

Implementation of Fixed-Point LMS Adaptive Filter with Efficient Area-Power-Delay

D. Senthilraja¹, K. Kalaiselvi²

¹Assistant Professor/ ECE, Excel Engineering College, Komarapalayam, Tamil Nadu 637303, India

²PG scholar (M.E., Applied Electronics), Excel Engineering College, Komarapalayam, Tamil Nadu 637303, India

Abstract: In this paper, we present an efficient architecture for the implementation of a delayed least mean square Adaptive filter. For achieving lower adaptation-delay and area-delay-power, we use a novel partial product generator and propose an optimized balanced pipelining across the time-consuming combinational blocks of the structure. We propose an efficient fixed-point implementation scheme in the proposed architecture. We present here the optimization of design to reduce the number of pipeline delays along with the area, sampling period, and energy consumption. The proposed design is found to be more efficient in terms of the power-delay product (PDP) and energy-delay product (EDP) compared to the existing structures.

Keywords: Least Mean Square (LMS) algorithms, Partial Product Generator, Weight updateblock, Modified DLMS adaptive filter.

1. Introduction

The filter is an important component in the communication world. It can eliminate unwanted signals from useful information. However, to obtain an optimal filtering performance, it requires 'a priori' knowledge of both the signal and its embedded noise statistical information. The classical approach to this problem is to design frequency selective filters, which approximate the frequency band of the signal of interest and reject those signals outside this frequency band. The removal of unwanted signals through the use of optimization theory is becoming popular, particularly in the area of adaptive filtering. These filters minimize the mean square of the error signal, which is the difference between the reference signal and the estimated filter output, by removing unwanted signals according to statistical parameters. The Least Mean Square (LMS) adaptive filter is the widely used filter because of its simplicity and performance. The Least Mean Square adaptive filter is used here because it differs from a traditional digital filter in many ways. A traditional digital filter has only one input signal $x(n)$ and one output signal $y(n)$. An adaptive filter requires an additional input signal $d(n)$ and returns an additional output signal $e(n)$. The filter coefficients of a traditional digital filter do not change over time. The coefficients of an adaptive filter change over time. Therefore, adaptive filters have a self-learning ability that traditional digital filters do not have.

Least mean squares (LMS) algorithms are a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean squares of the error signal (difference between the desired and the actual signal). It is stochastic gradient method in that the filter is only adapted based on the error at the current time. The LMS algorithm is the most popular method for adapting a filter, which have made it widely adopted in many applications. Applications include adaptive channel equalization, adaptive predictive speech coding, Noise Suppression and on-line system identification. Recently, because of the progress of digital signal processors, a variety of selective coefficient update of gradient-based adaptive

algorithms could be implemented in practice. The DLMS adaptive algorithm is introduced to achieve lower adaptation-delay. It can be implemented using pipelining. But it can be used only for large order adaptive filters. Typical DSP Programs with highly real-time, design hardware and or software to meet the application speed constraint. It also deals with 3-Dimensional Optimization (Area, Speed, and Power) to achieve required speed, area-power tradeoffs and power consumption. An efficient scheme is presented for implementing the LMS-based transversal adaptive filter in block floating-point (BFP) format, which permits processing of data over a wide dynamic range, at temporal and hardware complexities significantly less than that of a floating-point processor.

2. Delayed LMS Adaptive Filter

For every input sample, the LMS algorithm calculates the filter output and finds the difference between the computed output and the desired response. Using this difference the filter weights are updated in every cycle. During the n^{th} iteration, LMS algorithm updates the weights as follows:

$$W_{n+1} = W_n + \mu \cdot e(n) \cdot x(n) \quad (1)$$

Where,
 μ is the convergence-factor.
 $e(n) = d(n) - y(n)$

$$y(n) = w^T(n) \cdot x(n) \quad (2)$$

Here,
 $x(n)$ is the input vector, $d(n)$ is the desired response, and $y(n)$ is the filter output of the n^{th} iteration, $w(n)$ is the weight vector of an N^{th} order LMS adaptive filter at the n^{th} iteration, respectively, given by,

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

$$w_n = [w_n(0), w_n(1), \dots, w_n(N-1)]^T$$

$e(n)$ denotes the error computed in the n^{th} iteration which is used to update the weights.

The DLMS algorithm uses the delayed error $e(m)$, (i.e.) the error corresponding to $(n-m)^{\text{th}}$ iteration for updating the current weight. The weight-update equation of DLMS algorithm is given by,

$$W_{n+1} = W_n + \mu \cdot e(n-m) \cdot x(n-m) \quad (3)$$

where,

m is the adaptation-delay.

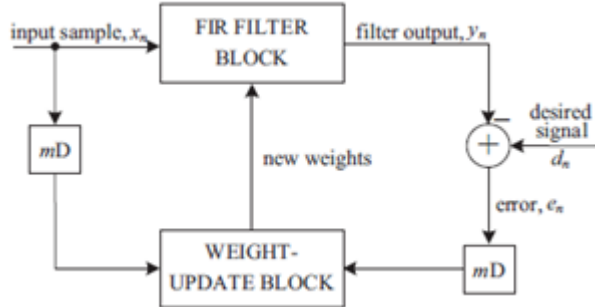


Figure 1: Structure of conventional delayed LMS adaptive filter

The structure of conventional delayed LMS adaptive filter is shown in Fig1. It can be seen that the adaptation-delay ' m ' is the number of cycles required for the error corresponding to any given sampling instant to become available to the weight adaptation circuit.

3. Proposed System

In the conventional DLMS algorithm, the adaptation delay of ' m ' cycles amounts to the delay introduced by the whole of adaptive filter structure consisting of FIR filtering and weight adaptation process. But instead, this adaptation delay could be decomposed into two parts. One part is the delay introduced due to the FIR filtering and the other part is due to the delay involved in weight adaptation.

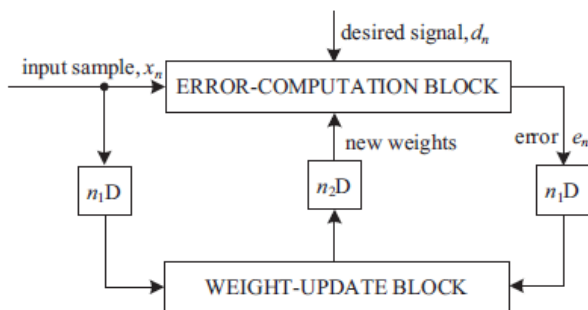


Figure 2: Structure of modified DLMS adaptive filter

The proposed adaptive filter architecture, shown in Fig.2, consists of two main computing blocks, namely the error computation block and weight-update block. The computation of filter output and the final subtraction to compute the feedback error are merged in the error computation unit to reduce the latency of error computation path.

A. Error-Computation Block

The proposed structure for error-computation unit of an N -tap DLMS adaptive filter is shown in Fig. 3. It consists of N number of 2-bit partial product generators (PPG) corresponding to N multipliers and a cluster of $L/2$ binary adder trees, followed by a single shift-add tree. Each sub block is described in detail.

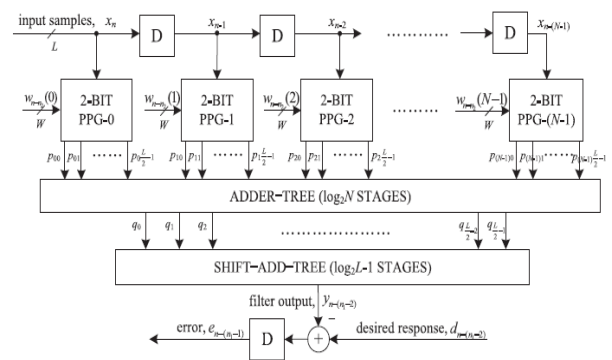


Figure 3: Structure of error-computation block.

1) Structure of PPG: The structure of each partial product generator (PPG) is shown in Fig.4. It consists of $L/2$ number of 2-to-3 decoders and the equal number of AND/OR cells (AOC). Each of the 2-to-3 decoders takes a 2-bit digit (u_1u_0) as input and produces three outputs $b_0 = u_0$, $b_1 = u_0 \cdot u_1$, and $b_2 = u_0 \cdot u_1$, such that $b_0 = 1$ for (u_1u_0) = 1, $b_1 = 1$ for (u_1u_0) = 2, and $b_2 = 1$ for (u_1u_0) = 3. The decoder output b_0 , b_1 and b_2 along with w , $2w$, and $3w$ are given to an AOC, where w , $2w$, and $3w$ are in 2's complement representation and sign-extended to have $(W+2)$ bits each.

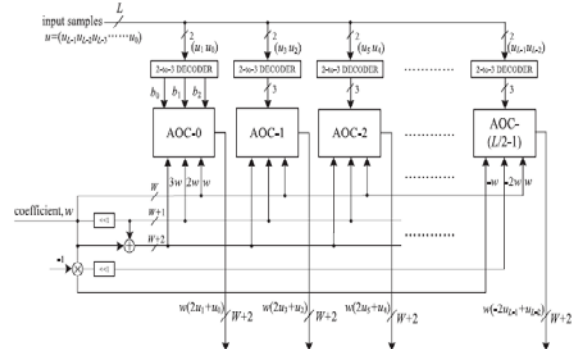


Figure 4: structure of partial product generator

2) Structure of AOCs: The structures of an AOC, as shown in Fig 5, consists of three AND cells and two OR cells. Each AND cell takes an n -bit input and a single bit input b , also consists of n AND gates. It distributes all the n bits of input D to its n AND gates as one of the inputs. The other inputs of all the n AND gates are fed with the single-bit input b . The output of an AOC is w , $2w$, and $3w$ corresponding to the decimal values 1, 2, and 3 of the 2-b input (u_1u_0). The decoder along with the AOC performs 2-bit multiplication and $L/2$ parallel multiplications with a 2-bit digit to produce $L/2$ partial products of the product word.

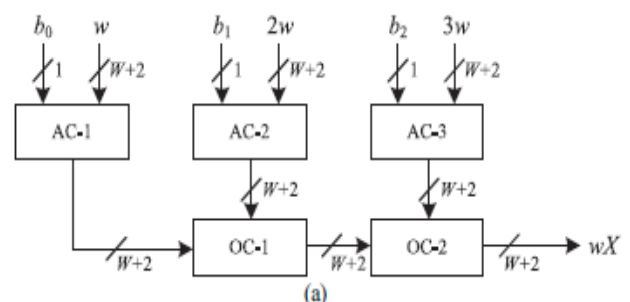


Figure 5: Structure and function of AND/OR cell

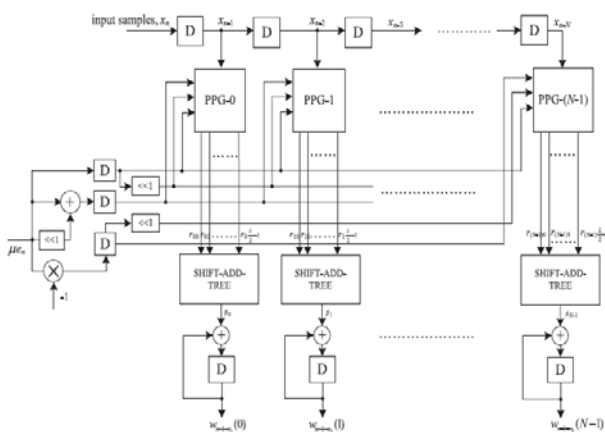
Table 1: Location of pipeline latches for L=8, N=8,16and32

N	Error-computation block		Weight-update block
	Adder tree	Shift-add tree	Shift-add tree
8	Stage-2	Stage-1 and 2	Stage-1
16	Stage-3	Stage-1 and 2	Stage-1
32	Stage-3	Stage-1 and 2	Stage-2

3)Structure of adder tree: The shifts-add operation on the partial products of each PPG gives the product value and then added all the N product values to compute the inner product output. However, the shift-add operation obtains the product value which increases the word length, and the adder size. To avoid increase in word size of the adders, we add all the N partial products of the same place value from all the 'N' PPGs by a single adder length. Table I, shows the pipeline latches for various filter lengths

B. Pipelined Structure Of Weight-Update block

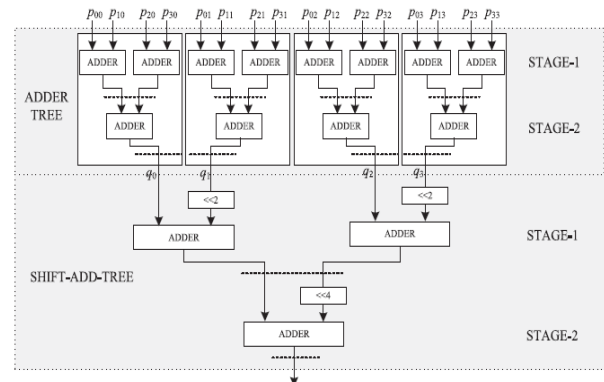
The proposed structure of weight-update block is shown in Fig.6. It performs N multiply-accumulate operations of the form $(\mu \times e) \times x_i + w_i$ to update N filter weights. The step size μ is taken as a negative power of 2 to realize the multiplication with recently available error by the shift operation. Each MAC unit performs the multiplication of the shifted value of error with the delayed input samples x_i followed by the additions with the corresponding old weight values w_i . All the MAC operations are performed by N PPGs, followed by N shift-add trees. Each of the PPGs generates L/2 partial products corresponding to the product of the recently shifted error value $\mu \times e$ with the number of 2-bit digits of the input word x_i . The sub expression can be shared across all the multipliers. This leads to a gradual reduction of adder in complexity. The final outputs of MAC units constitute updated weights to be used as inputs to the error-computation block and the weight-update block for the next iteration.

**Figure 6:** Structure of weight-update block

4. Adder Tree Optimization

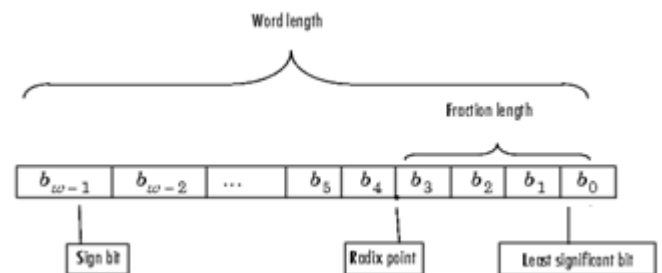
The adder tree and shift-add tree computation can be pruned for further optimization of area, delay, and power complexity. The adder tree structure is given in Fig.7. To reduce the computational complexity, some of the LSBs of inputs of the adder tree can be truncated and the guard bits can be used to minimize the impact of truncation on the

error performance of the adaptive filter. To have more hardware saving, the bits to be truncated are not generated by the PPGs, so the complexity of PPGs also gets reduced. To have more hardware saving, the bits to be truncated are not generated by the PPGs, so the complexity of PPGs also gets reduced.

**Figure 7:** Structure of adder tree

5. Fixed Point Considerations

The fixed-point implementation of the proposed DLMS adaptive filter shows the bit level pruning of the adder tree, to reduce the hardware complexity without the degradation of steady state MSE. For fixed-point implementation, the word lengths and radix points for input samples, weights, and internal signals are needed to be decided. Fig.8 shows the Fixed-point representation of a binary number. Table 3.2 shows the fixed-representation of the desired signals; its quantization is usually given as an input. For this purpose, the specific scaling/sign extension and truncation/zero padding are required. Since the LMS algorithm performs learning so that y has the same sign as d , the error signal e can also be set to have the same representation as y without overflow after the subtraction.

**Figure 8:** Fixed-point representation of a binary number**Table 2:** Fixed-point representation of the signals of the proposed DLMS adaptive filter

Signal Name	Fixed-Point Representation
X	(L, L _i)
W	(w, w _i)
P	(w+2, w _i +2)
Q	(w + 2 + log ₂ N, w _i + 2 + log ₂ N)
y,d,e	(w, w _i + L _i + log ₂ N)
μe	(w, w _i)
R	(w+2, w _i +2)
S	(w, w _i)

6. Result Analysis

This section evaluates the performance of the proposed modified least mean square (LMS) algorithm and shows the simulation results. The first result shows the output of the modified LMS adaptive filter. Following it is the output of the error-computation block and weight update block. The ModelSIM is the tool used here to check the performance of LMS adaptive filter. It is a complete HDL simulation environment that enables to verify the sources code and functional and timing models using test bench.

Fig 9 shows the output of the modified LMS adaptive filter. Two 8-bit inputs like $x_n=01001011$ and $d_n=01101010$ are given with 50 Hz clock frequency and setting $reset=0$. x_n and d_n are the input signal and desired signal. Again by setting $reset=1$, compilation is made and the obtained output $=00101111$, are displayed along with the results of other sub programs.

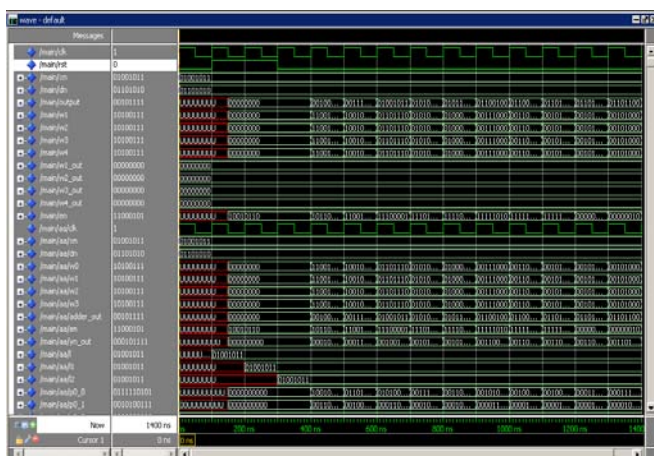


Figure 9: Output of the modified LMS adaptive filter

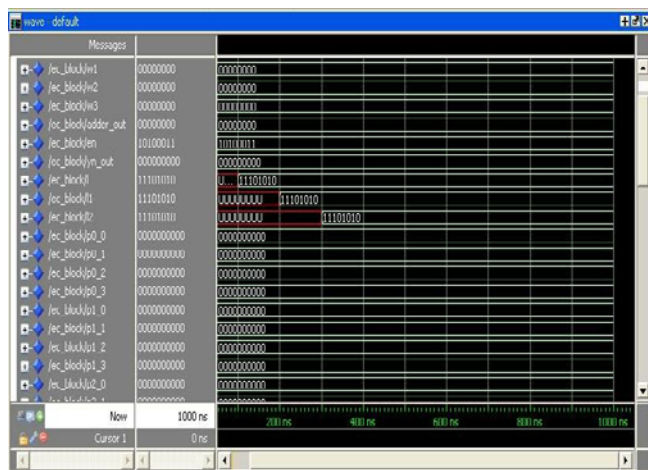


Figure 10: Output of the Error-Computation Block

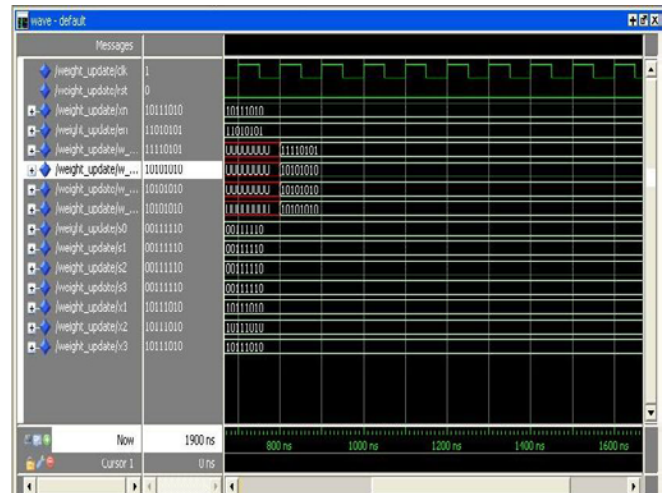


Figure 11: Output of the weight update block

7. Conclusion and Future Scope

In this paper, we proposed an efficient architecture for the design of a modified delayed LMS adaptive filter. By using a Partial Product Generator (PPG), the combinational blocks can achieve efficient area-delay product and energy-delay product. The proposed structure involved significantly less adaptation delay and provided significant saving of ADP and EDP compared to the existing structures.

The efficient addition scheme reduces the adaptation delay to achieve the faster performance and reduction in the critical path supports the high input-sampling rates. The future work involves that to reduce the adder complexity by replacing the various adders in adder blocks to achieve the better performance of area and power.

References

- [1] B. Widrow and S. D. Stearns, Adaptive Signal Processing. Englewood Cliffs, NJ, USA: Prentice-Hall, 1985.
- [2] S. Haykin and B. Widrow, Least-Mean-Square Adaptive Filters. Hoboken, NJ, USA: Wiley, 2003.
- [3] S. Ramanathan and V. Visvanathan, "A systolic architecture for LMS adaptive filtering with minimal adaptation delay," in Proc. Int Conf Very Large Scale Integr. (VLSI) Design, Jan. 1996, pp. 286-289.
- [4] Y. Yi, R. Woods, L.-K. Ting, and C. F. N. Cowan, "High speed FPGA-based implementation of delayed-LMS filters," J. Very Large Scale Integr. (VLSI) Signal Process., vol. 39, nos. 1-2, pp. 113-131, Jan. 2005.
- [5] L. D. Van and W. S. Feng, "An efficient systolic architecture for the DLMS adaptive filter and its applications," IEEE Trans. Circuits Syst. II, Analog Digital Signal Process., vol. 48, no. 4, pp. 359-366, Apr. 2001.
- [6] P. K. Mehar and M. Maheshwari, "A high speed FIR adaptive filter architecture using a modified delayed LMS algorithm," in Proc. IEEE Int Symp. Circuits Syst., May 2011.
- [7] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. New York, USA: Wiley, 1999.

- [8] C. Caraiscos and B. Liu, "A round off error analysis of the LMS adaptive algorithm," IEEE Trans. Acoust., Speech, Signal Process., vol. 32, no. 1, pp 34–41, Feb. 1984.
- [9] R. Rocher, D. Menard, O. Sentieys, and P. Scalart, "Accuracy evaluation of fixed-point LMS algorithm," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process, May 2004, pp. 237–240.
- [10] M. D. Meyer and D. P. Agrawal, "modular pipelined implementations of a delayed LMS transversal adaptive filter," in Proc. IEEE Int. Symp. Circuits Syst., May 1990, pp. 1943–1946.
- [11] G. Long, F. Ling, and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation," IEEE Trans. Acoust., Speech, Signal Process., vol. 37, no. 9, pp. 1397–1405, Sep. 1989.
- [12] M. D. Meyer and D. P. Agrawal, "A high sampling rate delayed LMS filter architecture," IEEE Trans. Circuits Syst. II, Analog Digital Signal Process., vol. 40, no. 11, pp. 727–729, Nov. 1993.