Pipelined Adaptive DFE Architectures Using Relaxed Look-Ahead

Naresh R. Shanbhag, *Member, IEEE,* and Keshab K. Parhi, *Senior Member, IEEE*

Abstract— **Fine-grain pipelined adaptive decision-feedback equalizer (ADFE) architectures are developed using the *relaxed look-ahead* technique. This technique, which is an approximation to the conventional *look-ahead computation*, maintains functionality of the algorithm rather than the input-output behavior. Thus, it results in substantial hardware savings as compared to either parallel processing or look-ahead techniques. Pipelining of the decision feedback loop and the adaptation loop is achieved by the use of *delay relaxation* and *sum relaxation*. Both the conventional and the predictor form of ADFE have been pipelined. Results of the convergence analysis of the proposed algorithms are also provided. The performance of the pipelined algorithms for the equalization of a magnetic recording channel is studied. It is shown that the conventional ADFE results in an SNR loss of about 0.6 dB per unit increase in the speed-up factor. The predictor form of ADFE is much more robust and results in less than 0.1 dB SNR loss per unit increase in the speed-up factor. Speed-ups of up to 8 and 45 have been demonstrated for the conventional and predictor forms of ADFE.**

I. Introduction

DESIGN OF DIGITAL signal processing (DSP) algorithms for high-throughput applications, such as video compression, are currently of great interest. In the area of digital communications, there is a growing need for high-speed equalizers for applications such as high-density magnetic storage systems, subscriber loop applications and mobile radio. The adaptive decision-feedback equalizer (ADFE) has been employed successfully for combating inter-symbol interference (ISI). However, the ADFE has remained difficult to pipeline and in this paper we propose a novel approach for fine-grain pipelining of the ADFE.

Two popular approaches for achieving high processing speed are pipelining [1] and parallel processing [2]. From a single chip implementation point of view the pipelining approach holds a distinct advantage due do its lower hardware cost. Recently, the utility of pipelined systems in low-speed, low-power applications such as speech codecs in mobile phones and other portable applications has also been observed [3]. Conventionally, algorithm transformation techniques [4] such as *look-ahead* have been employed to introduce concurrency in serial algorithms. Originally proposed for pipelining of recursive fixed-coefficient filters [1], [5], [6], the look-ahead technique has been successfully applied to pipeline two-dimensional recursive filters [7], dynamic programming [8], [9], finite state machines [10], quantizer loops [11], and adaptive digital filters [12], [13]. The implementation of a recursive filter chip capable of 86 million operations per second [14] has clearly demonstrated the feasibility of fine-grained pipelining.

The look-ahead technique, however, results in a large hardware overhead as it transforms a serial algorithm into an equivalent (in the sense of input–output behavior) pipelined algorithm. While past research demonstrated the use of look-ahead in the design of pipelined recursive signal processing algorithms, our current research is concerned with the design of inherently pipelinable algorithms. These algorithms have more than one delay element in any feedback loop and therefore can be pipelined without requiring any hardware increase as compared with the sequential ones. To this end we have developed the *relaxed look-ahead* technique [15] for the pipelining of adaptive digital filters. The relaxed look-ahead sacrifices the equivalence between the serial and pipelined algorithms at the expense of marginally altered convergence characteristics. Therefore, it maintains the functionality of the algorithm and is well suited for adaptive filtering applications.

Relaxed look-ahead involves approximating the algorithms obtained via look-ahead. A number of approximations are possible and each would result in a different algorithm. For example, in this paper we present the *delay relaxation* and the *sum relaxation* as two possible approximations, which can be used for pipelining of the ADFE. In the context of adaptive filtering, the approximations can be quite crude and yet result in minimal performance loss. However, in all cases, the resulting pipelined algorithm requires minimal hardware increase and achieves a higher throughput or requires lower power as compared to the serial algorithm. Note that, the relaxed look-ahead has already been employed for the pipelining of the least mean-squared (LMS) filter [16], the stochastic gradient lattice filter [17], the adaptive differential vector quantizer [18], and the adaptive differential pulse code modulation codec [16].

As mentioned before, fine-grain pipelining of the ADFE is known to be a difficult problem. This is mainly due to the fact that the ADFE has a nonlinear element (a quantizer) in the decision-feedback loop (DFL). The conventional ADFE (to be referred to as ADFE) in Fig. 1(a) and the predictor form of ADFE (to be referred to as predictor ADFE) in Fig. 1(b) consist of the feedforward filter (FFF), the feedback

Manuscript received September 8, 1993; revised November 10, 1994. This research was supported by the Army Research Office under Contract DAAL-90-G-0063. The associate editor coordinating the review of this paper and approving it for publication was Dr. J. Zeider.

N. R. Shanbhag is with AT&T Bell Laboratories in Murray Hill, NJ 07974 USA.

K. K. Parhi is with the Department of Electrical Engineering, University of Minnesota, Minneapolis, MN 55455 USA.

IEEE Log Number 9411201.

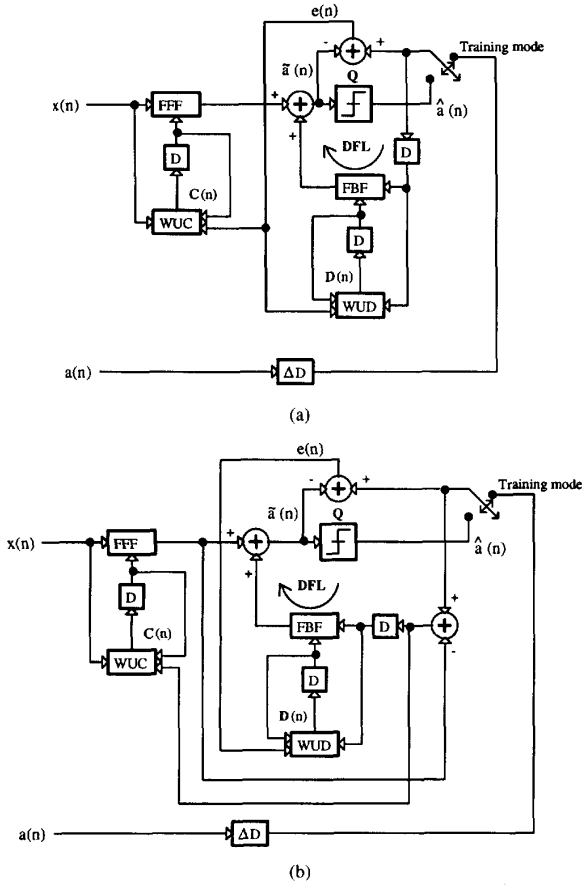


Fig. 1. Serial ADFE architectures: (a) Conventional; (b) predictor form.

filter (FBF), the quantizer (Q), and the coefficient update blocks WUC (for FFF) and WUD (for FBF). The delays Δ are employed to adjust the position of the main tap of FFF, which is usually the center-most tap. In addition to the DFL, the presence of the adaptation loop makes it even more difficult to achieve pipelining. Hence, past work [19]–[22] in high-speed ADFE architectures have almost exclusively adopted parallelization. In general, the algorithms in [20]–[22] result in a performance loss due to incorrect initialization of the feedforward filter (FFF) and a coding loss due to the initialization of feedback filter (FBF). The ADFE architecture in [21] has an advantage of offering a performance gain as it decomposes the channel into a set of parallel ISI free channels. However, it requires the knowledge of the channel coefficients. In [23], circuit and architectural techniques, such as the use of the transpose form for the FFF and the FBF, have been employed to achieve high-speed in a DFE. A VLSI implementation of an ADFE with two delays in the DFL has been proposed [24]. Performance degradation was observed in [24] due to the fact that the FBF cannot cancel the most significant ISI term.

From the discussion above it is clear that a fine-grain pipelined ADFE algorithm with minimal hardware overhead and a negligible performance loss would be desirable. In this paper, we employ relaxed look-ahead to develop such algorithms. To this end we first generalize an existing straightforward pipelining approach to obtain the pipelined ADFE architecture referred to as PIPADFE1 (to be read as Pipe ADFE1). Then we propose two new pipelined ADFE architectures referred to as PIPADFE2 and PIPADFE3 which are pipelined versions of the conventional and the predictor ADFE structures, respectively. We conclude that PIPADFE2 can converge much faster than PIPADFE1 and is an attractive alternative for implementation of conventional ADFE algorithms. Furthermore, we also show that PIPADFE3 is a robust pipelined architecture with respect to level of pipelining or speed-up factor. These algorithms require much smaller hardware overhead and are attractive from VLSI implementation point of view. In addition, these algorithms do not suffer from any coding loss. Similar to the parallel architectures [20], [22], the pipelined architectures do suffer from performance degradation as the speed-up increases although for a different reason. This degradation is mainly due to the coarseness of the approximations made in applying the relaxed look-ahead. Therefore, by improving these approximations it is possible to reduce the performance loss. This is a desirable flexibility offered by relaxed look-ahead. The performance of the proposed algorithms is compared with a generalized version of the algorithm proposed in [24].

This paper is organized as follows. In Section II, we present the relaxed look-ahead technique, which is then applied to pipeline the ADFE in Section III. In Section IV, we analyze the performance of the pipelined algorithms and compare them with that of the serial algorithm. Convergence analysis results are presented in Section V. Simulation results are presented for the equalization of a magnetic recording channel in Section VI.

II. The Relaxed Look-Ahead

In this section, we introduce the relaxed look-ahead as an approximation to the look-ahead. In order to do this, consider the following equations, which describe a linear adaptive estimator with a first-order weight-update recursion

$$\mathbf{W}(n) = \mathbf{W}(n-1) + \mu e(n)\mathbf{X}(n) \tag{1a}$$

$$e(n) = s(n) - \mathbf{W}^T(n-1)\mathbf{X}(n) \tag{1b}$$

where $\mathbf{W}(n)$ is a $N \times 1$ vector of coefficients of the filter FIR (see Fig. 2(a)), μ is the adaptation step-size, $e(n)$ is the estimation error, $\mathbf{X}(n)$ is the $N \times 1$ input vector, and $s(n)$ is the desired signal. The first-order recursion (1a) also describes the weight-update recursion of the ADFE. Therefore, the relaxed look-ahead pipelining results discussed in this section in the context of this first-order recursion can be directly applied to pipeline the ADFE (see Section III).

From Fig. 2(a) (and (1)), we can see that there are two major feedback paths, which present a bottleneck for high throughput applications. The first is called the error feedback path, which consists of the filter FIR, the adder, and the weight-update block WUC. The second path is the weight-update recursion defined by (1a). In order to break this bottleneck, we can apply the look-ahead [1] pipelining technique. An M-stage pipelined algorithm can be derived from (1) by the application of an

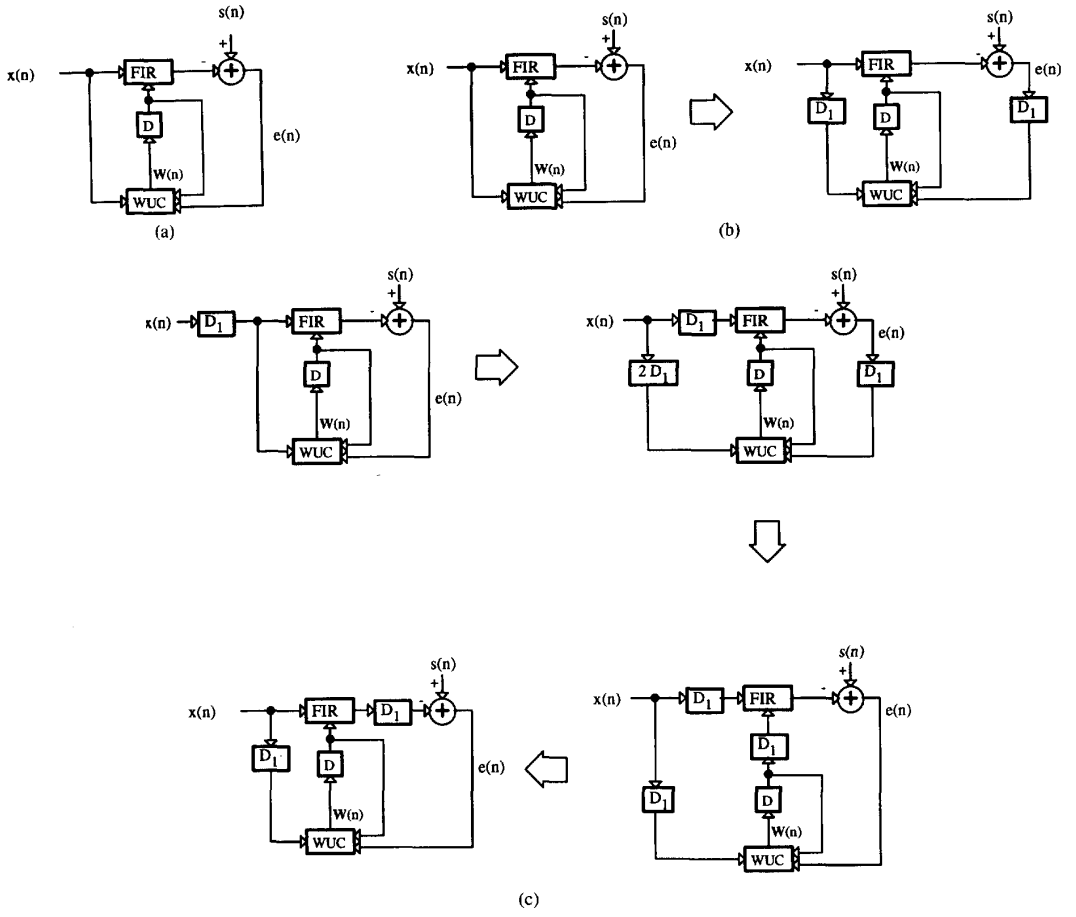


Fig. 2. Delay relaxations: (a) Original system; (b) delay relaxation; (c) its alternate form.

M -step look-ahead to (1). It can be easily checked that the hardware required to do so is quite large because this process involves computing $\mathbf{W}(n)$ from $\mathbf{W}(n-M)$. Note that the look-ahead transformation results in M latches in the recursive loop. These latches can be redistributed or *retimed* [25] to pipeline the feedback multiply-add operation by M levels.

However, by considering the error feedback path and weight-update recursion separately, we can pipeline the adaptive estimator in a hardware efficient manner. In particular, we pipeline the error feedback path by the delay relaxation, while the weight-update recursion is pipelined by the sum relaxation.

A. Delay Relaxation

The delay relaxation is shown in Fig. 2(b), where the error $e(n)$ and the input $\mathbf{X}(n)$ are delayed by D_1 samples before being employed in the WUC. This transformation is made on the basis of the assumption that the gradient estimate $e(n)\mathbf{X}(n)$ does not change substantially over D_1 clock-cycles. The delay relaxation has been employed in [26], [27] to develop the "delayed LMS" algorithm. A thorough convergence analysis of this kind of delayed adaptation scheme was also done in [26], [27]. It was concluded that, in a stationary environment and with a small step-size, the degradation in convergence speed and adaptation accuracy is negligible for small delays. As we shall see later, the degradation in convergence behavior is also dependent on the algorithm topology. However, from an architectural point of view the delay relaxation is an effective method for pipelining. This is because the D_1 delays can now be employed to pipeline the FIR and hardware overhead is just the pipelining latches. Note that the "delayed LMS" algorithm is a special case of the filtered X-LMS algorithm [28].

Another application of delay relaxation is illustrated in Fig. 2(c), where pipelining by placing latches at the inputs of the system is combined with the delay relaxation. This structure is retimed further to obtain the transformed system. As mentioned before, the structure in Fig. 2(c) can also be obtained from the filtered X-LMS algorithm [28] by replacing the plant $P(z)$ by D_1 delays. The delay relaxation can be justified as after convergence the weights do not change much. This implies that some degradation in performance is to be expected while the filter is converging. Note that the D_1 delays would be redistributed to pipeline the FIR.

Redistribution of D_1 delays results in the desired signal $s(n)$ being delayed by a certain amount D', where $D' \leq D_1$.

In Fig. 2, D' equals the number of delays needed to pipeline FIR. In data communications applications, FIR is an equalizer and $e(n)$ is the input to a slicer (or quantizer). In an echo-cancellation scenario, $e(n)$ is the sum of residual echo and the received signal. This leads to an additional end-to-end delay of D' symbol periods (assuming baud-rate processing) for both cases and this is usually not a problem. However, if the FIR is the FBF in an ADFE, any delay in its output implies that the FBF is unable to cancel D' post-cursor ISI terms and would result in a performance degradation if it is not compensated for.

B. Sum Relaxation

Even though the delay relaxation is sufficient to pipeline the FIR and part of the WUC, the weight-update loop (1a) remains to be pipelined. The computation time of (1a) is lower bounded by a single add time. In order to reduce this lower bound, we apply a D_2-step look-ahead to (1a) to obtain

$$\mathbf{W}(n) = \mathbf{W}(n - D_2) + \mu \sum_{i=0}^{D_2-1} e(n-i)\mathbf{X}(n-i). \quad (2)$$

Note that due to the inter-dependence of $e(n)$ and $\mathbf{W}(n-1)$, an exact implementation of $\mathbf{W}(n)$ in terms of $\mathbf{W}(n - D_2)$ only would result in a large computational complexity. Hence, we implicitly assume that (2) will be employed along with the delay relaxation.

In (2), the summation term represents the overhead. However, instead of taking the sum of D_2 terms in (2), we may retain only LA terms to get

$$\mathbf{W}(n) = \mathbf{W}(n - D_2) + \mu \sum_{i=0}^{LA-1} e(n-i)\mathbf{X}(n-i) \quad (3)$$

where the partial look-ahead factor LA maybe either less than or equal to D_2. The replacement of D_2 sum terms in (2) to LA sum terms in (3) is referred to as the *sum relaxation*. Note that the summation in (3) can be realized by computing the product $e(n)\mathbf{X}(n)$ and then passing it through an FIR filter whose coefficients are all equal to unity. This FIR filter can be realized in an equivalent transpose form. In that case, the computational delay for the summation would be independent of LA. There is, however, an overhead of $N(LA)$ adders for this relaxation.

Each of the relaxations can be applied individually or in combination. Therefore, unlike the conventional look-ahead technique [1], the relaxed look-ahead results in a rich variety of architectures. Depending on the application at hand certain approximations may be more appropriate than others. In addition, by tuning the adaptation parameters (such as the step-size, etc.) the degradation in performance due to the approximations can be minimized.

The pipelined architecture resulting from the application of relaxed look-ahead can always be clocked at a higher speed than the original one. This increase in throughput (also referred to as the *speed-up*), which is in direct proportion to the number of pipeline stages, can be exploited in many ways. If the input sample-rate is increased to match the throughput then an overall improvement in the absolute convergence time is possible. This will occur if the degradation in the convergence speed due to the relaxations is by a smaller factor than the speed-up. As we shall see later, this is true for all the pipelined architectures presented here. If the input sample-rate does not need to be increased, then the increase in throughput can be traded-off with area, by employing systematic *folding* transformations [29], or power [3].

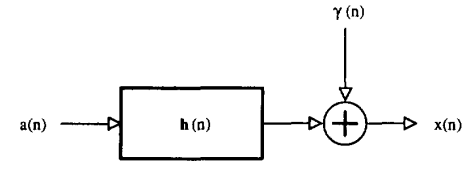


Fig. 3. Channel model.

III. Pipelined ADFE Architectures

In this section, we employ the relaxed look-ahead technique, which was described in the previous section, to develop three pipelined ADFE architectures. For the sake of simplicity and to demonstrate the technique of relaxed look-ahead, we only consider the equalization of channels with linear ISI. This, however, does not preclude the application of relaxed look-ahead to pipeline equalizers such as the RAM-DFE [30], which has been applied successfully to cancel nonlinear ISI. The channel model we consider is shown in Fig. 3, where $a(n)$ is the channel input at time instance n, $\mathbf{h}(n)$ is the channel coefficient vector, $\gamma(n)$ is white Gaussian noise and $x(n)$ is the received sample.

The first algorithm, referred to as PIPADFE1 (pipelined ADFE1), is an extension of the algorithm proposed in [24]. This algorithm has been derived simply for the sake of comparison with the other architectures. The second pipelined ADFE algorithm (PIPADFE2) is derived by the application of relaxed look-ahead to the ADFE in Fig. 1(a), while the third algorithm (PIPADFE3) is developed from the predictor ADFE (see Fig. 1(b)). For simplicity, the pipelined algorithms are derived assuming correct quantizer decisions.

First, we introduce some terminologies to define the equations which describe the serial ADFE of Fig. 1(a).

$$\tilde{a}(n) = a_F(n) + a_B(n) \quad (4a)$$

$$a_F(n) = \mathbf{C}^T(n-1)\mathbf{X}(n) \quad (4b)$$

$$a_B(n) = \mathbf{D}^T(n-1)\hat{\mathbf{a}}(n-1) \quad (4c)$$

$$e(n) = \hat{a}(n) - \tilde{a}(n) \quad (4d)$$

$$\hat{a}(n) = \mathbf{Q}[\tilde{a}(n)] \quad (4e)$$

$$\mathbf{W}(n) = \mathbf{W}(n-1) + \mu e(n)\mathbf{U}(n) \quad (4f)$$

where $a_F(n)$ is the output of FFF, $a_B(n)$ is the output of the FBF, $\mathbf{C}(n)$ is the vector of FFF coefficients, $\mathbf{D}(n)$ is the vector of FBF coefficients, $\mathbf{X}(n)$ is the vector of received samples, $\hat{\mathbf{a}}(n)$ is the vector of detected symbols, $\tilde{a}(n)$ is the input to quantizer Q and $\hat{a}(n)$ is the quantizer decision. In addition, let N_F and N_B represent the number of taps of the FFF and FBF, respectively. Note that (4f) is of the same form as (1a) and represents the familiar least mean-squared (LMS) algorithm

with μ being the adaptation step-size. The vector $\mathbf{W}(n)$ is the combined coefficient vector defined as

$$\mathbf{W}^T(n) = [\mathbf{C}^T(n) \quad \mathbf{D}^T(n)]. \tag{5}$$

The data vector $\mathbf{U}(n)$ is given by

$$\mathbf{U}^T(n) = [\mathbf{X}^T(n) \quad \hat{\mathbf{a}}(n-1)^T]. \tag{6}$$

Note that when correct decisions are made by the quantizer then

$$\hat{a}(n) = a(n-\Delta). \tag{7}$$

A. The PIPADFE1 Algorithm

The FFF and FBF can be pipelined by delaying their inputs by D_1 samples and then applying the form of delay transfer relaxation shown in Fig. 2(c). We shall see later that introducing D_1 delays in the DFL results in a substantial performance degradation because the FBF cannot employ past decisions to cancel the D_1 most-significant ISI terms. This fact has been observed in [24], where $D_1 = 2$ delays were introduced in the DFL. These steps result in (4a) being modified to

$$\tilde{a}(n) = a_F(n-D_1) + a_B(n-D_1) \tag{8}$$

while (4f) is modified to

$$\mathbf{W}(n) = \mathbf{W}(n-1) + \mu e(n)\mathbf{U}_1(n) \tag{9}$$

where the new data vector $\mathbf{U}_1(n)$ is defined as

$$\mathbf{U}_1^T(n) = [\mathbf{X}^T(n-D_1) \quad \hat{\mathbf{a}}(n-1)^T] \tag{10}$$

with $\hat{a}(n) = a(n-D_1-\Delta)$ for correct quantizer decisions. Applying sum relaxation to (9) results in

$$\mathbf{W}(n) = \mathbf{W}(n-D_2) + \mu \sum_{i=0}^{LA-1} e(n-i)\mathbf{U}_1(n-i). \tag{11}$$

Finally, we approximate (4b) by employing (11) as follows

$$\begin{aligned}\tilde{a}_F(n) &= \mathbf{C}^T(n-1)\mathbf{X}(n) \\ &= \Bigg[\mathbf{C}^T(n-D_2-1) \\ &\quad + \mu \sum_{i=0}^{LA-1} e(n-i-1)\mathbf{X}(n-D_1-i-1)\Bigg]\mathbf{X}(n) \\ &\approx \mathbf{C}^T(n-D_2)\mathbf{X}(n).\end{aligned} \tag{12}$$

Note that in deriving (12), we have assumed that the step-size μ is sufficiently small. In a similar fashion, we can approximate (4c) as

$$\tilde{a}_B(n) \approx \mathbf{D}^T(n-D_2)\hat{\mathbf{a}}(n-1). \tag{13}$$

These steps result in an algorithm which is a generalization of the algorithm presented in [24]. The equations describing

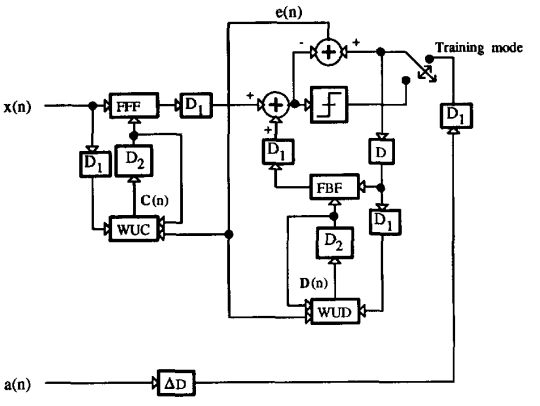


Fig. 4. PIPADFE1 architecture.

PIPADFE1 (see Fig. 4) are

$$\tilde{a}(n) = a_F(n-D_1) + a_B(n-D_1) \tag{14a}$$

$$a_F(n) = \mathbf{C}^T(n-D_2)\mathbf{X}(n) \tag{14b}$$

$$a_B(n) = \mathbf{D}^T(n-D_2)\hat{\mathbf{a}}(n-1) \tag{14c}$$

$$e(n) = \tilde{a}(n) - \hat{a}(n) \tag{14d}$$

$$\hat{a}(n) = \mathbf{Q}[\tilde{a}(n)] \tag{14e}$$

$$\mathbf{W}(n) = \mathbf{W}(n-D_2) + \mu \sum_{i=0}^{LA-1} e(n-i)\mathbf{U}_1(n-i) \tag{14f}$$

with $\mathbf{U}_1(n)$ defined as in (10). The hardware overhead for PIPADFE1 are the pipelining latches and $(LA-1)(N_F+N_B)$ adders. In order to demonstrate the increase in throughput due to pipelining, we consider the serial ADFE in Fig. 5(a), where the FFF and FBF are of second order and D represents a unit delay. Assuming a multiply time of 40 units and an add time of 20 units, it can be easily checked that the critical path in this architecture has a delay of 200 units. This implies that the input samples can be processed at a rate of once every 200 units. With $D_1 = 5$, we can pipeline the multiply-adds such that the critical path delay is reduced to 40 units. The retimed PIPADFE1 architecture, which operates five times faster than the serial ADFE, is shown in Fig. 5(b). Note that the architecture in [24] is a special case of PIPADFE1 with $D_1 = 2$ and $D_2 = 1$.

B. The PIPADFE2 Algorithm

Instead of introducing D_1 delays in the DFL directly as in the case of PIPADFE1, the PIPADFE2 is derived by pipelining the DFL via relaxed look-ahead. The computations of the DFL can be written as

$$\tilde{a}(n) = \mathbf{C}^T(n-1)\mathbf{X}(n) + \sum_{i=0}^{N_B-1} d_i(n-1)\hat{a}(n-i-1) \tag{15}$$

where $d_i(n)$'s are the FBF coefficients and N_B is its order. In order to apply look-ahead, we can either linearize the DFL or employ the technique in [11]. In order to minimize the hardware, we choose the former approach. Note that the

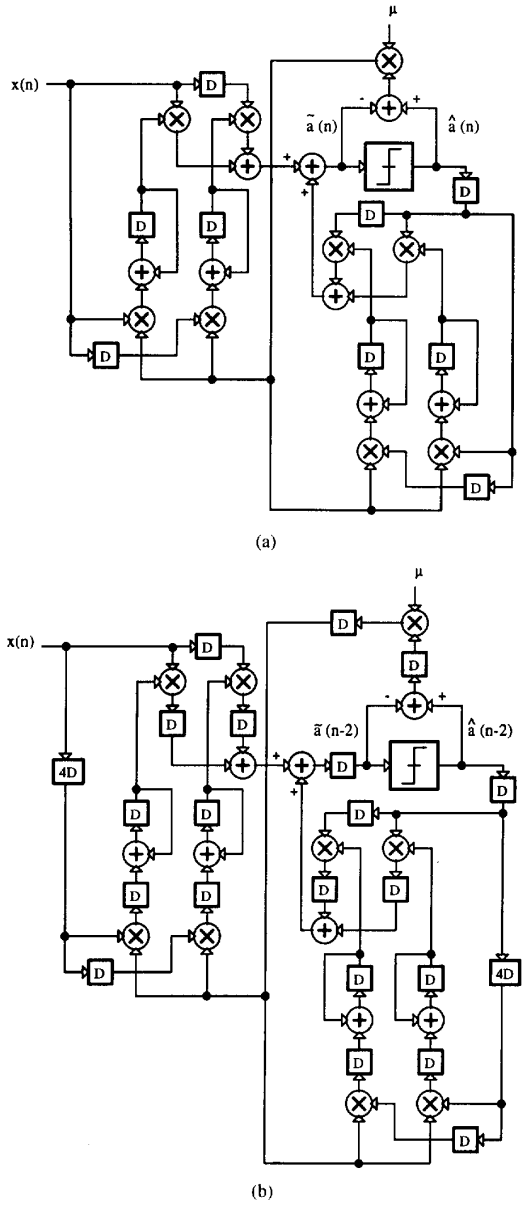


Fig. 5. Pipelining example: (a) Serial ADFE; (b) PIPADFE1 with a speedup of 5.

linearization of the DFL is simply an intermediate step in the process of developing PIPADFE2. We assume (1) $d_i(n)$'s and $c_i(n)$'s vary slowly, and (2) the quantization error is small, i.e., $\tilde{a}(n) \approx \hat{a}(n)$. Therefore, (15) can be approximated as

$$\begin{aligned}\tilde{a}(n) &= \mathbf{C}^T\mathbf{X}(n) + \sum_{i=0}^{N_B-1} d_i\tilde{a}(n-i-1) \\ &= \mathbf{C}^T\mathbf{X}(n) + \sum_{i=0}^{D_1-1} d_i\tilde{a}(n-i-1) \\ &\quad + \sum_{i=D_1}^{N_B-1} d_i\tilde{a}(n-i-1). \end{aligned} \tag{16}$$

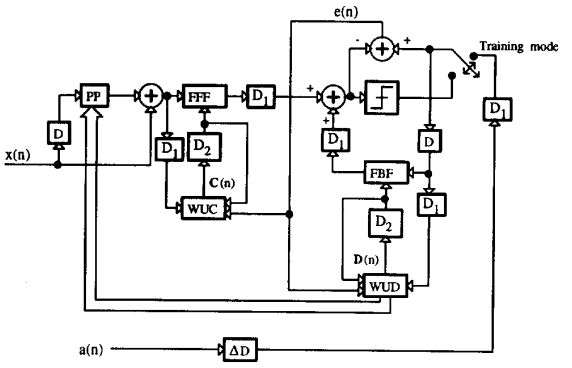


Fig. 6. PIPADFE2 architecture.

Note that we have implicitly assumed that $N_B - 1 > D_1$. We will now apply look-ahead to (16) and then approximate the resulting expression. Employing (16) itself to substitute for $\tilde{a}(n-i-1)$ in the first summation in (16), we get

$$\begin{aligned}\tilde{a}(n) &= \mathbf{C}^T\left[\mathbf{X}(n) + \sum_{i=0}^{D_1-1} d_i\mathbf{X}(n-i-1)\right] \\ &\quad + \sum_{i=0}^{D_1-1}\sum_{i_1=0}^{N_B-1} d_i d_{i_1}\tilde{a}(n-i-i_1-2) \\ &\quad + \sum_{i=D_1}^{N_B-1} d_i\tilde{a}(n-i-1) \\ &= \mathbf{C}^T\left[\mathbf{X}(n) + \sum_{i=0}^{D_1-1} d_i\mathbf{X}(n-i-1)\right] \\ &\quad + \left\{\sum_{i=0}^{D_1-1}\sum_{i_1=0}^{D_1-1} d_i d_{i_1}\tilde{a}(n-i-i_1-2)\right\} \\ &\quad + \left[\sum_{i=0}^{D_1-1}\sum_{i_1=D_1}^{N_B-1} d_i d_{i_1}\tilde{a}(n-i-i_1-2)\right. \\ &\quad \left. + \sum_{i=D_1}^{N_B-1} d_i\tilde{a}(n-i-1)\right]. \end{aligned} \tag{17}$$

In (17), the variable $\tilde{a}(n-1)$ has been eliminated. Similarly, we can carry on this process by substituting for $\tilde{a}(n-i-i_1-2)$ (in (17)) in the summation enclosed within braces.

Repeating the above process of repeated substitution until variables $\tilde{a}(n-1)$ to $\tilde{a}(n-D_1)$ are eliminated, we get

$$\tilde{a}(n) = \mathbf{C}^T A + B \tag{18}$$

where A represents the computations of the data preprocessing section (PP) defined as

$$\begin{aligned}A &= \mathbf{X}(n) + \sum_{i=0}^{D_1-1} d_i\mathbf{X}(n-i-1) \\ &\quad + \sum_{i=0}^{D_1-1}\sum_{i_1=0}^{D_1-1} d_i d_{i_1}\mathbf{X}(n-i-i_1-2)\end{aligned}$$

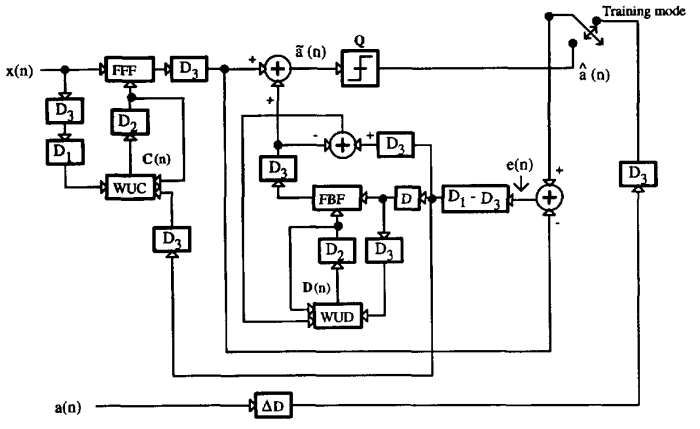


Fig. 7. PIPADFE3 architecture.

$$+\cdots+\sum_{i=0}^{D_1-1}\sum_{i_1=0}^{D_1-1}\cdots\sum_{i_{D_1}=0}^{D_1-1} d_i d_{i_1}\cdots d_{i_{D_1}} \cdot \mathbf{X}(n-i-i_1-\cdots-i_{D_1}-D_1-1) \tag{19}$$

and B represents the computations of the feedback section as shown below

$$B=\sum_{i=0}^{N_B-D_1-1} d_{D_1+i}\tilde{a}(n-D_1-i-1) +\sum_{i=0}^{D_1-1}\sum_{i_1=0}^{N_B-D_1-1} d_i d_{i_1+D_1}\tilde{a}(n-D_1-i-i_1-2) +\cdots+\sum_{i=0}^{D_1-1}\sum_{i_1=0}^{D_1-1}\cdots\sum_{i_{D_1}=0}^{D_1-1} d_i d_{i_1}\cdots d_{i_{D_1}} \cdot \tilde{a}(n-i-i_1-\cdots-i_{D_1}-D_1-1). \tag{20}$$

We would now like to approximate (19) and (20) by simpler expressions. In order to guide us in making these approximations, we employ the following observations:

1) The appearance of PP is in accordance with the well known fact that introduction of delays in the DFL necessitates the use of a higher order FFF to compensate for the ISI terms not canceled by the FBF. This is interesting in the light of the fact that only an algorithm transformation technique, i.e., look-ahead, was employed to arrive at this conclusion. This also provides some justification for linearizing the DFL in the first place.
2) The PP derives all its coefficients (except the one, which operates on the most recent input) from those of the FBF of the serial ADFE. In addition, the coefficients of FBF, which appear in PP, are d_i, $0 \le i < D_1$. Hence, these D_1 coefficients of the FBF are required to cancel the D_1 most significant ISI terms as in the serial case.
3) If the impulse response of the combined channel and FFF is such that the magnitude of all the ISI terms (and therefore the FBF coefficients) is less than unity, then the contribution from the third and higher terms in (19) becomes negligible. If the combined channel and FFF impulse response has terms greater than unity, then the contribution from the third and higher terms in (19) can be accounted for (in the adaptive case) by increasing the number of terms in the second summation.
4) The computation of B contains $\tilde{a}(n-D_1-i)$, $i>1$. This implies that D_1 delays have been introduced at the input to the FBF.
5) All the FBF coefficients of the serial ADFE appear in B.

In the light of points 2) and 3) above, we approximate A as follows

$$A=\mathbf{X}(n)+\sum_{i=0}^{D_1-1} d_i\mathbf{X}(n-i-1). \tag{21}$$

Note that (21) is obtained from (19) by neglecting all terms except the first two. From points 4) and 5), we approximate B as

$$B=\sum_{i=0}^{N_B-1} d_i\tilde{a}(n-D_1-i-1). \tag{22}$$

Even with this seemingly crude approximation, it will be shown via simulations that PIPADFE2 converges twice as fast as PIPADFE1. Clearly, the performance of PIPADFE2 can be enhanced further by having sophisticated approximations (at the expense of increased hardware) in place of (21) and (22).

The pipelined DFL computation for PIPADFE2 can be written as

$$\tilde{a}(n)=\mathbf{C}^T\left[\mathbf{X}(n)+\sum_{i=0}^{D_1-1} d_i\mathbf{X}(n-i-1)\right] +\sum_{i=0}^{N_B-1} d_i\hat{a}(n-D_1-i-1) \tag{23}$$

where we have replaced the quantizer in the DFL.

Next, we delay input to the PP (and the training input) by D_1 samples and then apply the delay transfer relaxation first to PP and then to FFF. This results in the presence of D_1 delays at the output of FFF. In a similar fashion, we apply the delay relaxation of Fig. 2(c) to FBF to transfer D_1 to its

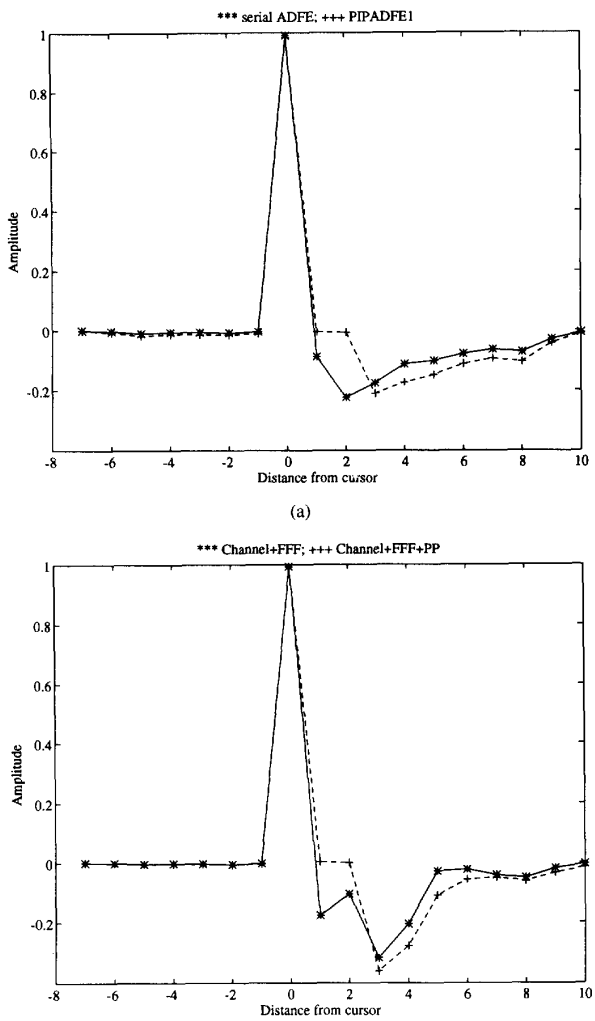


Fig. 8. Combined pulse response of (a) the channel+FFF for the serial ADFE (solid) and PIPADFE1 (dash-dot) with $D_1 = 2$, and (b) channel+FFF (solid) and channel+FFF+PP (dash-dot) for PIPADFE2 with $D_1 = 2$.

output. Finally, employing the sum relaxation as in the case of PIPADFE1, we obtain the following equations which describe PIPADFE2

$$\tilde{a}(n) = a_F(n - D_1) + a_B(n - D_1) \tag{24a}$$

$$a_F(n) = \mathbf{C}^T(n - D_2)\left[\mathbf{X}^T(n) + \sum_{i=0}^{D_1-1} d_i(n - D_2)\mathbf{X}^T(n - i - 1)\right] \tag{24b}$$

$$a_B(n) = \sum_{i=0}^{N_B-1} d_i(n - D_2)\hat{a}(n - i - 1) \tag{24c}$$

$$e(n) = \tilde{a}(n) - \hat{a}(n) \tag{24d}$$

$$\hat{a}(n) = \mathbf{Q}[\tilde{a}(n)] \tag{24e}$$

$$\mathbf{W}(n) = \mathbf{W}(n - D_2) + \mu \sum_{i=0}^{LA-1} e(n - i)\mathbf{U}_2(n - i) \tag{24f}$$

where

$$\mathbf{U}_2(n) = \left[\mathbf{X}^T(n - D_1) + \sum_{i=0}^{D_1-1} d_i(n - D_2) \cdot \mathbf{X}^T(n - D_1 - i - 1), \hat{a}^T(n - 1)\right]^T \tag{25}$$

and $\hat{a}(n) = a(n - D_1 - \Delta)$ for correct quantizer decisions.

The architecture for PIPADFE2 is shown in Fig. 6. The hardware overhead for PIPADFE2 are D_1 multipliers due to the PP, and $(LA-1)(N_F + N_B)$ adders (if the weight-update loop is also pipelined). This, however, does not reduce the clock frequency because the D_1 latches can be employed to pipeline PP as well. Note that it is possible to apply the delay relaxation to PP and FFF with a higher value of D_1 say D_1', while keeping the number of pipelining latches in the DFL at a constant value of D_1.

C. The PIPADFE3 Algorithm

In this section, we pipeline the serial predictor ADFE in Fig. 1(b), which is described by the following expressions

$$\mathbf{C}(n) = \mathbf{C}(n - 1) + \mu e(n)\mathbf{X}(n) \tag{26a}$$

$$\mathbf{D}(n) = \mathbf{D}(n - 1) + \mu[\hat{a}(n) - \tilde{a}(n)]\mathbf{E}(n - 1) \tag{26b}$$

$$e(n) = \hat{a}(n) - a_F(n) \tag{26c}$$

$$\tilde{a}(n) = a_F(n) + a_B(n) \tag{26d}$$

$$\hat{a}(n) = \mathbf{Q}[\tilde{a}(n)] \tag{26e}$$

$$a_F(n) = \mathbf{C}^T(n - 1)\mathbf{X}(n) \tag{26f}$$

$$a_B(n) = \mathbf{D}^T(n - 1)\mathbf{E}(n - 1) \tag{26g}$$

where $\mathbf{E}^T(n) = [e(n), e(n-1), \cdots, e(n - N_B + 1)]$ and $\hat{a}(n) = a(n - \Delta)$ for correct quantizer decisions. From (26a), (26c), and (26f), we see that the FFF adapts independently of the FBF. This feature of the predictor ADFE allows us to pipeline it to much higher levels than the conventional ADFE.

First, we transform (26b) into

$$\mathbf{D}(n) = \mathbf{D}(n - 1) + \mu[e(n) - a_B(n)]\mathbf{E}(n - 1) \tag{27}$$

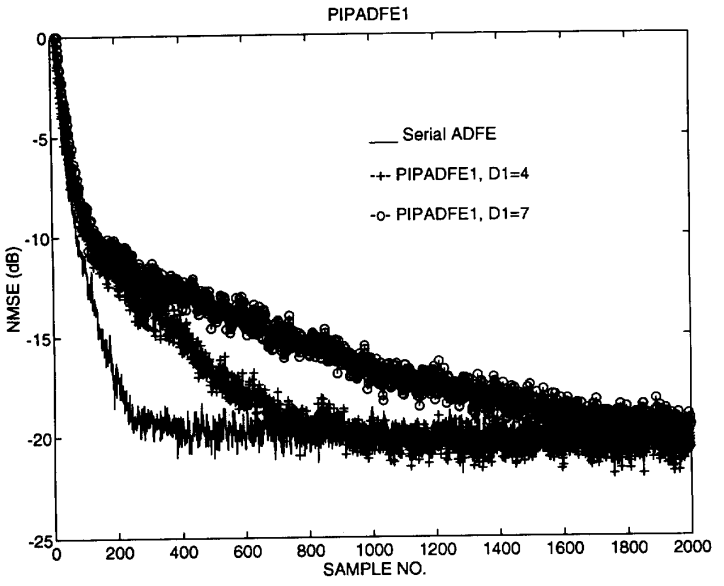
which can be derived by substituting for $\hat{a}(n)$ (from (26c) and $\tilde{a}(n)$ (from (26d) in (26b)).

We shall now pipeline the predictor ADFE by applying the delay relaxation and the sum relaxation. First, we apply the delay relaxation (see Fig. 2(b)) to the FFF weight-update (26a) to get

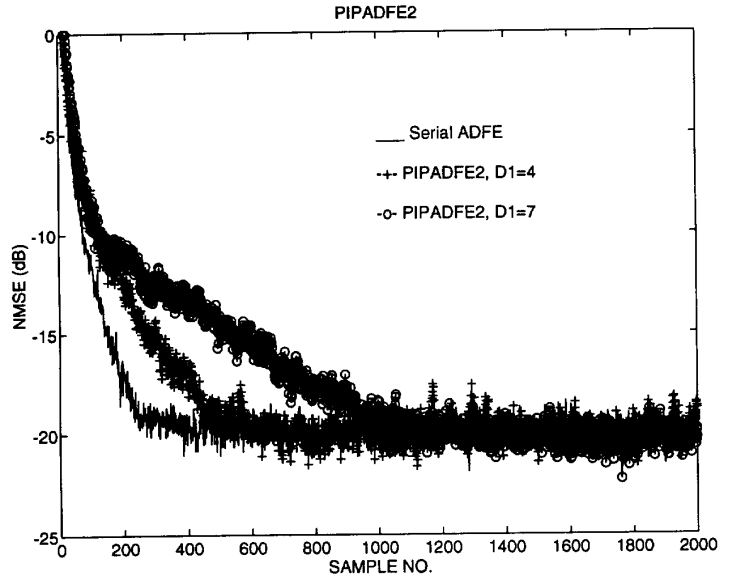
$$\mathbf{C}(n) = \mathbf{C}(n - 1) + \mu e(n - D_1)\mathbf{X}(n - D_1) \tag{28a}$$

$$\mathbf{D}(n) = \mathbf{D}(n - 1) + \mu[e(n - D_1) - a_B'(n)] \cdot \mathbf{E}(n - D_1 - 1) \tag{28b}$$

where $a_B'(n) = \mathbf{D}^T(n - 1)\mathbf{E}(n - D_1 - 1)$. Note that this step has already introduced D_1 latches in the DFL because $e(n)$ is present in it. In addition, (26b) has also been modified to



(a)



(b)

Fig. 9. Normalized MSE convergence plots: (a) PIPADFE1; (b) PIPADFE2; (c) PIPADFE3.

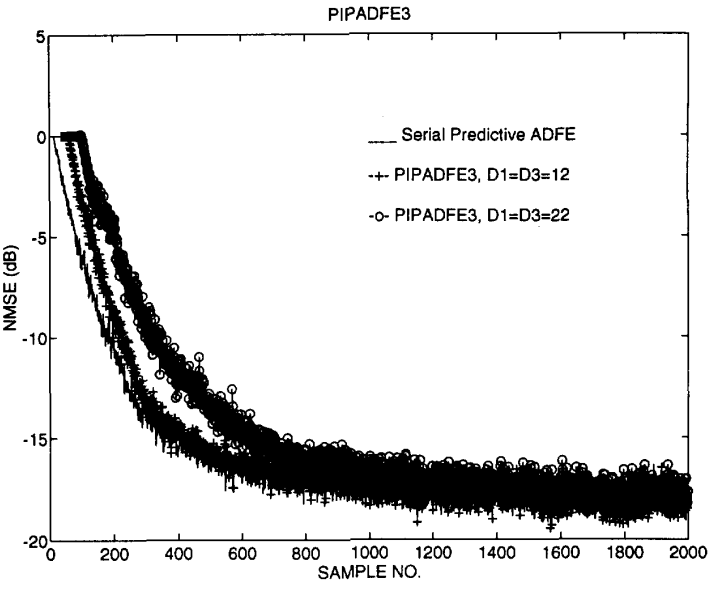
(28b) as $e(n)$ is an input to FBF. Next, we apply the sum relaxation to (28) to obtain

$$\mathbf{C}(n) = \mathbf{C}(n - D_2) + \mu \sum_{i=0}^{LA-1} \cdot e(n - D_1 - i)\mathbf{X}(n - D_1 - i) \quad (29a)$$

$$\mathbf{D}(n) = \mathbf{D}(n - D_2) + \mu \sum_{i=0}^{LA-1} \cdot [e(n - D_1 - i) - a'_B(n - i)] \cdot \mathbf{E}(n - D_1 - 1 - i). \quad (29b)$$

Finally, we delay the input to the FFF by D_3 delays, where $D_3 \leq D_1$ and apply the delay relaxation of the form shown in Fig. 2(c) to FFF and FBF. This results in D_3 delays at the output of the FFF and FBF. The pipelined predictor ADFE (referred to as PIPADFE3) is shown in Fig. 7 and the following equations describe it

$$\mathbf{C}(n) = \mathbf{C}(n - D_2) + \mu \sum_{i=0}^{LA-1} e(n - D_1 - i) \cdot \mathbf{X}(n - D_1 - D_3 - i) \quad (30a)$$



(c)

Fig. 9. *Continued.*

$$\mathbf{D}(n) = \mathbf{D}(n - D_2) + \mu \sum_{i=0}^{LA-1} [e(n - D_1 - i) - a''_B(n - D_3 - i)]\mathbf{E}(n - D_1 - 1 - i) \quad (30b)$$

$$e(n) = \hat{a}(n) - a_F(n - D_3) \quad (30c)$$

$$\tilde{a}(n) = a_F(n - D_3) + a_B(n - D_3) \quad (30d)$$

$$\hat{a}(n) = \mathbf{Q}[\tilde{a}(n)] \quad (30e)$$

$$a_F(n) = \mathbf{C}^T(n - D_2)\mathbf{X}(n) \quad (30f)$$

$$a''_B(n) = \mathbf{D}^T(n - D_2)\mathbf{E}(n - D_1 + D_3 - 1) \quad (30g)$$

with $\hat{a}(n) = a(n - D_3 - \Delta)$ for correct quantizer decisions.

In Fig. 1(b), we can easily confirm that the critical path consists of the FBF, adder, quantizer, adder and WUC. From Fig. 7, we find that this critical path contains $D_1 + D_3$ latches. Assuming uniformly pipelined stages, the speed-up achieved by PIPADFE3 over the serial architecture in Fig. 1(b) is $D_1 + D_3 + 1$. The hardware overhead for PIPADFE3 are the pipelining latches and $(LA - 1)[N_F + N_B]$ adders.

IV. Performance Analysis

In this section, we will analyze and compare, in qualitative terms, the performance of the serial ADFE, PIPADFE1, PIPADFE2, and PIPADFE3. To do this we have simulated the performance of the equalizers for a magnetic recording channel with a channel SNR of 20 dB. The channel coefficients ([0.2, 0.6, 1.0, −1.0, −0.6, −0.2]) were obtained from a Lorentzian pulse model [31] with the symbol period being one-half of the width of channel step response pulse at a height of 50% of the maximum.

The FFF in all the equalizers attempts to cancel the precursor ISI. To see this, we plot [see Fig. 8(a)] the pulse response of the combined channel and FFF for a serial ADFE. The nonzero postcursor ISI is canceled by the FBF coefficients. In the case of PIPADFE1 (and PIPADFE2), the FBF output is delayed by D_1 samples. Therefore, the FBF cannot cancel the first D_1 postcursor ISI terms and the burden of canceling them falls on the FFF. This is also indicated in Fig. 8(a), where the combined channel and FFF pulse response for PIPADFE1 with $D_1 = 2$ is shown. Clearly, as D_1 increases the performance of PIPADFE1 degrades and approaches that of a linear equalizer.

From the discussion above, it is clear that the performance of a pipelined ADFE algorithm can be improved substantially (especially for large D_1) if the FFF does not have to cancel the postcursor ISI. This is exactly what PIPADFE2 achieves. In Fig. 8(b), we show the combined channel and FFF for PIPADFE2 with $D_1 = 2$. Just as in the case of the serial ADFE, the FFF in PIPADFE2 only cancels the precursor ISI. However, in this case the first D_1 postcursor ISI's are canceled by the PP, whose coefficients are derived from those of FBF. This can be confirmed by plotting the pulse response of the system consisting of the channel, PP and FFF. Finally, the remaining postcursor ISI's are canceled by the FBF.

Even though it can be shown [32] that the predictor ADFE (see Fig. 1(b)) is equivalent to the conventional ADFE (see Fig. 1(a)) when the number of taps are infinite, in actual practice the predictor ADFE may perform worse than the conventional form. This is because in the predictor form the FFF and the FBF minimize two different error signals. Our interest, in this paper, is in comparing the performance of the serial predictor ADFE and PIPADFE3. However, it has been shown in [33] that for applications such as 1.544 Mbps asymmetric digital subscriber loop (ADSL), the predictor form does, in fact, perform better than the conventional ADFE. In the predictor form, the FFF adapts independently of the FBF and therefore, we were able to pipeline it by employing the delay relaxations and the sum relaxation. Thus, we should

expect any performance degradation due to pipelining to be similar to that of the pipelined LMS [16]. In [16], it was found that these relaxations result in a minimal loss in performance even for very high speed-ups. As we shall see later, via convergence analysis and simulations, that the same holds true for PIPADFE3.

In Section VI, we will further confirm the conclusions of this sub-section by comparing the performance of PIPADFE1, PIPADFE2 and PIPADFE3 as D_1 increases.

V. Convergence Analysis

In this section, we will analyze the convergence behavior of PIPADFE1, PIPADFE2, and PIPADFE3. In particular, the analytical expressions for the bounds on μ for convergence in the mean-squared sense and for the adaptation accuracy (in terms of the misadjustment $\mathcal{M}$) are provided. The misadjustment $\mathcal{M}$ [34] is defined as

$$\mathcal{M} = \frac{\varepsilon(\infty) - \varepsilon_{\min}}{\varepsilon_{\min}} \tag{31}$$

where $\varepsilon(n)$ is the average value of the mean-squared error at time instance n and $\varepsilon_{\min}$ is the minimum mean-squared error.

The analysis in this section is based on the results of the convergence analysis of the pipelined LMS algorithm in [16]. For mathematical tractability, we assume that the quantizer decisions are correct, and that the assumptions in the independence theory [35] are applicable. In addition, we only consider special cases of PIPADFE1 ($LA = 1$, $D_1 = KD_2$), PIPADFE2 ($LA = 1$, $D_1 = KD_2$) and PIPADFE3 ($LA = 1$, $D_1 + D_3 = KD_2$). The details of the derivation are provided in the Appendix, and only the results are presented in this section.

A. PIPADFE1

The bounds on μ for convergence in the mean-squared sense is given by

$$0 \le \mu \le \frac{\alpha_1 P + 2K - \sqrt{(\alpha_1 P + 2K)^2 - 8K(K+1)}}{2K(K+1)\sigma^2} \tag{32}$$

where

$$\alpha_1 = \frac{\mathrm{tr}\,[\mathbf{RR}]}{N_F(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2) + N_B} \tag{33a}$$

$$b = \frac{\mu}{N_F + N_B}\left[N_F(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2) + N_B\right] \tag{33b}$$

$$P = N_F + N_B + 2 \tag{33c}$$

$$\sigma^2 = \frac{b}{\mu} \tag{33d}$$

$$K = \frac{D_1}{D_2} \tag{33e}$$

where $\mathrm{tr}\,[.]$ represents the matrix trace operator and $\mathbf{R}$ is the autocorrelation matrix of the data vector $\mathbf{U}_3^T(n) = [\mathbf{X}^T(n), \mathbf{a}^T(n - D_1 - \Delta - 1)]$. The denominator and the numerator in (32) are a quadratic and linear functions, respectively, of K. This clearly indicates that as K or D_1 (with fixed D_2) increases, the upper bound on μ for convergence in the mean-squared sense is lowered. This is not a drawback of PIPADFE1 because practical values of μ are much smaller than this upper bound.

The misadjustment for PIPADFE1 is given by

$$\mathcal{M} = \frac{\alpha_1(N_F + N_B)b}{2 - (\alpha_1 P + 2K)b + K(K+1)b^2}. \tag{34}$$

Note that for very small values of μ, the third term in the denominator is negligible as compared to the first two terms. Hence, as K increases, the misadjustment of PIPADFE1 would increase. However, in practice, this increase in the misadjustment is negligible as seen in [16].

B. PIPADFE2

The bounds on μ for PIPADFE2 and the misadjustment are also given by (32) and (34), respectively, with

$$\alpha_1 = \frac{\mathrm{tr}\,[\mathbf{RR}]}{N_F(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2) + N_B + \delta(n)} \tag{35a}$$

$$b = \frac{\mu}{N_F + N_B}\left[N_F(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2) + N_B + \delta(n)\right] \tag{35b}$$

$$P = N_F + N_B + 2 \tag{35c}$$

$$\sigma^2 = \frac{b}{\mu} \tag{35d}$$

$$K = \frac{D_1}{D_2} \tag{35e}$$

where $\delta(n)$ is a time-varying function due to the presence of PP and $\mathbf{R}$ is the autocorrelation matrix of the data vector $\mathbf{U}_4(n)$ defined in (A20).

Further analysis of PIPADFE2 is made difficult due to the fact that it is not easy to characterize $\delta(n)$, which depends on the FBF coefficients and $\mathbf{X}(n)$.

C. PIPADFE3

Analysis of PIPADFE3 is made convenient by the fact that the FFF adapts independent of the FBF (see (30)). Hence, convergence analysis of the FFF can be done independently. For the FFF, the bounds on μ for convergence in the mean-squared sense is given by (32) with

$$\alpha_1 = \alpha_{fff} = \frac{S + N_F(2\sigma_\gamma^2\mathbf{h}^T\mathbf{h} + \sigma_\gamma^4)}{N_F(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2)} \tag{36a}$$

$$S = N_F r_0 + 2\sum_{i=1}^{N_F - 1}(N_F - i)r_i \tag{36b}$$

$$b = b_{fff} = \mu(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2) \tag{36c}$$

$$P = N_F + 2 \tag{36d}$$

$$\sigma^2 = \frac{b}{\mu} \tag{36e}$$

$$K = \frac{D_1 + D_3}{D_2} \tag{36f}$$

and r_i is given by

$$r_i = \begin{cases} \sum_{j=0}^{N-i-1} h_j h_{i+j}, & \text{if } i \le N-1 \\ 0, & \text{otherwise} \end{cases} \quad (37)$$

where N is the number of channel coefficients.

The misadjustment for FFF is given by

$$\mathcal{M}_{fff} = \frac{\alpha_{fff} N_F b_{fff}}{2 - (\alpha_{fff} P + 2K) b_{fff} + K(K+1) b_{fff}^2}. \quad (38)$$

The bounds on μ for the FBF are also given by (32) with

$$\begin{aligned} \alpha_1 &= \alpha_{fbf} \\ &= \{\mathrm{tr}[\mathbf{R}'\mathbf{R}']\}/\{N_B(1+\mathcal{M}_{fff}) \\ &\quad \cdot [1 - \mathbf{h}_{\Delta+1}^T (\mathbf{H}\mathbf{H}^T + \sigma_\gamma^2 \mathbf{I}_{N_F \times N_F})^{-1} \mathbf{h}_{\Delta+1}]\} \quad (39a) \\ b &= b_{fbf} \\ &= \mu(1+\mathcal{M}_{fff}) \\ &\quad \cdot [1 - \mathbf{h}_{\Delta+1}^T (\mathbf{H}\mathbf{H}^T + \sigma_\gamma^2) \mathbf{I}_{N_F \times N_F}]^{-1} \mathbf{h}_{\Delta+1} \quad (39b) \\ P &= N_B + 2 \quad (39c) \\ \sigma^2 &= \frac{b}{\mu} \quad (39d) \\ K &= \frac{D_3}{D_2} \quad (39e) \end{aligned}$$

where $\mathbf{R}'$ is the autocorrelation matrix of the vector $\mathbf{E}'^T(n) = [e(n - D_1 + D_2 - 1), e(n - D_1 + D_2 - 2, \cdots, e(n - D_1 + D_2 - N_B)]$ and the misadjustment is given by

$$\mathcal{M}_{fbf} = \frac{\alpha_{fbf} N_B b_{fbf}}{2 - (\alpha_{fbf} P + K) b_{fbf} + K(K+1) b_{fbf}^2}. \quad (40)$$

Note the dependence of (39a) and (39b) on the misadjustment of the FFF. This is to be expected because the input to the FBF is the error signal generated by the FFF.

VI. Simulation Results

In this section, we present extensive simulation results in order to compare the performance of the pipelined algorithms with each other and with the serial ADFE. All simulations for a magnetic recording channel have been performed with a Lorentzian model, whose coefficients are defined in Section IV. For PIPADFE1 and PIPADFE2, values of $N_F = 13$ and $N_B = 10$ were chosen, while the corresponding values for PIPADFE3 were $N_F = 20$ and $N_B = 10$. This choice of N_F and N_B were made in order to get a positive noise margin at the slicer for a nominal channel SNR, where the channel SNR is defined as the ratio of channel output power to the noise power. The nominal channel SNR for storage channels was found to be about 22 dB [36] and an output SNR (i.e., SNR at the slicer) of 16 dB was required for a byte error rate of 10^{-7} or less. Hence, we take this value of output SNR as the lower limit of acceptable performance.

The first four simulations study the effect of pipelining on the convergence speed (Experiment A), output SNR (Experiment B), performance in presence of channel nonstationarity (Experiment C) and the effectiveness of sum relaxation (Experiment D). In the fifth simulation, we consider an infinite impulse response (IIR) channel [22] and study the performance of PIPADFE1, PIPADFE2 and PIPADFE3.

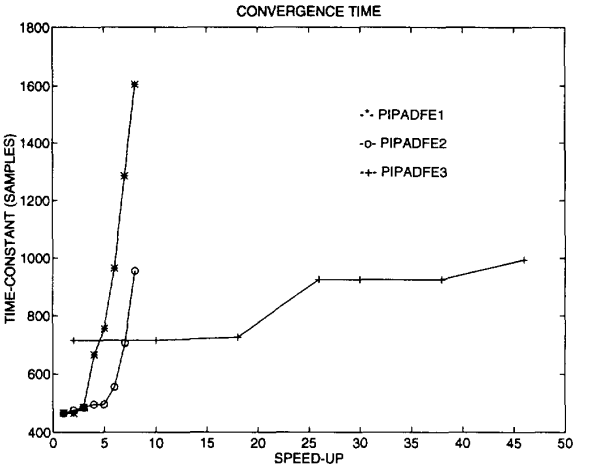


Fig. 10. Convergence time-constant versus speed-up for magnetic recording channel.

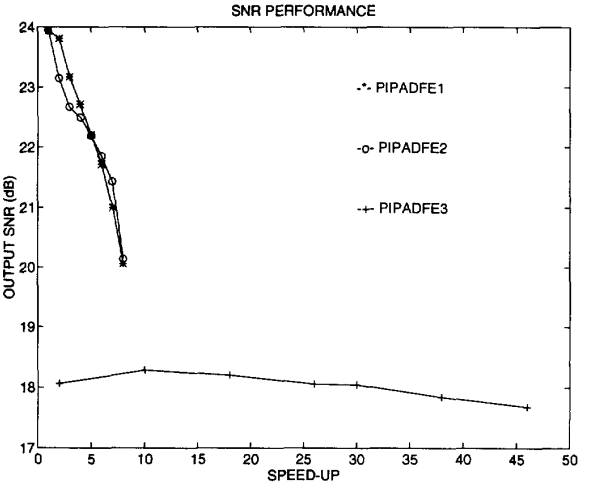
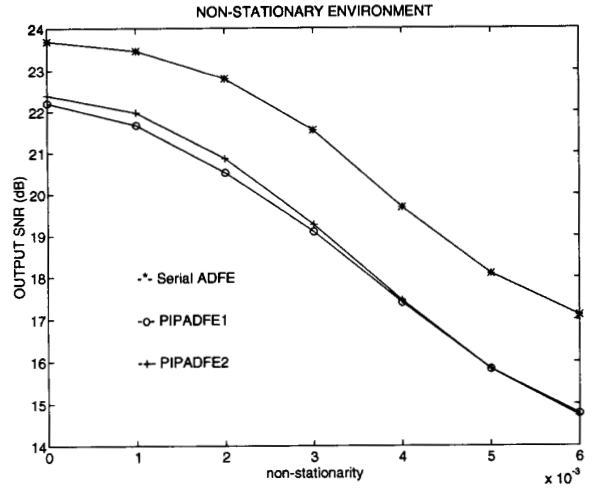


Fig. 11. Output SNR versus speed-up for magnetic recording channel.

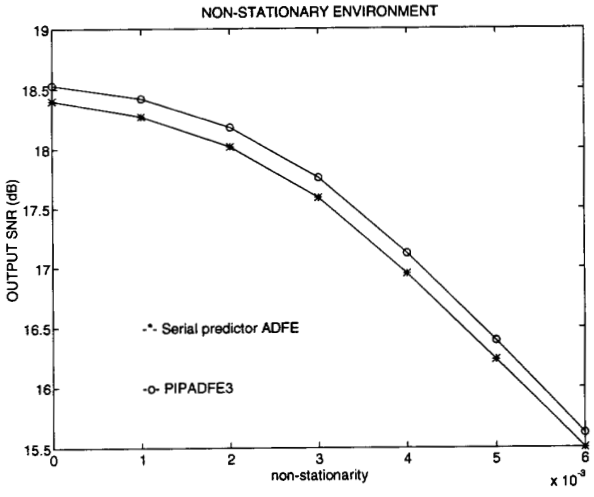
A. Convergence Speed

In order to study the effect of pipelining on the convergence speed, the step-sizes were chosen such that an output SNR of 20 dB is achieved. In Fig. 9(a), we plot the mean-squared error (MSE) curve for PIPADFE1. Note that as D_1 increases, the algorithm takes longer to converge. For $D_1 = 7$, PIPADFE1 takes about 3.5 times longer than the serial ADFE. In case of PIPADFE2 (see Fig. 9(b)), the degradation in convergence speed is reduced. In fact, PIPADFE2 achieves the same output SNR as PIPADFE1 in about half the number of samples.

With PIPADFE3 (see Fig. 9(c)), the degradation in convergence speed is negligible even with $D_1 = D_3 = 22$. Note that this value of D_1 and D_3 corresponds to a speed-up of 45. As mentioned before (in Section IV), the performance degradation of PIPADFE3 is expected to be similar to that of the pipelined LMS [16] algorithm. This is now verified.



(a)



(b)

Fig. 12. Output SNR versus channel nonstationarity: (a) Conventional ADFE; (b) predictor ADFE.

Finally, in Fig. 10, we plot the convergence time-constant (i.e., number of samples required to converge) for different speed-ups. It is clear that PIPADFE2 is twice as fast as PIPADFE1 for speed-ups greater than 3. In addition, PIPADFE3 can achieve very high speed-ups with negligible degradation in convergence speed.

B. Output SNR

The purpose of this experiment is to determine how the performance of PIPADFE1, PIPADFE2 and PIPADFE3 degrade as speed-up is increased. The step-sizes for the simulations in this sub-section were chosen such the convergence time-constant at different speed-ups is the same. In case of PIPADFE1, this time-constant was 1700 samples, while that for PIPADFE2 was 900. From Fig. 11, it can be seen that both PIPADFE1 and PIPADFE2 have an SNR loss of about 0.6 dB per unit increase in the speed-up. The maximum achievable speed-up depends upon the value of convergence time, which is acceptable. For the time-constants mentioned above, both PIPADFE1 and PIPADFE2 can achieve speed-ups of up to 8 with a 4 dB noise margin. From Fig. 11, we can also see that degradation in output SNR for PIPADFE3 is less than 0.1 dB for speed-ups of at least 45. The convergence time-constant for PIPADFE3 was kept fixed at 800. As PIPADFE3 can be pipelined at very high-speeds, therefore, we can afford to have higher number of taps for the FBF in order to improve its SNR performance and yet achieve significant speed-ups.

C. Nonstationarity

To study the performance of the algorithms in the presence of channel nonstationarity, we model a time-varying channel [20] as follows

$$\mathbf{h}(n) = \mathbf{h}(0) + \mathbf{G}(n) \tag{41a}$$

$$\mathbf{G}(n) = \mathbf{G}(n-1) + \mathbf{I}(n) \tag{41b}$$

$$\mathbf{G}(0) = \mathbf{0} \tag{41c}$$

$$\mathbf{h}(0) = [0.2 \quad 0.6 \quad 1.0 \quad -1.0 \quad -0.6 \quad -0.2]^T \tag{41d}$$

where $\mathbf{h}(n)$ is the vector of channel coefficients at time index n and $\mathbf{I}(n)$ is a vector of white Gaussian variables with standard deviation η.

In Fig. 12(a), we show performance of PIPADFE1 and PIPADFE2 with $D_1 = 4$ as function of η. The step-sizes were chosen to be the same as that in Experiment B. It is clear that as the channel nonstationarity increases, the output SNR of PIPADFE2 and PIPADFE1 degrade in the same fashion as that of the serial ADFE. This is also true for PIPADFE3 (see Fig. 12(b)). However, for the drop in SNR for PIPADFE3 is about half that of PIPADFE1 and PIPADFE2 for the same degree of channel nonstationarity.

D. Sum Relaxation

The sum relaxation, defined in (3), was employed to pipeline the weight-update equations for PIPADFE1 (14f), PIPADFE2 (24f), and PIPADFE3 (30). In this experiment, we show the effect of sum relaxation in improving the convergence speed for a given value of D_2 (the pipelining level of the weight-update loop) and given output SNR. All simulations were done with the conventional ADFE with $D_1 = 0$.

In Fig. 13, the MSE convergence plot for $D_2 = 1$ and $D_2 = 4$ is shown. With $D_2 = 1$ and $LA = 0$, which is essentially the serial ADFE, a convergence time-constant of about 600 samples is obtained. When D_2 is increased to 4 and $LA = 0$, the convergence time-constant is also increased by a factor of 4 to about 2400. In general, we find that (with the same step-size) the convergence time-constant is D_2 times that of the serial ADFE. It is clear that any attempt to improve the convergence speed by increasing the step-size would result in a worse output SNR. However, by employing sum relaxation with $LA = 3$ the convergence time-constant is made equal to that of the serial ADFE without any loss in the output SNR. This clearly implies that very fine pipelining of the weight-update loop is possible.

Recall that $D_1 = 0$ implies the equivalence of the serial ADFE, PIPADFE1 and PIPADFE2. Hence, the conclusions of

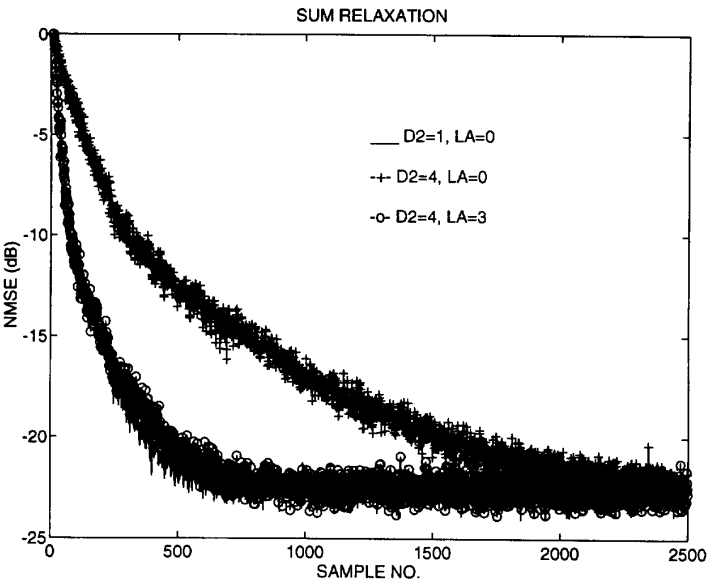


Fig. 13. Normalized MSE convergence plots with sum relaxation.

this experiment are valid for all the algorithms under consideration. Similar results were also obtained for PIPADFE3.

E. IIR Channel

In this experiment we consider the channel to be of the form [22],

$$H(z) = \frac{1 + 0.5z^{-1}}{1 - 0.5z^{-1}} \tag{42}$$

where $H(z)$ is the z-transform of $h(n)$. The values of $N_F = 10$ and $N_B = 7$, were chosen for PIPADFE1 and PIPADFE2. For PIPADFE3, the corresponding values were $N_F = 13$ and $N_B = 10$. The additive noise power was scaled such that the channel SNR was identical to that in Experiments A and B.

In Fig. 14, we have plotted the convergence time-constants for PIPADFE1, PIPADFE2, and PIPADFE3 with respect to the speed-up for a constant output SNR of 20 dB. As in the case of the magnetic recording channel (see Fig. 10), we find that PIPADFE2 converges faster than PIPADFE1 as the speed-up increases. In addition, the convergence time-constant for PIPADFE3 increases linearly by approximately 54 samples per unit increase in speed-up. However, unlike in the case of a magnetic recording channel, PIPADFE3 always converges faster than either PIPADFE1 or PIPADFE2, except in the case where the speed-up is unity.

Next, we plot (see Fig. 15) the output SNR with respect to speed-up for a given convergence time-constant (3000 samples in this case). As can be seen in Fig. 15, PIPADFE3 has less than 1 dB loss in performance for speed-ups of up to 45. The performance of PIPADFE1 and PIPADFE2 are more or less identical as was in the case of a magnetic recording channel. The loss in performance is less than 3 dB for speed-ups of up to 8. From Figs. 14 and 15, we conclude that not only does PIPADFE3 converge faster than PIPADFE1 and PIPADFE2, but that it also has a better output SNR for the same speed-up.

VII. Conclusions

The technique of relaxed look-ahead [15] was employed to develop two fine-grain pipelined ADFE algorithms for high-speed equalization applications. These algorithms are attractive from an implementation point of view due to their low hardware requirements as compared to the existing parallel processing schemes. Speed-ups of up to 8 (for the conventional ADFE) and 45 (for the predictor ADFE) have been demonstrated for the equalization of a magnetic storage channel.

This work is a part of our research on the development of inherently pipelinable DSP algorithms, which are hardware efficient. Work in fixed-coefficient, inherently concurrent direct-form and lattice recursive digital filters [37] has also been successful. The pipelined algorithms presented in this paper can be improved further by incorporating improved relaxations, especially in case of PIPADFE2. In addition, combining coding with equalization, as was done in [38], could also be exploited for the development of better pipelined equalization algorithms.

Appendix
Convergence Analysis

We employ the following results from [16] for a pipelined LMS algorithm described by

$$\mathbf{W}(n) = \mathbf{W}(n - D_2) + \mu e(n - KD_2)\mathbf{U}(n - KD_2) \tag{A1a}$$

$$e(n) = s(n) - \mathbf{W}^T(n - D_2)\mathbf{U}(n) \tag{A1b}$$

where $\mathbf{W}^T(n) = [w_1(n), w_2(n), \cdots, w_N(n)]$ is the vector of weights, $\mathbf{U}^T(n) = [u(n), u(n-1), \cdots, u(n-N+1)]$ is the input vector, μ is the adaptation constant, $s(n)$ is the desired

signal, $e(n)$ is the error and N is the filter order. Now define

$$\alpha_1 = \frac{\sum_{i=1}^{N} \lambda_i^2}{\sum_{i=1}^{N} \lambda_i} \tag{A2a}$$

$$v = \frac{E\{[u(n)]^4\}}{(E\{[u(n)]^2\})^2} \tag{A2b}$$

$$P = N + v - 1 \tag{A2c}$$

$$\sigma^2 = \frac{\sum_{i=1}^{N} \lambda_i}{N} \tag{A2d}$$

$$b = \mu\sigma^2 \tag{A2e}$$

where λ_i's ($i = 1, \cdots, N$) are the eigenvalues of the $\mathbf{R} = E[\mathbf{U}(n)\mathbf{U}^T(n)]$ with $E[.]$ representing the expectation operator. If $u(n)$ has a Gaussian distribution then $v = 3$ and therefore $P = N + 2$.

The bounds on μ for convergence in the mean-squared sense is given by

$$0 \le \mu \le \frac{\alpha_1 P + 2K - \sqrt{(\alpha_1 P + 2K)^2 - 8K(K+1)}}{2K(K+1)\sigma^2} \tag{A3}$$

and the misadjustment is given by

$$\mathcal{M} = \frac{\alpha_1 N b}{2 - (\alpha_1 P + 2K)b + K(K+1)b^2}. \tag{A4}$$

In the analysis to follow, we assume that the quantizer decisions are always correct and the channel input $a(n)$ is an uncorrelated ± 1 sequence. Therefore, the results of the analysis are applicable to the training mode and also to the decision-directed mode with correct quantizer decisions. In all cases, we will first recast the pipelined algorithm in the form of (A1) and then analyze the autocorrelation matrix of the resulting data vector. In particular, we will attempt to derive an analytical expression for the sum of the eigenvalues of the data correlation matrix. In order to do this we employ the well known fact that $\sum_{i=0}^{N-1} \lambda_i = \text{tr}[\mathbf{R}]$, where λ_i's are the eigenvalues of $\mathbf{R}$ and tr[.] is the matrix trace operator.

A. PIPADFE1

It is easy to show that with $D_1 = KD_2$, $LA = 1$ and $\hat{a}(n) = a(n - D_1 - \Delta)$, PIPADFE1 can be written as

$$\mathbf{W}(n) = \mathbf{W}(n - D_2) + \mu e'(n - D_1)\mathbf{U}_3(n - D_1) \tag{A5a}$$

$$e'(n) = a(n - \Delta) - \mathbf{W}^T(n - D_2)\mathbf{U}_3(n) \tag{A5b}$$

where

$$\mathbf{U}_3^T(n) = [\mathbf{X}^T(n), \mathbf{a}^T(n - D_1 - \Delta - 1)]. \tag{A6}$$

The autocorrelation matrix $\mathbf{R} = E[\mathbf{U}_3(n)\mathbf{U}_3^T(n)]$ is given by

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{bmatrix} \tag{A7}$$

where

$$\mathbf{R}_{11} = E[\mathbf{X}(n)\mathbf{X}^T(n)] \tag{A8a}$$

$$\mathbf{R}_{12} = E[\mathbf{X}(n)\mathbf{a}^T(n - D_1 - \Delta - 1)] = \mathbf{R}_{21}^T \tag{A8b}$$

$$\mathbf{R}_{22} = E[\mathbf{a}(n - D_1 - \Delta - 1)\mathbf{a}^T(n - D_1 - \Delta - 1)] = \mathbf{I}_{N_B \times N_B}. \tag{A8c}$$

Note that (A8c) assumes $a(n)$ to be an uncorrelated ± 1 sequence at the channel input. From (A7) and (A8c), we have

$$\text{tr}[\mathbf{R}] = \text{tr}[\mathbf{R}_{11}] + \text{tr}[\mathbf{R}_{22}] = \text{tr}[\mathbf{R}_{11}] + N_B. \tag{A9}$$

Assuming a stationary channel, the output $x(n)$ can be written as

$$x(n) = \mathbf{h}^T\mathbf{a}_N(n) + \gamma(n) \tag{A10}$$

where $\mathbf{h}^T = [h_0, h_1, h_2, \cdots, h_{N-1}]$ is the channel coefficient vector, $\mathbf{a}_N(n)^T = [a(n), a(n-1), \cdots, a(n-N+1)]$, and $\gamma(n)$ is a zero mean white Gaussian process with variance σ_γ^2. Employing (A10), we can express the vector $\mathbf{X}(n)$ as follows

$$\mathbf{X}(n) = \mathbf{H}\mathbf{a}_{N_1}(n) + \Gamma(n) \tag{A11}$$

where $\mathbf{a}_{N_1}^T(n) = [a(n), a(n-1), \cdots, a(n-N_1+1)]$ with $N_1 = N_F + N - 1$, $\Gamma^T(n) = [\gamma(n), \gamma(n-1), \cdots, \gamma(n - N_F + 1)]$ and $\mathbf{H}$ is a $N_F \times N_1$ matrix defined as follows

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}^T & 0 & 0 & \cdots & \mathbf{0}_{1\times N-1} \\ 0 & \mathbf{h}^T & 0 & \cdots & \mathbf{0}_{1\times N-1} \\ 0 & 0 & \mathbf{h}^T & \cdots & \mathbf{0}_{1\times N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \mathbf{h}^T \end{bmatrix}. \tag{A12}$$

Substituting (A11) into (A8a) and employing the fact that $\gamma(n)$ is uncorrelated with $a(n)$, we get

$$\mathbf{R}_{11} = \mathbf{H}\mathbf{H}^T + \sigma_\gamma^2\mathbf{I}_{N_F \times N_F} \tag{A13}$$

From (A12), it can be seen that

$$\text{tr}[\mathbf{H}\mathbf{H}^T] = N_F\mathbf{h}\mathbf{h}^T. \tag{A14}$$

Employing (A14), we first obtain tr[$\mathbf{R}_{11}$] from (A13) and then substitute the result in (A9) to get

$$\text{tr}[\mathbf{R}] = N_F(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2) + N_B. \tag{A15}$$

Thus, the bounds on μ for convergence in the mean-squared sense is given by

$$0 \le \mu \le \frac{\alpha_1 P + 2K - \sqrt{(\alpha_1 P + 2K)^2 - 8K(K+1)}}{2K(K+1)\sigma^2} \tag{A16}$$

where

$$\alpha_1 = \frac{\text{tr}[\mathbf{R}\mathbf{R}]}{N_F(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2) + N_B} \tag{A17a}$$

$$b = \frac{\mu}{N_F + N_B}[N_F(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2) + N_B] \tag{A17b}$$

$$P = N_F + N_B + 2 \tag{A17c}$$

$$\sigma^2 = \frac{b}{\mu} \tag{A17d}$$

$$K = \frac{D_1}{D_2} \tag{A17e}$$

and the misadjustment is given by

$$\mathcal{M} = \frac{\alpha_1(N_F + N_B)b}{2 - (\alpha_1 P + 2K)b + K(K+1)b^2}. \tag{A18}$$

B. PIPADFE2

The PIPADFE2 with $D_1 = KD_2$ and $\hat{a}(n) = a(n-\Delta-D_1)$ can also be recast into the same form as (A1) as shown below

$$\mathbf{W}(n) = \mathbf{W}(n - D_2) + \mu e'(n - D_1)\mathbf{U}_4(n - D_1) \tag{A19a}$$

$$e'(n) = a(n - \Delta) - \mathbf{W}^T(n - D_2)\mathbf{U}_4(n) \tag{A19b}$$

where

$$\mathbf{U}_4^T(n) = \left[\mathbf{X}^T(n) + \sum_{i=0}^{D_1-1} d_i(n - D_2)\mathbf{X}^T(n - i - 1), \right.$$
$$\left. \mathbf{a}^T(n - D_1 - \Delta - 1)\right]. \tag{A20}$$

Due to the presence of FBF coefficients in (A20), it is difficult evaluate $\mathrm{tr}[\mathbf{R}]$, where

$$\mathbf{R} = E[\mathbf{U}_4(n)\mathbf{U}_4^T(n)]. \tag{A21}$$

However, from (A20), it is possible to show that

$$\mathrm{tr}[\mathbf{R}] = \mathrm{tr}[\mathbf{R}_{11}] + N_B + \delta(n)$$
$$= N_F(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2) + N_B + \delta(n) \tag{A22}$$

where $\delta(n)$ is a time-varying term due to the presence of FBF coefficients in (A20). Note that we have employed (A13) to substitute for $\mathrm{tr}[\mathbf{R}_{11}]$ in (A22). Hence, for PIPADFE2, we get

$$0 \le \mu \le \frac{\alpha_1 P + 2K - \sqrt{(\alpha_1 P + 2K)^2 - 8K(K+1)}}{2K(K+1)\sigma^2} \tag{A23}$$

where

$$\alpha_1 = \frac{\mathrm{tr}[\mathbf{RR}]}{N_F(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2) + N_B + \delta(n)} \tag{A24a}$$

$$b = \frac{\mu}{N_F + N_B}[N_F(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2) + N_B + \delta(n)] \tag{A24b}$$

$$P = N_F + N_B + 2 \tag{A24c}$$

$$\sigma^2 = \frac{b}{\mu} \tag{A24d}$$

$$K = \frac{D_1}{D_2} \tag{A24e}$$

and the misadjustment is given by

$$\mathcal{M} = \frac{\alpha_1(N_F + N_B)b}{2 - (\alpha_1 P + 2K)b + K(K+1)b^2} \tag{A25}$$

which are the desired equations.

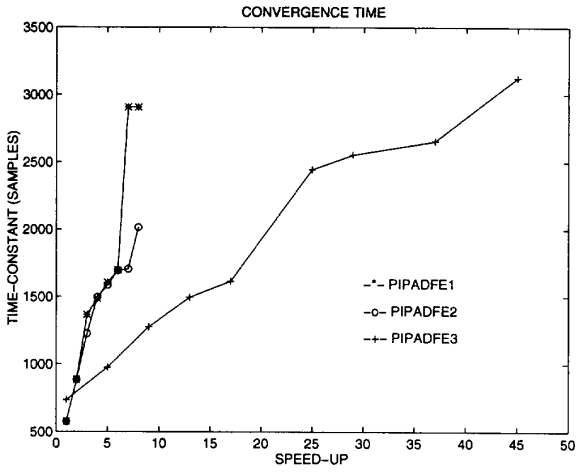


Fig. 14. Convergence time-constant versus speed-up for IIR channel.

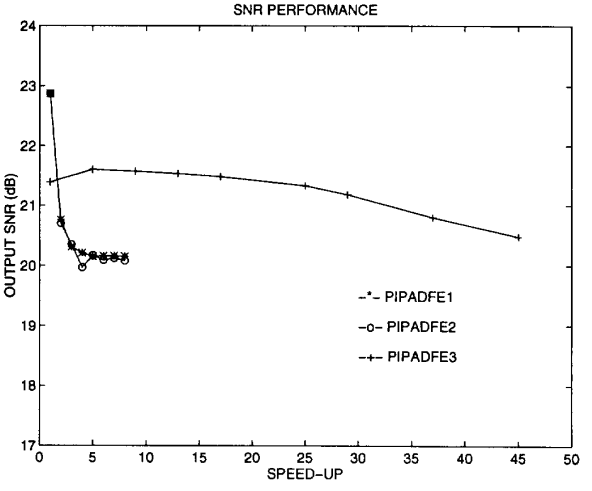


Fig. 15. Output SNR versus speed-up for IIR channel.

C. PIPADFE3

The PIPADFE3 equations can be written as follows

$$\mathbf{C}(n) = \mathbf{C}(n - D_2) + \mu e'(n - D_{13}) \cdot \mathbf{X}(n - D_{13}) \tag{A26a}$$

$$e'(n) = a(n - \Delta) - \mathbf{C}^T(n - D_2)\mathbf{X}(n) \tag{A26b}$$

$$\mathbf{D}(n) = \mathbf{D}(n - D_2) + \mu e''(n - D_3) \cdot \mathbf{E}'(n - D_3 - 1) \tag{A26c}$$

$$e''(n) = e'(n - D_1) - \mathbf{D}^T(n - D_2)\mathbf{E}'(n) \tag{A26d}$$

where $D_{13} = D_1 + D_3 = KD_2$ and $\mathbf{E}'^T(n) = [e'(n - D_1 - 1), e'(n - D_1 - 2), \cdots, e'(n - D_1 - N_B)]$.

From (A26a) and (A26b), it can be seen that the FFF adapts independent of the FBF and this adaptation is of same form as the pipelined LMS filter (A1). Hence, we first analyze the FFF and then the FBF.

As the input to the FFF is $\mathbf{X}(n)$, hence the trace of the autocorrelation matrix of the data vector for FFF is known

from (A13) and (A14). In addition, from (A13), we get

$$\begin{aligned}\mathrm{tr}\,[\mathbf{R}_{11}\mathbf{R}_{11}] &= \mathrm{tr}\,[\mathbf{H}\mathbf{H}^T\mathbf{H}\mathbf{H}^T] + 2\sigma_\gamma^2 \mathrm{tr}\,[\mathbf{H}\mathbf{H}^T] + \sigma_\gamma^4 N_B \\ &= S + 2\sigma_\gamma^2 N_F * \mathbf{h}^T\mathbf{h} + \sigma_\gamma^4 N_F \end{aligned} \tag{A27}$$

where $S = \mathrm{tr}\,[\mathbf{H}\mathbf{H}^T\mathbf{H}\mathbf{H}^T]$. Note that (A14) has been employed to substitute for the second term in (A27). Observing that $\mathbf{H}\mathbf{H}^T\mathbf{H}\mathbf{H}^T$ is a symmetric matrix, we compute the sum of the squares of all the entries in $\mathbf{H}\mathbf{H}^T$ to obtain S. Furthermore, $\mathbf{H}\mathbf{H}^T$ itself is a symmetric matrix and from the definition of $\mathbf{H}$ in (A12), we get the following expression for the elements $\mathbf{H}\mathbf{H}^T$ along the ith diagonal

$$r_i = \begin{cases} \displaystyle\sum_{j=0}^{N-i-1} h_j h_{i+j}, & \text{if } i \le N-1 \\ 0, & \text{otherwise} \end{cases}. \tag{A28}$$

The trace of $\mathbf{H}\mathbf{H}^T\mathbf{H}\mathbf{H}^T$ can be now written as

$$S = N_F r_0 + 2\sum_{i=1}^{N_F-1} (N_F - i) r_i. \tag{A29}$$

Thus, for the FFF, we get the bounds on μ for convergence in the mean-squared sense follows

$$\begin{aligned} 0 &\le \mu \\ &\le [\alpha_{fff} P + 2K \\ &\quad - \sqrt{(\alpha_{fff} P + 2K)^2 - 8K(K+1)}]/ \\ &\quad [2K(K+1)\sigma^2] \end{aligned} \tag{A30}$$

where

$$\alpha_{fff} = \frac{S + N_F(2\sigma_\gamma^2 \mathbf{h}^T\mathbf{h} + \sigma_\gamma^4)}{N_F(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2)} \tag{A31a}$$

$$b_{fff} = \mu(\mathbf{h}^T\mathbf{h} + \sigma_\gamma^2) \tag{A31b}$$

$$P = N_F + 2 \tag{A31c}$$

$$\sigma^2 = \frac{b}{\mu} \tag{A31d}$$

$$K = \frac{D_1 + D_3}{D_2} \tag{A31e}$$

and S is given by (A29). The misadjustment for FFF is given by

$$\mathcal{M}_{fff} = \frac{\alpha_{fff} N_F b_{fff}}{2 - (\alpha_{fff} P + 2K) b_{fff} + K(K+1) b_{fff}^2}. \tag{A32}$$

The FBF [see (A26c) and (A26d)] is also in the same form as (A1). We assume that a suitable value of μ has been chosen according to (A30) so that the FFF converges. Let $E[\mathbf{E}'(n)\mathbf{E}'^T(n)] = \mathbf{R}'$, then

$$\begin{aligned}\mathrm{tr}\,[\mathbf{R}'] &= N_B \sigma_e^2 \\ &\approx N_B(1 + \mathcal{M}_{fff}) e_{\min} \end{aligned} \tag{A33}$$

where σ_e^2 is the power of $e(n)$ and $e_{\min}$ is the minimum mean-squared error for the FFF, which is given by

$$\begin{aligned} e_{\min} &= 1 - \mathbf{P}_1^T \mathbf{R}_{11}^{-1} \mathbf{P}_1 \\ &= 1 - \mathbf{P}_1^T (\mathbf{H}\mathbf{H}^T + \sigma_\gamma^2 \mathbf{I}_{N_F \times N_F})^{-1} \mathbf{P}_1 \end{aligned} \tag{A34}$$

where $\mathbf{R}_{11}$ is the autocorrelation of the data in the FFF and $\mathbf{P}_1$ is the crosscorrelation between the desired signal $a(n-\Delta)$ and the data $\mathbf{X}(n)$. Employing (A11) to substitute for $\mathbf{X}(n)$, we may evaluate $\mathbf{P}_1$ as follows

$$\begin{aligned} \mathbf{P}_1 &= E[a(n-\Delta)\mathbf{X}(n)] \\ &= E[a(n-\Delta)\{\mathbf{a}_{N_1}^T(n)\mathbf{H}^T + \Gamma^T(n)]\} \\ &= \mathbf{h}_{\Delta+1} \end{aligned} \tag{A35}$$

where $\mathbf{h}_{\Delta+1}$ is the $(\Delta+1)$th row of $\mathbf{H}$. Hence, we get for the FBF

$$\begin{aligned} 0 &\le \mu \\ &\le [\alpha_{fbf} P + 2K \\ &\quad - \sqrt{(\alpha_{fbf} P + 2K)^2 - 8K(K+1)}]/ \\ &\quad [2K(K+1)\sigma^2] \end{aligned} \tag{A36}$$

where

$$\begin{aligned} \alpha_{fbf} &= \{\mathrm{tr}\,[\mathbf{R}'\mathbf{R}']\}/\{N_B(1 + \mathcal{M}_{fff}) \\ &\quad \cdot [1 - \mathbf{h}_{\Delta+1}^T(\mathbf{H}\mathbf{H}^T \\ &\quad + \sigma_\gamma^2 \mathbf{I}_{N_F \times N_F})^{-1} \mathbf{h}_{\Delta+1}]\} \end{aligned} \tag{A37a}$$

$$\begin{aligned} b_{fbf} &= \mu(1 + \mathcal{M}_{fff}) \\ &\quad \cdot [1 - \mathbf{h}_{\Delta+1}^T(\mathbf{H}\mathbf{H}^T \\ &\quad + \sigma_\gamma^2) \mathbf{I}_{N_F \times N_F}]^{-1} \mathbf{h}_{\Delta+1} \end{aligned} \tag{A37b}$$

$$P = N_B + 2 \tag{A37c}$$

$$\sigma^2 = \frac{b}{\mu} \tag{A37d}$$

$$K = \frac{D_3}{D_2} \tag{A37e}$$

and the misadjustment is given by

$$\mathcal{M}_{fbf} = \frac{\alpha_{fbf} N_B b_{fbf}}{2 - (\alpha_{fbf} P + K) b_{fbf} + K(K+1) b_{fbf}^2} \tag{A38}$$

which are the desired equations.

ACKNOWLEDGMENT

The authors would like to acknowledge the constructive criticisms of the anonymous reviewers, which led to significant improvements in the manuscript.

REFERENCES

[1] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters—Part I: Pipelining using scattered look-ahead and decomposition," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 37, pp. 1099–1117, July 1989.

[2] ——, "Pipeline interleaving and parallelism in recursive digital filters—Part II: Pipelined incremental block filtering," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 37, pp. 1118–1134, July 1989.

[3] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low power CMOS digital design," *IEEE J. Solid-State Circ.*, vol. 27, pp. 473–484, Apr. 1992.

[4] K. K. Parhi, "Algorithm transformation techniques for concurrent processors," *Proc. IEEE*, vol. 77, pp. 1879–1895, Dec. 1989.

[5] P. M. Kogge, "Parallel solution of recurrence problems," *IBM J. Res. Develop.*, vol. 18, pp. 138–148, Mar. 1974.

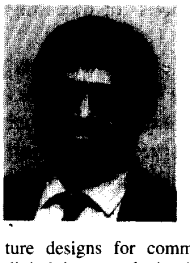
[6] H. H. Loomis and B. Sinha, "High speed recursive digital filter realization," *Circ. Syst. Signal Processing*, vol. 3, no. 3, pp. 267–294, 1984.

[7] K. K. Parhi and D. G. Messerschmitt, "Concurrent architectures for two-dimensional recursive digital filtering," *IEEE Trans. Circ. Syst.*, vol. 36, pp. 813–829, June 1989.
[8] K. K. Parhi, "Pipelining in dynamic programming architectures," *IEEE Trans. Signal Processing*, vol. 39, pp. 1442–1450, June 1991.
[9] ——, "High-speed VLSI architectures for Huffman and Viterbi decoders," *IEEE Trans. Circ. Syst. II: Analog and Digital Signal Processing*, vol. 39, pp. 385–391, June 1992.
[10] H.-D. Lin and D. G. Messerschmitt, "Finite state machine has unlimited concurrency," *IEEE Trans. Circ. Syst.*, vol. 38, pp. 465–475, May 1991.
[11] K. K. Parhi, "Pipelining in algorithms with quantizer loops," *IEEE Trans. Circ. Syst.*, vol. 38, pp. 745–754, July 1991.
[12] K. K. Parhi and D. G. Messerschmitt, "Concurrent cellular VLSI adaptive filter architectures," *IEEE Trans. Circ. Syst.*, vol. CAS-34, pp. 1141–1151, Oct. 1987.
[13] T. Meng and D. G. Messerschmitt, "Arbitrarily high sampling rate adaptive filters," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-35, pp. 455–470, Apr. 1987.
[14] M. Hatamian and K. K. Parhi, "An 85 MHz 4th order programmable IIR digital filter chip," *IEEE J. Solid-State Circ.*, vol. 27, pp. 175–183, Feb. 1992.
[15] N. R. Shanbhag and K. K. Parhi, *Pipelined Adaptive Digital Filters*. Boston, MA: Kluwer Academic Publishers, 1994.
[16] ——, "Relaxed Look-ahead pipelined LMS adaptive filters and their application to ADPCM coder," *IEEE Trans. Circ. Syst.—Part II*, vol. 40, pp. 753–766, Dec. 1993.
[17] ——, "A pipelined adaptive lattice filter architecture," *IEEE Trans. Signal Processing*, vol. 41, pp. 1925–1939, May 1993.
[18] ——, "A pipelined adaptive differential vector quantizer for real-time video compression," in *Proc. IEEE Workshop Visual Signal Processing Commun.*, Raleigh, NC, Sept. 1992, pp. 9–14.
[19] G. W. Davidson *et al.*, "An investigation of block adaptive decision feedback equalization for frequency selective fading," in *IEEE Int. Conf. Commun.*, 1988.
[20] A. Gatherer and T. H.-Y. Meng, "High sampling rate adaptive decision feedback equalizer," *IEEE Trans. Signal Processing*, vol. 41, pp. 1000–1005, Feb. 1993.
[21] J. M. Cioffi, P. Fortier, S. Kasturia, and G. Dudevoir, "Pipelining the decision feedback equalizer," in *Proc. IEEE DSP Workshop*, 1988.
[22] K. J. Raghunath and K. K. Parhi, "Parallel adaptive decision feedback equalizers," *IEEE Trans. Signal Processing*, vol. 41, pp. 1956–1961, May 1993.
[23] F. Lu and H. Samueli, "A 60 MBd, 480 Mb/s, 256-QAM decision-feedback equalizer in 1.2 μ CMOS," *IEEE J. Solid-State Circuits*, vol. 28, pp. 330–338, Mar. 1993.
[24] M. Schobinger *et al.*, "CMOS digital adaptive decision feedback equalizer chip for multilevel QAM digital radio modems," in *Proc. IEEE Int. Symp. Circ. Syst.*, May 1990, pp. 574–557.
[25] C. Leiserson and J. Saxe, "Optimizing synchronous systems," *J. VLSI Comput. Syst.*, vol. 1, pp. 41–67, 1983.
[26] G. Long, F. Ling, and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 37, pp. 1397–1405, Sept. 1989.
[27] ——, "Corrections to "The LMS algorithm with delayed coefficient adaptation"," *IEEE Trans. Signal Processing*, vol. 40, pp. 230–232, Jan. 1992.
[28] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
[29] K. K. Parhi, C.-Y. Wang, and A. P. Brown, "Synthesis of control circuits in folded pipelined DSP architectures," *IEEE J. Solid-state Circ.*, vol. 27, pp. 29–43, Jan. 1992.
[30] K. D. Fisher, J. M. Cioffi, W. L. Abbot, P. S. Bednarz, and C. M. Melas, "An adaptive RAM-DFE for storage channels," *IEEE Trans. Commun.*, vol. 39, pp. 1559–1568, Nov. 1991.
[31] I. Megory–Cohen, C. M. Melas, M. Hassner, and T. Howell, "An analytical model for magnetic readback pulses," IBM Res. Rep. RJ5236, July 1986.
[32] C. A. Belfiore and J. H. Park, Jr., "Decision feedback equalization," *Proc. IEEE*, vol. 67, pp. 1143–1156, Aug. 1979.
[33] W. Y. Chen, G. H. Im, and J. J. Werner, "Design of digital carrierless AM/PM transceivers," AT&T/Bellcore Contribution T1E1.4/92-149, Aug. 1992.
[34] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall.
[35] W. A. Gardner, "Learning characteristics of stochastic gradient descent algorithms: A general study, analysis and critique," *Signal Processing*, vol. 6, pp. 113–133, 1984.
[36] J. M. Cioffi, W. L. Abbot, H. K. Thapar, C. M. Melas, and K. D. Fisher, "Adaptive equalization in magnetic-disk storage channels," *IEEE Commun. Mag.*, pp. 14–29, Feb. 1990.
[37] J.-G. Chung and K. K. Parhi, "Pipelining of lattice IIR digital filters," *IEEE Trans. Signal Processing*, vol. 42, pp. 751–761, Apr. 1994.
[38] S. Kasturia, J. T. Aslanis, and J. M. Cioffi, "Vector coding for partial response channels," *IEEE Trans. Info. Theory*, vol. 36, pp. 741–753, July 1990.

Naresh R. Shanbhag (S'86–M'93) was born in 1966 in India. He received the B.Tech. degree from the Indian Institute of Technology, New Delhi, in 1988, the M.S. degree from Wright State University, Dayton, OH, in 1990, and the Ph.D. degree from the University of Minnesota in 1993, all in electrical engineering.

Since 1993, he has been working with the AT&T Bell Laboratories, Murray Hill, NJ, in the area of integrated circuits for asynchronous transfer mode. His research interests are in the general area of VLSI architectures and algorithms for speech and image compression, adaptive filtering, and channel equalization. In addition, he is also interested in efficient VLSI implementation methodologies for digital signal processing and communications algorithms.

Dr. Shanbhag received the 1994 Darlington Best Paper Award from the IEEE Circuits and Systems Society, and is co-author of the book *Pipelined Adaptive Digital Filters* (Kluwer, 1994).



Keshab K. Parhi (S'85–M'88–SM'91) received the B. Tech. (Hon.) degree from the Indian Institute of Technology, Kharagpur, in 1982, the M.S.E.E. degree from the University of Pennsylvania, Philadelphia, in 1984, and the Ph.D. degree from the University of California, Berkeley, in 1988.

He has been with the University of Minnesota since 1988, where he is currently a Professor of Electrical Engineering. He has also held short-term positions in several industries. His research interests include concurrent algorithm and architecture designs for communications, signal and image processing systems, digital integrated circuits, VLSI digital filters, computer arithmetic, high-level DSP synthesis, and multiprocessor prototyping and task scheduling for programmable software systems. He has published over 130 papers in these areas.

Dr. Parhi received the 1994 Darlington and the 1993 Guillemin-Cauer Best Paper awards from the IEEE Circuits and Systems Society, the 1991 Best Paper award from the IEEE Signal Processing Society, the 1991 Browder Thompson prize paper award from the IEEE, the 1992 Young Investigator Award of the National Science Foundation, the 1992–1994 McKnight-Land Grant professorship of the University of Minnesota, and the 1987 Eliahu Jury Award for Excellence in Systems Research at the University of California, Berkeley. He is a former Associate Editor of Image Processing and VLSI Applications in the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS and is a current Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING, an editor of the *Journal of VLSI Signal Processing*, and is a member of Eta Kappa Nu.