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

[1] **Mr.D.Senthilraja,** [2] **K.Kalaiselvi,**
[1]Assistant professor/ ECE, [2]PG scholar (M.E., Applied Electronics),
[1,2]Excel Engineering College, Komarapalayam-637 303
excelsen1@gmail.com , kalaikarthi145@gmail.com

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.**

I.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.

II. 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:

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

Where,

μ is the convergence-factor.

$$e(n) = d(n) - y(n)$$

$$y(n) = wTn \cdot x(n) \qquad (2)$$

Here,

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

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

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

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

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

where,

m is the adaptation-delay.

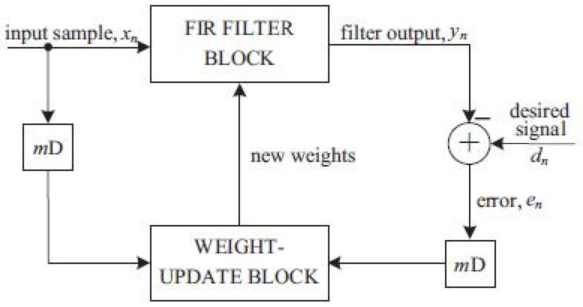

Fig 1. Structure of conventional delayed LMS adaptive filter.

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.

III. 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.

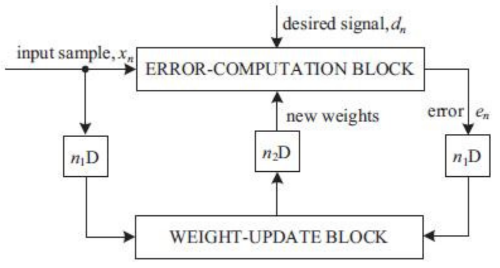

Fig 2. Structure of modified DLMS adaptive filter

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.

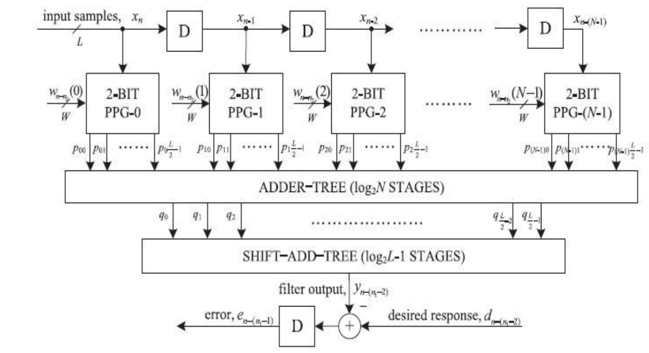
input samples, x_n
L
D
x_{n-1}
D
x_{n-2}
D
$x_{n-(N-1)}$
2-BIT PPG-0
2-BIT PPG-1
2-BIT PPG-2
2-BIT PPG-(N-1)
ADDER–TREE ($\log_2 N$ STAGES)
SHIFT–ADD–TREE ($\log_2 L$-1 STAGES)
filter output, $y_{n-(n_1-2)}$
error, $e_{n-(n_1-1)}$
D
desired response, $d_{n-(n_1-2)}$

Fig3.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 (u1u0) as input and produces three outputs b0 = u0. u1, b1 = u0 · u1, and b2 = u0 · u1, such that b0 = 1 for (u1u0) = 1, b1 = 1 for (u1u0) = 2, and b2 = 1 for (u1u0) = 3. The decoder output b0, b1 and b2 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.

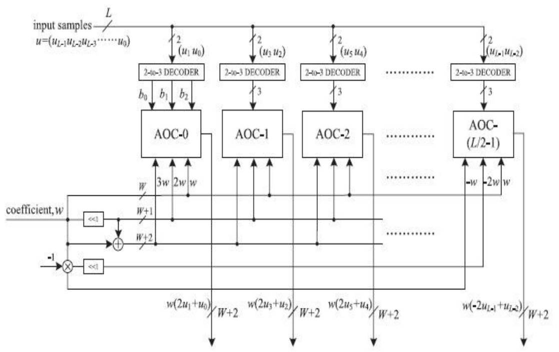

Fig.4 structure of partial product generator

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 (u1u0). 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.

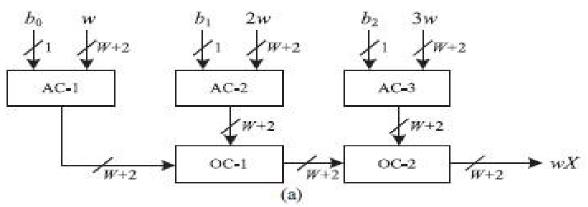

Fig 5. Structure and function of AND/OR cell

Table I
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 tree. Table I,shows the pipeline latches for various filter lengths

B. PIPELINED STRUCTURE OF WEIGHT-UPDATEBLOCK

The proposed structure of weight-update block is shown in Fig.6. It performs N multiply-accumulate operations of the form (μ × e) × 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 μ × e with the number of 2-bit digits of the input word xi. 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.

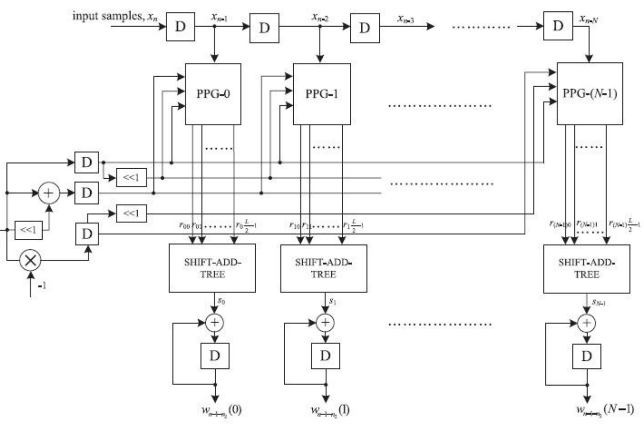

Fig.6. Structure of weight-update block

IV. 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.

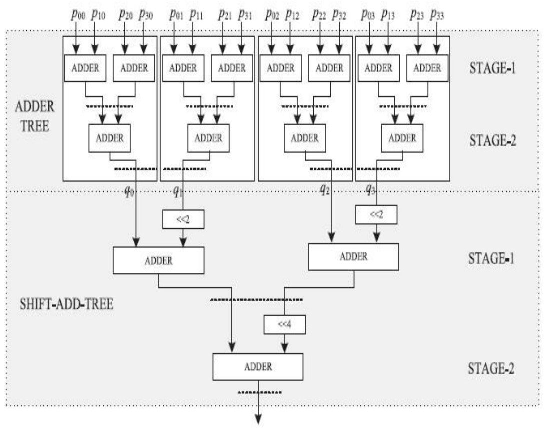
p_{00} p_{10} p_{20} p_{30} p_{01} p_{11} p_{21} p_{31} p_{02} p_{12} p_{22} p_{32} p_{03} p_{13} p_{23} p_{33}
ADDER TREE
ADDER
STAGE-1
STAGE-2
q_0 q_1 q_2 q_3
<<2
SHIFT-ADD-TREE
<<4

Fig 7. Structure of adder tree

V.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 need 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.

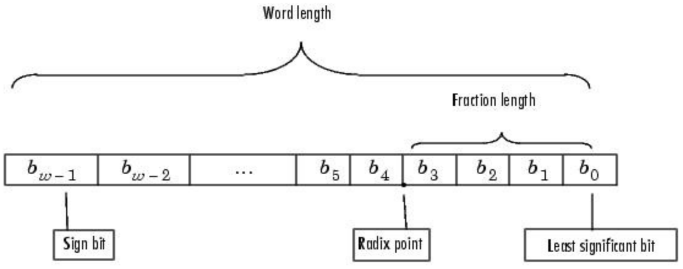

Fig 8.Fixed-point representation of a binary number

Table II.

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)**

VI. RESULT ANALYSIS

This section evaluates the performance of the proposedmodified least mean square (LMS) algorithm and shows thesimulation 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 sousrce 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=01001011and 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.

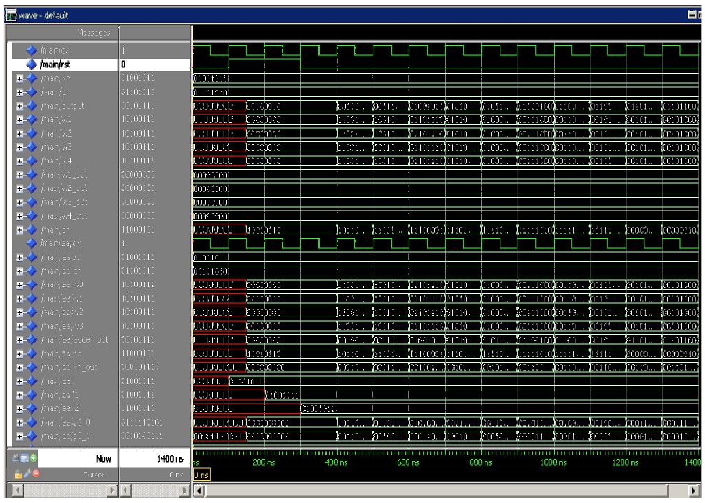
```
/main/rst    0
Now          1400 ns
200 ns   400 ns   600 ns   800 ns   1000 ns   1200 ns   1400
```

Fig9. Output of the modified LMS adaptive filter

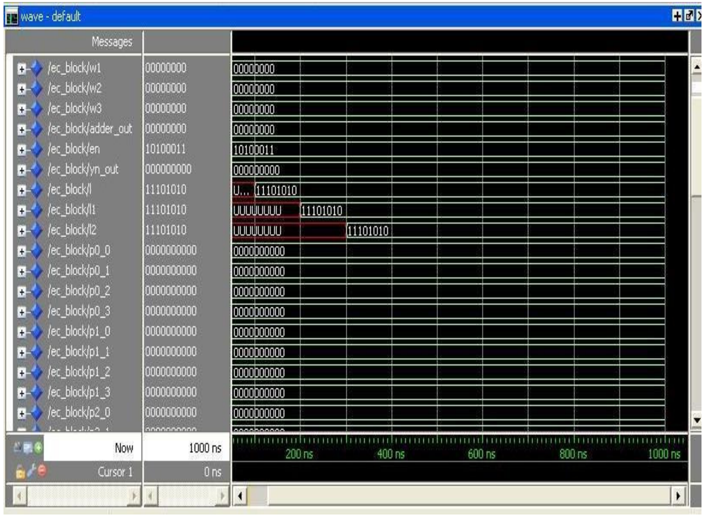
```
Messages
/ec_block/w1          00000000      00000000
/ec_block/w2          00000000      00000000
/ec_block/w3          00000000      00000000
/ec_block/adder_out   00000000      00000000
/ec_block/en          10100011      10100011
/ec_block/yn_out      000000000     000000000
/ec_block/l           11101010      U... 11101010
/ec_block/l1          11101010      UUUUUUUU 11101010
/ec_block/l2          11101010      UUUUUUUU 11101010
/ec_block/p0_0        0000000000    0000000000
/ec_block/p0_1        0000000000    0000000000
/ec_block/p0_2        0000000000    0000000000
/ec_block/p0_3        0000000000    0000000000
/ec_block/p1_0        0000000000    0000000000
/ec_block/p1_1        0000000000    0000000000
/ec_block/p1_2        0000000000    0000000000
/ec_block/p1_3        0000000000    0000000000
/ec_block/p2_0        0000000000    0000000000
Now        1000 ns      200 ns   400 ns   600 ns   800 ns   1000 ns
Cursor 1   0 ns
```

Fig10. Output of the Error-Computation Block

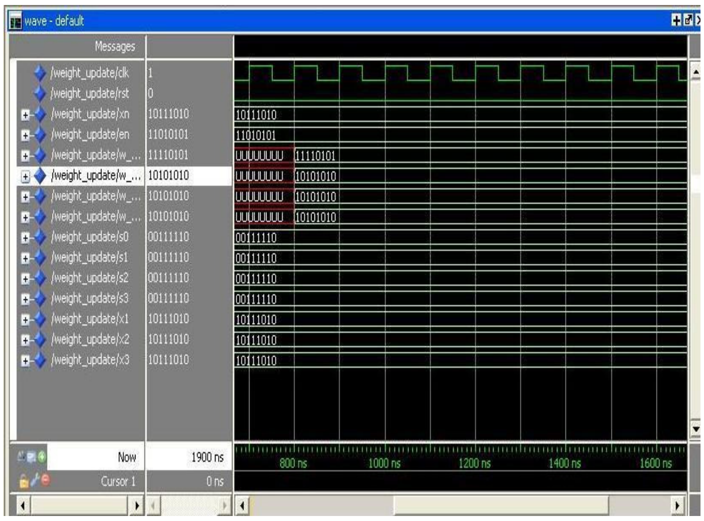
```
Messages
/weight_update/clk    1
/weight_update/rst    0
/weight_update/xn     10111010     10111010
/weight_update/en     11010101     11010101
/weight_update/w_...  11110101     UUUUUUUU 11110101
/weight_update/w_...  10101010     UUUUUUUU 10101010
/weight_update/w_...  10101010     UUUUUUUU 10101010
/weight_update/w_...  10101010     UUUUUUUU 10101010
/weight_update/s0     00111110     00111110
/weight_update/s1     00111110     00111110
/weight_update/s2     00111110     00111110
/weight_update/s3     00111110     00111110
/weight_update/x1     10111010     10111010
/weight_update/x2     10111010     10111010
/weight_update/x3     10111010     10111010
Now        1900 ns      800 ns   1000 ns   1200 ns   1400 ns   1600 ns
Cursor 1   0 ns
```

Fig11. Output of the weight update block

VII. CONCLUSION AND FUTURE SCOPE

In this paper, we proposed an efficient architecture for thedesign of a modified delayed LMS adaptive filter. By using aPartial Product Generator (PPG), the combinational blocks canachieve 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

i. B.Widrow and S. D. Stearns, Adaptive Signal Processing.Englewood Cliffs, NJ, USA: Prentice-Hall, 1985.

ii. S. Haykin and B. Widrow, Least-Mean-Square Adaptive Filters. Hobo-ken, NJ, USA: Wiley, 2003.

iii. S. Ramanathan and V. Visvanathan, “A systolic architecture for LMS adaptive filering with minimal adaptation delay,”in Proc. IntConf Very Large Scale Integr,(VLSI) Design,Jan.1996,pp. 286-289.

iv. Y. Yi, R. Woods, L.-K. Ting, and C. F. N. Cowan, “High speed FPGA-based implementations of delayed-LMS filters,” J. Very Large Scale Integr. (VLSI) Signal Process., vol. 39, nos. 1–2, pp. 113–131, Jan. 2005.

v. 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.

vi. P.K.Mehar and M.Maheshwari, “A high speed FIR adaptive filter architecture using a modified delayed LMS algorithm,” in Proc. IEEE Int Symp. Circu its Syst, May 2011

vii. K. K. Parhi, VLSI Digital Signal Procesing Systems: Design and Implementation. New York, USA: Wiley, 1999.

viii. C. Caraiscos and B. Liu, “A roundoff error analysis of the LMS adaptive algorithm,” IEEE Trans. Acoust., Speech, SignalProcess., vol. 32, no. 1, pp 34–41, Feb. 1984.

ix. R. Rocher, D. Menard, O. Sentieys, and P. Scalart,“Accuracy evaluation of fixed-point LMS algorithm,” in Proc. IEEEInt. Conf. Acoust., Speech, Signal Process., May 2004, pp. 237–240.

x. M. D. Meyer and D. P. Agrawal, “A modular p ipelinedimplementation of a d elayed LMS transversal adaptive filter,” inProc. IEEE Int. Symp. Circuits Syst., May 1990, pp. 1943–1946.

xi. G. Long, F. Ling, and J. G. Proakis, “The LMS algorithmwith delayed coefficien t adaptatio n,” IEEE Trans. Acoust., Speech,Signal Process., vol. 37, no. 9, pp. 1397–1405, Sep. 1989.

xii. M. D. Meyer and D. P. Agrawal, “A high sampling ratedelayed LMS filter architecture,” IEEE Trans. Circuits Syst. II,Analog Digital Signal Process., vol. 40, no. 11, pp. 727–729, Nov.1993.