ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

High Throughput, Low Area, Low Power Distributed Arithmetic Formulation for Adaptive Filter

G.Selvapriya[1], M.Mano[2], K.RekhaSwathiSri[3], Mr S.Karthick[4]

PG Scholars, Department of ECE, Bannari Amman Institute of Technology, India [, 1, 2, 3]

Assistant Professor (Sr. G), Department of ECE, Bannari Amman Institute of Technology, India[4]

ABSTRACT--DA formulation employed for two separate blocks weight update block and filteringoperations requires larger area and is not suited for higher order filters therefore causes reduction in the throughput.These problems have been overcome by efficient distributed formulation of Adaptive filters. LMS adaptation performed on a sample-by-sample basis is replaced by a dynamic LUT update using a weight update scheme. Further, parallelLUT update and concurrent implementation of filtering and weight-update operations significantly increases throughput rate. Adder based shift accumulation for inner product computation replaced by conditional signed carry-save accumulation reduces the sampling period and area complexity. Fast bit clock reduction for carry-save accumulation reduces power consumption. It involves the same number of multiplexers, smaller LUT, and nearly half the number of adders compared to the previous DA-based design.

KEYWORDSs-Distributed arithmetic, Adaptive FIR, LMS, LUT, FPGA, MAC

I. INTRODUCTION

Adaptive Finite Impulse Response (AFIR) digital filters are extensively used due to their key role in various digital signal processing (DSP) and Communication applications for the virtues of providing linear phase, system stability and adaptability. AFIR weight updating is performed by a widely used Least Mean Square(LMS) algorithm due to its superior convergence performance and simple calculation [2]. The direct form configuration on the forward path of the FIRfilter results in a long critical path due to an inner-product computation to obtain a filter output. Therefore, when the input signal has a high sampling rate, it is necessary to reduce the critical path of the structure so that the critical path could not exceed the sampling period.Thepipeline implementation of LMS-based ADF [3] uses correction terms for updating the filter weights of the current iteration calculated from the error corresponding to a past iteration this briefs Delayed LMS (DLMS) algorithm.

For some applications of the adaptive finite impulse response (FIR) filtering, the adaptation algorithm can be implemented only with a delay in the coefficient update, DLMS can be transformed into LMS [4]. Multiplier less DA based technique [5] due to its high-throughput processing capability and regularity, result in cost-effective and area–time efficient computing structures. Hardware-efficient DA-based design of adaptive filter [6] uses two separate lookup tables (LUTs) for filtering and weight update, usesbit-serial operations and look-up tables (LUTs) to implement high throughput filters that use only about one cycle per bit of resolution regardless of filter length. It also uses an auxiliary LUT with special addressing; the efficiency and throughput of DA adaptive filters can be of the same order as fixed DA filters. [7], [8] have improved the design in [6] by using only one LUT for filtering as well as weight updating. P. K. Meher and S. Y. Park, [9] proposedhigh-throughput pipelined realization of adaptive FIR filter based on distributed arithmetic is discussed.

The adaptation delay of two cycles, however, does not make noticeable degradation of the convergence performance [9].Offset binary coding is popularly used to reduce the LUT size to half for area-efficient implementation of DA [4], [7], which can be applied to conventional design as well.However, the structures in [8], [9], [10] do not support high sampling rate since they involve several cycles for LUT updates for each new sample. Efficient architecture for high-speed DA-based adaptive filter with very low adaptation delay [9] can be applied to conventional work.A novel DA-based architecture for low- power, low-area, and high-throughput pipelined implementation of adaptive filter with very low adaptation [1] briefs various implementation approaches.

This brief is organized as follows. Section II deals with LMS Adaptive Algorithm. Section III presents the formulation of DA Adaptive filter. Analyzing of DA Structure and its complexity is concluded in Section IV. Further, extension of work is discussed in Section V.

II. LMS ADAPTIVE ALGORITHM

For every cycle, the LMS algorithm computes a filter output and an error value that is equal to the difference between the current filter output and the desired response. The estimated error is then used to update the filter weights in every training cycle. The weights of LMS adaptive filter during the nth iteration are updated according to the following equations:

$w(n+1) = w(n) + \mu \cdot e(n) \cdot x(n)$ (1)
$e(n) = d(n) - y(n)$ (2)

$y(n) = w^{qT}(n) \cdot x(n)$ (3)

The input vector x(n) and the weight vector w(n) at the nth training iteration are respectively given by

$x(n) = [x(n), x(n-1), \ldots, x(n-N+1)]^T$ (4)

$w(n) = [w0(n), w1(n), \ldots, w_{N-1}(n)]^T$ (5)

d(n) is the desired response, and y(n) is the filter output of the nth iteration. e(n) denotes the error computed during the nth iteration, used in updating the weights, μ is the convergence factor, and N is the filter length.

The feedback error e (n) becomes available after certain number of cycles called "adaptation delay" in case of pipelined designs. In case of pipelined architectures, use the delayed error e (n−m) for updating the current weight instead of the most recent error, where m is the adaptation delay. The weigh

t-update equation of such delayed LMS adaptive filter is given by

$w(n+1) = w(n) + \mu \cdot e(n-m) \cdot x(n-m)$ (6)

III. DA ADAPTIVE FILTER

A. DA FOR INNER PRODUCT COMPUTATION

The LMS adaptive filter, in each cycle, needs to perform an inner-product computation which contributes to the most of the critical path. For simplicity of presentation, let the inner product of (3) be given by

$y = \sum_{k=0}^{N-1} w_k\, x_k$ (7)

Where w_k and x_k for $0 \le k \le N-1$ form the N-point vectors corresponding the current weights and most recent N −1 input, respectively. Assuming L to be the bit width of the weight, each component of the weight vector may be expressed in two's complement representation

$w_k = -w_{k0} + \sum_{l=1}^{N-1} w_{kl} \cdot 2^{-l}$ (8)

where w_{kl} denotes the lth bit of w_k. Substituting (8), we can write (7) in an expanded form,

$y = -\sum_{k=0}^{N-1} x_k . w_{k0} + \sum_{k=0}^{N-1} x_k . [\sum_{l=1}^{N-1} w_{kl} . 2^{-l}]$ (9)

To convert the sum-of-products form of (7) into a distributed form, the order of summations over the indices k and l in (9) can be interchanged to have

$y = -\sum_{k=0}^{N-1} x_k . w_{k0} + \sum_{l=1}^{N-1} 2^{-l} . \sum_{k=0}^{N-1} x_k . w_{kl}$(10)
and the inner product given by (7) can be computed as

$y = (\sum_{l=1}^{L-1} 2^{-l} . y_l) - y_0,$

$y_l = \sum_{k=0}^{N-1} x_k . w_{kl}$(11)

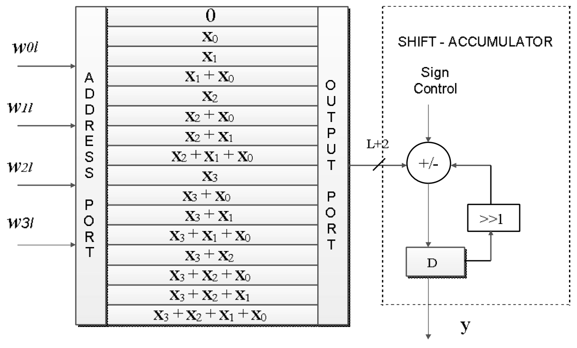


Figure.1 Conventional DA for 4-point inner product Structure

The partial sum y_l for l =0 ,1,...,L−1 can have 2N possible values since any element of the N-point bit sequence {w_{kl} for 0 ≤ k ≤ N −1} can either be zero or one. If all the 2N possible values are precomputed and stored in a LUT, the partial sums y_l can be read out from the LUT using the bit sequence{w_{kl}}as address bits for computing the inner product. Since the shift accumulation in Fig.1.involves significant critical path, the shift accumulation can be performed using carry-save accumulator, as shown in Fig. 2. The bit slices of vector w are fed one after the other in the least significant bit (LSB) to the most significant bit (MSB) order to the carry-save accumulator.

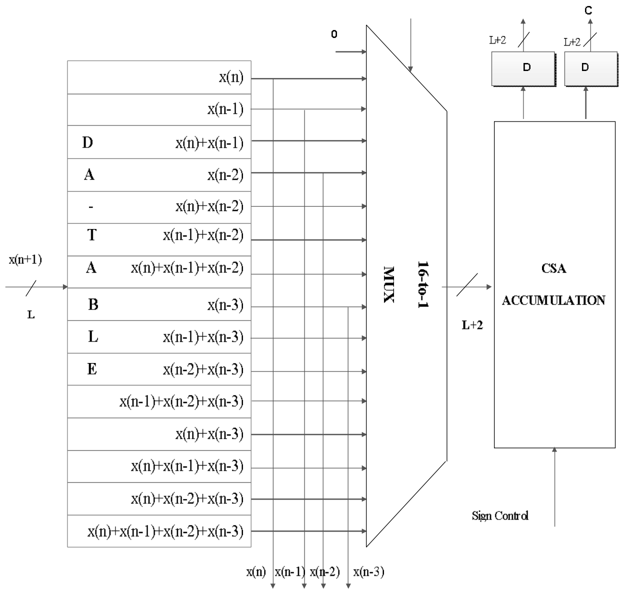


Figure.2 DA based 4-point inner product with CSA

However, the negative (two's complement) of the LUT output needs to be accumulated in case of MSB slices. Therefore, all the bits of LUT output are passed through XOR gates with a sign-control input which is set to one only when the MSB slice appears as address. The XOR gates thus produce the one's complement of the LUT output corresponding to the MSB slice but do not affect the output for other bit slices. Finally, the sum and carry words obtained after L clock cycles are required to be added by a final adder (not shown in the figure), and the input carry of the final adder is required to be set to one to account for the two's complement operation of the LUT output corresponding to the MSB slice. The content of the k_{th}LUT location can be expressed as

$$c_k = \sum_{j=0}^{N-1} x_j \cdot k_j \quad (12)$$

wherek_j is the (j + 1)th bit of N-bit binary representation of integer k for $0 \le k \le 2^N - 1$. Note that c_k for $0 \le k \le 2^N - 1$ can be precomputed and stored in RAM-based LUT of 2^N words. However, instead of storing 2^N words in LUT, we store $(2^N - 1)$ words in a DA table of $2^N - 1$ registers.

The four-point inner-product block includes a DA table consisting of an array of 14 registers which stores the partial inner products y_l for $0 < l \le 14$ and a 16:1 multiplexer (MUX) to select the content of one of those registers. Bit slices of weights A = {w3l w2l w1l w0l} for $0 \le l \le L-1$ are fed to the MUX as control in LSB-to-MSB order, and the output of the MUX is fed to the carry-save accumulator After L bit cycles, the carry-save accumulator shift accumulates all the partial inner products and generates a sum word and a carry word of size (L + 2)bit each.

B. DA FOR INNER ADAPTIVE FILTER STRUCTURE

The computation of adaptive filters of large orders needs to be decomposed into small adaptive filtering blocks since DA- based implementation of inner product of long vectors requires a very large LUT [4]. Therefore, here the DA-based structures of small order LMS adaptive filters are constructed for constructing higher order filters.

The inner-product computation of (7) can be decomposed into N/P (assuming that N = PQ) small adaptive filtering blocks1 of filter length P as

$$y = \sum_{k=0}^{P-1} w_k . x_k + \sum_{k=P}^{2P-1} w_k . x_k \,..+ \sum_{k=N-P}^{N-1} w_k . x_k \quad (13)$$

Each of these P-point inner-product computation blocks will update P weights accordingly by their respective weight-increment unit. The structure for N = 16 and P =4 is shown in Figure.3. It consists of four inner-product blocks of length P =4 , which is shown in Figure.2. The (L + 2)-bit sum and carry produced by the four blocks are added by two separate binary adder trees.

Four carry-in bits should be added to sum words which are output of four 4-point inner-product blocks. Since the carry words are of double the weight compared to the sum words, two carry-in bits are set as input carry at the first level binary adder tree of carry words, which is equivalent to inclusion of four carry-in bits to the sum words. Assuming thatμ =1/N, we truncate the four LSBs of e(n) for N = 16 to make the word length of sign-magnitude separator be L bit.

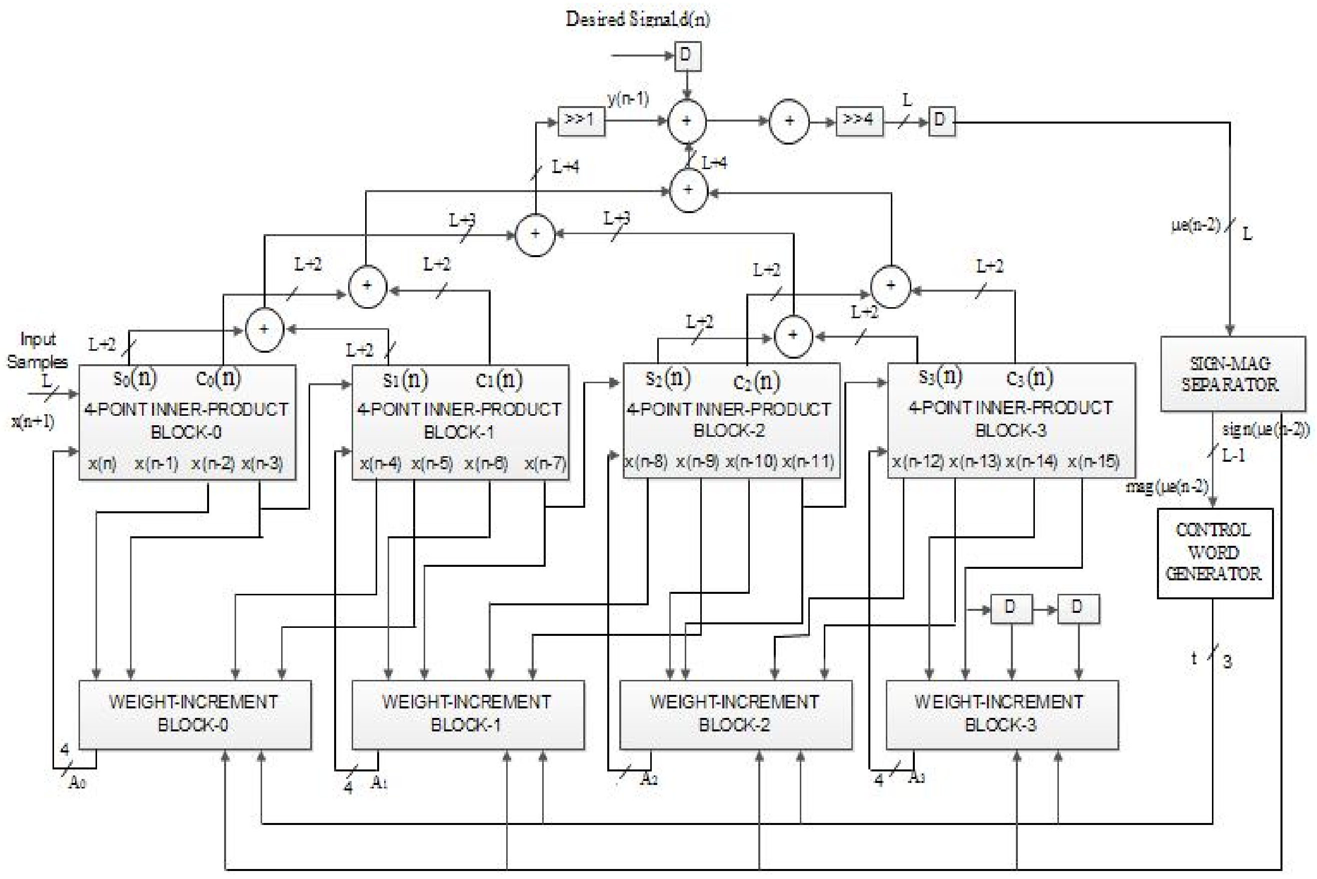


Figure.3 Structure of DA based LMS adaptive filter of length N=16

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

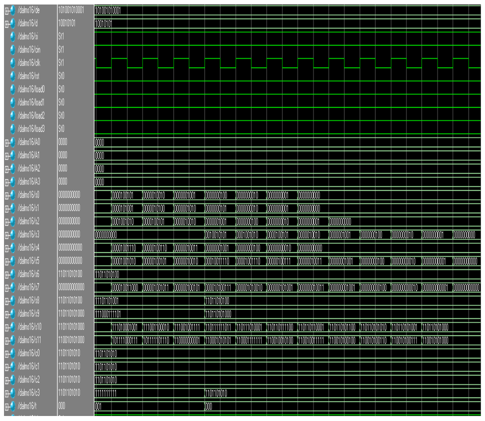
Organized by

Department Of CSE, JayaShriram Group Of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

The design uses two clocks, namely, the bit clock and the byte clock. The duration of the byte clock is the same as the sampling period. The bit clock is used in carry save accumulation units and word-parallel bit-serial converters, while the byte clock is used in the rest of the circuit. The duration of bit clock is given by $T_{BC} = 4_{TM} + T_{FA} + T_{XOR} + T_D$. The duration of the sample period (byte clock) of the proposed design is $L \times T_{BC,}$ as one output per cycle is obtained the throughput per unit time of design is found to be much higher.

IV. SIMULATION RESULT

DA LMS ADAPTIVE FILTER FOR N=16



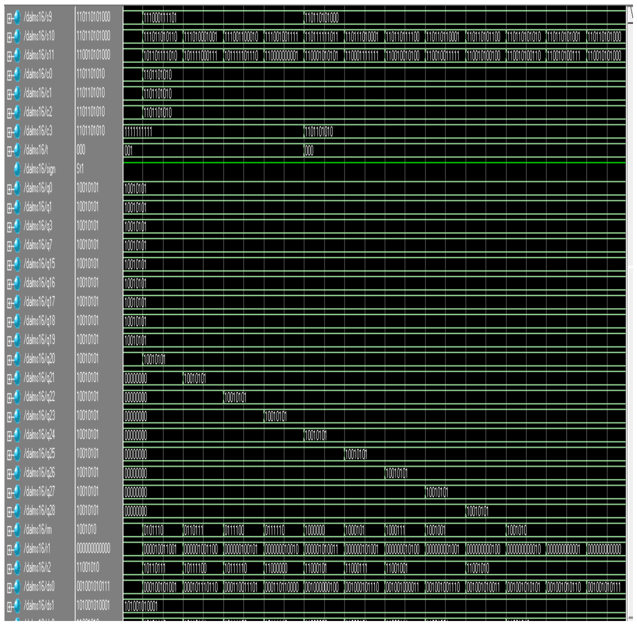


TABLE I SYNTHESIS REPORT FOR COMPARISON BETWEEN EXISTNG AND PROPOSED

Parameters	**Existing**	**Conventional**
Length	**16**	**16**
Throughput (per µs)	**77.39**	**300.5**
Power (mw)	**21.16**	**10.41**
Area (sq. µs)	**20347**	**17159**

V. CONCLUSION

It has been a rewarding experience in more than one way we have gained an insight while analyzing this design. The 16 bit structure of DA based adaptive filter is divided into many modules. The number system used here is signed value system this can also be implementable for more than 16 bit taps. But the input value to the filter must be 2's complement number format. DA filtering is done by serial architecture; latency occurs so, fix the clock properly and selects the sampling frequency. Throughput rate is significantly enhanced by parallel LUT update and concurrent processing of filtering operation and weight-update operation. A carry-save accumulation scheme of signed partial inner products for the computation of filter output is implemented. No auxiliary LUT with special addressing is required. We have coded the different modules of the design using Verilog and verified by ModelSim SE 5.7g and the power consumption, area, delay is analyzed using Xilinx software and its reports are shown in table I for analyzing the 16 bit DA based Adaptive filter.

VI. FUTURE WORK

Till now all the DA based adaptive filter was implemented in an ordinary look up table so the development here is constructing the look up table in efficient manner that is Distributed arithmetic by offset binary coding. In this off set binary coding, the look up table is exactly reduced by half from the actual look up table. So, this is somewhat area filter based on LUT, named as Adaptive DA Filter using Offset Binary Coding.

Conventional method can be improved with lower adaptation-delay and area-delay-power efficientimplementation, using novel partial product generator and strategy for optimizedbalanced pipelining across the time-consuming combinational blocks of the structure can also be implemented.

REFERENCES

[1] Sang Yoon Park and Pramod Kumar Meher, "Low-Power, High-Throughput, and Low-Area Adaptive FIR Filter Based on Distributed Arithmetic", IEEE Transactions on Circuits and Systems—II: vol. 60, no. 6, June 2013

[2] S. Haykin and B. Widrow, "Least-Mean-Square Adaptive Filters". Hoboken, NJ: Wiley-Interscience, 2003.

[3] 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, Apr. 1990, pp. 1273–1276.

[4] R. D. Poltmann, "Conversion of the delayed LMS algorithm into the LMS algorithm,"IEEE Signal Process. Lett., vol.2,p.223,Dec.1994.

[5] S. A. White, "Applications of the distributed arithmetic to digital signal processing: A tutorial review," IEEE ASSP Mag., vol. 6, no. 3, pp. 4–19, Jul. 1989.

[6] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, "LMS adaptive filters using distributed arithmetic for high throughput," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 42, no. 7, pp. 1327–1337, Jul. 2004.

[7] R. Guo and L. S. DeBrunner, "Two high-performance adaptive filter implementation schemes using distributed arithmetic," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 48, no. 9, pp. 600–604, Sep. 2011.

[8] R. Guo and L. S. DeBrunner, "A novel adaptive filter implementation scheme using distributed arithmetic," in Proc. Asilomar Conf. Signals, Syst., Comput., Nov. 2011, pp. 160–164.

[9] P. K. Meher and S. Y. Park, "High-throughput pipelined realization of adaptive FIR filter based on distributed arithmetic," in VLSI Symp. Tech. Dig., Oct. 2011, pp. 428–433.

[10] M. D. Meyer and P. Agrawal, "A modular pipelined implementation of a delayed LMS transversal adaptive filter," in Proc. IEEE Int. Symp. Circuits Syst., May 1990, pp. 1943–1946.

[11] Paulo S.R. Diniz , Adaptive filtering Algorithms and Practical Implementations., ISBN 978-1-4614-4105-2.