# Low-Power Adaptive Filter Architectures and Their Application to 51.84 Mb/s ATM-LAN

Naresh R. Shanbhag, *Member, IEEE*, and Manish Goel

***Abstract*— In this paper, we present low-power and high-speed algorithms and architectures for complex adaptive filters. These architectures have been derived via the application of *algebraic* and *algorithm transformations*. The *strength reduction* transformation is applied at the algorithmic level as opposed to the traditional application at the architectural level. This results in a power reduction by 21% as compared with the traditional cross-coupled structure. A fine-grained pipelined architecture for the strength-reduced algorithm is then developed via the *relaxed lookahead* transformation. This technique, which is an approximation of the conventional *lookahead computation*, maintains the functionality of the algorithm rather than the input–output behavior. Convergence analysis of the proposed architecture has been presented and supported via simulation results. The pipelined architecture allows high-speed operation with negligible hardware overhead. It also enables an additional power savings of 39 to 69% when combined with power-supply reduction. Thus, an overall power reduction ranging from 60–90% over the traditional cross-coupled architecture is achieved. The proposed architecture is then employed as a receive equalizer in a communication system for a data rate of 51.84 Mb/s over 100 m of UTP-3 wiring in an ATM-LAN environment. Simulation results indicate that speedups of up to 156 can be achieved with about a 0.8-dB loss in performance.**

## I. Introduction

DIGITAL communications systems are currently being developed for high-bit rate transmission over bandlimited channels. These applications include asymmetric digital subscriber loop [8], [22] (ADSL), high-speed digital subscriber loop (HDSL) [20], [26], [45], very high-speed digital subscriber loop (VHDSL) [7], [17], ATM-LAN [18] and interactive multimedia television (IMTV) [19], high-density magnetic recording [9], wireless systems [1], and digital high-definition TV (HDTV) transmission [29], [30]. In each of these applications, the bandlimited nature of the channel and the required performance levels necessitate the use of highly complex digital communications algorithms.

In addition to the increasing computational complexity, the requirements on a silicon implementation have also become stringent at the same time. There is no doubt that a cost-effective silicon implementation is critical for a successful deployment of any new technology. Therefore, constraints from a very large scale integration (VLSI) implementation perspective such as power dissipation, area, speed, and reliability also come into the picture. Design of high-speed and low-power algorithms and architectures is in great demand for all the above-mentioned applications. In particular, the advent of mobile applications has generated a great amount of interest in the design of low-power VLSI communications systems. Even in tethered applications, a low-power solution provides the added benefits of increased reliability and reduced packaging costs.

Design of low-power VLSI systems is presently an active area of research [5], [6], [16]. Power-reduction techniques have been proposed at all levels of the design hierarchy, beginning with algorithms and architectures and ending with circuits and technological innovations. Existing techniques include those at the algorithmic level (such as reduced complexity algorithms [5]), architectural level (such as pipelining [25], [32] and parallel processing [33]), logic (logic minimization [43] and precomputation [2]), circuit (reduced voltage swing [28] and adiabatic logic [3], [12]) and technological level [11]. It is now well recognized that an astute algorithmic and architectural design can have a large impact on the final power dissipation characteristics of the fabricated VLSI solution. In this paper, we will investigate algorithms and architectures for low-power and high-speed adaptive filters.

Adaptive equalizers are a major component of receivers in modern day communications systems accounting for up to 90% of the gate count [38]. They are employed to combat various channel impairments such as intersymbol interference (ISI), channel variations, crosstalk, timing jitter, etc. With the drive toward increasingly higher transmission rates, there is a corresponding increase in the complexity of the adaptive receivers. Hence, there is a tremendous need for power, area, and speed optimized adaptive equalizer architectures.

Traditionally, the focus in algorithm design has been to obtain performance in terms of better signal-to-noise ratios (SNR's) and/or bit-error rates (BER's). The present trend is to trade off a small amount of performance via *algorithm transformation* techniques [31] for a much superior VLSI architecture. Algorithm transformation techniques [6], [31] such as *lookahead* [32], *relaxed lookahead* [37], *block processing* [33], *associativity* [36], *unfolding* [15], [34], *folding* [35], and *retiming* [21] have all been employed to design high-speed algorithms and architectures. Low-power operation was then achieved by trading off excess speed with power.

A class of transformations known as *algebraic transformations* is of particular interest [36]. These transformations have been proposed to achieve arbitrarily high speedups in recursive algorithms. Strength reduction [5] is an algebraic

Manuscript received March 10, 1996; revised December 9, 1996. This work was supported by Analog Devices, Inc. and the University of Illinois Research Board. The associate editor coordinating the review of this paper and approving it for publication was Dr. K. J. Ray Liu.

The authors are with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA.

Publisher Item Identifier S 1053-587X(97)03340-0.

transformation, which has been applied at the architectural level to trade off multiplications with additions. This results in an overall savings in area and power as multipliers are more expensive (both in terms of area and power) than adders. A key contribution of this paper is the application of the strength reduction transformation at the algorithmic level (instead of the architectural level) to obtain low-power adaptive filter algorithms. An algorithmic level application of strength reduction is shown to be much more effective in achieving power reduction as compared with an architectural level application.

While the application of strength reduction reduces the number of computations, it does not reduce the critical path computation time. In fact, application of strength reduction increases the critical path computation time. This results in a throughput limitation, which is undesirable in a high-sample rate environment. We address this problem via the application of relaxed lookahead transformation [37], which seeks to develop fine-grained pipelined architectures by approximating the architectures obtained via the lookahead technique. The relaxed lookahead technique results in a negligible performance loss while providing a hardware-efficient pipelined adaptive filter architecture. This technique has been applied to pipeline numerous adaptive algorithms including the least mean-squared (LMS) algorithm [39], the pipelined stochastic gradient lattice filter [40], and the pipelined adaptive decision feedback equalizer [41].

In a related work [13], it has been shown that a complex filter can be implemented with three real filters. Our work differs from [13] in that we also optimize the weight-update section, employ pipelining to reduce the critical path, and present a convergence analysis of the pipelined architecture.

In this paper, the relaxed lookahead is employed to pipeline the strength-reduced adaptive filter algorithm and, thus, achieve high speed. The application of relaxed lookahead technique allows one to increase the throughput with a negligible increase in hardware, which is very attractive from a VLSI implementation point of view.

Note that in this paper, we have chosen to work with the conventional direct-form full-LMS algorithm because it has the longest critical path and is therefore more difficult to pipeline. Nevertheless, it is possible to apply strength reduction and relaxed lookahead pipelining to reduced complexity LMS algorithms such as the sign LMS. In addition, the transpose LMS adaptive filter is another candidate to which the proposed approach in this paper can be applied.

We demonstrate an application of the proposed low-power adaptive filter architecture in a high-speed communication system. In particular, the proposed filter is employed as an equalizer for the 51.84-Mb/s transmission over unshielded twisted pair (UTP-3) wiring using a 16-CAP (for carrierless amplitude/phase) modulation scheme. It must be mentioned that the 16-CAP line code was recently chosen as an ATM-LAN standard over UTP-3 at 51.84 Mb/s [18]. Therefore, this study is of great interest as it can lead to low-power and cost-effective ATM-LAN transceivers.

The organization of this paper is as follows. In Section II, we present a review of algebraic transformations and the relaxed lookahead pipelining technique. The strength reduction transformation is applied to the conventional cross-coupled filter architecture in Section III. Relaxed lookahead pipelined filter architecture is then developed in Section IV. Section V presents simulations results for verifying the convergence analysis of the pipelined architecture. Finally, in Section VI, we demonstrate the application of the proposed architecture for 51.84 Mb/s ATM-LAN environment.

## II. Preliminaries

In this section, we will review algebraic transformations and relaxed lookahead pipelining. We start with the description of the strength reduction transformation and its relation to low-power operation. Next, we will illustrate the relaxed lookahead form of pipelining and present the pipelined LMS algorithm [37].

### A. Algebraic Transformations

Algebraic transformations are an important class of architectural level transformations, which have been proposed for high speed [36] and for low power [5]. These transformations rely on the fact that most linear DSP algorithms can be expressed in terms of multiply–add operations. Hence, algebraic transformations such as associativity [6], [36], distributivity [36], common subexpression replication, common subexpression elimination, manifest expression elimination, and commutativity can be employed to improve either the throughput or reduce the complexity of the algorithm under consideration. In particular, the strength reduction transformation trades off high-complexity multiply operations with low-complexity add operations, thus achieving low power. In this paper, we will consider the strength reduction transformation and its role in achieving low power.

Consider the problem of computing the product of two complex numbers $(a+jb)$ and $(c+jd)$ as follows:

$$(a+jb)(c+jd) = (ac-bd) + j(ad+bc). \tag{2.1}$$

From (2.1), a direct-mapped architectural implementation would require a total of four real multiplications and two real additions to compute the complex product. However, it is possible to reduce this complexity via strength reduction [4], [5]. Application of strength reduction involves reformulating (2.1) as follows:

$$(a-b)d + a(c-d) = ac - bd \tag{2.2a}$$

$$(a-b)d + b(c+d) = ad + bc. \tag{2.2b}$$

As can be seen from (2.2), the number of real multiplications is three, and the number of additions is five. Therefore, this form of strength reduction reduces the number of multipliers by one at the expense of three additional adders. Typically, multiplications are more expensive than additions, and hence, we achieve an overall savings in hardware.

Comparing (2.1) and (2.2), we find that the strength reduction transformation also increases the critical path length, where the critical path is defined as the longest path from the input to the output. The critical path computation time of the

original system $(T_{c,o})$ and that of the strength-reduced system $(T_{c,sr})$ is given by

$$T_{c,o} = T_m + T_a \tag{2.3a}$$

$$T_{c,sr} = T_m + 2T_a \tag{2.3b}$$

where $T_m$ and $T_a$ are two-operand multiply and add times, respectively. This is a drawback of the strength-reduction transformation, which makes it undesirable in high-speed applications of interest in this paper. However, this problem can be easily solved by employing throughput enhancing techniques such as pipelining.

The dynamic power dissipation $P_D$ in CMOS technology is given by

$$P_D = C_L V_{dd}^2 f \tag{2.4}$$

where

$C_L$ average capacitance being switched,
$V_{dd}$ supply voltage,
$f$ frequency of operation.

Most of the existing power reduction techniques involve reducing one or more of the three quantities $C_L, V_{dd}$, and $f$. The strength reduction transformation achieves low power by reduction of arithmetic operations, which corresponds to the reduction of $C_L$ in (2.4).

In order to estimate the power savings due to this transformation, we assume that the effective capacitance of a two-operand multiplier is a factor $K_C$ times that of a two-operand adder. The factor $K_C$ depends on the relative precisions of the multiplier and the adder and their respective implementation styles. It can be seen from (2.1) and (2.2) that strength reduction results in a power savings factor PS of

$$\text{PS} = \frac{P_D(\text{original}) - P_D(\text{strength-reduced})}{P_D(\text{original})} \tag{2.5a}$$

$$= \frac{K_C - 3}{2(2K_C + 1)} \tag{2.5b}$$

where $P_D(\text{original})$ and $P_D(\text{strength-reduced})$ are the dynamic power dissipation of the original [see (2.1)] and strength-reduced [see (2.2)] algorithms. From (2.5b), it is clear that the strength-reduced architecture will achieve power savings as long as $K_C > 3$. Furthermore, it is clear from (2.5) that the power savings approach an asymptotic value of 25% as $K_C$ increases. If we assume array-based multiplier structures, then $K_C$ is approximately equal to $N_B$, where $N_B$ is the number of bits required to represent one input operand. Hence, it would be beneficial to employ the proposed transformation as long as the adders and multipliers have inputs with four or more bits. This is typically the case in the applications of interest where the required SNR dictates 7 to 8 bits of input precision. It can be easily checked that initially, the power savings increase rapidly as a function of $K_C$ with more than 15% savings obtained with $K_C = 10$.

### *B. Pipelining with Relaxed Lookahead*

The relaxed lookahead pipelining technique [37] allows very high sampling rates to be achieved with minimal hardware overhead. As mentioned before, the relaxed lookahead technique is an approximation to the lookahead technique [32]. Many approximations (which are also referred to as relaxations) can be formulated. However, we will consider only the *delay* and *sum* relaxations, which have proved to be very effective in pipelining the LMS [39] algorithm.

Consider the first-order recursion

$$w(n) = w(n-1) + a(n)x(n). \tag{2.6}$$

The computation time of (2.6) is lower bounded by a single add time. Next, we apply an $M$-step lookahead to (2.6) in the time-domain and obtain

$$w(n) = w(n-M) + \sum_{i=0}^{M-1} a(n-i)x(n-i). \tag{2.7}$$

This transformation introduces $M$ latches into the recursive loop, which can be retimed [21] to attain $M$-level pipelining of the add operation. Note that this transformation has not altered the input–output behavior. This invariance with respect to the input–output behavior has been achieved at the expense of the lookahead overhead term [the second term in (2.7)], which can be expensive. The relaxed lookahead technique involves approximating architectures such as those described by (2.7), which have been derived via lookahead technique. Delay and sum relaxations are two possible approximations, which will be described next.

The delay relaxation involves the use of delayed input $x(n - D_1)$ and delayed coefficient $a(n - D_1)$ in (2.7). If the average value of the product $a(n)x(n)$ is more or less constant over $D_1$ samples, then (2.7) can be approximated as

$$w(n) = w(n-M) + \sum_{i=0}^{M-1} a(n-D_1-i)x(n-D_1-i). \tag{2.8}$$

Note that this approximation results in the "delayed LMS" [23], [24] algorithm when applied to the traditional LMS algorithm [46]. In general, this is a reasonable approximation for stationary or slowly varying product $a(n)x(n)$ in (2.7).

Application of the sum relaxation to (2.7) involves taking $LA$ terms from (2.7), where $LA \leq M$, to get

$$w(n) = w(n-M) + \frac{M}{LA} \sum_{i=0}^{LA-1} a(n-i)x(n-i). \tag{2.9}$$

This relaxation can be justified if the average value of the product $a(n)x(n)$ is slowly varying, and simulations for LMS filters indicate this to be a good approximation.

In addition to the two relaxations presented above, other relaxations can be defined by approximating the algorithm obtained via application of lookahead. The application of these relaxations, either individually or in different combinations, results in a rich variety of architectures. However, these

architectures will have different convergence properties, and it is necessary to analyze their convergence behavior.

The delay and sum relaxations have been employed to pipeline the LMS algorithm [37]. Consider the serial LMS (SLMS) filter described by the following equations:

$$\begin{aligned} \boldsymbol{W}(n) &= \boldsymbol{W}(n-1) + \mu e(n)\boldsymbol{X}(n); \\ e(n) &= d(n) - \boldsymbol{W}^T(n-1)\boldsymbol{X}(n) \end{aligned} \tag{2.10}$$

where

| | |
|---|---|
| $\boldsymbol{W}(n)$ | weight vector, |
| $\boldsymbol{X}(n)$ | input vector, |
| $e(n)$ | adaptation error, |
| $\mu$ | step size, |
| $d(n)$ | desired signal. |

The critical path for the serial LMS [39] is given by

$$T_{c,\mathrm{SLMS}} = 2T_m + (N+1)T_a \tag{2.11}$$

where $N$ is equal to the number of taps in the filter block (or **F**-block).

The computation time of the critical path can be reduced by developing a pipelined LMS architecture. The relaxed lookahead pipelined LMS architecture (see [39] for details) is given by

$$\begin{aligned} \boldsymbol{W}(n) &= \boldsymbol{W}(n-D_2) + \mu \sum_{i=0}^{LA-1} e(n-D_1-i)\boldsymbol{X}(n-D_1-i); \\ e(n) &= d(n) - \boldsymbol{W}^T(n-D_2)\boldsymbol{X}(n) \end{aligned} \tag{2.12}$$

where $D_1$ delays are introduced via the delay relaxation, and $D_2$ delays are introduced via the sum relaxation. The $D_1$ and $D_2$ delays can be employed to pipeline the hardware operators in an actual implementation. In fact, the strategic location of $D_1$ and $D_2$ delays enables pipelining of all the arithmetic operations at a fine-grain level. Relaxed lookahead pipelined filters have found practical applications in the design of a 100-MHz adaptive differential pulse code modulation (ADPCM) video codec chip [42], the 51.84-Mb/s ATM-LAN [18], [38], and IMTV transceiver chip sets [19].

The application of relaxed lookahead requires a subsequent convergence analysis of the pipelined filter. This analysis has been done in [39], and the interested reader is referred to [37] and [39] for details. In this paper, we will employ the relaxed lookahead pipelined LMS filter to obtain pipelined filter architectures. Note that the increased throughput due to pipelining can be employed to do the following:

1) meet the speed requirements,
2) reduce power (in combination with power supply scaling)
3) reduce area (in combination with folding transformation [35]).

## III. Algebraic Transformations for Low Power

Algebraic transformations have been applied at the architectural level in the past [6], [36]. While the strength reduction transformation in (2.2) can also be applied at the architectural level for increased hardware savings, we propose to apply them at the algorithmic level. It will be seen that the impact on the hardware requirements is much greater when the proposed strength reduction transformation is applied at the algorithmic level. In particular, we will assume that a passband digital communication system such as quadrature amplitude modulation (QAM) [14] or carrierless amplitude/phase (CAP) modulation [45] is being employed. In this situation, the receiver processes a 2-D signal with a 2-D filter. This results in the traditional cross-coupled structure, which will be the starting point of our work.

### A. Traditional Cross-Coupled Equalizer Architecture

The output of the filtering block in an LMS algorithm can be written as

$$y(n) = \boldsymbol{W}^T(n-1)\boldsymbol{X}(n). \tag{3.1}$$

Clearly, if the input $\boldsymbol{X}(n)$ and the filter $\boldsymbol{W}(n)$ are complex quantities, then we can apply the strength reduction transformation (2.2) to the polynomial multiplication in (3.1) to obtain a low-power architecture.

Modulation schemes such as quadrature amplitude modulation QAM [14] and CAP [45] employ a 2-D signal constellation, which can be represented as a complex signal. If a complex filter is to be implemented, then we can represent its output as a complex polynomial product. Furthermore, if the transformation in (2.2) is employed, then we would need only three real filters [instead of four as in (2.1)]. Each real filter requires $N$ multiplications and $N-1$ additions. Therefore, the application of the proposed transformation in (2.2) would then save a substantial amount of hardware.

Let the filter input be a complex signal $\tilde{\boldsymbol{X}}(n)$ defined as

$$\tilde{\boldsymbol{X}}(n) = \boldsymbol{X}_r(n) + j\boldsymbol{X}_i(n) \tag{3.2}$$

where $\boldsymbol{X}_r(n)$ and $\boldsymbol{X}_i(n)$ are the real and imaginary parts, respectively. Furthermore, if the filter is also complex i.e., $\tilde{\boldsymbol{W}}(n) = \boldsymbol{c}(n) + j\boldsymbol{d}(n)$, then its output $\tilde{y}(n)$ can be obtained as follows:

$$\begin{aligned} \tilde{y}(n) &= \tilde{\boldsymbol{W}}^H(n-1)\tilde{\boldsymbol{X}}(n) \\ &= [\boldsymbol{c}^T(n-1) - j\boldsymbol{d}^T(n-1)][\boldsymbol{X}_r(n) + j\boldsymbol{X}_i(n)] \\ &= [\boldsymbol{c}^T(n-1)\boldsymbol{X}_r(n) + \boldsymbol{d}^T(n-1)\boldsymbol{X}_i(n)] \\ &\quad + j[\boldsymbol{c}^T(n-1)\boldsymbol{X}_i(n) - \boldsymbol{d}^T(n-1)\boldsymbol{X}_r(n)] \\ &= y_r(n) + jy_i(n) \end{aligned} \tag{3.3}$$

where $\tilde{\boldsymbol{W}}^H$ represents the Hermitian (transpose and complex conjugate) of the matrix $\tilde{\boldsymbol{W}}$. A direct implementation of (3.3) results in the traditional cross-coupled structure shown in Fig. 1. This structure requires four FIR filters and two output adders, which amounts to $4N-2$ adders and $4N$ multipliers. If the channel impairments include severe ISI and/or multipath, then the number of taps necessary can be quite large, resulting in a high-complexity and high power dissipation.

In the adaptive case, a weight-update (WUD) block would be needed to automatically compute the coefficients of the filter. This can be done by implementing a complex version of (2.10) as follows:

$$\tilde{\boldsymbol{W}}(n) = \tilde{\boldsymbol{W}}(n-1) + \mu\tilde{e}^*(n)\tilde{\boldsymbol{X}}(n) \tag{3.4}$$

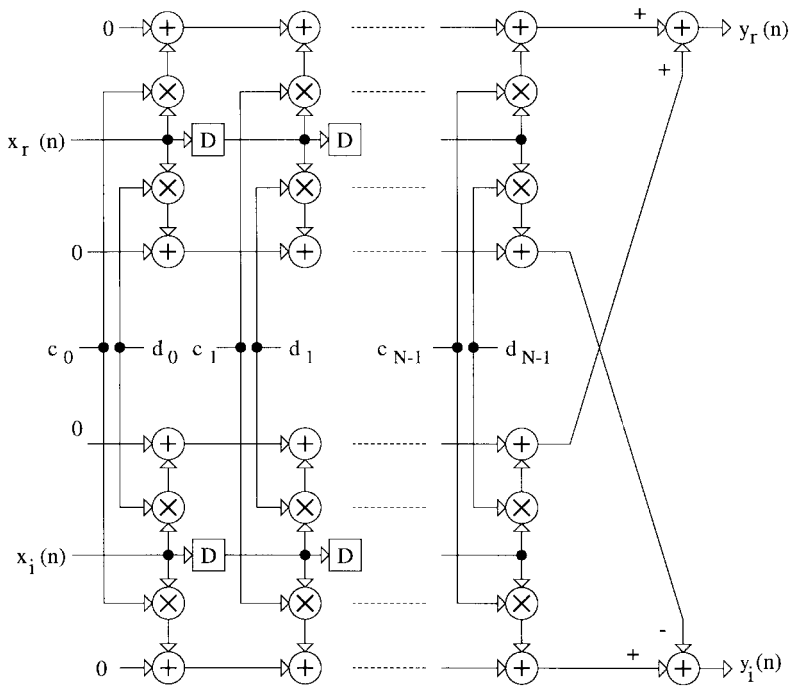



Fig. 1. Traditional cross-coupled filter architecture.

where

$$\tilde{e}(n) = e_r(n) + je_i(n), \quad e_r(n) = Q[y_r(n)] - y_r(n),$$
$$e_i(n) = Q[y_i(n)] - y_i(n)$$

where $Q[\cdot]$ is the output of the slicer, and $\tilde{e}^*$ represents the complex conjugate of $\tilde{e}$. Next, we substitute these definitions of $\tilde{\boldsymbol{W}}(n), \tilde{e}(n)$, and $\tilde{\boldsymbol{X}}(n)$ into (3.4) to obtain the following two real update equations:

$$\boldsymbol{c}(n) = \boldsymbol{c}(n-1) + \mu[e_r(n)\boldsymbol{X}_r(n) + e_i(n)\boldsymbol{X}_i(n)] \tag{3.5a}$$
$$\boldsymbol{d}(n) = \boldsymbol{d}(n-1) + \mu[e_r(n)\boldsymbol{X}_i(n) - e_i(n)\boldsymbol{X}_r(n)]. \tag{3.5b}$$

The WUD block architecture for computing (3.5) is shown in Fig. 2. It is clear that the hardware requirements are $4N+2$ adders and $4N$ multipliers for an $N$-tap 2-D filter. In the next subsection, we will present a low-power adaptive filter architecture using strength reduction.

### *B. Low Power Equalizer Architecture via Strength Reduction*

Observing (3.3) and (3.4), it is clear that strength reduction transformation (2.2) can be applied to the two complex multiplications present in it. We will see that this application of the transformation at the algorithmic level is much more effective in reducing power as opposed to an architectural level application. Applying the proposed transformation to (3.3) first, we obtain

$$\begin{aligned}\tilde{y}(n) &= \tilde{\boldsymbol{W}}^H(n-1)\tilde{\boldsymbol{X}}(n) = \tilde{\boldsymbol{X}}^T(n)\tilde{\boldsymbol{W}}^*(n-1)\\ &= [\boldsymbol{X}_r^T(n) + j\boldsymbol{X}_i^T(n)][\boldsymbol{c}(n-1) - j\boldsymbol{d}(n-1)]\\ &= [y_1(n) + y_3(n)] + j[y_2(n) + y_3(n)]\end{aligned} \tag{3.6}$$

where

$$\begin{aligned}y_1(n) &= [\boldsymbol{c}^T(n-1) + \boldsymbol{d}^T(n-1)]\boldsymbol{X}_r(n)\\ &= \boldsymbol{c}_1^T(n-1)\boldsymbol{X}_r(n)\end{aligned} \tag{3.7a}$$
$$\begin{aligned}y_2(n) &= [\boldsymbol{c}^T(n-1) - \boldsymbol{d}^T(n-1)]\boldsymbol{X}_i(n)\\ &= \boldsymbol{d}_1^T(n-1)\boldsymbol{X}_i(n)\end{aligned} \tag{3.7b}$$
$$\begin{aligned}y_3(n) &= -\boldsymbol{d}^T(n-1)[\boldsymbol{X}_r(n) - \boldsymbol{X}_i(n)]\\ &= -\boldsymbol{d}^T(n-1)\boldsymbol{X}_1(n)\end{aligned} \tag{3.7c}$$

where

$$\boldsymbol{X}_1(n) = \boldsymbol{X}_r(n) - \boldsymbol{X}_i(n), \quad \boldsymbol{c}_1(n) = \boldsymbol{c}(n) + \boldsymbol{d}(n) \quad \text{and}$$
$$\boldsymbol{d}_1(n) = \boldsymbol{c}(n) - \boldsymbol{d}(n).$$

The proposed architecture (see Fig. 3) requires three filters and two output adders. This corresponds to $4N$ adders and $3N$ multipliers, which is approximately a 25% reduction in the hardware, as compared with the traditional structure (see Fig. 1). It, therefore, represents an attractive alternative from a VLSI perspective.

We now consider the adaptive version and specifically analyze the WUD block. From (3.7) and Fig. 3, it seems that an efficient architecture may result if

$$\boldsymbol{c}_1(n-1) = [\boldsymbol{c}(n-1) + \boldsymbol{d}(n-1)] \quad \text{and}$$
$$\boldsymbol{d}_1(n-1) = [\boldsymbol{c}(n-1) - \boldsymbol{d}(n-1)]$$

are adapted instead of $\boldsymbol{c}(n-1)$ and $\boldsymbol{d}(n-1)$. In order to see if this is the case, we will derive the update equation for $\boldsymbol{c}_1(n-1)$ and $\boldsymbol{d}_1(n-1)$ next.

Adding (3.5a) to (3.5b), we obtain the update equation for $\boldsymbol{c}_1(n-1)$ as follows:

$$\begin{aligned}\boldsymbol{c}_1(n) = \boldsymbol{c}_1(n-1) + \mu[&e_r(n)(\boldsymbol{X}_r(n) + \boldsymbol{X}_i(n))\\ &- e_i(n)(\boldsymbol{X}_r(n) - \boldsymbol{X}_i(n))].\end{aligned} \tag{3.8}$$

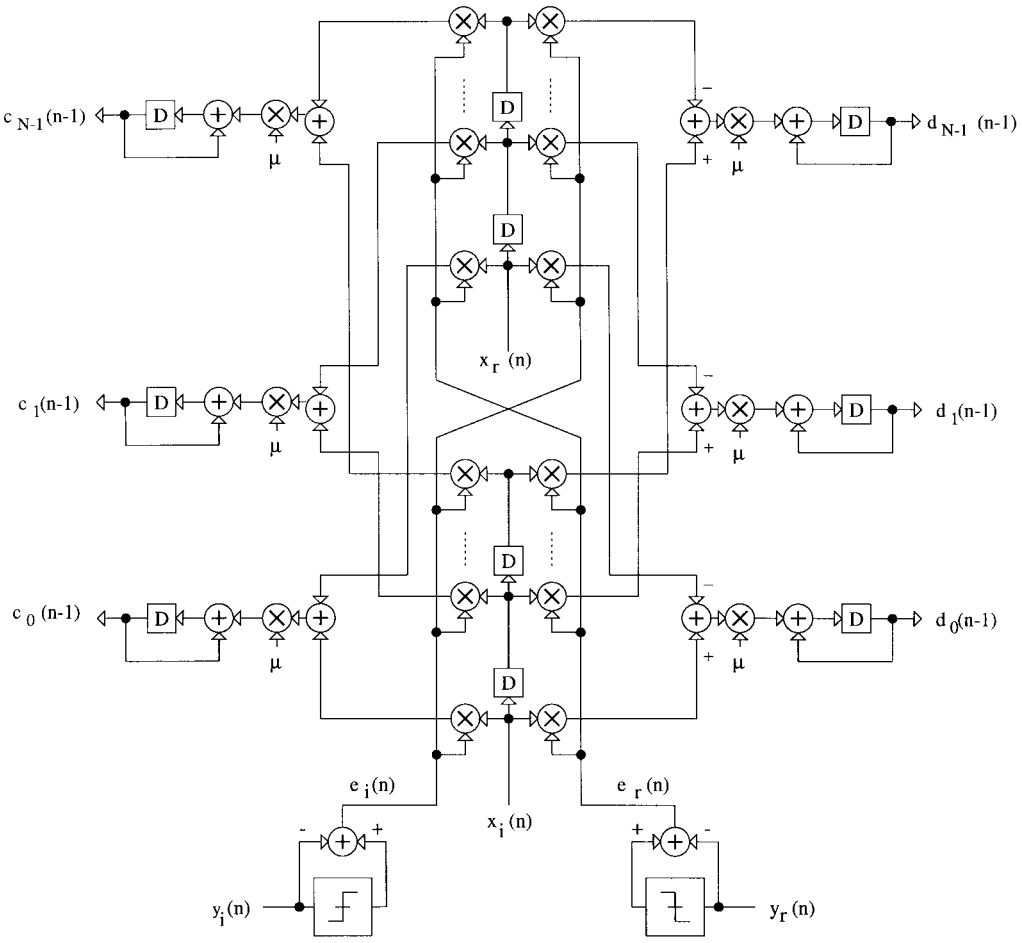



Fig. 2. Traditional weight-update block architecture.

In a similar fashion, subtracting (3.5b) from (3.5a) provides us with the corresponding equation for updating $\boldsymbol{d}_1(n-1)]$ as follows:

$$\boldsymbol{d}_1(n) = \boldsymbol{d}_1(n-1) + \mu[e_r(n)(\boldsymbol{X}_r(n) - \boldsymbol{X}_i(n)) + e_i(n)(\boldsymbol{X}_r(n) + \boldsymbol{X}_i(n))]. \tag{3.9}$$

It is now easy to show that (3.8) and (3.9) can be written in the following complex form:

$$\tilde{\boldsymbol{W}}_1(n) = \tilde{\boldsymbol{W}}_1(n-1) + \mu\tilde{e}(n)[(\boldsymbol{X}_r(n) + \boldsymbol{X}_i(n)) + j(\boldsymbol{X}_r(n) - \boldsymbol{X}_i(n))] \tag{3.10}$$

where $\tilde{\boldsymbol{W}}_1(n) = \boldsymbol{c}_1(n) + j\boldsymbol{d}_1(n)$. We can now apply the strength reduction transformation to the complex product in (3.10) to obtain a low-power WUD architecture. Doing so results in the following set of equations that describe the strength-reduced WUD block

$$\tilde{\boldsymbol{W}}_1(n) = \tilde{\boldsymbol{W}}_1(n-1) + \mu[\boldsymbol{eX}_1(n) + \boldsymbol{eX}_3(n) + j(\boldsymbol{eX}_2(n) + \boldsymbol{eX}_3(n))] \tag{3.11}$$

where

$$\boldsymbol{eX}_1(n) = 2e_r(n)\boldsymbol{X}_i(n) \tag{3.12a}$$

$$\boldsymbol{eX}_2(n) = 2e_i(n)\boldsymbol{X}_r(n) \tag{3.12b}$$

$$\boldsymbol{eX}_3(n) = [e_r(n) - e_i(n)][\boldsymbol{X}_r(n) - \boldsymbol{X}_i(n)] = e_1(n)\boldsymbol{X}_1(n) \tag{3.12c}$$

where $e_1(n) = e_r(n) - e_i(n)$, and $\boldsymbol{X}_1(n) = \boldsymbol{X}_r(n) - \boldsymbol{X}_i(n)$. The architecture corresponding to (3.11) and (3.12) is shown in Fig. 4. It can be seen that this WUD architecture requires only $3N$ multipliers and $4N + 3$ adders. Thus, the number of multipliers are reduced by one fourth at the expense of an additional adder as compared with the traditional WUD architecture (see Fig. 2).

Combining the architecture for the **F**-block in Fig. 3 and that for the WUD block in Fig. 4, we obtain the proposed strength-reduced low-power adaptive filter architecture in Fig. 5. A complete description of the low-power adaptive filter architecture is given by (3.6) and (3.7) as well as (3.11) and (3.12). In Fig. 5, we show the overall block diagram of the adaptive filter, where the FR block and the WUDR block compute (3.7a) and (3.12a), respectively. Similarly, the FI block and the WUDI block compute (3.7b) and (3.12b), respectively. Furthermore, the FRI and WUDRI blocks compute (3.7c) and (3.12c). Note that in Fig. 5, we have separated the slicer and the error computation adders from the WUDR and WUDI blocks. This is done only to depict the error feedback loop clearly. Henceforth, we will refer to the FR, FI, and FRI blocks as the **F**-blocks and the WUDR, WUDI, and WUDR blocks as the WUD blocks.

### *C. Comparison*

A comparison of the hardware requirements for the traditional cross-coupled structure and the proposed architecture is

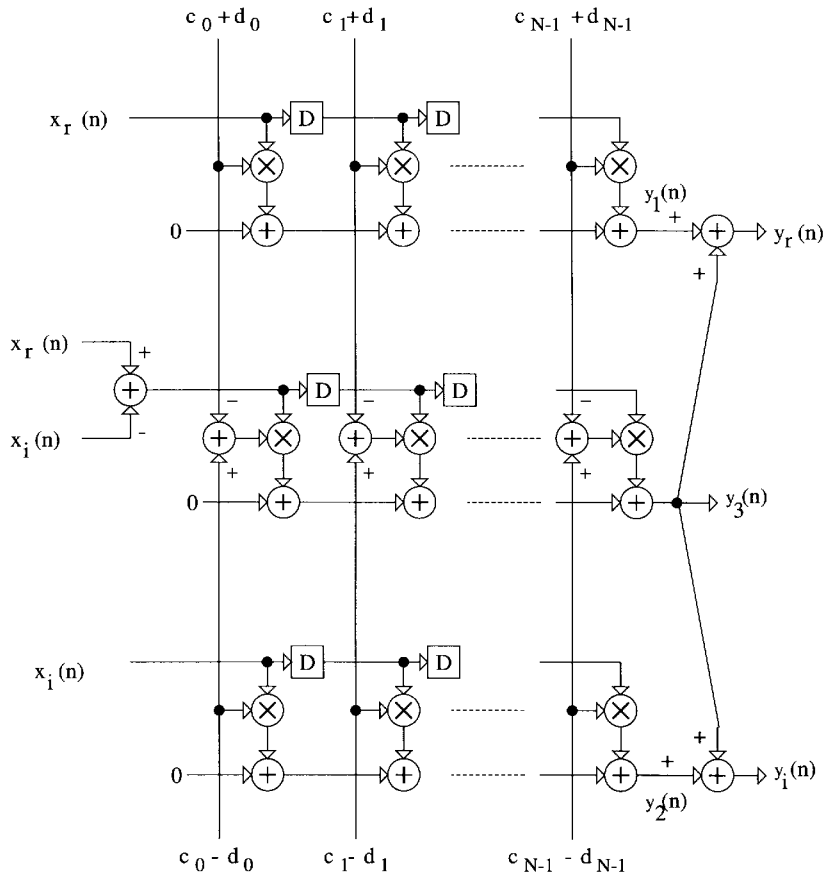



Fig. 3. Proposed filter architecture obtained via strength reduction.

TABLE I
HARDWARE REQUIREMENTS OF THE CROSS-COUPLED AND PROPOSED FILTER ARCHITECTURE

| ARCHITECTURE | F-BLOCK | | WUD-BLOCK | | TOTAL | |
|---|---|---|---|---|---|---|
| | # Mult. | # Add. | # Mult. | # Add. | # Mult. | # Add. |
| Cross-coupled | 4N | 4N-2 | 4N | 4N+2 | 8N | 8N |
| Proposed | 3N | 4N | 3N | 4N+3 | 6N | 8N+3 |

presented in Table I. It can be seen that for large values of $N$, the proposed architecture results in a 25% reduction in the number of multipliers at the expense of three additional adders. This reduction in hardware provides the dual benefits of lower area and lower power. Power reduction is derived from the fact that the switching capacitance in (2.4) is reduced.

We will now derive a more accurate estimate of the power savings achieved by the proposed low-power filter architecture. In order to do so, it is necessary to take into account the fact that in Figs. 1–5, the precision requirements on the adders in the WUD blocks (excluding the adders which compute the error) are typically twice that of the rest of the adders. There are $4N$ such adders in the traditional and the proposed structure. From Table I, we can see that the traditional architecture will have a switching capacitance, which is proportional to $8NK_C + 12N$. On the other hand, the switching capacitance for the proposed architecture can be shown to be proportional to $6NK_C + 12N + 3$. Substituting these values into the definition of PS in (2.5a), we can show that the power savings PS due to the proposed filter architecture is given by

$$\text{PS} = \frac{(2NK_C - 3)}{4(2NK_C + 3N)} \tag{3.13}$$

where $K_C$ is the ratio of the effective capacitance of a two-operand multiplier to that of a two-operand **F**-block adder. Note that the savings predicted by (3.13) are somewhat optimistic as the effect of latches have not been included. From (3.13), we can see that for large values of $K_C$ and $N$, the power reduction approaches an asymptotic value of 25%. A plot of (3.13) would indicate that most of the power savings would be obtained for values of $N$ approximately equal to 10. Even with a typical values of $K_C = 8$ and $N = 32$ in (3.13), the resulting area and power savings equal 21%. Thus, the benefits of the proposed architecture are obtained quite easily.

Applying the strength-reduction transformation to the traditional cross-coupled architecture (see Figs. 1 and 2) at the architectural level is also possible. We will now show that an architectural application of strength reduction is not as effective in reducing power dissipation as an algorithmic application proposed here. We note that in the traditional cross-coupled architecture, there are $N$ complex multipliers and $N-1$ complex adders for the **F**-block and $N$ complex multipliers and $N+1$ complex adders ($N$ adders being double precision) for the WUD block. This gives the total

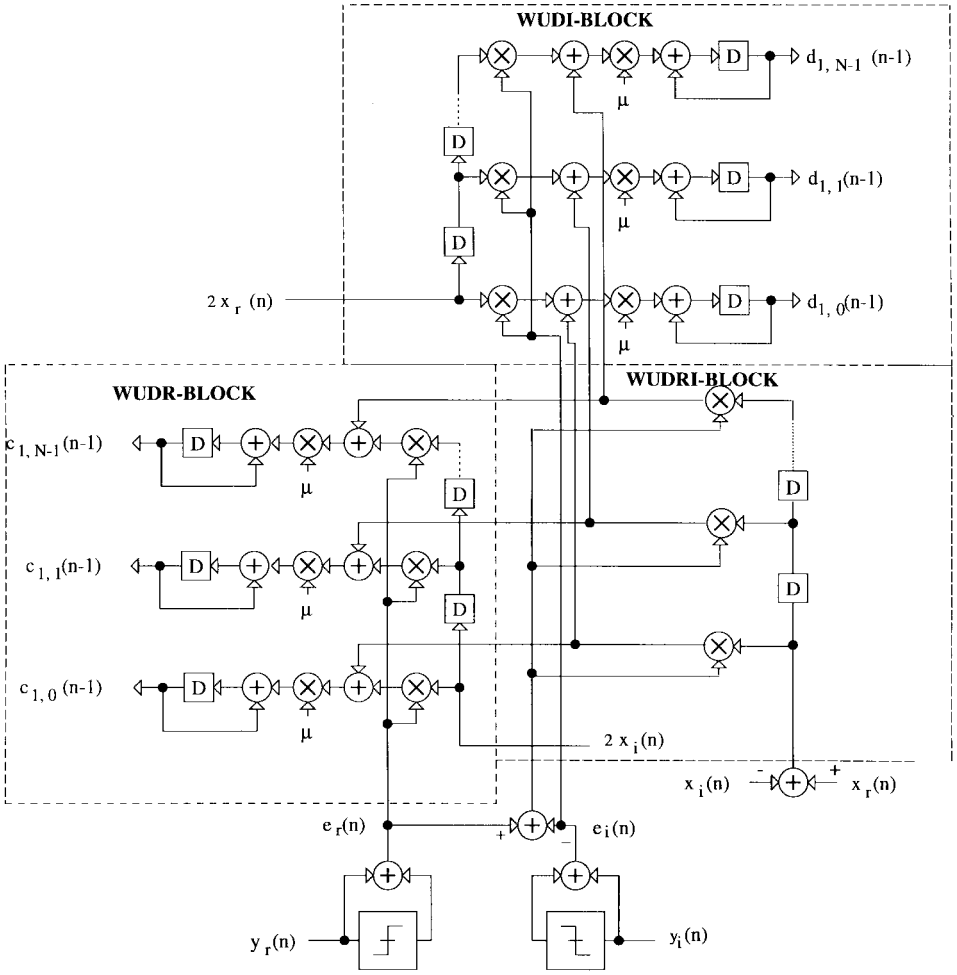



Fig. 4. Proposed weight-update block architecture obtained via strength reduction.

capacitance of $8NK_C + 12N$ times the capacitance of one **F**-block adder. Now, if we apply strength reduction to each of the complex multiplications in the **F**-block and WUD block, we get the switched capacitance equal to $6NK_C + 21N$ times the capacitance of one **F**-block adder. Substituting the switched capacitances for the two architectures into (2.5a), we obtain

$$\text{PS} = \frac{2K_C - 9}{8K_C + 12} \tag{3.14}$$

which is equal to 9.21% for $K_C = 8$. Thus, the application of the transformation at the algorithmic level is much more effective in reducing power as compared with the application at the architectural level.

## IV. Relaxed Lookahead Pipelined Architectures

In the previous section, the traditional cross-coupled architecture (see Figs. 1 and 2) and the proposed low-power architecture (see Fig. 5) were described and compared. It was seen that the proposed architecture provides a power savings of approximately 25% over the traditional structure. In the adaptive case, both architectures suffer from a throughput bottleneck due to the error feedback loop. This is clearly seen (Fig. 5) in the case of the strength-reduced architecture. In this section, we propose a solution to this problem by developing a pipelined version of the strength-reduced architecture shown in Fig. 5. We shall see that this process will allow us to trade off area and power for speed, thus achieving further power and area savings.

### A. Serial Equalizer Architecture (SEA)

We will refer to the architecture in Fig. 5 as the serial (or unpipelined) filter architecture (SEA), which is also described by (3.6) and (3.7) and by (3.11) and (3.12). In order to pipeline the SEA architecture, we will rewrite the equations for SEA as follows:

$$y_1(n) = \boldsymbol{c}_1^T(n-1)\boldsymbol{X}_r(n) \tag{4.1a}$$
$$\boldsymbol{c}_1(n) = \boldsymbol{c}_1(n-1) + \mu[2e_r(n)\boldsymbol{X}_i(n) + e_1(n)\boldsymbol{X}_1(n)] \tag{4.1b}$$
$$y_2(n) = \boldsymbol{d}_1^T(n-1)\boldsymbol{X}_i(n) \tag{4.1c}$$
$$\boldsymbol{d}_1(n) = \boldsymbol{d}_1(n-1) + \mu[2e_i(n)\boldsymbol{X}_r(n) + e_1(n)\boldsymbol{X}_1(n)] \tag{4.1d}$$
$$y_3(n) = -\boldsymbol{d}^T(n-1)\boldsymbol{X}_1(n) \tag{4.1e}$$
$$y_r(n) = y_1(n) + y_3(n); \quad y_i(n) = y_2(n) + y_3(n) \tag{4.1f}$$
$$e_r(n) = Q[y_r(n)] - y_r(n); \quad e_i(n) = Q[y_i(n)] - y_i(n). \tag{4.1g}$$

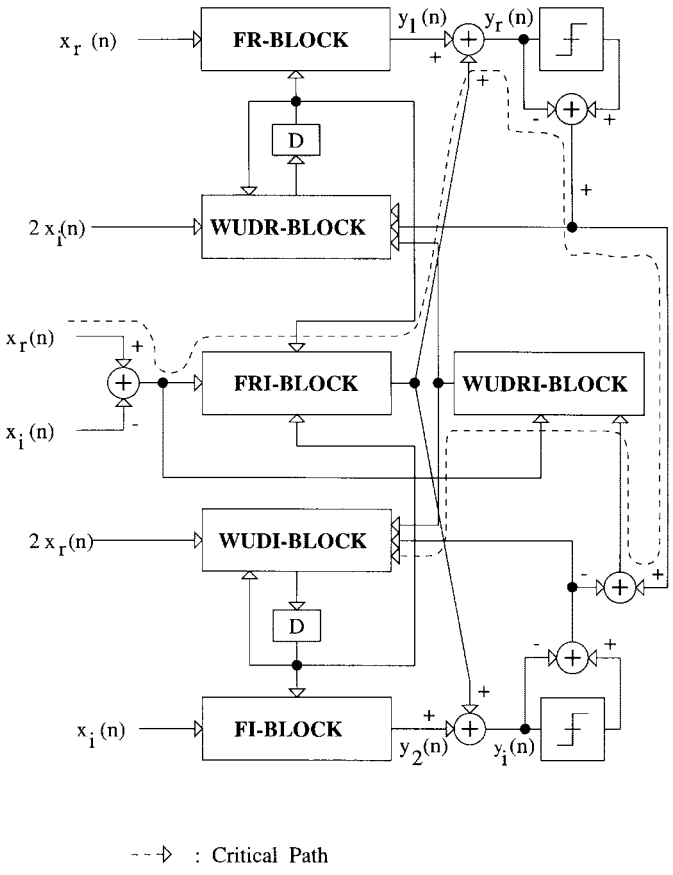



Fig. 5. Block diagram of the proposed adaptive filter architecture.

From Fig. 5, we observe that the input sample-period $T_{\text{SEA}}$ would be greater than the sum of the computation times of the FR(FI) block, the error computation, and the WUDR(WUDI) block. The critical path is indicated in Fig. 5. Obtaining the computation time for the FR(FI)-block from Fig. 3 and that for the WUDR(WUDI)-block from Fig. 4, we get

$$T_{\text{SEA}} \geq 2T_m + (N+7)T_a \tag{4.2}$$

where $T_m$ and $T_a$ are two-operand multiply and single-precision add times, respectively. For applications that require large values of $N$, the lower bound on $T_{\text{SEA}}$ in (4.2) may prevent a feasible implementation. This is particularly true for the high bit rate communications systems mentioned in the introduction. Note that this problem is also present in case of the traditional cross-coupled architecture in Figs. 1 and 2. We present a solution to this problem in the next subsection, where a pipelined filter architecture (PEA) is derived.

### *B. Pipelined Equalizer Architecture (PEA)*

In order to derive the PEA, we will start with the SEA equations (4.1) and then apply relaxed lookahead [37]. Observe that the two equation pairs [(4.1a)–(4.1b)] and [(4.1c)–(4.1d)] have a form similar to that of the traditional LMS described by (2.10). Hence, pipelining of SEA can be achieved via inspection of the relaxed lookahead pipelined LMS algorithm [39] given by (2.12). Doing so results in the following equations,

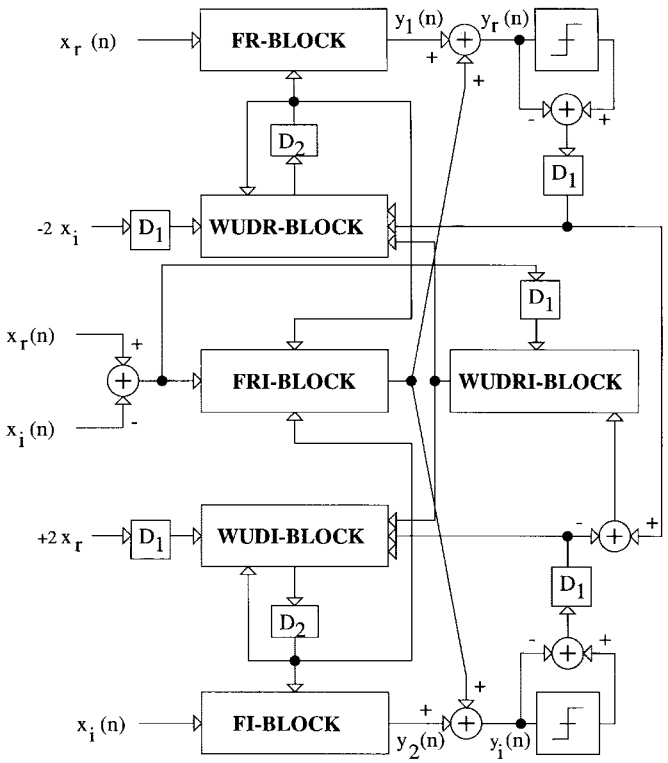



Fig. 6. Block diagram of the relaxed lookahead pipelined adaptive filter architecture.

which describe PEA:

$$y_1(n) = \boldsymbol{c}_1^T(n-D_2)\boldsymbol{X}_r(n) \tag{4.3a}$$

$$\begin{aligned}\boldsymbol{c}_1(n) = \boldsymbol{c}_1(n-D_2) + \mu \sum_{i=0}^{LA-1} [&2e_r(n-D_1-i) \\ &\cdot \boldsymbol{X}_i(n-D_1-i) + e_1(n-D_1-i) \\ &\cdot \boldsymbol{X}_1(n-D_1-i)]\end{aligned} \tag{4.3b}$$

$$y_2(n) = \boldsymbol{d}_1^T(n-D_2)\boldsymbol{X}_i(n) \tag{4.3c}$$

$$\begin{aligned}\boldsymbol{d}_1(n) = \boldsymbol{d}_1(n-D_2) + \mu \sum_{i=0}^{LA-1} [&2e_i(n-D_1-i) \\ &\cdot \boldsymbol{X}_r(n-D_1-i) + e_1(n-D_1-i) \\ &\cdot \boldsymbol{X}_1(n-D_1-i)]\end{aligned} \tag{4.3d}$$

$$y_3(n) = -\boldsymbol{d}^T(n-D_2)\boldsymbol{X}_1(n) \tag{4.3e}$$

$$y_r(n) = y_1(n) + y_3(n); \quad y_i(n) = y_2(n) + y_3(n) \tag{4.3f}$$

$$e_r(n) = Q[y_r(n)] - y_r(n); \quad e_i(n) = Q[y_i(n)] - y_i(n) \tag{4.3g}$$

where $LA$ is the lookahead factor ($LA \leq D_2$), and $D_1$ and $D_2$ are algorithmic delays that can be employed for hardware pipelining. The block diagram for PEA is shown in Fig. 6. Note that in a practical implementation, $D_1$ delays will be employed to pipeline the FR, FI, FRI blocks and the WUDR and WUDI blocks. The $D_2$ delays can be employed to pipeline the adder in the recursive loop in the WUDR and WUDI blocks. Thus, all the operations in the PEA can be pipelined at a fine-grain level. Note that the summation in (4.3b) and (4.3d) can be realized by computing the product within the summation 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 due to the summation would be independent of $LA$. However, an overhead of $2N(LA-1)$ adders will result.

The input sample period $T_{\mathbf{PEA}}$ depends on the manner in which the algorithmic delays $D_1$ and $D_2$ have been retimed [21]. Assuming that retiming has been done in a uniform fashion (i.e., all stages have equal computation times), the lower bound on $T_{\mathbf{PEA}}$ is given by

$$T_{\mathbf{PEA}} \geq \frac{2T_m + (N + 2LA + 5)T_a}{(D_1 + D_2)}. \tag{4.4}$$

Thus, higher values of $D_1$ and $D_2$ imply higher speed ups. Practical maximum values of $D_1$ and $D_2$ are a function of the desired algorithmic performance (i.e., BER and/or SNR at the slicer). In Section VI, we will demonstrate how typical values of $D_1$ and $D_2$ can be obtained. For most communications applications, $D_1$ and $D_2$ delays would result in an increased delay from the transmitter to the receiver. For example, in the ATM-LAN chip-sets [38], pipelining resulted in an additional two-symbol period delay, which was a small fraction of the overall point-to-point delay. This would be true for most of the applications mentioned in the introduction, and hence, pipelining is an effective method to enhance the throughput.

### C. Convergence Analysis

Convergence analysis of the pipelined strength reduced architecture can be done in a fashion similar to that of the pipelined LMS [36]. For the sake of mathematical tractability, we have analyzed a special case of the proposed architecture, where $LA = 1$ and $D_1$ is a multiple of $D_2$, i.e., $D_1 = KD_2$. For details of the convergence analysis, the reader is referred to [36]. We will present only the final results in this subsection.

The following definitions are necessary before presenting the analytical expressions:

$$\lambda_{\text{rms}}^2 = \frac{\sum_{i=1}^{N} \lambda_i^2}{N} \tag{4.5a}$$

$$\lambda_{\text{av}} = \frac{\sum_{i=1}^{N} \lambda_i}{N} = \sigma^2 \tag{4.5b}$$

$$\alpha = \frac{\lambda_{\text{rms}}^2}{\lambda_{\text{av}}^2} \tag{4.5c}$$

$$v = \frac{E[|x(n)|^4]}{(E[|x(n)|^2])^2} \tag{4.5d}$$

$$P = N + v - 1 \tag{4.5e}$$

$$b = \mu\sigma^2 \tag{4.5f}$$

where $\lambda_i$'s $(i = 1, \cdots, N)$ are the eigenvalues of $\boldsymbol{R}$. Here, $\boldsymbol{R}$ is the autocorrelation matrix of the filter input.

As shown in [36], the upper bound on the step size $\mu$ for convergence in the mean-squared error (MSE) was found to be

$$0 \leq \mu \leq \frac{\alpha P + 2K - \sqrt{(\alpha P + 2K)^2 - 8K(K+1)}}{2K(K+1)\sigma^2}. \tag{4.6}$$

Note that (4.6) is identical to the corresponding expression for the pipelined LMS algorithm [36]. Hence, from (4.6), we see that the upper bound on the step size reduces as $K$ increases. This fact is confirmed via simulations in Section V. From (4.6), we also conclude that the upper bound can be kept constant with respect to $D_1$ if $K$ is kept constant.

The misadjustment is defined as follows:

$$\mathcal{M} = \frac{\epsilon(\infty) - \epsilon_{\min}}{\epsilon_{\min}}, \tag{4.7}$$

where $\epsilon(n)$ is $E(J(n))$, and $J(n)$ is the mean-squared error at time instance $n$. The notation $\epsilon_{\min}$ refers to the minimum mean-squared error, which would be obtained if the filter weight vector $\boldsymbol{W}(n)$ equaled the Wiener solution $\boldsymbol{W}_o$. The misadjustment for the proposed architecture was found to be

$$\mathcal{M} = \frac{\alpha N b}{2 - (\alpha P + 2K)b + K(K+1)b^2}. \tag{4.8}$$

In (4.8), the linear term in $b$ dominates the quadratic term. Hence, the misadjustment increases with $K$. In Section VI, we will see that the misadjustment does not change substantially as $K$ varies and, therefore, can be considered to be approximately constant.

### D. Power Reduction

As mentioned in Section II-B, pipelining along with power supply voltage reduction has been proposed [5] as a technique for reducing the power dissipation. In CMOS technology, scaling the power supply by a factor $K_V$ can be shown to reduce the speed of operation by the same factor, especially for small values of $K_V$. However, pipelined architectures can easily compensate for this loss in speed by the reduction of the critical path length. Hence, some of the increase in throughput due to pipelining can be traded off with power reduction. An implicit assumption in this approach to low-power operation is the requirement that the pipelining overhead be minimal. As was seen in this section, relaxed lookahead pipelining results in an overhead of $2N(LA-1)$ adders and $5D_1+2D_2$ latches (without retiming). This implies that the average switching capacitance in (2.4) would increase. Employing the fact that these additional adders are double precision, we get the power savings PS with respect to the cross-coupled architecture as in (4.9), shown at the bottom of the page, where $K_L$ is the ratio of the effective capacitance of a 1-b latch to that of a 1-b adder, and $K_V > 1$ is the factor by which the power supply is

$$\text{PS} = \frac{2NK_C(4K_V^2 - 3) + 2N(6K_V^2 - 2LA - 4) - (5D_1 + 2D_2)K_L - 3}{K_V^2(8NK_C + 12N)} \tag{4.9}$$

TABLE II
EXPERIMENT A

| | K | $\mu_{max}$ | | Misadjustment (%) $\mu$=0.001 | |
|---|---|---|---|---|---|
| | | Theory | Measured | Theory | Measured |
| $D_2 = 2$ | 1 | 0.037 | 0.032 | 2.47 | 2.16 |
| | 2 | 0.035 | 0.030 | 2.47 | 2.16 |
| | 4 | 0.032 | 0.028 | 2.48 | 2.16 |
| | 8 | 0.027 | 0.024 | 2.50 | 2.17 |
| | 16 | 0.022 | 0.019 | 2.54 | 2.21 |
| $D_2 = 4$ | 1 | 0.037 | 0.034 | 2.47 | 2.20 |
| | 2 | 0.035 | 0.032 | 2.47 | 2.19 |
| | 4 | 0.032 | 0.029 | 2.48 | 2.21 |
| | 8 | 0.027 | 0.024 | 2.50 | 2.23 |
| | 16 | 0.022 | 0.018 | 2.54 | 2.27 |

scaled. Employing typical values of $K_V = 5V/3.3V, K_C = 8, K_L = 1/3, N = 32, D_1 = 48, D_2 = 2$ and $LA = 3$ in (4.9), we obtain a total power savings of approximately 60% over the traditional cross-coupled architecture. Clearly, 21% of the power savings is obtained from the strength reduction transformation, whereas the rest (39%) is due to power-supply scaling. Note that this increased power savings is achieved in spite of the additional $2N(LA - 1)$ adders required due to relaxed lookahead pipelining.

Based on the transistor threshold voltages, it has been shown in [5] that values of $K_V = 3$ are possible with present CMOS technology. With this value of $K_V$, (4.9) predicts a power savings of 90%, which is a significant reduction. Thus, a judicious application of algebraic transformations (strength reduction), algorithm transformations (pipelining), and power-supply scaling can result in substantial power reduction.

## V. SIMULATION RESULTS

In this section, we present simulation results for verifying the performance of the proposed adaptive filter architecture. In Experiment A, we employ the proposed architecture in a system identification setup in order to verify (4.6) and (4.8). In Experiment B, we demonstrate the use of sum relaxation to improve the convergence speed as the level of pipelining increases.

### A. Experiment A

In this experiment, we employ a system identification setup in order to verify (4.6) and (4.8). Such a setup emulates those communications systems that employ echo cancellers or near-end cross talk (NEXT) cancellers. The system to be identified was a 50th-order complex FIR filter, and the proposed adaptive filter had $N = 24$ complex taps. The results were averaged over 50 independent trials and are shown in Table II for values of $D_2 = 2$ and $D_2 = 4$. The theoretical and measured values of $\mathcal{M}$ and $\mu_{\max}$ (the maximum value of $\mu$ for convergence) are tabulated in Table II.

From Table II, we notice that the theoretical and measured values of $\mathcal{M}$ and $\mu_{\max}$ match closely and have the same trend as the level of pipelining increases. As predicted by (4.6) and (4.8), we can observe a slight degradation in $\mathcal{M}$ and a decrease in $\mu_{\max}$ as $K$ is increased.

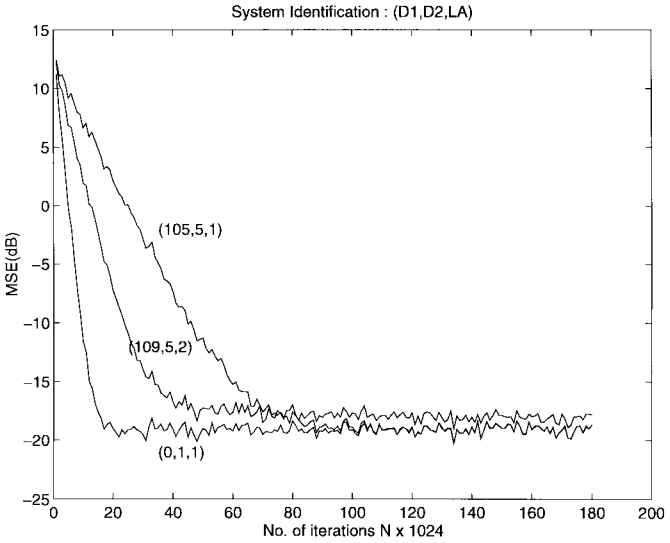



Fig. 7. Sum relaxation.

### B. Experiment B

Consider the system identification setup of Experiment A with $N = 32$. Further, assume that $T_m = 20$ ns and $T_a = 10.26$ ns, where $T_m$ and $T_a$ were defined in Section IV as the computation times of two-operand multiply and single-precision add operations, respectively. From (4.2), we obtain the clock period of the serial architecture (see Fig. 5) as $T_{\text{SEA}} = 440$ ns. If the application demands a sample period of 4 ns, then the serial architecture cannot meet this throughput rate. Hence, we need to employ the pipelined architecture in Fig. 6 with a speedup of 110. In particular, the double precision adders in the WUD blocks need to be pipelined by five stages, implying a value of $D_2 = 5$. Substituting $D_2 = 5, N = 32, T_m = 20, T_a = 10.26$ ns, $LA = 1$, and $T_{\text{PEA}} = 4$ into (4.4), we see that $D_1$ should be at least 105. In Fig. 7, we plot the MSE plots for the serial $((D_1, D_2, LA) = (0, 1, 1))$ and pipelined architectures $((D_1, D_2, LA) = (105, 5, 1))$. It is clear that the pipelined architecture has a slower convergence time. However, by employing sum relaxation with $LA = 2$, we obtain the third MSE plot (109, 5, 2) in Fig. 7, where the convergence speed is now significantly improved. Note that the sum relaxed architecture has a higher value of $D_1 = 109$. This is due to the fact that with $LA = 2$, the critical path time of the pipelined architecture is increased [see (4.4)] and therefore needs to be accounted for.

## VI. APPLICATION TO 51.84 Mb/s ATM-LAN

In this section, we will study the performance of the proposed low-power adaptive filter architecture in a high-speed digital communication system. In particular, we will employ the proposed architecture as an equalizer in a CAP-QAM modulation scheme for a data rate of 51.84 Mb/s over 100 m of unshielded twisted-pair (UTP3) wiring. As mentioned

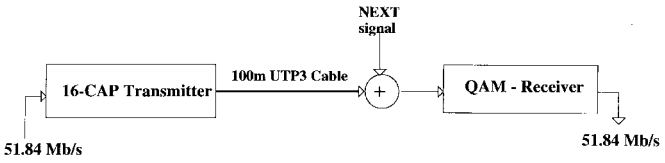



Fig. 8. Communication system block diagram for 51.84 Mb/s ATM-LAN.

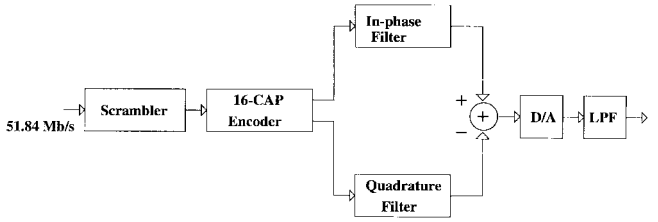
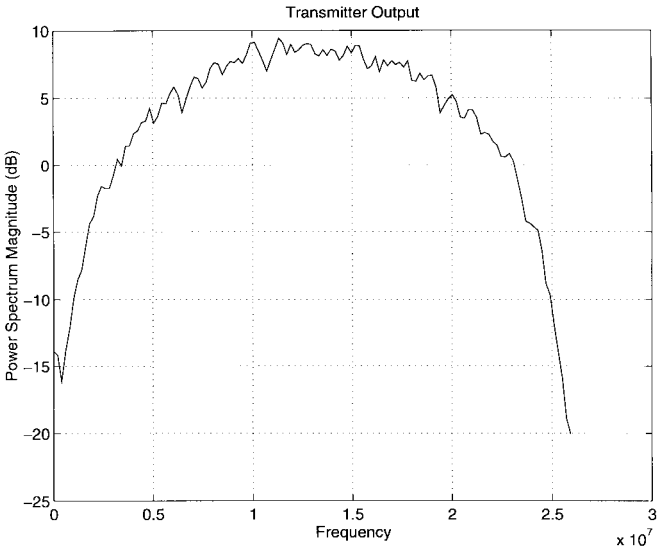
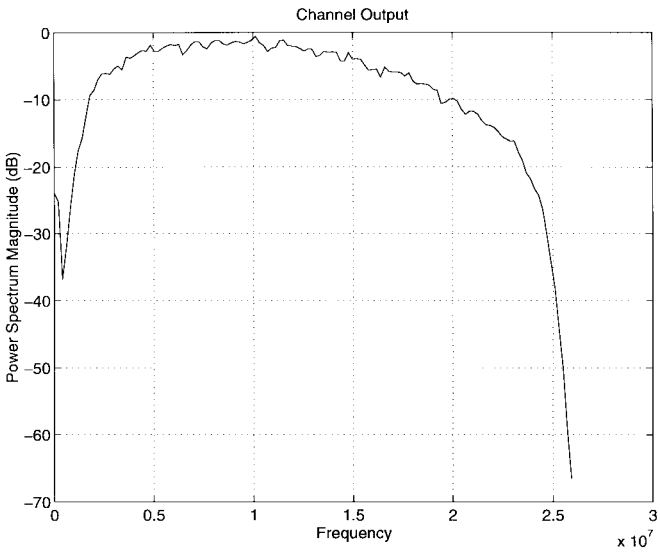



Fig. 9. CAP transmitter.




(a)




(b)

Fig. 10. Signal spectrum. (a) Transmitter output. (b) Channel output.

before, 16-CAP is currently the line code of choice [18] in this application.

While the standard does specify the line code to be 16-CAP, there is a lot of flexibility in deciding the transmitter and receiver structures. Hence, in this section, we have assumed a CAP transmitter and a QAM receiver. Our only reason for choosing QAM at the receiver is to be able to apply the proposed architecture to the QAM equalizer, which has been traditionally implemented as a cross-coupled structure. The overall communication link is shown in Fig. 8, where the input bit stream is accepted by a CAP transmitter and transformed into a format suitable for transmission over the channel. In addition to attenuation and dispersion, the received signal has NEXT superimposed on it. The NEXT impairment occurs due to electromagnetic coupling between the local transmitted signal and the received signal. This coupling is caused by the physical proximity of the wire pairs for transmission in the two directions. The models for NEXT can be obtained from [45].

Except for the essentials, we will skip most of the details regarding CAP [45] and QAM [14]. In Section VI-A, we briefly describe the CAP transmitter, and then, the QAM receiver is described in Section VI-B. Finally, the simulation results are presented in Section VI-C.

### *A. The CAP Transmitter*

The block diagram of a digital CAP transmitter is shown in Fig. 9. The bit stream to be transmitted is first passed through a scrambler. The scrambled bits are then fed into an encoder, which maps blocks of $m$ bits onto one of $k = 2^m$ different complex symbols $a(n) = a_r(n) + ja_i(n)$ for a $k$-CAP line code. In this study, we have employed $k = 16$. The symbols $a_r(n)$ and $a_i(n)$ are processed by digital shaping filters. This requires that the shaping filters be operated at a sampling frequency $f_s$, which is at least twice the maximum frequency component of the transmit spectrum. The outputs of the filters are subtracted, and the result is passed through a digital-to-analog (D/A) converter, which is followed by an interpolating lowpass filter (LPF). It can be seen that most of the signal processing at the transmitter (including transmit shaping) is done in the digital domain, which permits a robust VLSI implementation.

The signal at the output of the CAP transmitter (see Fig. 9) can be written as

$$s(t) = \sum_{n=-\infty}^{\infty} [a_r(n)p(t - nT) - a_i(n)\tilde{p}(t - nT)] \quad (6.1)$$

where $T$ is the symbol period, $a_r(n)$ and $a_i(n)$ are discrete multilevel symbols, which are sent in symbol period $nT$, and $p(t)$ and $\tilde{p}(t)$ are the impulse responses of in-phase and quadrature passband shaping filters, respectively. The passband pulses $p(t)$ and $\tilde{p}(t)$ in (6.1) are defined as

$$p(t) \triangleq g(t)\cos(2\pi f_c t) \quad \tilde{p}(t) \triangleq g(t)\sin(2\pi f_c t) \quad (6.2)$$

where $g(t)$ is a baseband pulse, and $f_c$ is a frequency that is larger than the largest frequency component in $g(t)$. The two impulse responses in (6.2) form a Hilbert pair, i.e., their

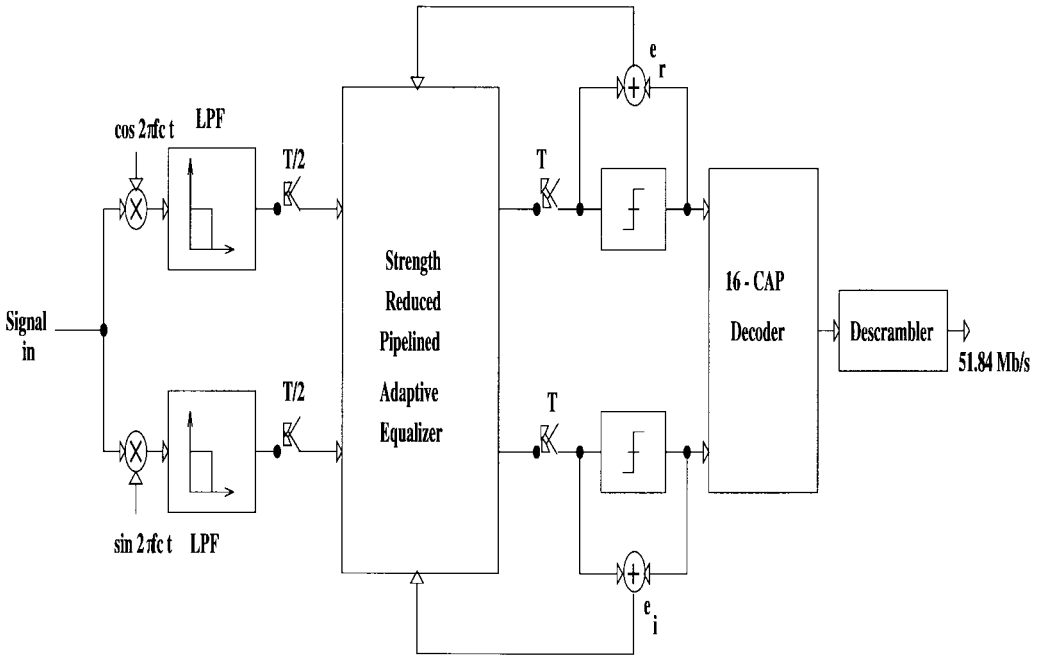



Fig. 11. QAM receiver.

Fourier transforms have the same amplitude characteristics, whereas their phase characteristics differ by $90°$. Typically, the baseband pulse $g(t)$ is a square-root raised cosine [14] pulse.

The output spectrum is broadband with a bandwidth of 25.92 MHz [see Fig. 10(a)]. While larger bandwidths are possible, the FCC Class B requirements restrict the signal energy to below 30 MHz. The bit rate of 51.84 Mb/s and 16-CAP signal constellation imply a symbol rate of 12.96 Mbaud. Hence, the chosen transmit spectrum has 100% excess bandwidth, as shown in Fig. 10(a). The received spectrum at the output of the channel [see Fig. 10(b)] indicates the extent of propagation loss.

### *B. The QAM Receiver*

The QAM receiver in Fig. 11 first demodulates the received signal (which is sampled at 51.84 Msamples/s) such that the signal at the output of the lowpass filters (LPF) has energy from DC to 12.96 MHz. This allows us to downsample the LPF output by a factor of two. The resulting complex signal can then be filtered via the traditional cross-coupled architecture (Figs. 1 and 2) or the proposed architecture (Fig. 5) operating at 25.92 Msamples/s. The equalizer outputs are sampled at the symbol rate of 12.96 MHz, which are then sliced to generate the detected symbols. The error across the slicer is employed to adapt the equalizer coefficients once every symbol period. The detected symbols are also decoded to generate the received bit stream.

### *C. Simulation Results for 51.84 Mb/s ATM-LAN*

Achieving the specified BER of $10^{-10}$ with 16-CAP translates to a SNR at the slicer $(\mathrm{SNR}_o)$ of 23.25 dB. The values of the step size $\mu$ employed in the simulations were deliberately made powers of two so that the hardware implementation requires only shift-right operations. In particular, we employed gear shifting whereby the value of $\mu$ was halved after the first 120 000 symbols and again after 240 000 symbols. Furthermore, the receive equalizer was chosen to have a span of 32 symbol periods to obtain a noise margin of about 2 dB. For simplicity, we have assumed that a training sequence is available for equalizer training.

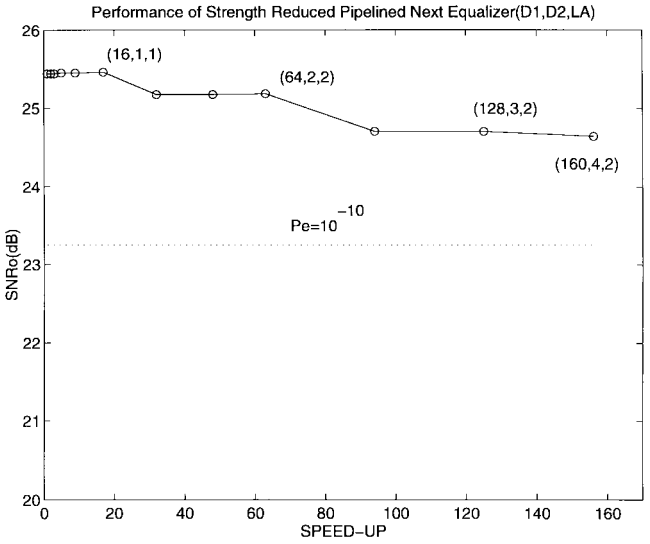



Fig. 12. $\mathrm{SNR}_o$ versus speedup.

Based on the convergence analysis results of Section IV-C, we can conclude that $\mathrm{SNR}_o$ will degrade as the level of pipelining is increased. In Fig. 12, we plot the $\mathrm{SNR}_o$ with respect to the speedup, where the speedup is defined as the ratio of $T_{\mathrm{SEA}}$ [see (4.2)] to $T_{\mathrm{PEA}}$ [see (4.4)]. It is clear from Fig. 12 that the SNR degrades by less than 0.8 dB for speedups

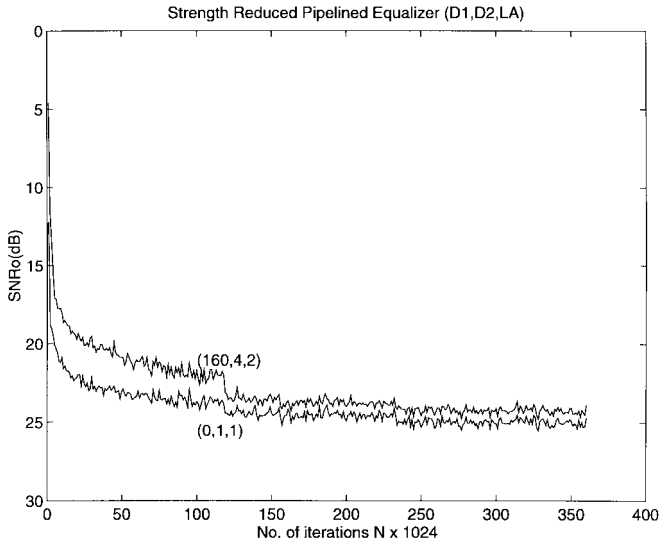



Fig. 13. Convergence curves for the error across the slicer.

of up to 156. Speedups of up to 50 or 60 may suffice for most applications. In that case, the loss in performance is less than 0.27 dB. Thus, the proposed architecture allows substantial speedups with negligible performance loss.

For all cases, $SNR_o$ increases to 20 dB within 4 ms (approx 50 000 symbols). This is indicated in Fig. 13, where the convergence plot for the serial and pipelined (speedup of 156) cases are shown. The fact that the indicated values of SNR was achieved by running the simulation for 360 000 symbols, which corresponds to approximately 28 ms, is also worth noting. For the application at hand, a convergence speed of a few hundred milliseconds is deemed acceptable.

Thus, we conclude that the proposed architecture is a viable alternative for QAM-based receivers, especially in an ATM-LAN environment.

## VII. Conclusions

Application of strength reduction transformation [4], [5] at the algorithmic level (as opposed to the architectural level) has resulted in a low-power complex adaptive filter architecture. Power and area savings of approximately 21% were shown to be achievable. Relaxed lookahead [37] pipelined architectures were then developed for achieving high-speed operation. An additional 39% power savings was achieved by scaling down the power supply.

It must be mentioned that the low-power architecture presented in this paper is applicable to any communication system, which employs a 2-D signal constellation. While we have demonstrated the application of the proposed architecture for 51.84 Mb/s ATM-LAN, numerous other applications exist. Our current research is being directed toward the design of low-power NEXT cancellers and adaptive DFE's for higher speed digital subscriber loops (such as 100–155 Mb/s). Future extensions of this work include the study of finite-precision effects of the proposed architecture and, eventually, an integrated circuit implementation. Development of optimal folding strategies is also a future goal in order to tradeoff area with speed so that power, area, and speed optimal systems can be implemented.

## References

[1] M. Abdulrahman, A. U. H. Sheikh, and D. D. Falconer, "Decision feedback equalization for CDMA in indoor wireless communications," *IEEE J. Select. Areas Commun.*, vol. 12, pp. 698–706, May 1994.

[2] M. Alidina, J. Monterio, S. Devdas, A. Ghosh, and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low-power," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 426–436, Dec. 1994.

[3] W. C. Athas, L. J. Svensson, J. G. Koller, N. Tzartzanis, and E. Y.-C. Chou, "Low-power digital systems based on adiabatic switching principles," *IEEE J. Solid-State Circuits*, vol. 29, pp. 398–407, Dec. 1994.

[4] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*. Reading, MA: Addison-Wesley, 1987.

[5] A. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proc. IEEE*, vol. 83, pp. 498–523, Apr. 1995.

[6] A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing power using transformations," *IEEE Trans. Comput.-Aided Design*, Jan. 1995.

[7] G. Cherubini, S. Olcer, and G. Ungerboeck, "A quarternary partial-response class-IV system for 125 Mb/s data-transmission over unshielded twisted-pair," in *Proc. IEEE Int. Conf. Commun.*, vol. 3, pp. 1814–1819, May 1993.

[8] P. S. Chow, J. C. Tu, and J. M. Cioffi, "Performance evaluation of a multichannel transceiver system for ADSL and VHDSL services," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 909–919, Aug. 1991.

[9] J. M. Cioffi *et al.*, "Adaptive equalization in magnetic disk storage channels," *IEEE Commun. Mag.*, pp. 14–29, Feb. 1990.

[10] B. Daneshrad, and H. Samueli, "A 1.6 Mbps digital-QAM system for DSL transmission," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1600–1610, Dec. 1995.

[11] B. Davari, R. H. Dennard, and G. G. Shahidi, "CMOS scaling for high-performance and low-power—The next ten years," in *Proc. IEEE*, vol. 83, pp. 595–606, Apr. 1995.

[12] A. G. Dickinson and J. S. Denker, "Adiabatic dynamic logic," in *Proc. Custom Integrated Circuits Conf.*, 1994.

[13] A. Gersho, "Reduced complexity implementation of passband adaptive equalizers," *IEEE J. Select. Areas Commun.*, vol. JSAC-2, pp. 778–779, Sept. 1984.

[14] R. D. Gitlin, J. F. Hayes, and S. B. Weinstein, *Data Communications Principles*. New York: Plenum, 1992.

[15] R. Hartley and P. Corbett, "Digit-serial processing techniques," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 707–719, 1990.

[16] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-power digital design," in *Proc. 1994 IEEE Symp. Low Power Electron.*, San Diego, CA, Oct. 10–12, pp. 8–11.

[17] G. H. Im and J. J. Werner, "Bandwidth-efficient digital transmission up to 155 Mb/s over unshielded twisted-pair wiring," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1643–1655, Dec. 1995.

[18] G. H. Im *et al.*, "51.84 Mb/s 16-CAP ATM LAN standard," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 6120–632, May 1995.

[19] G. H. Im and L. J. Smithwick, "Performance of a local distribution system for interactive multimedia TV," *IEEE Globecom '95*, Nov. 1995

[20] J. W. Lechleider, "High bit rate digital subscriber lines: A review of HDSL progress," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 769–784, Aug. 1991.

[21] C. Leiserson and J. Saxe, "Optimizing synchronous systems," *J. VLSI Comput. Syst.*, vol. 1, pp. 41–67, 1983.

[22] D. W. Lin, C.-T. Chen, and T. R. Hsing, "Video on phone lines: Technology and applications," *Proc. IEEE*, vol. 83, pp. 175–193, Feb. 1995.

[23] 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.

[24] ——, "Corrections to "The LMS algorithm with delayed coefficient adaptation," *IEEE Trans. Signal Processing*, vol. 40, pp. 230–232, Jan. 1992.

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

[26] F. Lu and H. Samueli, "A 60 Mbd, 480-Mb/s, 256-QAM decision-feedback equalizer in 1.2 $\mu$m CMOS," *IEEE J. Solid-State Circuits*, vol. 28, pp. 330–338, Mar. 1993.

[27] J. Monteiro, S. Devdas, and A. Ghosh, "Retiming sequential circuits for low power," *IEEE Int. Conf. Comput.-Aided Design*, pp. 398–402.
[28] Y. Nakagome, K. Itoh, M. Isoda, K. Takeuchi, and M. Aoki, "Sub-1-V swing internal bus architecture for future low-power ULSI's," *IEEE J. Solid-State Circuits*, vol. 28, pp. 414–419, Apr. 1993.
[29] A. N. Netravalli and A. Lippman, "Digital television: A perspective," *Proc. IEEE*, vol. 83, pp. 834–842, June 1995.
[30] A. N. Netravalli and B. G. Haskell, *Digital Pictures: Representation and Compression*. New York: Plenum, 1988.
[31] K. K. Parhi, "Algorithm transformation techniques for concurrent processors," *Proc. IEEE*, vol. 77, pp. 1879–1895, Dec. 1989.
[32] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters—Part I: Pipelining using scattered lookahead and decomposition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1099–1117, July 1989.
[33] ——, "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.
[34] K. K. Parhi, "A systematic approach for the design of digit-serial signal processing architectures," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 358–375, Apr. 1991.
[35] K. K. Parhi, C. Y. Wang, and A. P. Brown, "Synthesis of control circuits in folded pipelined DSP architectures," *IEEE J. Solid-State Circuits*, vol. 27, pp. 29–43, Jan. 1992.
[36] M. Potkonjak and J. Rabaey, "Fast implementation of recursive programs using transformations," in *Proc. ICASSP*, San Francisco, CA, Mar. 1992, pp. V-569–V-572.
[37] N. R. Shanbhag and K. K. Parhi, *Pipelined Adaptive Digital Filters*. Boston, MA: Kluwer, 1994.
[38] N. R. Shanbhag, G. A. Wilson, R. Shariatdoust, J. Kumar, R. Townsend, "VLSI complexity of the 16-CAP transceiver," *ATM Forum PHY Subworking Group Contribution*, ATM-Forum/93-994, Nov. 16–19, 1993, Stockholm, Sweden.
[39] N. R. Shanbhag and K. K. Parhi, "Relaxed lookahead pipelined LMS adaptive filters and their application to ADPCM coder," *IEEE Trans. Circuits Syst.*, vol. 40, pp. 753–766, Dec. 1993.
[40] ——, "A pipelined adaptive lattice filter architecture," *IEEE Trans. Signal Processing*, vol. 41, pp. 1925–1939, May 1993 .
[41] ——, "Pipelined adaptive DFE architectures using relaxed lookahead," *IEEE Trans. Signal Processing*, vol. 43, pp. 1368–1385, June 1995.
[42] ——, "VLSI implementation of a 100 MHz pipelined ADPCM codec chip," in *Proc. IEEE VLSI Signal Processing Workshop*, Veldhoven, The Netherlands, Oct. 1993, pp. 114–122.
[43] A. Shen, A. Ghosh, S. Devdas, and K. Keutzer, "On average power dissipation and random pattern testability of CMOS combinational logic networks," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, 1992, pp. 402–407.
[44] E. A. Vittoz, "Future of analog in the VLSI environment," *Proc. ISCAS*, 1990, pp. 1372–1375.
[45] J. J. Werner, "The HDSL environment," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 785–800, Aug. 1991.
[46] B. Widrow *et al.*, "Stationary and nonstationary learning characteristics of the LMS adaptive filter," *Proc. IEEE*, vol. 64, pp. 1151–1162, 1976.

**Naresh R. Shanbhag** (M'93) received the B.Tech. degree from the Indian Institute of Technology, New Delhi, in 1988 and the Ph.D. degree from University of Minnesota, Minneapolis, in 1993, all in electrical engineering.

From July 1993 to August 1995, he worked at AT&T Bell Laboratories, Murray Hill, NJ, in the Wide-Area Networks Group, where he was responsible of development of VLSI algorithms, architectures and implementation for high-speed data communications applications. In particular, he was the lead chip architect for AT&T's 51.84 Mb/s transceiver chips over twisted-pair wiring for the asynchronous transfer mode (ATM)-LAN and the interactive multimedia television (IMTV) transitter-receiver chipset. In August 1995, he joined the Coordinated Science Laboratory and the Electrical and Computer Engineering Department, the University of Illinois, Urbana-Champaign, as an assistant professor. His research interests (see URL http://uivlsi.csl.uiuc.edu/~shanbhag) are in the area of VLSI architectures and algorithms for signal processing and communications. This includes the design of high-speed and/or low-power algorithms for speech and video processing, adaptive filtering, and high-bit rate digital communications systems. In addition, he is also interested in efficient VLSI implementation methodologies for these applications.

Dr. Shanbhag received the NSF CAREER Award in 1996, the 1994 Darlington best paper award from the IEEE Circuits and Systems Society and is the co-author of the research monograph *Pipelined Adaptive Digital Filters* (Boston, MA: Kluwer, 1994).

**Manish Goel** was born on May 15, 1973 in India. He received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, New Delhi, in 1994. He is currently pursuing the Ph.D. degree in electrical and computer engineering at the University of Illinois, Urbana-Champaign.

His research interests are in the area of VLSI for signal processing and communication applications, which include low-power adaptive filter design, algorithmic transformations, and implementation methodologies. In addition, he is also interested in low-complexity transceiver design for modern digital communication systems such as high-speed digital subscriber loop and wireless CDMA systems.

Mr. Goel received the 1993 Motorola Undergraduate Project Award and the 1994 Best Undergraduate Project Award during his undergraduate studies at the Indian Institute of Technology.