PIPELINED ARCHITECTURES FOR THE TD-LMS ADAPTIVE FILTER

George-Othon Glentis

Technological Education Institute of Krete, Branch at Chania, Department of Electronics,
3, Romanou Str, Chalepa, 73133 Chania, Krete, Greece.
e-mail: gglentis@chania.teiher.gr

ABSTRACT

In this paper, efficient pipelined architectures for the implementation of the Transform Domain LMS algorithm, are presented. Pipelining of the TD-LMS algorithm is achieved by introducing an amount of time delay into the original adaptive scheme. The adaptation delay introduced to the TD-LMS algorithm allows for the development of pipelined architectures. By retiming the delays existing in the error feedback loop, two efficient pipelined implementations of the delayed TD-LMS algorithm are developed.

1. INTRODUCTION

The design of adaptive filters and system identification algorithms with optimum learning, in the sense of minimizing the accumulated squared error between the output signal and a desired response, has been the subject of major research for a long time, [1].

The introduction of the Delayed LMS algorithm gave rise to the development of high throughput pipelineable and/or parallel schemes that have been proposed for the implementation of the LMS algorithm on ASIC VLSI systolic or wavefront array processors, [4]. A major drawback of the D-LMS algorithm is the degradation that appears on the convergence performance, due to the adaptation delay imposed by the algorithm. The remedy to this drawback is the introduction of correction terms that compensate for the adaptation delay and give results identical to the original LMS algorithm, subject to a certain amount of output delay, [5].

In an attempt to improve the performance of the LMS algorithm, unitary transformations on the input data vector, have been used, [6]. The resulting algorithms may have increased convergence rate for some classes on input signals, yet the computational complexity remains similar to that of the original LMS scheme. The algorithmic family established and the variations followed are known as the *Transform Domain Adaptive Filtering* algorithms, [2],[3],[6].

This work was supported by the Greek Secretary of Research and Technology, PENED-99EΔ83, 'ISODIA' project

2. THE DELAYED TD-LMS

Given an input signal $x(n)$ and a desired response signal $y(n)$, the Delayed Transform Domain LMS (D-TD-LMS) is described by the set of equations

$$\mathbf{f}_M(n) = \mathbf{S}_M \mathbf{x}_M(n) \tag{1}$$

$$e(n) = y(n) - \mathbf{C}_M^H(n-1)\mathbf{f}_M(n) \tag{2}$$

$$\mathbf{p}_M(n) = \lambda \mathbf{p}_M(n) + (1-\lambda)\mathrm{diag}(f_1^2(n)\dots f_M^2(n)) \tag{3}$$

$$\mathbf{F}_M(n) = \mu \mathbf{p}_M^{-1}(n)\mathbf{f}_M(n) \tag{4}$$

$$\mathbf{C}_M(n) = \mathbf{C}_M(n-1) + \mathbf{F}_M(n-D)e^*(n-D) \tag{5}$$

$\mathbf{C}_M = [C_1\ C_2\ \dots C_M]^T$ is the vector that carries the transformed filter coefficients. $\mathbf{x}_M(n) = [x(n)\ x(n-1)\ \dots x(n-M+1)]^T$ is the regressor vector. $\mathbf{S}_M$ is a unitary transform (within a constant scalar) of order M, i.e., $\mathbf{S}_M\mathbf{S}_M^H = \beta\mathbf{I}_M$. H stands for the Hermitian operator (conjugate and transpose). D is a proper amount of adaptation delay.

When the DFT is used as the unitary transformation, $\mathbf{f}_M(n)$ is the sliding window Fourier transform of the input data $\mathbf{x}_M(n)$, and each element of $\mathbf{C}_M(n)$ is associated with a specific frequency band. The sliding window DFT algorithm can be efficiently implemented either using a sliding FFT algorithm, or a frequency-sampling filter structure. In both cases, the computational complexity is M complex multiplications per iteration period. However, the later case is more suitable for the VLSI implementation, since it has a regular structure. It is implemented using a set of first order recursive equations of the form

$$\begin{aligned} f_{m+1}(n) &= \rho e^{-j\frac{2\pi m}{M}} f_{m+1}(n-1) + \\ & x(n) - \rho^M x(n-M), \quad m = 0, 1, \dots M-1 \end{aligned} \tag{6}$$

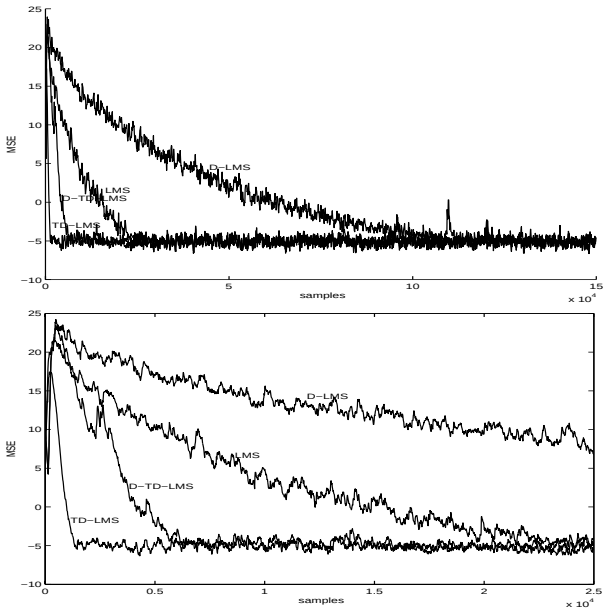

Figure 1: Simulation results

$\rho \in (0,1)$ is a stabilization factor that is used to compensate for the marginal stability of the original realization, [2].

A transform domain delayed LMS algorithm and architecture has been proposed in [7]. That method uses block processing operations (such as FFT/IFFT's) and concepts of multirate filtering to reduce complexity and increase the throughput rate, at the expense of performance degradation. Our method is based on the sample by sample transform domain LMS adaptive filter, thus fast convergence rate and tracking characteristics are retained. The pipelined architectures that are developed in Section 3, allow the maximum throughput rate, given a library of computational hardware elements. They have regular and modular structure, with local interconnections, being suitable for implementation of a massively parallel computer or on special purpose VLSI array processors.

The performance of the D-TD-LMS is illustrated by a typical system identification experiment. Consider an FIR filter of order $M = 128$. The impulse response was a typical impulse response of the acoustic echo path of a car enclosure. A stationary AR process of order 2, driven by a white noise signal, was used as an input to the FIR filter. At the output of the FIR system white gaussian noise was added, resulting to an SNR equal to about 30dB. The eigenvalue spread of the sampled autocorrelation matrix of the input signal which is controlled by the parameters of the AR process, was in our case $O(10^3)$. Four algorithms have been tested, namely, the LMS, the D-LMS with a delay D=M-1, the TD-LMS and the D-TD-LMS with a delay D=M-1. The filtering error power for each case was computed by averaging the squared instantaneous filtering errors over an exponentially decaying window with effective memory equal to 128 time instances. The simulation results are shown in Figure 1. Clearly, the convergence rate of the D-LMS algorithm is very slow. On the other hand, the D-TD-LMS algorithm converges at an acceptable rate, faster than the LMS algorithm.

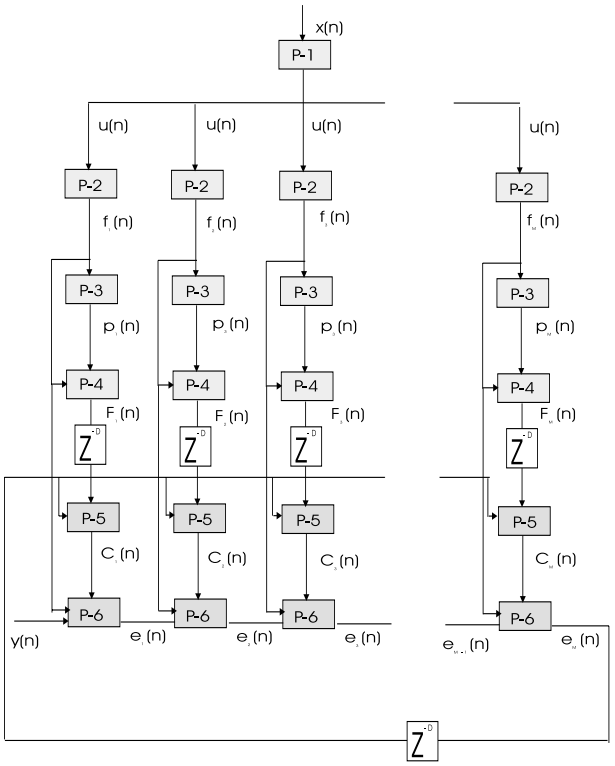

Figure 2: Data-flow graph of the D-TD-LMS algorithm

3. PIPELINED DELAYED TD-LMS ARCITECTURES

The data-flow graph of the D-TD-LMS algorithm is depicted in Figure 2. It is composed by a set of identical Processing Elements (PE), denoted by P-1 to P-6. The operations performed by each PE is summarized in Table 1. PE's P-1 to P-4 involve feedforward interconnections. Thus, pipelining of these PE's can be achieved by placing delay latches in between. On the other hand, P-5 and P-6 are connected via a long feedback loop, and as a result, some extra effort is required for the pipelining of these elements.

A pipelined architecture can readily be derived by replacing the serial inner product estimation of the er-

P.E.	Operation
P-1	$u(n) = x(n) - \rho^M x(n-M)$
P-2	$f(n) = \rho e^{-j\frac{2\pi}{M}m} f(n-1) + u(n)$
P-3	$p(n) = \lambda p(n-1) + (1-\lambda)\|f(n)\|^2$
P-4	$F(n) = \mu f(n)/p(n)$
P-5	$C(n) = C(n-1) + F(n)e^*(n)$
P-6	$e_i(n) = e_{i-1}(n) - C^*(n-1)f(n)$

Table 1: Computational tasks of the Processing Elements

ror signal by a binary tree adder scheme. The presence of the adaptation delay in the error feedback loop of the original data-flow graph (Figure 2), can be used for the efficient pipelining of the binary tree adder that estimates the error signal, [8]. Obviously, the amount of adaptation delay that is required for full pipelining of the D-TD-LMS algorithm is $D = [log_2(M)] + 1$. The data-flow graph of the pipelined D-TD-LMS algorithm is shown in Figure 3. Pipeline latches introduced between the stages of the binary tree adder are depicted by small black rectangular boxes. Further improvement can be achieved by pipelining PE's P1 to P3, by introducing pipelining latches in between, and by delaying the input sequence $y(n)$ by the same amount of time. These latches are depicted by small gray rectangular elements. Notice that, PE's P5 and P6 can work on parallel. Thus, the critical path is now determined to be $T_C = max\{T_D, T_M + T_A\}$ where, T_D is the time required for the division, and T_M, T_A is the time required for the multiplication and the addition operation, respectively.

In the sequel, an alternative pipelined architecture for the D-TD-LMS algorithm will be presented, that avoids the use of a binary tree adder, allowing, thus, for a systolic implementation. The data-flow graph that is depicted in Figure 2 consists of M identical columns of PE's. Consider the last (i.e. the M-th) column of PE's. The delay element that appears at the output of P-6 can be transferred to all three inputs of P-6. The delay element associated to the error input signal is divided to two parts, a unit delay (denoted by a solid black box in Figure 4), and an element that consists of D-1 delays units. The delay elements associated with the other two inputs of P-6, are moved upwards to the top of the data-flow graph, passing through P-5 to P-1. This movement causes the appearance of a delay element at the input of P-5, which is decomposed accordingly to a unit delay and a part that consists of the rest. The block of D-1 delays that appears in the input of P-5 and P-5 of the M-th column of PE's serve as output delays of the corresponding PE's of the (M-1)-th column of

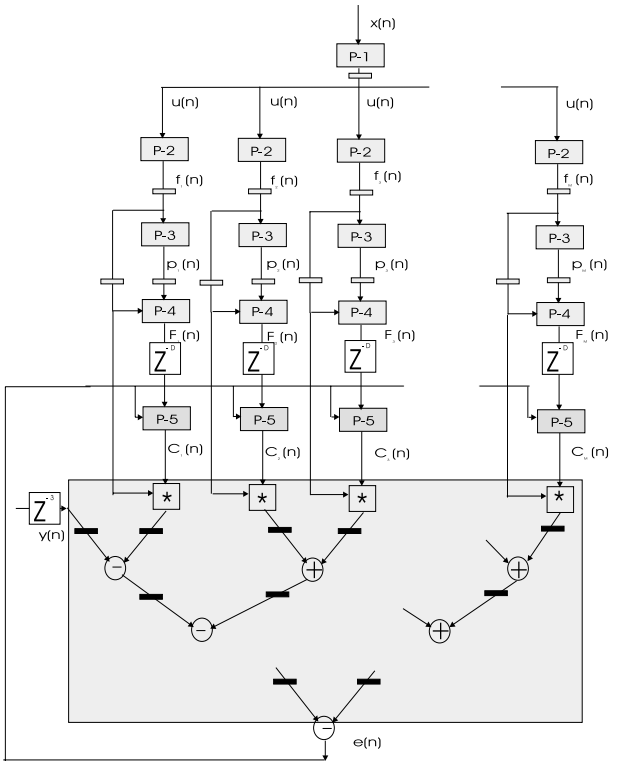

Figure 3: A pipelined architecture for the D-TD-LMS using a binary tree adder

PE's. Thus delay elements are transferred in a similar way to the output of the first column of PE's. Each column of PE's is fed with a delayed input signal. In our example, the last column is fed by the signal $x(n-D)$. Thus, to allow for a full pipelining, $D = M-1$ adaptation delays should be introduced to eq. (5). The resulting architecture is shown in Figure (6). The critical path in this case is easily shown to be equal to the former architecture, i.e., $T_C = max\{T_D, T_M + T_A\}$.

Both pipelined architectures are very attractive for the implementation of the D-TD-LMS algorithm on massively parallel general purpose computers or on special purpose hardware. They both have a regular structure with local interconnections. The overall performance of the D-TD-LMS algorithm, measured by means of computational complexity, tracking performance and degree of pipelining, is compared competitively to other algorithms that have be proposed for high speed adaptive filtering applications. A comparison among different pipelined adaptive algorithms is given in Table 2, for the case of real and complex data. The complexity is measured by means of real multiplications and divisions. Notice, that when real data are processed, the D-TD-LMS algorithm can be implemented using a half-length signal-flow graph, i.e. (M/2) columns of

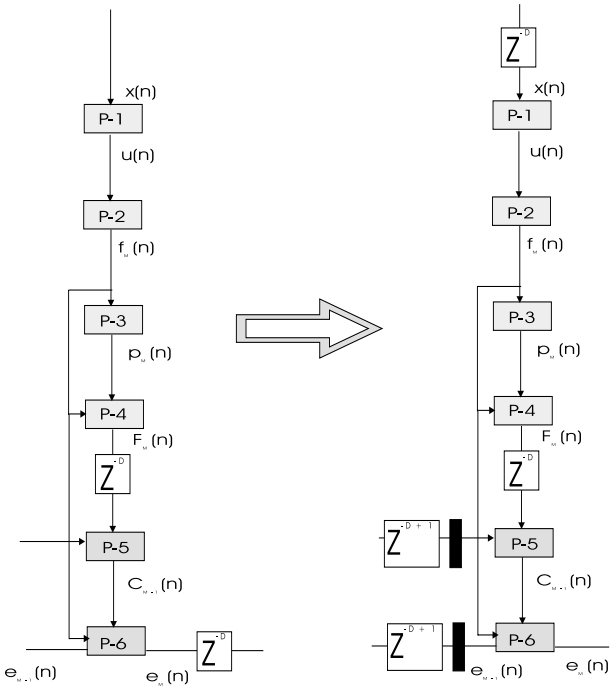

Figure 4: Transfer and decomposition of the output delay elements

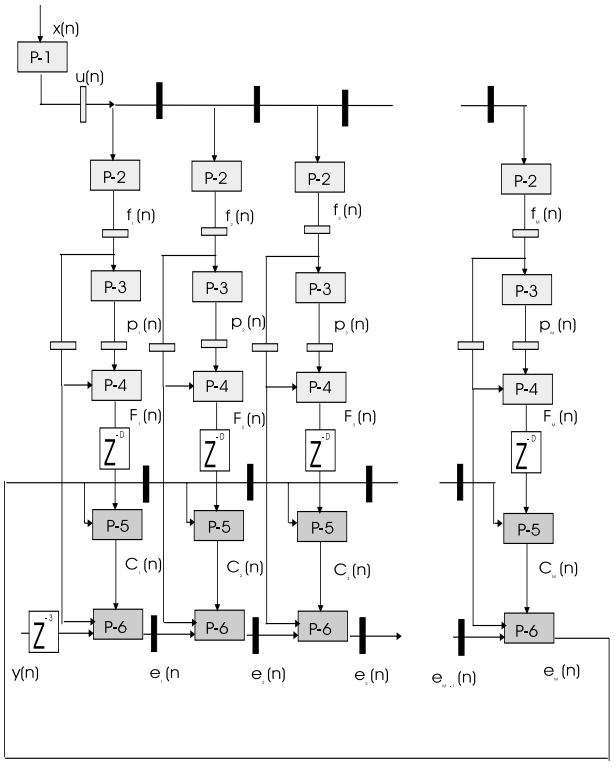

Figure 5: A fully pipelined architecture for the D-TD-LMS algorithm

Method	Real	Complex	D	Latency
D-LMS	2M	8M	M	M, 0
D-LMS [5]	5M	20M	0	M
D-TL-MS	5.5M	16M	M	M
SGL	9M	28M	0	M
LSL	14M	37M	0	M

Table 2: Comparison among competitive pipelined adaptive algorithms

PE's are required in this case, [6].

4. CONCLUSION

In this paper, efficient architectures for the pipelined implementation of the Transform Domain LMS algorithm have been considered. The pipelined operation of the algorithm have been achieved introducing a proper amount of adaptation delay to the original algorithm, resulting to a Delayed TD-LMS scheme. Two pipelined architectures have been proposed that allows for full pipelining of the algorithm, resulting to an adaptation delay of $[log_2(M)] + 4$ and $M + 2$, for the first and the second method, respectively. The critical path has been reduced to $T_c = max\{T_D, T_M + T_A\}$.

5. REFERENCES

[1] S. Haykin, Adaptive Filter Theory. Third Edition, Prentice Hall, 1996

[2] J. Shynk, 'Frequency-domain and multirate adaptive filtering,' IEEE Signal Processing Magazine, pp. 14-39, Jan. 1992.

[3] G. Glentis, K. Berberidis, and S. Theodoridis, 'Efficient least squares adaptive algorithms for FIR transversal filtering: a unified view,' IEEE Signal Processing Magazine, pp. 13-42, July 1999.

[4] H. Herzberg, R. Haimi-Cohen, and Y. Beery, 'A systolic array realization of an LMS adaptive filter and the effects of delayed adaptation,' IEEE Trans. Signal Processing, pp. 2799-2803, Nov. 1992.

[5] S. Douglas, Q. Zhu, and K. Smith, 'A pipelined LMS adaptive FIR filter architecture without adaptation delay,' IEEE Trans. Signal Processing, pp. 775-779, March 1998.

[6] S. Narayan, A.M. Peterson, and M.J. Narasimba, 'Transform domain LMS algorithm,' IEEE Trans. Acoust. Speech, Signal Proc., vol. ASSP-31, pp. 609-615, June 1983.

[7] A. Wu and C. Wu, 'Transform-Domain Delayed LMS algorithm and architecture,' ICCAS-98, pp.V.194-197, 1998

[8] P. Pirsch, Architectures for digital Signal Processing, Wiley 1996