International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org Volume 3 Issue 4 ǁ April 2014 ǁ PP.15-20

An Efficient Adaptive Fir Filter Based On Distributed Arithmetic

[1,]M.Usha , [2,]R.Ramadoss
[1,] P.G.Scholar , [2,] Assistant Professor
[1,2],*Dept. of Electronics and communication Engineering Sri Muthukumaran Institute Of Technology Chennai,Tamilnadu,India*

ABSTRACT: *Adaptive filtering constitutes an important class of DSP algorithms employed in several hand held mobile devices for applications such as echo cancellation, signal de-noising, and channel equalization. The throughput of the proposed design is increased by parallel lookup table (LUT) update.The 16:1 multiplexer is replace by a 8:1 and 2:1 MUX. The conventional adder-based shift accumulation for DA-based inner-product computation is replaced by conditional signed carry-save accumulation in order to reduce the area complexity; the power consumption of the proposed design is reduced by using a fast bit clock for all operations. It involves the same number of multiplexors, smaller LUT, and nearly half the number of adders compared to the existing DA-based design. The proposed architecture is found to involve significantly 29% less area-13% less power and throughput compared with the existing DA-based implementations of FIR filter and a increase in operating frequency of 12MHZ is achieved.*

KEYWORDS: *Adaptive filter, circuit optimization, distributed arithmetic (DA), least mean square (LMS) algorithm*

I. INTRODUCTION

Adaptive filters are widely used in several digital signal processing (DSP) applications. The tap-delay line finite impulse response (FIR) filters whose weights are updated by the famous Widrow-Hoff least mean square (LMS) algorithm [1] is the most popularly used adaptive filter not only due to its simplicity but also due to its satisfactory convergence performance [2]. The direct form FIR filter configuration for the implementation of LMS adaptive filter results in either zero or lower adaptation-delay but involves a large critical-path due to an inner-product computation to obtain the filter output. Therefore, when the input signal has high sample-rate, the critical-path could exceed the sample period. In such cases, it is necessary to reduce the critical-path by pipelined implementation. Since the conventional LMS algorithm does not support pipelined implementation due to its recursive behavior, it is modified to a form called delayed LMS algorithm [3], which allows pipelined implementation of different sections of the adaptive filter. State-of-the-art hardware implementation of adaptive filters typically involves DSP microprocessors or custom logic design using one or more hardware multiply-accumulate (MAC) units. While an implementation employing DSP microprocessors provides easy programmability, the serial implementation on a single processing unit adversely affects the throughput of these filters. This is especially true for higher order filters. Custom logic design using one or more hardware MAC units may be used to parallelize the implementation and thus improve the throughput but at the cost of increased logic complexity, chip area usage, and power consumption [4].

Various types of DSP operations are employed in practice. Filtering is one of the most widely used signal processing operations [5]. For FIR filters, output y(n) is a linear convolution of weights w_n and inputs. Distributed arithmetic (DA) is one way to implement convolution without multiplier, where the MAC operations are replaced by a series of LUT access and summations. Techniques, such as ROM decomposition [6] and offset-binary coding (OBC) [7] can reduce the LUT size, which would otherwise increase exponentially with the filter lengthN+1for conventional DA. However, in many applications such as echo cancelation and system identification, coefficient adaptation is needed. This adaptation makes it challenging to implement DA-based adaptive filters with low cost due to the necessity of updating LUTs. Several approaches have been developed for DA-based adaptive filters, i.e., from the point of view of reducing logic complexity [8]. Recently, a DA-based FIR adaptive filter implementation scheme has been presented in [9], which uses extra "auxiliary" LUTs to help in the updating; however, memory usage is doubled. The rest of this paper is organized as follows. Section 2 describes the Review of LMS Adaptive Algorithms. Section 3 introduces the proposed architecture and Proposed DA based approach for inner product computation, Section 4 describes the simulation results for proposed system. Conclusions are finally drawn in Section 5.

II. EXISTING SYSTEM

A. Review of LMS Adaptive Algorithms

During each 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:

$$w(n+1) = w(n) + \mu \cdot e(n) \cdot x(n) \quad (1)$$

where

$$e(n) = d(n) - y(n) \quad (2)$$
$$y(n) = w\ qT(n) \cdot x(n). \quad (3)$$

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

$$x(n = [x(n),x(n-1),...,x(n-N+1)]T \quad (4)$$
$$w(n)= [w0(n),w1(n),...,wN-1(n)]T. \quad (5)$$

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, which is used to update the weights, μ is the convergence factor, and N is the filter length.In the case of pipelined designs, the feedback error e(n) becomes available after certain number of cycles, called the "adaptation delay." The pipelined architectures therefore use the delayed error e(n−m) for updating the current weight instead of the most recent error, where m is the adaptation de-lay. The weight-update equation of such delayed LMS adaptive filter is given by

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

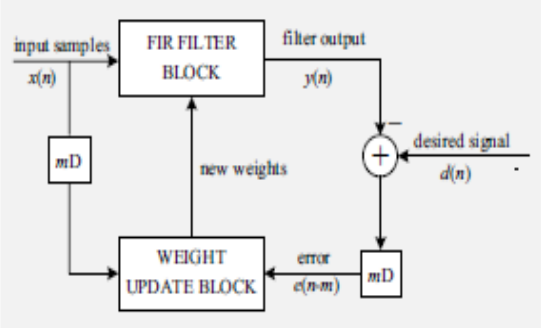

Fig. 1. Existing System Block Diagram

III. PROPOSED SYSTEM

A. Proposed DA based Adaptive Filter

The proposed structure of DA-based adaptive filter of length N=4 is shown in Fig. 2. It consists of a four-point inner-product block and a weight-increment block along with additional circuits for the computation of error value e(n) and control word t for the barrel shifters. The four-point inner-product block Fig. 3 includes a DA table consisting of an array of 15 registers which stores the partial inner products y_l for $0 < l \leq 15$and a 16 : 1 multiplexor (MUX) to select the content of one of those registers. Bit slices of weights A={w3l w2l w1l w0l} for $0 \leq l \leq L-1$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 (shown in Fig. 2). 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. The carry and sum words are shifted added with an input carry "1" to generate filter output which is subsequently subtracted from the desired output d(n)to obtain the error e(n). The magnitude of the computed error is decoded to generate the control word t for the barrel shifter. The logic used for the generation of control word t to be used for the barrel shifter. The convergence factor μ is usually taken to be O(1/N). We have taken μ= 1/N. However, one can take μ as $2^{-i}/N$, where i is a small integer. The number of shifts t in that case is increased by i, and the input to the barrel shifters is pre shifted by i locations accordingly to reduce the hardware complexity.

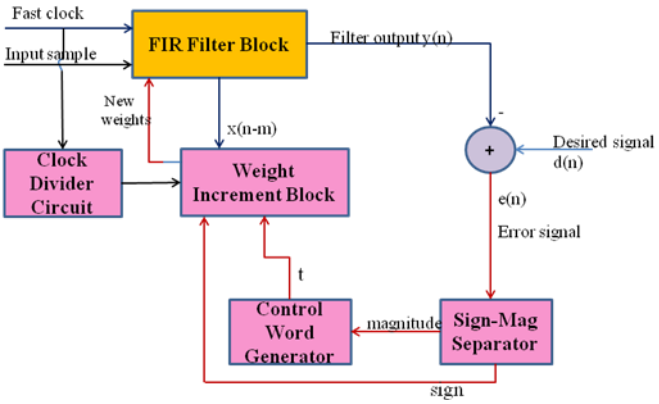

Fig. 2. Proposed System Structure

The weight-increment unit consists of four barrel shifters and four adder/subtractor cells. The barrel shifter shifts the different input values x_k for k= 0,1,...,N−1 by appropriate number of locations (deter-mined by the location of the most significant one in the estimated error). The barrel shifter yields the desired increments to be added with or subtracted from the current weights. The sign bit of the error is used as the control for adder/subtractor cells such that, when sign bit is zero or one, the barrel-shifter output is respectively added with or subtracted from the content of the corresponding current value in the weight register.

B. Proposed DA based approach 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

$$Y = \sum_{K=0}^{N-1} W_K \, . X_K \qquad (7)$$

Where Wk and X_K for $0 \leq k \leq N-1$ 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

$$Y = -w_{k0} + \sum_{l=1}^{L-1} W_{Kl} \, . 2^{-l} \qquad (8)$$

Where W_{kl} denotes the l^{th} bit of W_k. Substituting (8), we can write (7) in an expanded form

$$Y = -\sum_{K=0}^{N-1} x_k \, . w_{k0} + \sum_{K=0}^{N-1} x_k \sum_{l=1}^{L-1} W_{Kl} \, . 2^{-l} \qquad (9)$$

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

$$Y = -\sum_{K=0}^{N-1} x_k \, . w_{k0} + \sum_{K=0}^{N-1} 2^{-l} \sum_{l=1}^{L-1} W_{Kl} \, . x_k \qquad (10)$$

and the inner product given by (10) can be computed as
where

$$y_l = \sum_{l=1}^{L-1} W_{Kl} \, . x_k \qquad (11)$$

Since any element of the N-point bit sequence $\{w_{kl}$ for $0\leq k\leq N-1\}$can either be zero or one, the partial sum y_l for l=0,1,...,L−1can have 2^N possible values. If all the 2^N possible values of yl are precomputed and stored in a LUT, the partial sums yl can be read out from the LUT using the bit sequence{wkl}as address bits for computing the inner product.The inner product can be calculated in L cycles of shift accumulation, followed by LUT-read operations corresponding to L number of bit slices$\{w_{kl}\}$for $0\leq l\leq L-1$.The bit slices of vector ware fed one after the next in the least significant bit (LSB) to the most significant bit (MSB) order to the carry-save accumulator. 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 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

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

where k_j is the $(j+1)^{th}$ bit of k^{th} LUT location can be expressed as N-bit binary representation of integer k for $0\leq k\leq 2^N-1$. Note that ck for $0\leq k\leq 2^N-1$ can be pre computed and stored in RAM-based LUT of 2^N words.

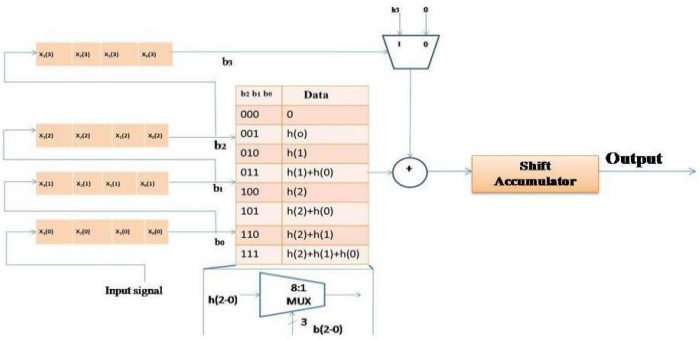

Fig. 3. . DA table for generation of possible sums of input samples.

However, instead of storing 2^N words in LUT, an example of such a DA table for N=4 is shown in Fig. 4. It contains only 15 registers to store the as address. The XOR gates thus produce the one's pre computed sums of input words. Seven new values of ck are computed by seven adders in parallel.

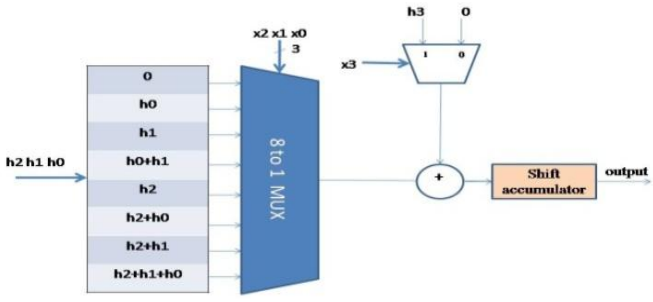

Fig. 4 Internal blocks of inner product computation block.

IV. RESULTS AND DISCUSSION

The proposed System was executed on Windows XP operating system at an operating frequency of 2.0GHz using Xillinx simu lator.

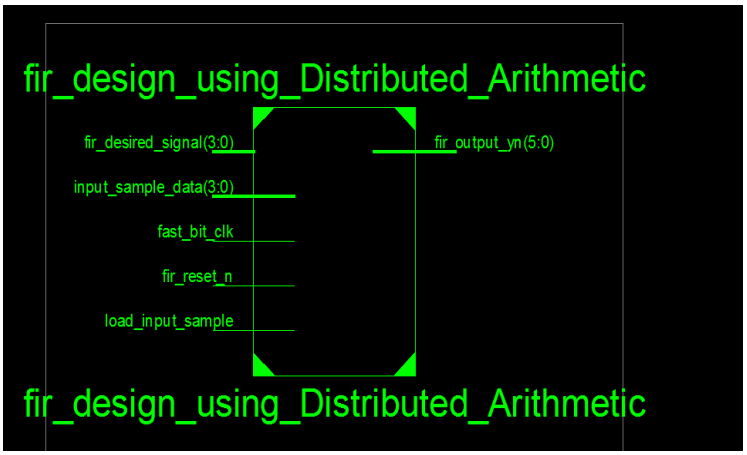

Fig. 5. Proposed method pin diagram without slow clock

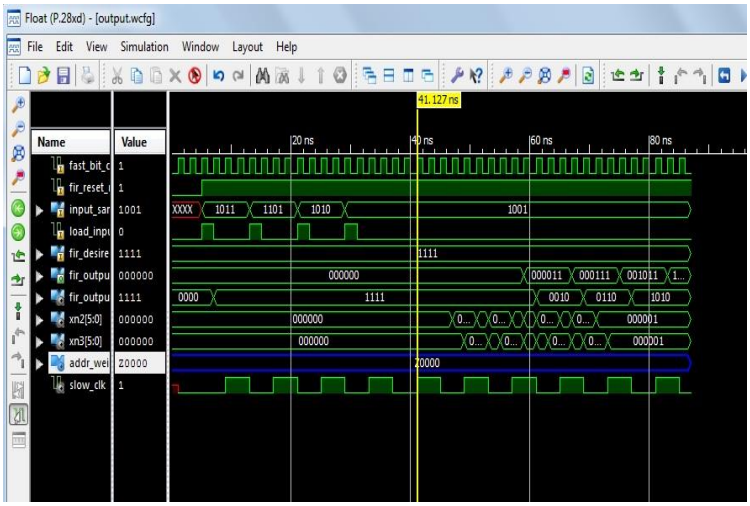

Fig. 6. Filter output wave form

Its seen from the synthesis result that the frequency of the proposed system is increased by 12 MHZ.

V. CONCLUSION

We have suggested an efficient pipelined architecture for low-power, high-throughput, and low-area implementation of DA-based adaptive filter. Throughput rate is significantly enhanced by parallel LUT update and concurrent processing of filtering operation and weight-update operation. We have also proposed a carry-save accumulation scheme of signed partial inner products for the computation of filter output. From the synthesis results, we find that the proposed design consumes 13% less power and 29% less ADP over our previous DA-based FIR adaptive filter in average for filter lengths N=16 Compared to the best of other existing designs, our proposed architecture provides 9.5 times less power and 4.6 times less ADP. Offset binary coding is popularly used to reduce the LUT size to half for area-efficient implementation of DA which can be applied to our design as well.

REFERENCES

[1] B. Widrow and S. D. Stearns, Adaptive signal processing. Prentice Hall, Englewood Cliffs, NJ, 1985.

[2] S. Haykin and B. Widrow, Least-mean-square adaptive filters. Wiley-Interscience, Hoboken, NJ, 2003.

[3] M. Meyer and P. Agrawal, "A modular pipelined implementation of a delayed LMS transversal adaptive filter," in Proc. IEEE International Symposium on Circuits and Systems, ISCAS '09, May 1990, pp. 1943–1946.

[4] D. Allred, V. Krishnan, W. Huang, and D. Anderson, "Implementation of an LMS adaptive filter on an FPGA employing multiplexed multiplier architecture," in Proc. Asilomar Conf. Signals Systems Computers, Nov. 2003, pp.918-921.

[5] S. K. Mitra, Digital Signal Processing: A Computer-Based Approach, 2nd. New York: McGraw-Hill, 2001.

[6] K.K.Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. Hoboken, NJ: Wiley, 1999.

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

[8] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, "An FPGA implementation for a high throughput adaptive filter using distributed arithmetic," in Proc. 12th Annu. IEEE Symp. Field-Programmable Custom Comput. Mach., 2004, pp. 324–325

[9] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, "A novel high performance distributed arithmetic adaptive filter implementation on an FPGA," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., 2004, vol. 5, pp. V-161–V-164