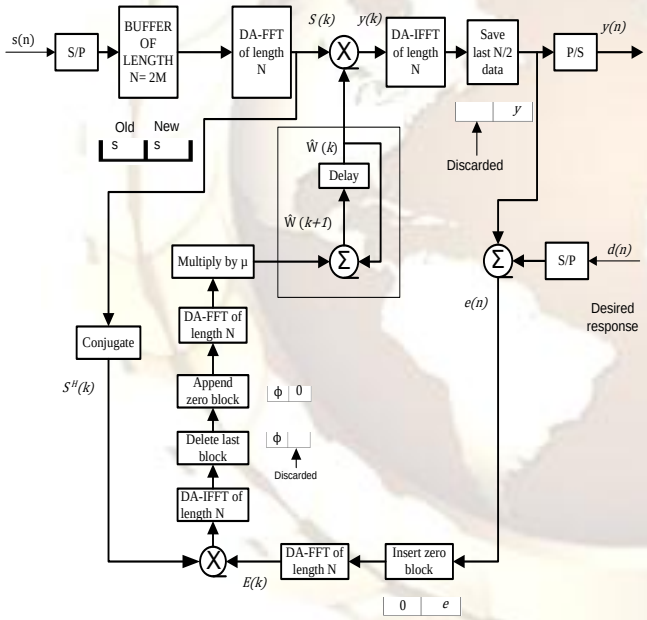

Figure 1. Proposed DA based FBLMS Adaptive filter

FFT and IFFT are the blocks that include large number of multipliers. Introduction of DA eliminates the need of that multipliers by a mechanism that generates partial products and then sums the products together and resulting system will have high area efficiency.

IV. INTRODUCTION TO FFT USING DA

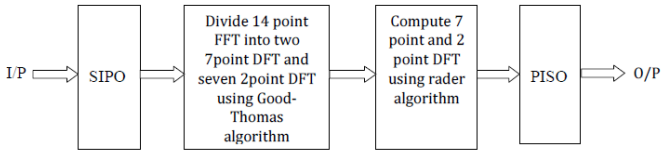

Figure 2. FFT Blocks

Using DA, FFT can be efficiently calculated by jointly employing the Good-Thomas and Rader algorithms.

A. *Good-Thomas Index Mapping*

Good-Thomas algorithm re-expresses the discrete Fourier transform (DFT) of a size $N = N_1N_2$ as a two-dimensional N_1N_2 DFT, but only for the case where N_1 and N_2 are relatively prime.
The index mapping suggest by Good and Thomas for n is

$$n = N_2n_1 + N_1n_2 \bmod N \begin{cases} 0 \le n_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases} \quad (9)$$

and as index mapping for k results

$$k = N_2\langle N_2^{-1}\rangle_{N_1}k_1 + N_1\langle N_1^{-1}\rangle_{N_2}k_2 \bmod N \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le k_2 \le N_2 - 1 \end{cases} \quad (10)$$

A $N = N_1N_2$-point DFT can be computed according to following steps:

1) Index transform of input sequence, according to equation for n.
2) Computation of N_2 DFTs of length N_1 using Rader algorithm.
3) Computation of N_1 DFTs of length N_2 using Rader algorithm.
4) Index transform of input sequence, according to equation for k.

To find the 14 point FFT block, Fig. 3 shows a schematic of a block processing system.

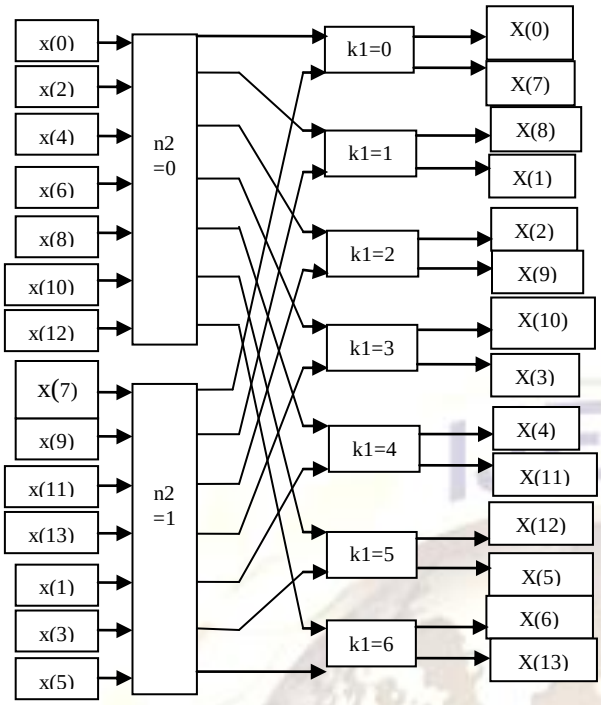

Figure 3. Mapping of 14 point FFT using Good-Thomas Algorithm

From Fig. 3, we realize that first stage has 2 DFTs each having 7-points and second stage has 7 DFTs each having of length 2. Here multiplication with twiddle factors between the stages is not required.

B. Rader Algorithm

It is easily seen from the definition of the DFT 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. Algorithm presented here first decomposes the one dimensional DFT into a multidimensional DFT using the index map proposed by Good. Next, a method which is based on the index permutation proposed by Rader is used to convert the short DFTs into convolution. This method changes a prime length N DFT of real data into two convolutions of length $(N-1)/2$.

Now consider $N_1 = 7$ 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.

$$X(0) = \sum_{n=0}^{N-1} x_n \qquad (11)$$

In matrix form, we write

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

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

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

If real and imaginary parts of W matrix are separated, a simplification is possible. Consider first the real part using notation in matrix that k stands for $cos(2\pi k/7)$. The real part becomes

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

Using the notation of k for $sin(2\pi k/7)$ gives, for the imaginary part

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

These are cyclic convolution relation. Since in our problem we always convolve with the same coefficients, arithmetic efficiency can be improved by precalculating 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.

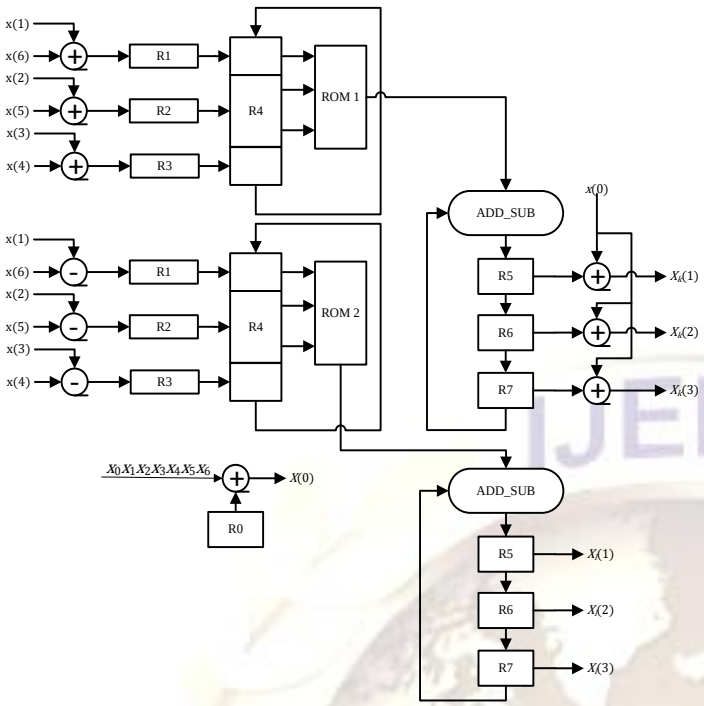

Figure 5. Architecture for FFT using DA.

V. EXPERIMENTAL RESULTS

Verilog codes are written for both FFT using DA and shift and add method and synthesized using Xilinx 13.2 version. Family of device was Spartan 3E and target device was XC3S500E. Fig. 6 shows the logic utilization of both the architectures and Table 1 shows Delay in ns. From these results, it is clear that clear that our proposed architecture based adaptive filter is faster than that used by shift and add method and Area utilization using DA is very much less than that by shift and add method. Thus the power utilization is also less in case of FBLMS using DA

Table 1. Delay Comparison

Design	Minimum Period in *ns*
16×16 DA Multiplier	35.281
16×16 SA Multiplier	36.296

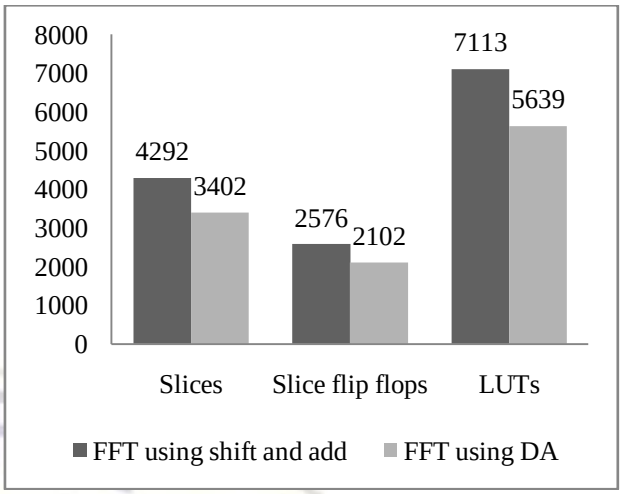

Figure 6. Comparison of resource utilization in FPGA

VI. CONCLUSION

In this paper, we have proposed a new area efficient FBLMS adaptive filter. FBLMS is the fastest and computationally efficient adaptive algorithm and FFT, the main computational block in FBLMS can be efficiently calculated by DA. 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.

REFERENCES

[1] S. Haykin, *Adaptive Filter Theory*, 4th ed., T. Kailath, Ed. Pearson Education, 2008.

[2] U Meyer Baese, *Digital Signal Processing With field Programmable Gate Arrays*,3rd ed.

[3] Sudhanshu Baghel , Rafiahamed Shaik, "FPGA Implementation of Fast Block LMS Adaptive Filter Using Distributed Arithmetic for High Throughput", p443-p447, 2011.

[4] Arman Chahardahcherik, Yousef S. Kavian, Otto Strobel, and Ridha Rejeb, "Implementing FFT Algorithms on FPGA", *IJCSNS International Journal of Computer Science and Network Security*, VOL.11 No.11, pp. 148 – 156, November 2011.

[5] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Magazine*, July 1989.

[6] 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.

[7] D.J. 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.

[8] 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.

[9] C. M. Rader, "Discrete fourier transforms when the number of data samples is prime," *IEEE Proceedings*, vol. 56, pp. 1107-1108, June 1968.