PIPELINED ARCHITECTURES FOR THE FREQUENCY DOMAIN LINEAR EQUALIZER

GEORGE-OTHON GLENTIS, KRISTINA GEORGOULAKIS

Department of Telecommunications, University of Peloponnese
Terma Karaiskaki 22100, Tripoli, Greece
e-mail: gglentis@uop.gr

In this paper, novel pipelined architectures for the implementation of the frequency domain linear equalizer are presented. The Frequency Domain (FD) LMS algorithm is utilized for the adaptation of equalizer coefficients. The pipelining of the FD LMS linear equalizer is achieved by introducing an amount of time delay into the original adaptive scheme, and following proper delay retiming. Simulation results are presented that illustrate the performance of the effect of the time delay introduced into the adaptation algorithm. The proposed architectures for efficient pipelining of the FD LMS linear equalization algorithm are suitable for implementation on special purpose hardware by means of the ASIC, ASIP or FPGA VLSI processors.

Keywords: adaptive equalization, frequency domain LMS, pipelined implementation

1. Introduction

The design of high-bit rate adaptive equalizers has been the subject of major research and development, for high-speed digital communication over satellite, microwave, mobile, and unshielded twisted pair channels (Azadet and Nicole 1998; Maginot *et al.*, 1991; Rofougaran *et al.*, 1998; Shanbhag and Im, 1998). Analog channels deliver corrupted and transformed versions of their input waveforms, which result in the degradation of communication system performance. To recover the data signal, equalization techniques that combat the channel distortions are employed at the receiver. Adaptive equalization refers to a particular case where the design of the equalizer is performed at the receiver, on the basis of the available data (received signal and/or training signal). In this case, temporal characteristics of the physical channel that may vary with time are captured into the equalizer design (Quereshi, 1985; Proakis, 1995; Benedetto and Biglieri, 1999). Adaptive equalizers are implemented by means of adaptive signal processing algorithms. Fast convergence speed and the tracking ability with respect to time varying statistics, low computational complexity, parallelism and pipelining, modularity and local communication are issues related to performance, when very high-speed implementation of adaptive equalizers, on the ASIC or ASIP VLSI processors, is under consideration (Haykin, 1996; Kalouptsidis and Theodoridis, 1993; Parhi, 1999; Pirsch, 1998).

Linear equalization in the frequency domain has been proposed in the past, as an improvement over Least Mean Squared (LMS) error based adaptive linear equalizers (Qureshi, 1985; Picchi and Prati, 1984). LMS-like algorithms are popular due to low computational complexity and simplicity in hardware realization of the underlying algorithmic structure. However, the convergence rate of the LMS-based adaptive equalizer heavily depends on the eigenvalue spread of the correlation matrix of the input data (Haykin, 1996). In an attempt to improve the convergence rate of the original scheme, a Discrete Fourier Transform (DFT) on the equalizer input data vector was used, resulting in the Frequency Domain (FD) LMS adaptive linear equalizer (Picchi and Prati, 1984). FD LMS may have increased the convergence rate for some classes of input signals, yet the computational complexity remains similar to that of the original LMS scheme. Frequency domain adaptive equalization has been considered extensively in major telecommunications schemes (Benvenuto and Tomasin, 2001; Berberidis *et al.*, 2004; Bilcu *et al.*, 2002; 2003; Huang and Benesty, 2003; Moreli *et al.*, 2005; Shamma, 2002; Son *et al.*, 2006; Ting *et al.*, 2005; Yang *et al.*, 2004).

In this paper, efficient pipelined architectures for the implementation of the FD LMS adaptive linear equalizer are presented. The unitary transform utilized is the Discrete Fourier Transform (DFT), and it is implemented by means of a sliding window DFT, which allows full pipelining. An amount of time delay is subsequently introduced into the original adaptive scheme, resulting in the Delayed FD LMS adaptive algorithm. Proper retiming of the existing delays results in a fully pipelined architecture, which is suitable for parallel implementation on a general

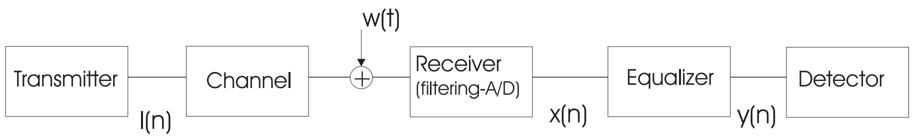


Fig. 1. Channel equalilzation setup.

purpose parallel machine or on dedicated VLSI hardware, using a systolic or a wavefront array of processors. VLSI implementation issues are also discussed.

2. Delayed FD-LMS Adaptive Linear Equalizer

InterSymbol Interference (ISI) is a major impairment in today's high bit rate communication systems (Arslan and Bottomley, 2001; Benedetto and Biglieri, 1999; Qureshi, 1985; Proakis, 1995). Channel equalizers used in the receiver part aim to suppress the effect of ISI. In most cases, the communication channel is unknown and the design of the equalizer is performed on the basis of a known training sequence of information bits.

The channel equalization setup adopted in this paper is illustrated in Fig. 1. The transmitted waveform has the form

$$u(t) = \sum_{k=-\infty}^{\infty} I(k) g_t(t - kT_s). \tag{1}$$

$I(k)$ is an equiprobable sequence of the transmitted data taken from a binary alphabet, i.e., $I(k) \in \{\pm 1\}$, $g_t(k)$ is the pulse shape and T_s is the symbol period.

The symbol spaced sampled, discrete time received signal $x(n)$ of ISI and a noise impaired linear channel is written as

$$x(n) = \sum_{i=0}^{L} h_i(n) I(n-i) + w(n). \tag{2}$$

Here L is an integer that represents the memory of the channel. The sequence $h_i(n)$ $i = 0, 1, \dots, L$ represents the impulse response of the discrete time composite channel, reflecting the influence of the transmit filter, the communications channel and the receive filter. Moreover, $w(n)$ is an Additive White Gaussian Noise (AWGN) sequence.

The linear equalizer aims at reducing the effect of ISI on the received data. It is implemented by means of an FIR (Finite Impulse Response) digital filter of the form

$$y(n) = \sum_{i=0}^{M-1} c_i^* x(n-i), \tag{3}$$

where $x(n)$ is the input signal, $y(n)$ is the output signal of the equalizer, and M is an integer that denotes the equalizer length (Benedetto and Biglieri, 1999). The above equation can be written in a compact way as

$$y(n) = \boldsymbol{c}_M^H \boldsymbol{x}_M(n), \tag{4}$$

where

$$\boldsymbol{x}_M(n) = [x(n)\; x(n-1)\; \dots\; x(n-M+1)]^T \tag{5}$$

is the data vector, and

$$\boldsymbol{c}_M = [c_0\; c_1\; c_2\; \dots c_{M-1}]^T \tag{6}$$

is the vector of equalizer coefficients (a^* denotes the conjugate of a variable a and the superscript ^{H} stands for the Hermitian operator (conjugate and transpose)).

Given a set of training data, the coefficients of the linear equalizer are estimated by minimizing the cost function

$$J(\boldsymbol{c}_M) = \boldsymbol{\mathcal{E}}\big(|I(n-\delta) - y(n)|^2\big), \tag{7}$$

where $I(n)$ is a sequence of the known transmitted data, $\delta > 0$ is the equalizer's delay, and $\boldsymbol{\mathcal{E}}(\cdot)$ denotes the expectation operator. Once $\boldsymbol{c}_M$ is estimated, the equalizer operates in the so-called decision-directed mode, where the transmitted data are detected using the following decision rule:

$$\hat{I}(n-\delta) = \operatorname{dec}\big(y(n)\big). \tag{8}$$

Usually, a small amount of training data is available for tuning the equalizer parameters. The operation of the equalizer is afterwards turned into the decision-directed mode, where the adaptation of parameters is carried out using decisions as the desired response signal (decisions-directed adaptation).

One of the most common algorithms for channel estimation and channel equalization is Widrow's LMS algorithm (Glentis *et al.*, 1999; Haykin, 1996). It has the form

$$y(n) = \boldsymbol{c}_M^H(n-1) \boldsymbol{x}_M(n), \tag{9}$$

While in training mode

$$e(n) = I(n-\delta) - y(n), \tag{10}$$

Otherwise

$$\hat{I}(n-\delta) = \operatorname{dec}\big(y(n)\big), \tag{11}$$

$$e(n) = \hat{I}(n-\delta) - y(n), \tag{12}$$

WhileEnd

$$\boldsymbol{c}_M(n) = \boldsymbol{c}_M(n-1) + \mu_{\text{LMS}} \boldsymbol{x}_M(n) e^*(n). \tag{13}$$

The parameter μ_{LMS} is a positive constant that regulates the convergence speed of the adaptation algorithm. Despite their low computational complexity, the LMS algorithm converges slowly to the optimum solution, especially in the case when the input signal is highly correlated. Several algorithms have been proposed in the past for accelerating the performance of the LMS scheme. A comprehensive presentation of various algorithms for adaptive filtering is provided in the tutorial paper by Glentis *et al.* (1999). In the sequel, the frequency domain adaptive scheme will be adopted in the context of linear equalization.

2.1. Frequency Domain Adaptive Linear Equalizer.

The simplest form of the LMS algorithm offers adaptive filtering with a cost proportional to the equalizer filter size. However, the convergence rate of the algorithm heavily depends on the eigenvalue spread of the correlation matrix of the input data. In an attempt to improve the performance of the LMS algorithm, unitary transformations on the input data vector have been used (Farhang-Boroujeny *et al.*, 1996; Narayan *et al.*, 1993; Picchi and Prati, 1984; Shynk, 1992). The resulting algorithms may have an increased convergence rate for some classes on input signals, yet their computational complexity remains similar to that of the original LMS scheme.

Let $x(n)$ and $y(n)$ be the equalizer input and output signals, respectively. Let $I(n)$ and $\hat{i}(n)$ denote the training data and the detected data after equalization, respectively. The frequency domain LMS adaptive equalizer of (Picchi and Prati, 1984) is a transform domain LMS linear equalizer, where the unitary transform utilized is the DFT. It is described as follows:

$$\boldsymbol{f}_M(n) = \boldsymbol{W}_M \boldsymbol{x}_M(n), \tag{14}$$

$$\boldsymbol{F}_M(n) = \boldsymbol{p}_M^{-1} \boldsymbol{f}_M(n), \tag{15}$$

$$y(n) = \boldsymbol{C}_M^H(n-1)\boldsymbol{f}_M(n), \tag{16}$$

While in training mode

$$e(n) = I(n-\delta) - y(n), \tag{17}$$

Otherwise

$$\hat{I}(n-\delta) = \mathrm{dec}\left(y(n)\right), \tag{18}$$

$$e(n) = \hat{I}(n-\delta) - y(n), \tag{19}$$

WhileEnd

$$\boldsymbol{C}_M(n) = \boldsymbol{C}_M(n-1) + \mu_{\mathrm{FDLMS}} \boldsymbol{F}_M(n) e^*(n). \tag{20}$$

Here $\boldsymbol{W}_M$ denotes the DFT transform of order M. The DFT transform of the input data $\boldsymbol{x}_M(n)$ is denoted by $\boldsymbol{f}_M(n)$. It is a vector with the elements

$$\boldsymbol{f}_M(n) = \left[f_1(n)\ f_2(n)\ \cdots f_M(n)\right]^T, \tag{21}$$

called thereafter the frequency domain regressor vector. $\boldsymbol{C}_M = [C_1\ C_2\ \cdots C_M]^T$ is the vector that carries the transformed equalizer coefficients. Each element of $\boldsymbol{C}_M(n)$ is associated with a specific frequency band. Moreover, μ_{FDLMS} is a positive constant that controls the convergence speed of the algorithm.

Here $\boldsymbol{p}_M$ is the diagonal matrix with the entries being the signal powers associated with consecutive each frequency bins. It has the form

$$\boldsymbol{p}_M = \mathrm{diag}\left[p_1,\ p_1, \ldots, p_M\right], \tag{22}$$

where p_m is the signal power at the k-th frequency bin,

$$p_{m+1} = \boldsymbol{\mathcal{E}}\left[|f_{m+1}(n)|^2\right], \quad m = 0, 1, \ldots, M-1. \tag{23}$$

The role of $\boldsymbol{p}^{-1}$ in (15) is to reduce the eigenvalue spread of the corresponding system matrix. In practice, $\boldsymbol{p}$ is a time varying matrix whose elements are calculated in terms of the available data, e.g., using an exponentially weighed power estimator, implemented by the difference equation (Shynk, 1992):

$$\begin{aligned}\boldsymbol{p}_M(n) &= \lambda \boldsymbol{p}_M(n-1)\\ &\quad + (1-\lambda)\mathrm{diag}\left(|f_1(n)|^2, \ldots, |f_M(n)|^2\right),\end{aligned} \tag{24}$$

where $\lambda \in (0,1)$ is a smoothing factor. Clearly, $\lim_{n\to\infty} \boldsymbol{\mathcal{E}}[\boldsymbol{p}_M(n)] = \boldsymbol{p}_M$ for a stationary input signal.

2.2. Pipelined Implementation Aspects.

The inner product computations involved in the error feedback loop of the FD-LMS linear equalizer, i.e., Eqns. (16) and (20) prohibit full pipelining and/or parallelism of the algorithm. A remedy to this bottleneck is the introduction of an adaptation delay into the coefficients update equation (20), similarly to that introduced by Long *et al.* (1989) in the original LMS adaptive algorithm. Thus, (20) is now modified to allow for an adaptation delay of size D as

$$\boldsymbol{C}_M(n) = \boldsymbol{C}_M(n-1) + \mu_{\mathrm{DFDLMS}} \boldsymbol{F}_M(n-D) e^*(n-D). \tag{25}$$

Equations (14)–(19) together with (25) constitute a new adaptive scheme for the estimation of the frequency domain linear equalizer, called thereafter the Delayed Frequency Domain LMS (D-FD LMS) adaptive linear equalizer. The parameter μ_{DFDLMS} determines the convergence properties of the algorithm. The delayed frequency domain adaptive algorithm was introduced by Glentis (2001). The statistical properties, as well as conditions on the convergence of delayed frequency domain adaptive algorithms, have been studied by Glentis (2005).

The presence of a time delay in the error feedback loop permits for the development of high throughput pipelineable and/or parallel schemes for the implementation of the D-FD LMS algorithm on ASIC VLSI systolic or wavefront array processors.

The sliding window DFT algorithm implied by (14) can be efficiently implemented using either a sliding FFT algorithm (Farhang-Boriujeny *et al.*, 1996) or a frequency-sampling filter structure (Shynk, 1992). In both cases, the computational complexity is M complex multiplications per iteration period. However, the latter case is more suitable for VLSI implementation, since it has a regular structure. It is implemented using a set of first-order recursive equations of the form

$$\begin{aligned} f_{m+1}(n) &= \rho e^{-j\frac{2\pi m}{M}} f_{m+1}(n-1) \\ &\quad + x(n) - \rho^M x(n-M), \\ &\quad m = 0,1,\dots,M-1, \end{aligned} \tag{26}$$

where $\rho \in (0,1)$ is a stabilization factor that is used to compensate for the marginal stability of the original realization (Shynk, 1992). This particular way of computing the sliding window DFT is suitable for pipelined implementations, and it will be adopted in the sequel.

The D-FD LMS linear equalizer is summarized in Table 1.

Table 1. Delayed requency domain adaptive linear equalizer.

```
$f_{m+1}(n) = \rho e^{-j\frac{2\pi m}{M}} f_{m+1}(n-1) + x(n) - \rho^M x(n-M),$
        $m = 0,1,\dots,M-1,$
$\boldsymbol{p}_M(n) = \lambda \boldsymbol{p}_M(n-1) + (1-\lambda)\,\mathrm{diag}(|f_1(n)|^2,\cdots,|f_M(n)|^2),$
$\boldsymbol{F}_M(n) = \boldsymbol{p}_M^{-1}(n)\boldsymbol{f}_M(n),$
$y(n) = \boldsymbol{C}_M^H(n-1)\boldsymbol{f}_M(n),$
While in training mode
    $e(n) = I(n-\delta) - y(n),$
Otherwise
    $\hat{I}(n-\delta) = \mathrm{dec}\,(y(n)),$
    $e(n) = \hat{I}(n-\delta) - y(n),$
End While
$\boldsymbol{C}_M(n) = \boldsymbol{C}_M(n-1) + \mu_{\mathrm{DFDLMS}} \boldsymbol{F}_M(n-D) e^*(n-D).$
```

2.3. Division-Free Implementation. The division operations that appear in (15) can be implemented using the standard division circuitry, (Pirsch, 1998). Alternatively, a time-recursive division scheme, similar to those described in (Denyer and Renshaw, 1985; Thomas 1996), can be applied. The time-recursive division method approximates the division that appears in (15) using a first-order Taylor series, taken in conjunction with the recursive estimation of the reciprocal of (24), thus permiting the design of a simple, pipelined (approximate) division unit.

Let us consider the computations performed by (24), element-wise, $m = 0,1,\dots,M-1$, i.e.,

$$p_{m+1}(n) = \lambda p_{m+1}(n-1) + (1-\lambda)|f_{m+1}(n)|^2.$$

We define the reciprocal power variables

$$r_{m+1}(n) = \frac{1}{p_{m+1}(n)}, \quad m = 0,1,2,\dots,M-1. \tag{27}$$

Based on the above, we consider the computational scheme

$$F_{m+1}(n) = f_{m+1}(n) r_{m+1}(n). \tag{28}$$

The variables $r_{m+1}(n)$ can be efficiently estimated by a first-order approximation of the Taylor series expansion as

$$\begin{aligned} r_{m+1}(n) &\approx w r_{m+1}(n-1) \\ &\quad - w(w-1)|f_{m+1}(n) r_{m+1}(n-1)|^2, \end{aligned} \tag{29}$$

where $w = 1/\lambda$. Introducing further approximation in order to reduce the computations, we get

$$\begin{aligned} r_{m+1}(n) &\approx w r_{m+1}(n-1) \\ &\quad - w(w-1)|f_{m+1}(n-1) r_{m+1}(n-1)|^2 \\ &= w r_{m+1}(n-1) \\ &\quad - w(w-1)|F_{m+1}(n-1)|^2. \end{aligned} \tag{30}$$

Finally, a first-order delay relaxation is applied to the above formulae (Parhi, 1999) in order to facilitate pipelining,

$$r_{m+1}(n) \approx w r_{m+1}(n-1) - w(w-1)|F_{m+1}(n-2)|^2. \tag{31}$$

Equations (28) and (31) can be used instead of the original updating scheme imposed by Eqns. (24) and (15) for the estimation of the search vector $\boldsymbol{F}_M(n)$, thus allowing division-free implementation of the FD-D LMS adaptive linear equalizer.

2.4. Simulation Results. The performance of the proposed D-FD LMS adaptive linear equalizer is illustrated by a typical channel equalization experiment. Consider Channel (a) of (Proakis, 1985, p. 616), driven by an i.i.d. (independently identically distributed) binary input signal. The SNR ratio was set equal to 30 dB. The system's output is equalized by an adaptive linear equalizer of size $M = 31$. The equalization delay is set as to $\delta = 15$. The eigenenvalue spread of the input data autocorrelation

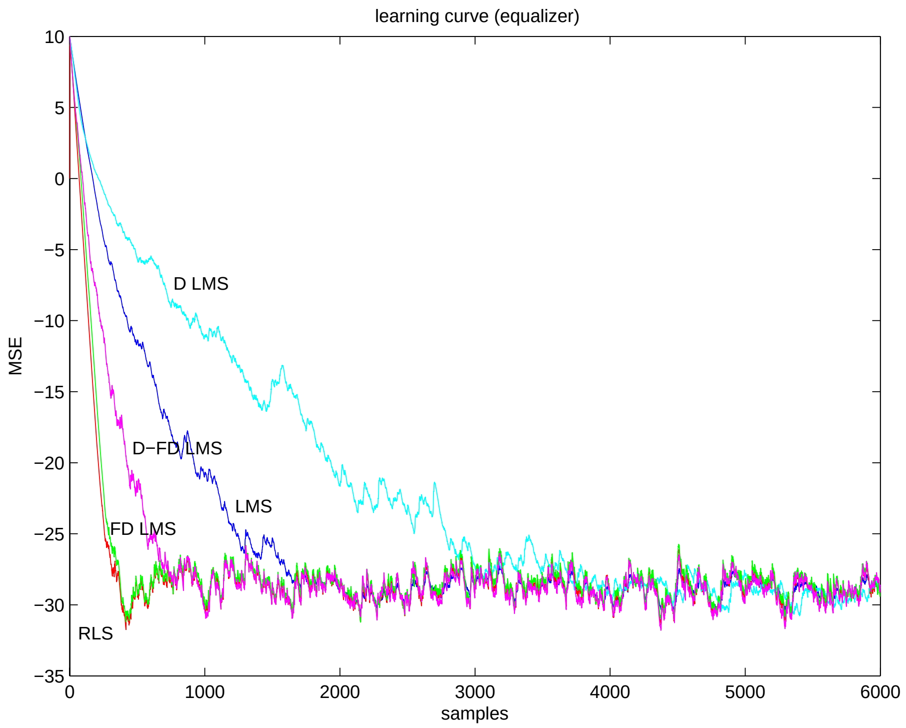


Fig. 2. MSE convergence rate for the RLS, LMS, D-LMS, FD-LMS and D-FD LMS adaptive equalizers.

matrix, $\boldsymbol{R}_M = \mathcal{E}\left(\boldsymbol{x}_M(n)\boldsymbol{x}^*_M(n)\right)$, estimated by means of the ratio of the maximum over the minimum eigenvalue of $\boldsymbol{R}_M$, was found to be approximately equal to 6.

Five adaptive algorithms were tested for the equalizer training, namely, RLS (Recursive Least Squares), LMS, delayed (D) LMS, FD LMS and D-FD LMS. The RLS algorithm was implemented using the standard exponentially weighed matrix inversion format (Haykin, 1996), and the corresponding exponentially forgetting factor was set to $\lambda_{RLS} = 0.992$. The amount of the adaptation delay introduced to both D-LMS and D-FD LMS adaptive algorithms was $D = 31$ time units. The tuning variable for the LMS algorithm was set to $\mu_{\mathrm{LMS}} = 0.005$, while the tuning variable for delayed LMS was fixed at $\mu_{\mathrm{DLMS}} = 0.002$. The tuning variable for the FD LMS algorithm was set to $\mu_{\mathrm{FDLMS}} = 0.01$, while the tuning variable for delayed FD LMS was set equal to $\mu_{\mathrm{DFDLMS}} = 0.005$. The forgetting factor λ that appears in (24) was set as $\lambda = 0.98$ and, finally, the stabilization factor ρ that appears in (26) was fixed at $\rho = 0.9999$. In all cases, the learning curve was used as a performance index, i.e., the Mean Squared Error (MSE) of the difference between the desired response signal $I(n-\delta)$ and the equalizer output $y(n)$,

$$J(n) = \boldsymbol{\mathcal{E}}\left(I(n-\delta) - y(n)\right)^2.$$

The expectation was computed by averaging the squared instantaneous estimation errors over an exponentially decaying window with an effective memory equal to 128 time instants. The learning curves for all the tested algorithms are depicted in Fig. 2. Clearly, the FD-LMS adaptive equalizer has almost the same performance as the RLS algorithm, which has much higher computational complexity, even when fast schemes are utilized (Haykin, 1996). Although the performance of the D-FD LMS adaptive equalizer has been affected by the presence of the adaptation delay in the error feedback loop, its convergence rate is much faster than that of its LMS counterpart. The D-LMS adaptive equalizer has the worst performance out of all the tested algorithms.

The performance of the proposed division-free implementation of the D-FD LMS equalizer is very close to that of the original D-FD LMS scheme. The approximate division method of (31) can be used instead of the original scheme imposed by (24) and (27) without affecting the overall performance of the algorithm. The error norm

$$J_P(n) = \frac{1}{M}\sum_{m=0}^{M-1} \boldsymbol{\mathcal{E}}\left(1/p_{m+1}(n) - r_{m+1}(n)\right)^2,$$

which expresses the difference between the inverse of $p_{m+1}(n)$ and its division-free estimate given by (31), is depicted in Fig. 3(b). Clearly, the mean approximate division error $J_P(n)$ remains at reasonable low levels after the initial convergence. The effect of the approximate division on the overall performance of the D-FD LMS algorithm is illustrated in Fig. 3(b), where the learning curve of the original D-FD LMS algorithm

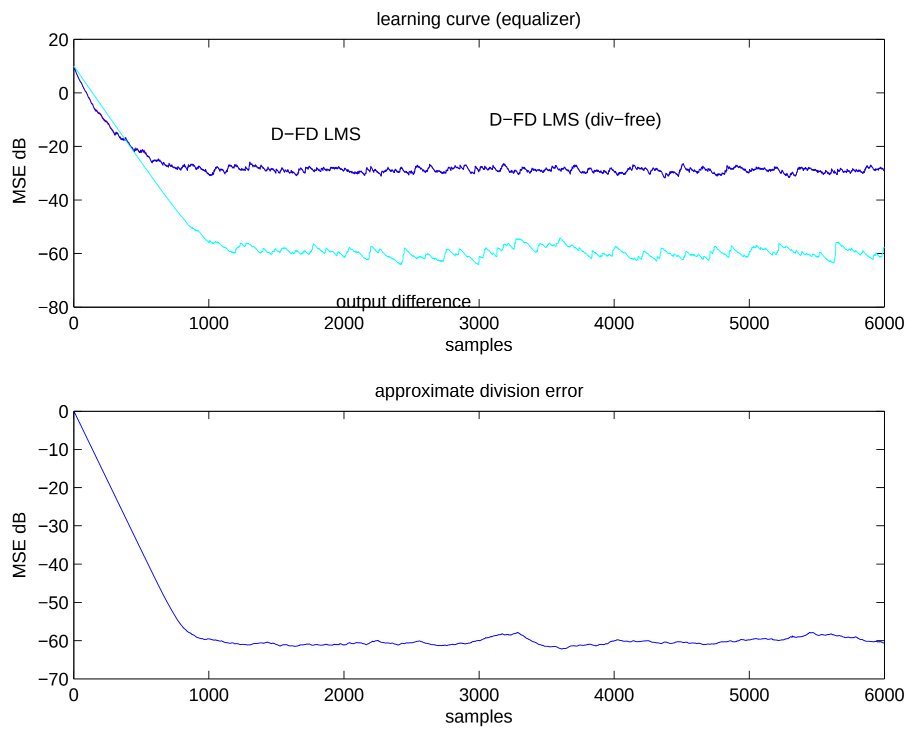


Fig. 3. MSE convergence rate for the division-free D-FD LMS adaptive equalizer.

and its division-free implementation, as well as the MSE of the difference between the two equalizers output, i.e., $\mathcal{E}\left(y_{\text{D-FD LMS}}(n) - y^{\text{div-free}}_{\text{D-FD LMS}}(n)\right)^2$, are depicted. Clearly, the learning curve of the division-free approach is (almost) identical to the original one.

Finally, the performance of the D-FD LMS adaptive equalizer with respect to the delay parameter D is illustrated in Fig. 4. It is clear that large values of the delay parameter D affect the convergence speed of the algorithm, although in all cases the D-FD LMS adaptive equalizer converges much faster than its D LMS counterpart.

3. Pipelined Architectures for the D-FD LMS Equalizer

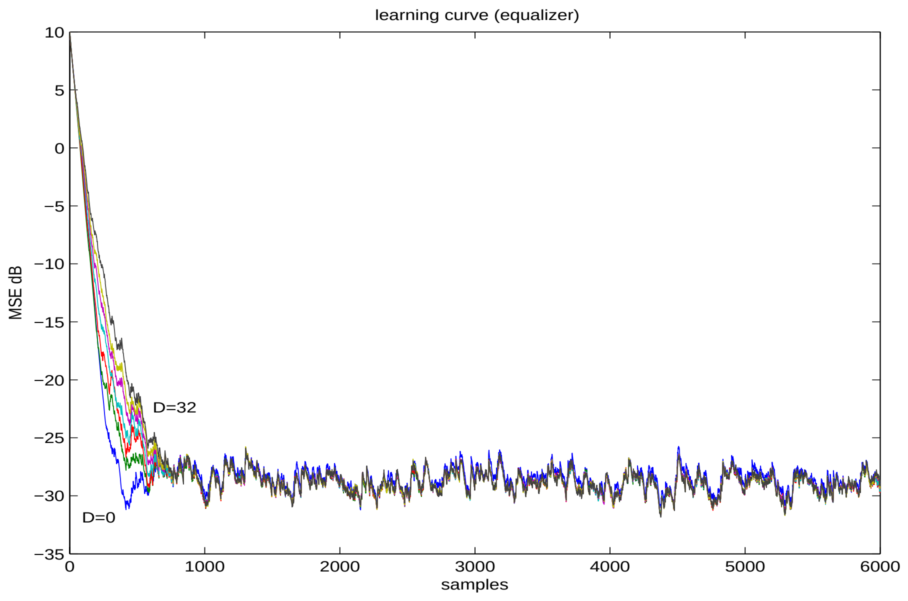
The data-flow graph of the D-FD LMS adaptive equalizer is depicted in Fig. 5. It is organized in a columnwise way, using a set of elementary processing units. Six types of processing elements (PE) are utilized, namely, P-1 up to P-6, each performing elementary complex operations. The sliding DFT is performed by PEs P-1 and P-2. The power normalization in the filtering domain is performed by P-3 and P-4. Finally, the filtering operation and the equalizer coefficients update are performed by P-6 and P-5, respectively. A detailed description of the computational tasks performed by each processing element is given in Table 2.

Table 2. Computational units of the D-FD LMS adaptive linear equalizer for division-free implementation.

$$\begin{aligned}
&\textbf{P-1}: \; u(n) = x(n) - \rho^M x(n-M)\\
&\qquad \text{FOR} \quad m = 0,1,\ldots,M-1,\\
&\textbf{P-2}: \; f_{m+1}(n) = \rho e^{-j\frac{2\pi m}{M}} f_{m+1}(n-1) + u(n),\\
&\textbf{P-3}: \; r_{m+1}(n) = w r_{m+1}(n-1)\\
&\qquad\qquad - w(w-1)|F_{m+1}(n-2)|^2,\\
&\textbf{P-4}: \; F_{m+1}(n) = f_{m+1}(n) r_{m+1}(n),\\
&\textbf{P-5}: \; C_{m+1}(n) = C_{m+1}(n-1)\\
&\qquad\qquad + \mu_{\text{DFDLMS}} F_{m+1}(n) e^*(n),\\
&\textbf{P-6}: \; y_{m+1}(n) = y_m(n) - C^*_{m+1}(n-1) f_{m+1}(n).
\end{aligned}$$

PEs P-1 to P-4 involve feedforward interconnections. Thus, the pipelining of these PEs can be achieved by placing delay latches in between. On the other hand, PEs 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. By retiming the delays existing in the error feedback loop, efficient pipelined implementations of the D-FD LMS adaptive equalizer are developed.

The filtering operation associated with the D-FD LMS adaptive equalizer is implemented by a set of M PEs, namely, P-6. $M-1$ consecutive additions have to be performed in order to compute the filter output $y(n)$. This particular set of operations results in a very long crit-



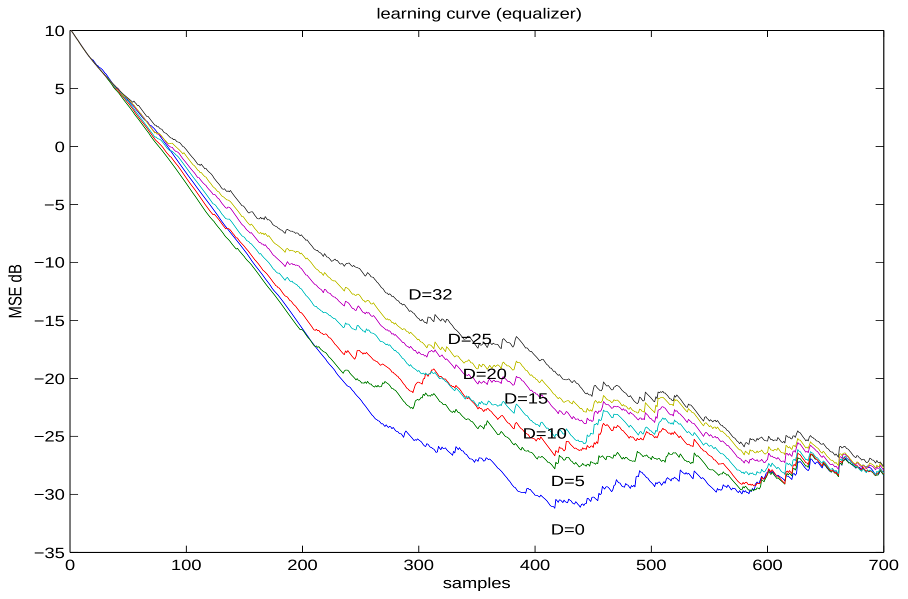


Fig. 4. MSE convergence rate for the D-FD LMS adaptive equalizer for different values of the adaptation delay ($D = 0, 5, 10, 15, 20, 25, 32$).

ical path that slows down the iteration period of the overall architecture. Reduction in the size of the critical path can be achieved by retiming the adaptation delays available at the error feedback loop. In this way, pipelined processing units of the short critical path are utilized, and the iteration period of the overall architecture is drastically reduced.

A pipelined architecture can readily be derived by replacing the serial additions implied in the computation of the filter output $y(n)$ by a binary tree adder scheme. The presence of the adaptation delay in the error feedback loop of the original data-flow graph (Fig. 5) can be used for efficient pipelining of the binary tree adder that estimates the error signal (see Fig. 6). The amount of adaptation delay that is required to fully pipeline the D-FD LMS algorithm is

$$D_1 = [\log_2(M)] + 1. \tag{32}$$

The pipelined architecture of the D-TD-LMS algorithm using a binary tree adder for the implementation of the filtering computations requires the smallest amount of adaptation delay, namely, $D = [\log_2(M)] + 1$. However, data broadcasting is required. The signals $u(n)$ and $e(n)$ are globally transmitted to all PEs P-2 and P-5 simultaneously. This structure introduces the smallest amount of the pipelining delay, and this property has to be taken into account when fast convergence speed and small output latency are of primary importance.

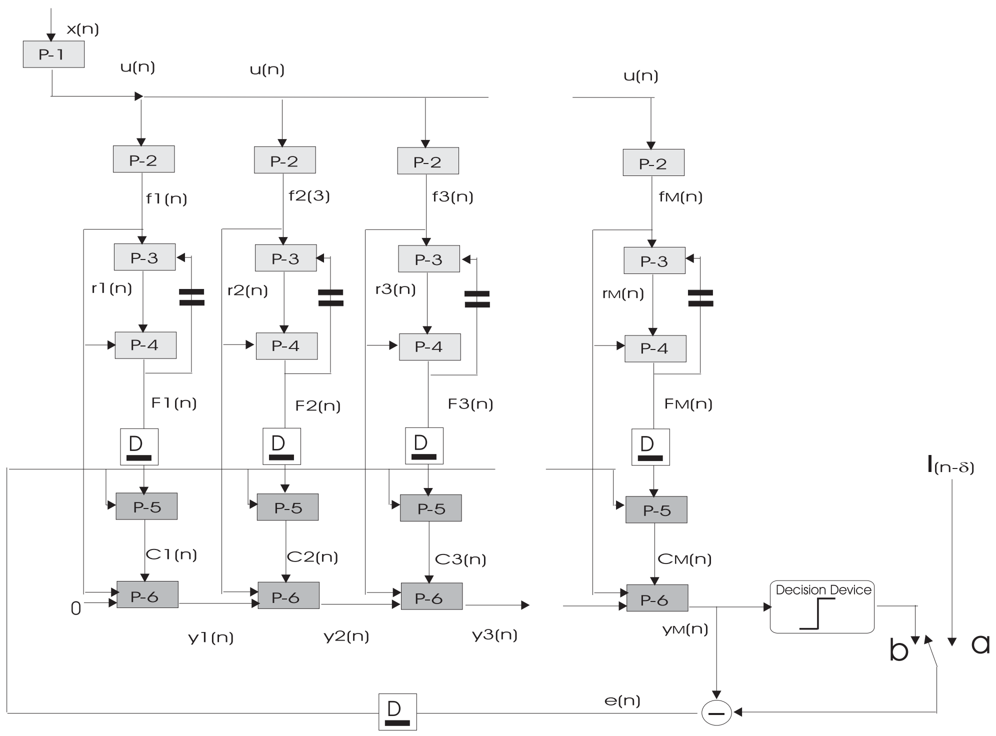


Fig. 5. Data flow graph for the D-FD LMS linear equalizer. During the training period, the switch is set to the position (a). When the equalizer operates in the decision directed mode, the switch is set to the position (b).

An alternative pipelined architecture for the D-FD LMS adaptive equalizer can be derived that avoids the use of a binary tree adder, thus allowing systolic implementation. The data-flow graph that is depicted in Fig. 5 consists of M identical columns of PEs. The adaptation delay units that appear in the error feedback loop are retimed by proper vertical cut sets. Specifically,

$$D_2 = M - 1 \tag{33}$$

adaptation delays are required for full pipelining of the D-FD LMS algorithm. The resulting architecture is shown in Fig. 7. The iteration period in this case is easily shown to be equal to the former architecture, i.e.,

$$T_{\text{pipe},2} = T_{\text{pipe},1}. \tag{34}$$

The output latency is $D_{O,2} = D_2 + 3$. The fully pipelined D-FD LMS adaptive equalizer has a modular structure and requires local data communication. It can be easily transformed into a locally recursive algorithm using the canonical mapping methodology and, hence, into efficient VLSI array processor implementation in an either systolic or wavefront architecture. However, it requires the maximum amount of adaptation delay. The large amount of adaptation delays required for pipelining affects the convergence speed and the tracking performance of the adaptive algorithm, and results in increased hardware requirements.

The computational complexity of the proposed pipelined frequency domain linear equalizer is subsequently analyzed for the case of complex valued input and output signals. The complexity is measured in terms of real valued multiplications and divisions (RVMD), assuming that complex valued multiplications are implemented by the standard method, where four real valued multipliers are engaged (Pirsch, 1998). Taking this fact into account, $16M$ nontrivial RVMDs are required for the implementation of the proposed algorithm (multiplication by the constants ρ and w can be replaced by digital shifts, provided that these values are selected to be a power of 2). On the other hand, fast RLS implementation by means of the fast (and pipelineable) adaptive lattice algorithm (Haykin, 1996) requires an amount of $37M$ RVMDs. The LMS and D-LMS adaptive schemes require $8M$ RVMDs (Long *et al.*, 1989; Parhi, 1999; Ramanathan and Visvanathan, 1999; Thomas, 1996), while the complexity of the modified D-LMS algorithms ranges between a minimum of $8M + 4\log_2(M)$ RVMDs and a maximum of $20M$ RVMDs, depending on the pipelining strategy adopted (Douglas *et al.*, 1998; Hadara *et al.*, 1998; Matsubara *et al.*, 1999). Notice that when the input signal and the desired response signal are real valued, the computational complexity of all methods discussed above is cut down approximately by a factor of 3. From the analysis conducted above, it is evident that the computational complexity of the proposed frequency domain linear equalizer

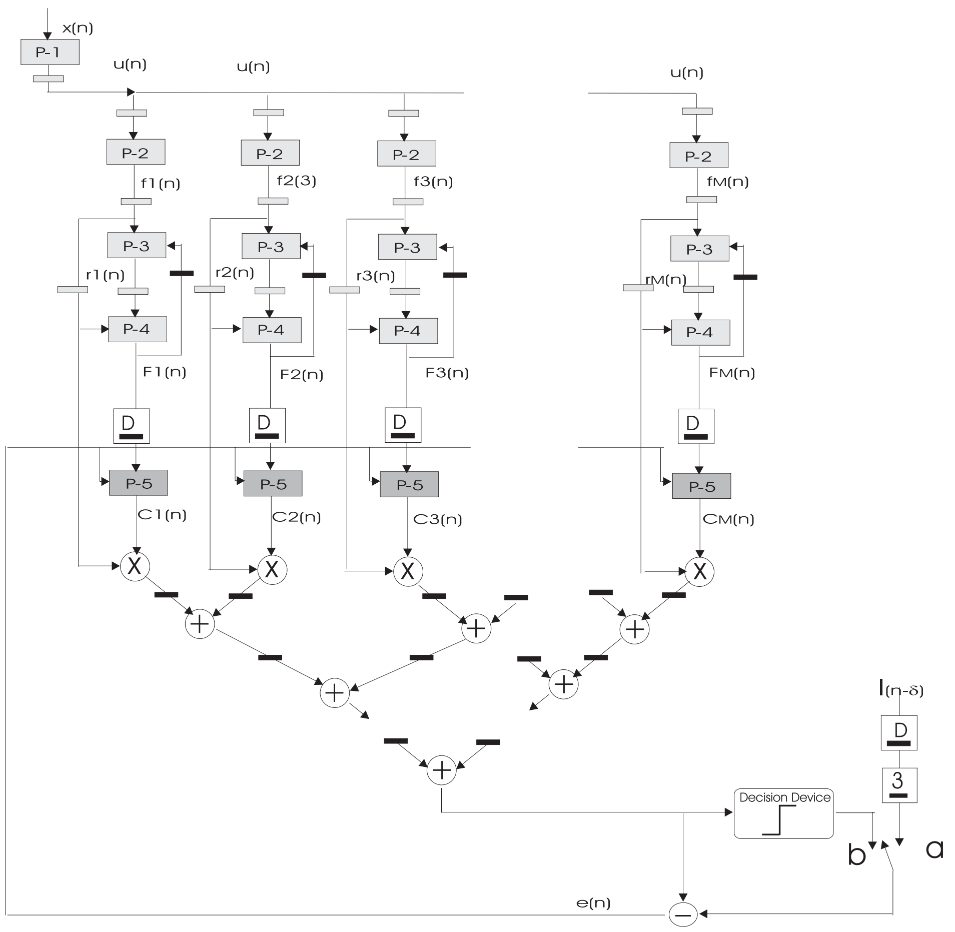


Fig. 6. Pipelined array architecture for the D-FD LMS linear equalizer. Small black and grey rectangular boxes denote unit time delays. Black or grey rectangular boxes, followed by an integer number or symbol, indicate multiple unit time delays.

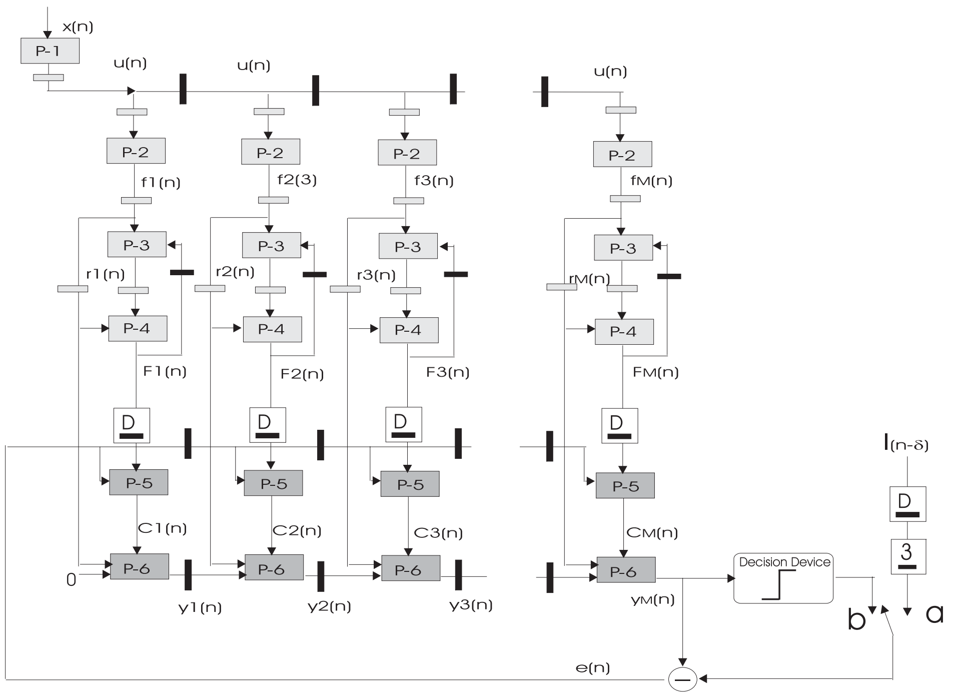


Fig. 7. Systolic architecture for the D-FD LMS linear equalizer.

is in between the low complexity LMS type schemes and the high complexity fast RLS methods.

The proposed architectures for efficient pipelining of the D FD-LMS linear equalization algorithm are suitable for implementation on special purpose hardware by means of the ASIC, ASIP or FPGA VLSI processors (Azadet and Nicole, 1998; Chen and Zhang, 2005; Kim *et al.*, 2003; Rofougaran *et al.*, 1998; Santha and Vaidehi, 2004; Ting *et al.*, 2005; Van and Feng, 2001; Yi and Woods, 2006).

4. Conclusion

In this paper, efficient architectures for pipelined implementation of the frequency domain LMS linear equalizer were considered. The pipelined operation of the algorithm was achieved by introducing a proper amount of adaptation delay to the original algorithm, resulting in a delayed frequency domain LMS scheme. By retiming the adaptation delay that was introduced in the error feedback loop, pipelined architectures were proposed that allow full pipelining of the algorithm. The resulting adaptation delay varies between the smallest value of $D = [\log_2(M)] + 1$ and the largest value of $D = M - 1$ time units. The critical path was reduced to $T_{\rm cri} = \tau_{\rm RMUL} + \tau_{\rm RADD}$ for all cases, where $\tau_{\rm RADD}$ and $\tau_{\rm RMUL}$ denote the times required for the computation of addition and multiplication, respectively. The proposed architectures are suitable for parallel implementations on dedicated hardware on the ASIC or ASIP VLSI processors.

References

Arslan H. and Bottomley G.E. (2001): *Channel estimation in narrowband wireless communication systems*. — Wireless Comm. Mobile Comput., Vol. 1, No. 2, pp. 201–219.

Azadet K. and Nicole C. (1998): *Low-power equalizer architectures for high-speed modems*. — IEEE Comm. Mag., Vol. 36, No. 10, pp. 118–126.

Benedetto S. and Biglieri E. (1999): *Principles of Digital Transmission: Width: Wireless Applications*. — New York: Kluwer.

Benvenuto N. and Tomasin S. (2001): *Frequency domain DFE: System design and comparison with OFDM*. — Proc. IEEE 8-th Symp. *Commun. and Vehic. Tech., SCVT*, Benelux, Delft, The Netherlands.

Berberidis K., Rantos S. and Palicot J. (2004): *A step-by-step quasi-Newton algorithm in the frequency domain and its application to adaptive channel equalization*. — IEEE Trans. Signal Process., Vol. 52, No. 12, pp. 3335–3344.

Bilcu R., Kuosmanen P. and Egiazarian K. (2002): *Channel equalization using a new transform domain LMS algorithm with adaptive step-size*. — WSEAS Trans. Circ., Vol. 1, No. 1, pp. 113–118.

Bilcu R., Kuosmanen P. and Egiazarian K. (2003): *Tracking time-varying channels with adaptive step-size transform domain LMS algorithm*, In: Recent Advances in Intelligent Systems and Signal Processing (Mastorakis N. *et al.*, Eds.). — Athens: WSEAS Press, pp. 104–109.

Chen S. and Zhang T. (2005): *Self-timed dynamically pipelined adaptive signal processing system: A case study of DLMS equalizer for real channel*. — IEEE Trans. Circuits Syst. I, Vol. 52, No. 7, pp. 1338–1347.

Denyer P. and Renshaw D. (1985): *VLSI Signal Processing. A bit serial approach*. — Boston, MA: Addison-Wesley.

Douglas S.C., Zhu Q. and Smith K. (1998): *A pipelined LMS adaptive FIR filter architecture without adaptation delay*. — IEEE Trans. Signal Process., Vol. 46, No. 3, pp. 775–779.

Farhang-Boroujeny B., Lee Y. and Ko C.C. (1996): *Sliding transforms for efficient implementation of transform domain adaptive filters*. — Signal Process., Vol. 52, pp. 83–96.

Glentis G. (2001): *Pipelined architectures for the TD LMS adaptive filter*. — Proc. IEEE Int. Conf. *Acoust. Speech, Signal Proc., ICASSP*, Salt Lake City, USA, pp. 1081–1084.

Glentis G. (2005): *Pipelined architectures for transform domain LMS adaptive filtering*. — J. Circ. Syst. Comput., Vol. 14, No. 3, pp. 553–580.

Glentis G., Berberidis K. and Theodoridis S. (1999): *Efficient least squares adaptive algorithms for FIR transversal filtering: A unified view*. — IEEE Signal Process. Mag., Vol. 16, No. 4, pp. 13–42.

Hadara A., Nishikawa K. and Kiya H. (1998): *Pipelined architecture of the LMS adaptive digital filter with the mimimum output latency*. — IEICE Trans. Fundam., Vol. E81-A, No. 8, pp. 1578–1584.

Haykin S. (1996): *Adaptive Filter Theory, 3rd Edition*. — New Jersey: Prentice Hall.

Huang Y. and Benesty J. (2003): *A class of frequency-domain adaptive approaches to blind multichannel identification*. — IEEE Trans. Signal Process., Vol. 51, No. 1, pp. 11–24.

Kalouptsidis N. and Theodoridis S. (1993): *Adaptive System Identification and Signal Processing Algorithms*. — Englewood Cliffs: Prentice Hall.

Kim C.H., Soeleman H. and Oy K. (2003): *Ultra-low-power DLMS adaptive filter for hearing aid applications*. — IEEE Trans. VLSI Syst., Vol. 11, No. 6, pp. 1058–1067.

Long G., Ling F. and Proakis J. (1989): *The LMS algorithm with delayed coefficients adaptation*. — IEEE Trans. Acoust. Speech Signal Process., pp. 1397–1405.

Maginot S., Balestro F., Joanblanq C., Senn P. and Palicot J. (1991): *A general-purpose high speed equalizer*. — IEEE J. Solid State Circ., Vol. 26, pp. 209–215.

Matsubara K., Nishikawa K. and Kiya H. (1999): *Pipelined LMS adaptive filter using a new look-ahead transformation*. — IEEE Trans. Circuits Syst. II, Vol. 46, No. 1, pp. 61–55.

Moreli M., Sanguinetti L. and Mengali U. (2005): *Channel estimation for adaptive frequency domain equalization*. — IEEE Trans. Wireless Comm., Vol. 4, No. 5, pp. 2508–2518.

Narayan S., Peterson A.M. and Narasimba M.J. (1983): *Transform domain LMS algorithm*. — IEEE Trans. Acoust. Speech, Signal Processing, Vol. 31, pp. 609–615.

Quereshi S.U.H. (1985): *Adaptive equalization*. — Proc. IEEE, Vol. 73, No. 9, pp. 1349–1387.

Parhi K. (1999): *VLSI Digital Signal Processing Systems: Design and Implementation*. — New York: Wiley.

Picchi G. and Prati G. (1984): *Self-orthogonalizing adaptive equalization in the discrete frequency domain*. — IEEE Trans. Commun., Vol. 32, No. 4, pp. 371–379.

Pirsch P. (1998): *Architectures for Digital Signal Processing*. — Chichester: Wiley.

Proakis J. (1995): *Digital Communications. 3-rd Ed*. — New York: McGraw-Hill.

Ramanathan S. and Visvanathan V. (1999): *Low-power pipelined LMS adaptive filter architectures with minimal adaptation delay*. — Integration VLSI, Vol. 27, No. 1, pp. 1–32.

Rofougaran A., Chang G., Rael J.J., Chang J. Y.-C., Rofougaran M., Chang P.J., Djafari M., Min J., Roth E.W., Abidi A.A. and Samueli H. (1998): *A single chip 900 MHz spread spectrum wireless transceiver in iμm CMOS. Parts I and II*. — IEEE J. Solid-State Circuits, Vol. 33, No. 4, pp. 515–547.

Santha K.R. and Vaidehi V. (2004): *Design of synchronous and asynchronous architectures for DFT based adaptive equalizer*. — Proc. IEEE Conf. *SoutheastCon*, Greensboro, NC, pp. 383–389.

Shamma M. (2002): *Improving the speed and performance of adaptive equalizers via transform based adaptive filtering*. — 14-th Int. Conf. *Digital Signal Processing, DSP*, Santorini-Hellas, Greece, Vol. 2, pp. 1301–1305.

Shanbhag N. and Im G.H. (1998): *VLSI systems design of 51.84 Mb/s transceivers for ATM-LAN and broadband access*. — IEEE Trans. Signal Process., Vol. 46, Issue 5, pp. 1403–1416.

Shynk J. (1992): *Frequency-domain and multirate adaptive filtering*. — IEEE Signal Process. Mag., Vol. 9, Issue 1, pp. 14-37.

Son S., Kim J., Lee Y., Kim H. and Park S. (2006): *Frequency-domain equalization for distributed terrestrial DTV transmission environments*. — IEEE Trans. Consum. Electron., Vol. 52, No. 1, pp. 59–67.

Thomas J. (1996): *Pipelined systolic architectures for DLMS adaptive filtering*. — J. VLSI Signal Process., Vol. 12, No. 3, pp. 223–246.

Ting L., Woods R. and Cowan C. (2005): *Virtex FPGA implementation of a pipelined adaptive LMS predictor for electronic support measures receivers*. — IEEE Trans. VLSI Syst., Vo. 13, No. 1, pp. 86–95.

Van L. and Feng W. (2001): *An efficient systolic architecture for the DLMS adaptive filter and its applications*. — IEEE Trans. Circ. Syst. II, Vol. 48, No. 4, pp. 359–366.

Yang Y., Park C. and Song J. (2004): *Fast constant modulus in the DFT domain*. — Proc. IEEE Conf. *Radio and Wireless, RAWCON2004*, Atlanta, GA, pp. 19–22.

Yi Y. and Woods R. (2006): *Hierarchical synthesis of complex DSP functions using IRIS*. — IEEE Trans. Computer. Aided Des. Integr. Circ. Syst., Vol. 25, No. 5, pp. 806–820.

Received: 8 November 2005
Revised: 16 September 2006