Real-Time Algorithms and Architectures for Multiuser Channel Estimation and Detection in Wireless Base-Station Receivers

Sridhar Rajagopal, *Student Member, IEEE*, Srikrishna Bhashyam, *Member, IEEE*, Joseph R. Cavallaro, *Member, IEEE*, and Behnaam Aazhang, *Fellow, IEEE*

***Abstract*—This paper presents algorithms and architecture designs that can meet real-time requirements of multiuser channel estimation and detection in future code-division multiple-access-based wireless base-station receivers. Sophisticated algorithms proposed to implement multiuser channel estimation and detection make their real-time implementation difficult on current digital signal processor-based receivers. A maximum-likelihood based multiuser channel estimation scheme requiring matrix inversions is redesigned from an implementation perspective for a reduced complexity, iterative scheme with a simple fixed-point very large scale integration (VLSI) architecture. A reduced-complexity, bit-streaming multiuser detection algorithm that avoids the need for multishot detection is also developed for a simple, pipelined VLSI architecture. Thus, we develop real-time solutions for multiuser channel estimation and detection for third-generation wireless systems by: 1) designing the algorithms from a fixed-point implementation perspective, without significant loss in error rate performance; 2) task partitioning; and 3) designing bit-streaming fixed-point VLSI architectures that explore pipelining, parallelism, and bit-level computations to achieve real-time with minimum area overhead.**

***Index Terms*—Digital signal processor, multiuser channel estimation, multiuser detection, real-time implementation, very large scale integration, wideband code-division multiple-access.**

I. Introduction

THIRD-GENERATION (3G) wireless cellular systems [1]–[3], are being designed to support extremely high data rates (in Mb/s) and quality-of-service (QoS) guarantees that are required for multimedia communication. Wideband code-division multiple-access (W-CDMA) [2] has been chosen as the multiple-access protocol to support these features. The existing narrowband CDMA IS-95 standard supports only voice and low-data rates up to 9.6 kb/s and uses single-user algorithms at the base-station receiver that ignore multiple access interference (MAI) between different users. To achieve improved performance at these high data rates, highly sophisticated and complex multiuser algorithms for channel estimation and detection need to be implemented. These algorithms combat MAI by jointly processing the signals of all users at the base-station receiver. The multiuser algorithms involving matrix multiplications and inversions [4], [5] require block-based computations and floating point accuracy and significantly increase the implementation complexity of the receiver [6]. A direct implementation of these multiuser algorithms using current generation digital signal processor (DSP)-based base-station receivers fails to meet third-generation real-time requirements [7]. Therefore, only single user algorithms for channel estimation and detection [8], [9] have been implemented in all current practical CDMA systems, such as IS-95.

Implementations for multiuser detection for the base station have been studied in [6] and [10] while low power versions targeted at mobile handsets have been studied in [11] and [12]. However, these detector implementations either assume perfect channel estimation or assume single user estimation using sliding-correlator type structures [8]. The detector implementations also assume that channel estimation is done in real-time and the data rates are considered to be dependent only on the detector. However, many advanced multiuser channel estimation schemes have high computational complexity, even more than that for multiuser detection, due to matrix inversions involved and cannot be performed in real-time. Also, algorithms for estimation and detection are block-computation based due to the need for repeated inversion updates for estimation [13] and multishot detection [5], [14], which make their real-time implementation more difficult. Matrix-inversion free schemes such as those based on conjugate gradient descent and recursive least squares (RLS) [15]–[17] exist in the literature. We have evaluated the applicability of such schemes for multiuser channel estimation and presented one such scheme with low computational complexity and suitable for implementation. Jointly performing multiuser channel estimation and detection is shown to have lower computational complexity and better error rate performance than performing multiuser estimation and detection separately [13]. Hence, we shall consider this joint algorithm for multiuser channel estimation and detection for redesign from a very large scale integration (VLSI) architecture perspective. Similar work on a joint channel estimation and detection scheme for time division multiple access (TDMA) systems with a systolic implementation for Kalman filtering is

Manuscript received August 2, 2000; revised May 3, 2001, and July 28, 2001; accepted July 28, 2001. The editor coordinating the review of this paper and approving it for publication is C. Nicol. This work was supported in part by Nokia, Texas Instruments Incorporated , the Texas Advanced Technology Program under Grant 1999-003604-080, and by National Science Foundation (NSF) under Grant ANI-9979465.

S. Rajagopal, J. R. Cavallaro, and B. Aazhang are with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: sridhar@rice.edu; cavallar@rice.edu; aaz@rice.edu).

S. Bhashyam was with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA. He is now with Qualcomm Inc., Campbell, CA 95008 USA (e-mail: skrishna@rice.edu).

Publisher Item Identifier 10.1109/TWC.2002.800545.

presented in [18]. They have also studied word-length effects and provided comparisons with least mean square (LMS) and RLS schemes.

In this paper, we present efficient algorithms for multiuser channel estimation and detection, designed from an implementation perspective and their mapping to real-time VLSI architectures. We redesign a multiuser channel estimation algorithm [13], based on the maximum-likelihood (ML) principle and present an iterative scheme, which is computationally effective, suitable for a fixed point implementation and is equivalent to matrix inversion in terms of error rate performance. A new bit-streaming multiuser detection scheme based on parallel interference cancellation is presented that avoids the need for multishot detection [5], [14], [19] for a simple bit-streaming pipelined VLSI architecture. Fixed-point implementations of the redesigned algorithms are presented. First, we determine the maximum data rate achievable with no area constraints. Then, we obtain the data rate achieved by an area-constrained architecture. Finally, we present area-time tradeoffs for real-time VLSI architectures to achieve the targeted data rates with minimum area overhead. Thus, the main contribution of this paper is to show real-time performance for multiuser algorithms by: 1) designing the algorithms from a fixed-point architecture perspective, without significant loss in error rate performance; 2) task partitioning; and 3) designing bit-streaming fixed point VLSI architectures to exploit available pipelining, parallelism and bit-level computations.

II. Multiuser Channel Estimation and Detection

A. Real-Time Requirements

Data transmission in 3G wireless systems such as third-generation partnership project (3GPP) or universal mobile telecommunications systems (UMTSs) is possible at varying rates such as from 32 kb/s to 2 Mb/s depending on the spreading factor (N) which varies from 256 (for vehicular traffic) to 4 (for indoor environments), respectively (for example, see [3]). The standards assume a chip rate of 4.096 Mcps and quadrature phase-shift keying (QPSK) modulation (2 bits/symbol). We have assumed binary phase-shift keying (BPSK) modulation (1 bit/symbol) in our work for simplicity. Hence, we target data rates in the range of 16 kb/s to 1 Mb/s. However, our proposed algorithms as well as our work on fixed-point analysis, pipelining, and parallelism can be extended to higher modulation schemes as well. We propose different architectures which explore area-time trade-offs in order to achieve these data rates. We seek to design architectures that meet real-time requirements to within an order-of-magnitude. Specifically, we target architecture designs for different spreading gains ($N = K = 4$, 16, 32, 128, 256) to achieve data rates of 16 kb/s, 64 kb/s, 128 kb/s, 256 kb/s, and 1 Mb/s, respectively. Note that the reference to 3G systems is solely as an example to illustrate important system features such as the varying data rates which we seek to target and the use of training sequences for channel estimation.

B. Received Signal Model

We assume BPSK modulation and use direct sequence spread spectrum signaling, where each active mobile unit possesses a unique signature sequence (short repetitive spreading code) to modulate the data bits (± 1). The base station receives a summation of the signals of all the active users after they travel through different paths in the channel. The multipath is caused due to reflections of the transmitted signal that arrive at the receiver along with the line-of-sight component. These channel paths induce different delays, attenuations and phase-shifts to the signals and the mobility of the users causes fading in the channel. Moreover, the signals from different users interfere with each other in addition to the additive white Gaussian noise (AWGN) present in the channel. Multiuser channel estimation refers to the joint estimation of these unknown parameters for all users to mitigate these undesirable effects and accurately detect the received bits of different users. Multiuser detection refers to the detection of the received bits for all users jointly by canceling the interference between the different users. The performance of multiuser detection depends greatly on the accuracy of the channel estimates. The model for the received signal at the output of the multipath channel [13] can be expressed as

$$\mathbf{r}_i = \mathbf{A}\underline{\mathbf{d}}_i + \mathbf{n}_i \tag{1}$$

where $\mathbf{r}_i \in \mathbb{C}^N$ is the received signal vector after chip-matched filtering [5], [20], $\mathbf{A} \in \mathbb{C}^{N \times 2K}$ is the effective spreading code matrix, containing information about the spreading codes (of length N), attenuation and delays from the various paths, $\underline{\mathbf{d}}_i \in \{-1, +1\}^{2K} = [d_{1,i-1}, d_{1,i}, \ldots, d_{K,i-1}, d_{K,i}]^\top$ are the bits of K users to be detected, $\mathbf{n}_i$ is AWGN and i is the time index. The size of the data bits of the users $\underline{\mathbf{d}}_i$ is $2K$ as we assume that all paths of all users are coarse synchronized to within one symbol period from the arbitrary timing reference. Hence, only two symbols of each user will overlap in each observation window. This model can be easily extended to include more general situations for the delays [21], without affecting the derivation of the channel estimation algorithms. The estimate of the matrix $\mathbf{A}$ contains the effective spreading code of all active users and the channel effects and is used for accurately detecting the received data bits of different users. We will call this estimate of the effective spreading code matrix, $\hat{\mathbf{A}}$, our channel estimate as it contains the channel information directly in the form needed for detection. This approach [13] is chosen as it provides: 1) a single framework for both channel estimation and detection and 2) both computational and performance gains. Most other multiuser channel estimation techniques try to estimate the individual channel attenuations and delays instead of the effective spreading code.

C. Multiuser Channel Estimation and Tracking

The block diagram of the base-station receiver is shown in Fig. 1. The multiuser channel estimation algorithm proposed in [13] is redesigned for implementation in this paper. The ML channel estimate is obtained using the knowledge of training symbols. Most proposed 3G systems [3] allow for the use of training symbols. When training symbols are not available the channel can be updated, to track time-variations, using decision feedback from the detector. This approach provides a simple linear channel estimation technique and its properties are similar to those associated with the ML approach discussed in [22].

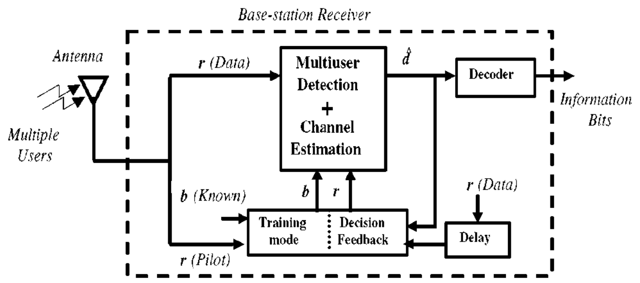


Fig. 1. Simplified view of the base station receiver. This figure shows the multiuser channel estimation and detection blocks in the receiver. A training sequence (pilot) is used for channel estimation and decision feedback is used to update the estimates in the absence of a pilot.

A basic summary of the algorithm and its computational aspects are presented here. More details can be found in [13]. Consider L observations of the received vector $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_L$ corresponding to the known training bit vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_L$. Given the knowledge of the training bits, the discretized received vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_L$ are independent and each of them is Gaussian distributed. Thus, the likelihood function becomes

$$\mathrm{p}(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_L | \mathbf{A}, \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_L) = \frac{1}{\pi^{NL}} \exp\left\{ -\sum_{i=1}^{L} (\mathbf{r}_i - \mathbf{A}\mathbf{b}_i)^H (\mathbf{r}_i - \mathbf{A}\mathbf{b}_i) \right\}.$$

After eliminating terms that do not affect the maximization, the log likelihood function becomes

$$\left\{ \sum_{i=1}^{L} (\mathbf{r}_i - \mathbf{A}\mathbf{b}_i)^H (\mathbf{r}_i - \mathbf{A}\mathbf{b}_i) \right\}. \tag{2}$$

The estimate $\hat{\mathbf{A}}$, that maximizes the log likelihood, satisfies the following:

$$\mathbf{R}_{bb}\hat{\mathbf{A}} = \mathbf{R}_{br}. \tag{3}$$

The matrices $\mathbf{R}_{bb}$ and $\mathbf{R}_{br}$ are defined as follows:

$$\mathbf{R}_{bb} = \sum_{i=1}^{L} \mathbf{b}_i \mathbf{b}_i^H \qquad \mathbf{R}_{br} = \sum_{i=1}^{L} \mathbf{b}_i \mathbf{r}_i^H. \tag{4}$$

Thus, the computations required to obtain the estimate $\hat{\mathbf{A}}$ are: 1) the computation of the correlation matrices $\mathbf{R}_{bb}$ and $\mathbf{R}_{br}$ and 2) the computation required to solve the linear equation in (3).

D. Multiuser Detection

Multiuser detection cancels the interference from other users to improve the error rate performance, compared with the traditional single user detection using only a matched filter [20]. We implement multistage detection [14], based on the principle of Parallel Interference Cancellation. This scheme cancels the interference from different users, iteratively in stages and is shown to have computational complexity quadratic with the number of users. It is also possible to feed the channel estimate matrix directly into the multistage detector instead of explicitly extracting the parameters.

The channel matrix $\mathbf{A}$ is rearranged into its odd and even columns $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{C}^{N\times K}$ which corresponds to the successive bit vectors $\mathbf{d}_{i-1}$ and $\mathbf{d}_i$; where $\mathbf{d}_i \in \{-1, +1\}^K = [d_{1,i}, \ldots, d_{K,i}]^\top$ are the bits of the K users at time instant i that need to be detected. In vector form, the received signal is

$$\mathbf{r}_i = [\mathbf{A}_0 \mathbf{A}_1] \begin{bmatrix} \mathbf{d_{i-1}} \\ \mathbf{d_i} \end{bmatrix} + \mathbf{n}_i. \tag{5}$$

1) Matched Filter (MF) Detector: The bits, $\mathbf{d}_i$, of the K users to be detected lie between the received signal $\mathbf{r}_i$ and $\mathbf{r}_{i-1}$ boundaries. The MF detector [5], [20] does a correlation of the input bits with the received bits. Hence, the MF detector can be represented as

$$\hat{\mathbf{d}}_i = sign\left(\Re\left[\mathbf{A}_1^H \mathbf{r}_{i-1} + \mathbf{A}_0^H \mathbf{r}_i\right]\right). \tag{6}$$

The multistage detector uses the MF to get an initial estimate of the bits and then iteratively subtracts the interference from all other users.

2) Multistage Detector: The multistage detector [14], [23] performs parallel interference cancellation iteratively in stages. The desired user's bits suffers from interference caused by the past or future overlapping symbols of different asynchronous users. Detecting a block of bits simultaneously (multishot detection) can give performance gains [5]. However, in order to do multishot detection, the above model should be extended to include multiple bits. Let us consider D bits at a time ($i = 1, 2, \ldots, D$). So, we form the multishot received vector $\mathbf{r} \in \mathbb{R}^{ND}$ by concatenating D vectors ($\mathbf{r}_i, i = 1, 2, \ldots, D$)

$$\mathbf{r} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & 0 & 0 \\ 0 & \mathbf{A}_0 & \mathbf{A}_1 & 0 \\ \vdots & \ddots & \ddots & \mathbf{A}_1 \\ 0 & 0 & \mathbf{A}_0 & \mathbf{A}_0 \end{bmatrix} \begin{bmatrix} \mathbf{d_1} \\ \mathbf{d_2} \\ \vdots \\ \mathbf{d_D} \end{bmatrix} + \mathbf{n}_i. \tag{7}$$

Let $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{ND\times KD}$ represent the new multishot channel matrix. The initial soft decision outputs $\mathbf{y}^{(0)} \in \mathbb{R}^{KD}$ and hard decision outputs $\hat{\mathbf{d}}^{(0)} \in \mathbb{R}^{KD}$ of the detector are obtained from a MF using the channel estimates as

$$\mathbf{y}^{(0)} = \Re\left[\boldsymbol{\mathcal{A}}^H \mathbf{r}\right] \tag{8}$$

$$\hat{\mathbf{d}}^{(0)} = \mathrm{sign}(\mathbf{y}^{(0)}) \tag{9}$$

$$\mathbf{y}^{(l)} = \mathbf{y}^{(0)} - \Re\left[\boldsymbol{\mathcal{A}}^H\boldsymbol{\mathcal{A}} - \mathrm{diag}\left(\boldsymbol{\mathcal{A}}^{\mathrm{H}}\boldsymbol{\mathcal{A}}\right)\right] \hat{\mathbf{d}}^{(l-1)} \tag{10}$$

$$\hat{\mathbf{d}}^{(l)} = \mathrm{sign}(\mathbf{y}^{(l)}) \tag{11}$$

where $\mathbf{y}^{(l)}$ and $\hat{\mathbf{d}}^{(l)}$ are the soft and hard decisions, respectively, after each stage of the multistage detector. These computations are iterated for $l = 1, 2, \ldots, M$ where M is the maximum number of iterations chosen for desired performance. The structure of $\boldsymbol{\mathcal{A}}^H\boldsymbol{\mathcal{A}} \in \mathbb{C}^{KD\times KD}$ is as shown

$$\begin{bmatrix} \mathbf{A}_0^H\mathbf{A}_0 & \mathbf{A}_0^H\mathbf{A}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_1^H\mathbf{A}_0 & \mathbf{A}_0^H\mathbf{A}_0 + \mathbf{A}_1^H\mathbf{A}_1 & \mathbf{A}_0^H\mathbf{A}_1 & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_1^H\mathbf{A}_0 & \mathbf{A}_0^H\mathbf{A}_0 + \mathbf{A}_1^H\mathbf{A}_1 \end{bmatrix}. \tag{12}$$

The block tri-diagonal nature of the matrix arises due to the assumption that the asynchronous delays of the different users are coarse synchronized within one symbol duration [13], [21]. If the channel is static, the matrix is also block-Toeplitz. We exploit the block tri-diagonal nature of the matrix later, for reducing the complexity and pipelining the algorithm effectively. The hard decisions, $\hat{\mathbf{d}}$, made at the end of the final stage, are fed back to the estimation block in the decision feedback mode for tracking in the absence of the pilot signal. Detectors using differencing methods have been proposed [23] to take advantage of the convergence behavior of the iterations. If there is no sign change of the detected bit in succeeding stages, the difference is zero and this fact is used to reduce the computations. However, the advantage is useful only in case of sequential execution of the detection loops, as in DSPs. Hence, we do not implement the differencing scheme in our design for a VLSI architecture.

III. Real-Time Algorithms for Multiuser Channel Estimation and Detection

A. Iterative Scheme for Channel Estimation

A direct computation of the ML based channel estimate $\hat{\mathbf{A}}$ involves the computation of the correlation matrices $\mathbf{R}_{bb}$ and $\mathbf{R}_{br}$, and then the computation of the solution to (3), $\mathbf{R}_{bb}^{-1}\mathbf{R}_{br}$, at the end of the pilot. A direct inversion at the end of the pilot is computationally expensive and delays the start of detection beyond the pilot. This delay limits the information rate. In our iterative algorithm, we approximate the ML solution based on the following ideas.

1) The product $\mathbf{R}_{bb}^{-1}\mathbf{R}_{br}$ can be directly approximated using iterative algorithms such as the gradient descent algorithm [16]. This reduces the computational complexity and is applicable in our case because $\mathbf{R}_{bb}$ is positive definite (as long as $L \geq 2K$).
2) The iterative algorithm can be modified to update the estimate as the pilot is being received instead of waiting until the end of the pilot. Therefore, the computation per bit is reduced by spreading the computation over the entire training duration. During the ith bit duration, the channel estimate, $\hat{\mathbf{A}}$, is updated iteratively in order to get closer to the ML estimate for training length of i. Therefore, the channel estimate is available for use in the detector immediately after the end of the pilot sequence.

The computations in the iterative scheme during the ith bit duration are given below

$$\mathbf{R}_{bb}^{(i)} = \mathbf{R}_{bb}^{(i-1)} + \mathbf{b}_i\mathbf{b}_i^T \tag{13}$$

$$\mathbf{R}_{br}^{(i)} = \mathbf{R}_{br}^{(i-1)} + \mathbf{b}_i\mathbf{r}_i^H \tag{14}$$

$$\hat{\mathbf{A}}^{(i)} = \hat{\mathbf{A}}^{(i-1)} - \mu\left(\mathbf{R}_{bb}^{(i)} * \hat{\mathbf{A}}^{(i-1)} - \mathbf{R}_{br}^{(i)}\right). \tag{15}$$

The term $(\mathbf{R}_{bb}^{(i)} * \hat{\mathbf{A}}^{(i-1)} - \mathbf{R}_{br}^{(i)})$ in step 3 is the gradient of the likelihood function in (2) at $\hat{\mathbf{A}}^{(i-1)}$ for a training length of i. The constant μ is the step size along the direction of the gradient. Since the gradient is known exactly, the iterative channel estimate can be made arbitrarily close to the ML estimate by repeating step 3 and using a value μ that is lesser than the reciprocal of the largest eigenvalue of $\mathbf{R}_{bb}^{(i)}$. In our simulations, we observe that a single iteration during each bit duration is sufficient in order to reach very close to the true ML estimate by the end of the training sequence. The solution converges monotonically to the true estimate with each iteration and the final error is negligible for realistic system parameters. A detailed analysis of the deterministic gradient descent algorithm can be found in [16] and [17] and a similar iterative algorithm for channel estimation for long code CDMA systems is analyzed in [24].

An important advantage of this iterative scheme is that it lends itself to a simple fixed point implementation, which was difficult to achieve using the previous inversion scheme based on ML [13]. The multiplication by the convergence parameter μ can be implemented as a right shift, by making it a power of two as the algorithm converges for a wide range of μ [24].

The proposed iterative channel estimation can also be easily extended to track slowly time-varying channels. During the tracking phase, bit decisions from the multiuser detector are used to update the channel estimate. Only a few iterations need to be performed for a slowly fading channel and the previous estimate serves as a very good initialization. The correlation matrices are maintained over a sliding window of length L as follows:

$$\mathbf{R}_{bb}^{(i)} = \mathbf{R}_{bb}^{(i-1)} + \mathbf{b}_i\mathbf{b}_i^T - \mathbf{b}_{i-L}\mathbf{b}_{i-L}^T \tag{16}$$

$$\mathbf{R}_{br}^{(i)} = \mathbf{R}_{br}^{(i-1)} + \mathbf{b}_i\mathbf{r}_i^H - \mathbf{b}_{i-L}\mathbf{r}_{i-L}^H. \tag{17}$$

B. Performance Comparisons

Iterative algorithms have been proposed earlier for channel estimation and detection in [15] and [25]–[28]. In [15] and [25], several iterative methods for general adaptive filter and equalizer applications are discussed in detail. Specific algorithms applicable for CDMA systems are developed in [26]–[29]. Most of these algorithms are based on the method of gradient descent or the method of least squares. These papers mainly target bit-error rate (BER) performance and they do not consider hardware complexity for a real-time implementation. In this paper, we propose an iterative channel estimation algorithm for multiuser channel estimation suitable for real-time implementation and we show that it has almost the same performance as schemes based on least squares.

As discussed in [15], the gradient descent algorithms can be broadly classified into two categories, deterministic and stochastic gradient descent. The well known least mean square (LMS) algorithm is a stochastic gradient algorithm, where the actual gradient is not known and is approximated by an estimated noisy gradient. In this paper, we use the deterministic gradient descent algorithm from [15]–[17], where the gradient of the objective function is known exactly, to solve the linear equation in (3).

The proposed iterative algorithm to obtain the ML estimate is related to the RLS approach for minimum mean-square-error (MMSE) estimation. In both cases, the estimate for preamble length l aims to minimize the squared error for that particular length l. However, we use the known gradient to obtain the estimate as opposed to the RLS algorithm which does not rely on gradient descent. Another difference between our iterative approach and RLS is that we use a sliding window update as opposed to RLS which uses an exponential weight factor update (λ). For the case of AWGN noise, we note that the ML and

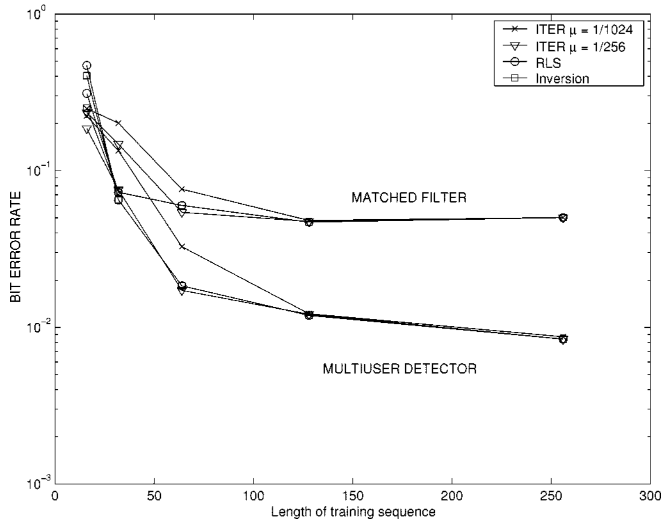


Fig. 2. BER performance comparison of the iterative scheme with RLS and true inversion for different preamble lengths. This figure shows the error performance for two detectors, a MF detector and a multiuser detector.

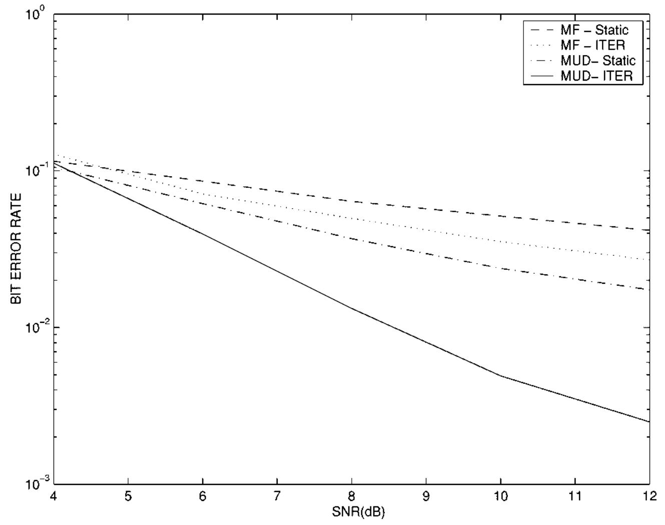


Fig. 3. Error rate performance in a multipath fading channel. This figure shows the error performance of both estimation schemes in the presence of slow fading at 10 km/h mobile velocity at a carrier frequency of 1.8 GHz. The matrix inversion based scheme assumes a static channel and is not updated with decision feedback, while the iterative scheme is updated every bit. The convergence parameter, μ, is chosen as 1/1024. A pilot sequence of 128 bits was used initially to obtain the channel estimates.

MMSE estimation approaches lead to the same solution for obtaining the channel estimate.

A comparison of the performance of our iterative scheme against the RLS algorithm is shown in Fig. 2. The simulations were performed for 8 equal power users with a spreading code of length 16 for a AWGN channel having three multipath reflections at 10 dB signal-to-noise ratio (SNR). The BER is calculated using the channel estimates after the end of the pilot phase for two types of detectors, a MF detector [5], [20] and a multistage multiuser detector (MUD) [14]. The users are all transmitting at the same power over a static channel with three paths of relative strengths 1, 0.5, and 0.33. Although the detection algorithm can handle the near–far problem, we simulated the equal power scenario as it generates the worst case for multistage detection. To use a sliding window update, we choose $\lambda = 1$ as the exponential weighting factor for RLS in our simulations. From Fig. 2, it can be seen that our iterative scheme (ITER) performs almost as well as the RLS algorithm and the actual matrix inversion. The value of μ should be less than the reciprocal of the largest eigenvalue of $\mathbf{R}_{bb}^{(i)}$ for convergence.

Since the maximum eigenvalue of $\mathbf{R}_{bb}^{(i)}$ increases with i, a larger μ is possible for a smaller preamble length. Therefore, faster convergence can be achieved for smaller preambles. The maximum value of μ that can provide stability for a given preamble can chosen at the receiver for fastest convergence. Therefore, the performance of our iterative algorithm is almost the same as that achieved by the RLS algorithm or the exact ML algorithm. From Fig. 2, we can see that the performance curves almost flatten out after a window length of 128 and, henceforth, we use $L = 128$ as our window length for simulations. Since for this window length, $\mu = 1/256$ and $\mu = 1/1024$ have the same performance, we will use $\mu = 1/1024$ henceforth in our simulations for greater stability.

Our iterative scheme is less computationally complex than RLS as we avoid the computation of the gain vector with every iteration. The RLS algorithm uses the matrix inversion lemma [15] to avoid matrix inversion but requires scalar division. Though the order of complexity in terms of multiplication and addition is the same for both the iterative scheme and RLS [$O(K^2N)$ per bit], the RLS scheme requires $O(KN)$ more divisions. The complexity difference may be thought of as the additional complexity to find a new μ (gain vector) for every iteration in RLS compared with the fixed μ used in our iterative scheme. Our iterative scheme is also more suitable for a hardware implementation than RLS. In a systolic implementation, our proposed iterative algorithm uses only truncated multipliers and adders and does not require any special boundary cells. For implementation of RLS, matrix decomposition techniques such as QR have been used [15]. The QR decomposition can also be implemented efficiently in fixed-point using systolic arrays [30], [31]. However, the cells in the array (especially, the boundary cells, which need to compute the Givens rotation) [15], [31] have more computational complexity than the cells used in our iterative algorithm.

Thus, we show that our proposed iterative algorithm has a lower computational complexity than RLS and is also more suitable for a hardware implementation. We now evaluate the performance of the iterative scheme with respect to the original ML scheme for different SNRs and for fading channels.

The analysis of the system for a multipath fading channel with tracking is as shown in Fig. 3. Here, we see that the proposed tracking scheme based on the update of (16) and (17) is able to effectively track the time-varying channel. The poor performance of the static channel assumption for this Rayleigh fading channel (with mobile velocity 10 km/h) at a carrier frequency of 1.8 GHz shows the importance of tracking. The simulation was done for 15 equal power users with a window length of 128 (and preamble length of 128). For faster fading, the window length needs to be decreased appropriately. The original channel estimation scheme requires a matrix inversion and matrix multiplication for every update while the iterative scheme reduces the complexity to a matrix multiplication per update.

C. Pipelined Detection

The multishot detection scheme [14], [32] proposed in the earlier section is block-based. Such a block-based implementation needs a windowing strategy and has to wait until all the bits needed in the window are received and are available for computation. This results in taking a window of D bits and using it to detect $D-2$ bits as the edge bits are not detected accurately due to windowing effects. Thus, there are two additional computations per block and per iteration that are not used. The detection is done in blocks and the two edge bits are thrown away and recalculated in the next iteration. However, the stages in the multistage detector can be efficiently pipelined [19] to avoid edge computations and to work on a bit-streaming basis. This is equivalent to the normal detection of a block of infinite length, detected in a simple pipelined fashion. Also, the computations can be reduced to work on smaller matrix sets. This can be done due to the block tri-diagonal nature of the matrix $\hat{\mathcal{A}}^H\hat{\mathcal{A}}$ as seen from (12). The computations performed on the intermediate bits reduce to

$$\mathbf{L} = \Re\left[\hat{\mathbf{A}}_1^H\hat{\mathbf{A}}_0\right] \tag{18}$$

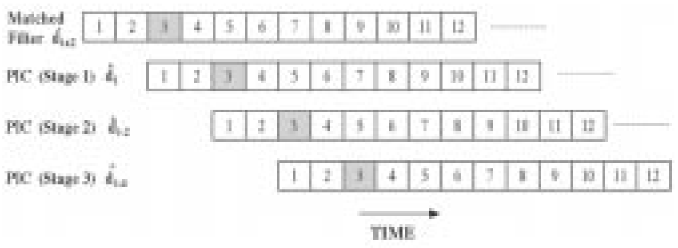
$$\mathbf{C} = \Re\left[\hat{\mathbf{A}}_0^H\hat{\mathbf{A}}_0 + \hat{\mathbf{A}}_1^H\hat{\mathbf{A}}_1 - \mathrm{diag}\left(\hat{\mathbf{A}}_0^{\mathrm{H}}\hat{\mathbf{A}}_0 + \hat{\mathbf{A}}_1^{\mathrm{H}}\hat{\mathbf{A}}_1\right)\right] \tag{19}$$

$$\mathbf{y}_i^{(l)} = \mathbf{y}_i^{(0)} - \mathbf{L}\hat{\mathbf{d}}_{i-1}^{(l-1)} - \mathbf{C}\hat{\mathbf{d}}_i^{(l-1)} - \mathbf{L}^H\hat{\mathbf{d}}_{i+1}^{(l-1)} \tag{20}$$

$$\hat{\mathbf{d}}_i^{(l)} = \mathrm{sign}\left(\mathbf{y}_{\mathrm{i}}^{(1)}\right). \tag{21}$$

Equation (20) may be thought of as subtracting the interference from the past bits of users, who have more delay, and the future bits of the users, who have less delay than the desired user. The left matrix $\mathbf{L} \in \mathbb{R}^{K\times K}$, stands for the partial correlation between the past bits of the interfering users and the desired user, the right matrix $\mathbf{L}^H$, stands for the partial correlation between the future bits of the interfering users and the desired user. The center matrix $\mathbf{C} \in \mathbb{R}^{K\times K}$, is the correlation of the current bits of interfering users and the diagonal elements are made zeros since only the interference from other users, represented by the nondiagonal elements, needs to be canceled. The lower index, i, represents time, while the upper index, l, represents the iterations. The initial estimates are obtained from the matched filter. Equation (20) is similar to the model chosen for output of the matched filter for multiuser detection in [32]. Equations (20) and (21) are equivalent to (10) and (11), where the block-based nature of the computations are replaced by bit-streaming computations.

The detection can now be pipelined as shown in Fig. 4. An example highlighting the calculation of bit 3 in the detector is shown. An initial estimate of the received signal is done using a MF detector, which depends only on the current and the past received bits. The stages of the multiuser detector need bits 2 and 4 of all users to cancel the interference for bit 3. Hence, the first-stage can cancel the interference only after the bits 2 and 4 estimates of the matched filter are available. The other stages have a similar structure. Hence, while bit 3 is being estimated from the final stage, the matched filter is estimating bit 9, the first-stage bit 7 and the second-stage bit 5. There are no edge bit computations in this scheme and, hence, they can be avoided and we get $2/D$ savings in computation per detection stage, where D is the detection window length including the edge bits. Also, instead of detecting a block of bits, each bit is detected in a streaming fashion, reducing the worst case latency by the detection window length $D/2$ and eliminating the memory requirements of block computation by a factor of D^2.



Matched Filter $\hat{d}_{1,i}$
PIC (Stage 1) $\hat{d}_i$
PIC (Stage 2) $\hat{d}_{i,2}$
PIC (Stage 3) $\hat{d}_{i,3}$
1 2 3 4 5 6 7 8 9 10 11 12
TIME

Fig. 4. Pipelined bit-streaming detection. This figure shows how the detection process can be streamlined to work on a bit basis rather than in blocks. As soon as the immediate future bits are available, the next iteration of detection is carried out. Bit 3 is highlighted as an example for pipelined detection.

D. Fixed-Point Implementation

We developed a model of the system in C++ using fixed-point "classes" in order to study the performance of the system with different precision requirements. The multiplications and addition operations were "over-loaded" so as to saturate if the available precision were to be exceeded. Since the received signal amplitude depends on the number of users in the system, the number of multiple path reflections, the spreading gain and the SNR, the amount of precision required by the A/D converter is given by

$$\text{precision (in bits)} = \left\lceil \log_2\left(K * \sum_{p=1}^{P}\frac{1}{p} + 4*10^{-(\mathrm{SNR}/20)} * \frac{\sqrt{N}}{\sqrt{2}}\right)\right\rceil + 4. \tag{22}$$

Equation (22) is due to the fact that the maximum value of the received signal would be $K*\sum_{p=1}^{P}(1/p)$, where K is the number of users and P is the number of multipath reflections. The noise would be less than $4*\sigma*(\sqrt{N}/\sqrt{2})$ with a probability of more than 0.99, where σ is the variance of the noise and N is the spreading gain. Four more bits for additional precision are provided with one bit for the sign. This gives precisions in the range of 8–12 bits for different users and spreading gains which is possible with current A/D converters.

We study the effects of finite precision on the estimation and detection algorithms based on their performance using simulations. A detailed analysis of the algorithms for finite precision (as in [33]) is challenging and is not the focus of this paper. We present two simulation results of the algorithms for finite precision with different spreading gains. Fig. 5 shows the BER performance of the channel estimation and detection algorithms for a spreading gain of 16 with 8 users. Fig. 6 shows the performance for a spreading gain of 32 with 15 users. In each case, we choose a preamble length of 128 and a μ of 1/1024 [chosen to be smaller than the reciprocal of the largest eigenvalue of $\mathbf{R}_{bb}^{(i)}$ for all i in order to ensure convergence].

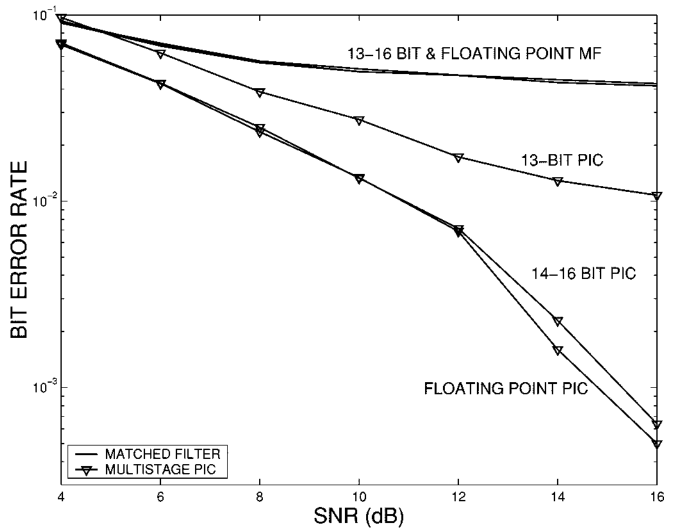


Fig. 5. Fixed point error rate performance for $N = 16$, $K = 8$. The figure shows the effects of quantization on the MF and MUD for different precisions.

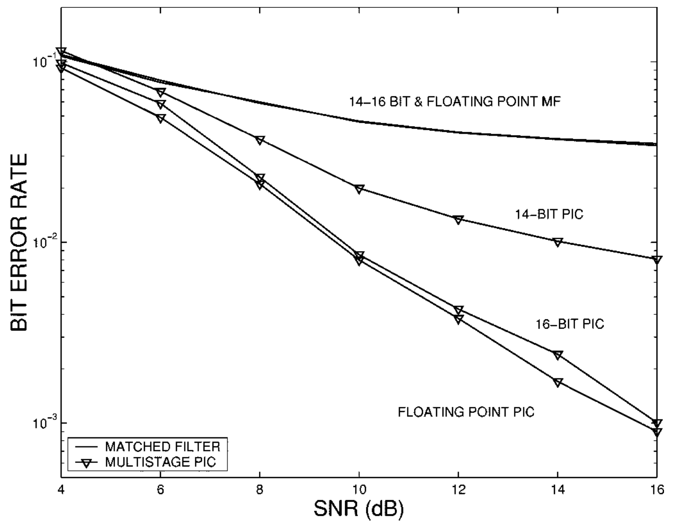


Fig. 6. Fixed point error rate performance for $N = 32$, $K = 15$. The figure shows the effects of quantization on the MF and MUD for different precisions.

Based on the simulations performed, we have made the following observations:

1) We see that 16-bit fixed point multiuser channel estimation and detection performs almost as well as floating point precision multiuser estimation and detection. In fact, for $N = 16$ and $K = 8$ the performance begins to degrade only at 13-bit precision and for $N = 32$ and $K = 15$ the performance degrades at 14-bit precision.
2) The A/D quantization of the received chip-matched filter output does not require as much precision as required for the computations. Reasonable precision of 8–12 bits for A/D conversion is sufficient. For very high SNR, there could be some degradation due to the A/D quantization as the quantization noise could be significant compared with the background noise.
3) The finite precision of the computations has greater impact on the performance of multiuser algorithms than on single-user algorithms. The matched filter receiver starts degrading only at 8-bit precision. This is reasonable to expect as the computations required for interference cancellation are more complex than that for matched filter detection. While matched filter detection requires just an inner product computation, multiuser detection requires us to solve a linear equation. Furthermore, significant performance gain is achieved in multiuser detection (compared with matched filter detection) with the extra precision.
4) Higher spreading gains and larger number of users implies larger number of multiply-and-accumulates, which may easily saturate the multipliers and adders. Hence, we see that going from $N = 16$ to $N = 32$ shows a slight increase in precision requirements (from 14 to 16).

IV. Task Decomposition and VLSI Architectures

A. *Task Decomposition of Multiuser Channel Estimation and Detection*

The various subblocks in the joint multiuser channel estimation and detection algorithm are as shown in Fig. 7. The figure shows the blocks required for channel estimation, the glue matrices $\mathbf{L}$, $\mathbf{C}$ between channel estimation and detection and the blocks in the detector. The blocks that are pipelined are shown on the horizontal time axis while the blocks that have coarse-grained parallelism are shown along the vertical axis. The dynamic range of the input is dependent on SNR, the MAI, and the number of users in the system. We assume a 16-bit precision for the architectures. The area and time requirements of the architecture do not vary significantly with the precision. Also note that the blocks $\mathbf{r}$, $\mathbf{R}_{br}$ and $\hat{\mathbf{A}}$ are complex-valued while $\mathbf{b}$ and $\mathbf{R}_{bb}$ are real-valued. For the sake of convenience, we henceforth represent the current inputs $\mathbf{b}_i$, $\mathbf{r}_i$ as $\mathbf{b}$, $\mathbf{r}$ and $\mathbf{b}_{i-L}$, $\mathbf{r}_{i-L}$ as $\mathbf{b}_0$, $\mathbf{r}_0$, respectively. All the architectures assume a single-cycle multiplication and addition as both multiplication and addition can be implemented in $\log(n)$ type computations [34] where n is the number of bits and the single cycle assumption also helps us with the DSP comparisons. We assume that a Wallace or Dadda multiplier tree [34] is used for multiplication requiring $O(n^2)$ 1-bit full adders (FA) for an n-bit multiplication. Since the multiplication by μ in (15) (implemented as a shift) results in truncation of the output, a truncated multiplication using significantly less hardware [35] can be used. The delays of blocks such as multiplexers and gates are assumed to be included in the single-cycle delay. For an area estimate of the architectures, we consider the number of 1-bit FA cells in the design. It can be observed from Fig. 7 that the bottlenecks in the pipeline are the matrix multiplications $\mathbf{R}_{bb} * \hat{\mathbf{A}}$ for channel estimation and the calculation of the $\mathbf{L}$, $\mathbf{C}$ matrices for multiuser detection.

We explore different area-time tradeoffs to develop real-time architectures with minimum area overhead. We explain the design in detail for a time-constrained architecture which shows the upper bound on data rates with no constraints on hardware and then show that by constraining hardware, we are able to design different architectures to meet real-time requirements with minimum area overhead. We have considered only the computational complexity for our analysis and have ignored the analysis of the memory requirements. This is because the focus of

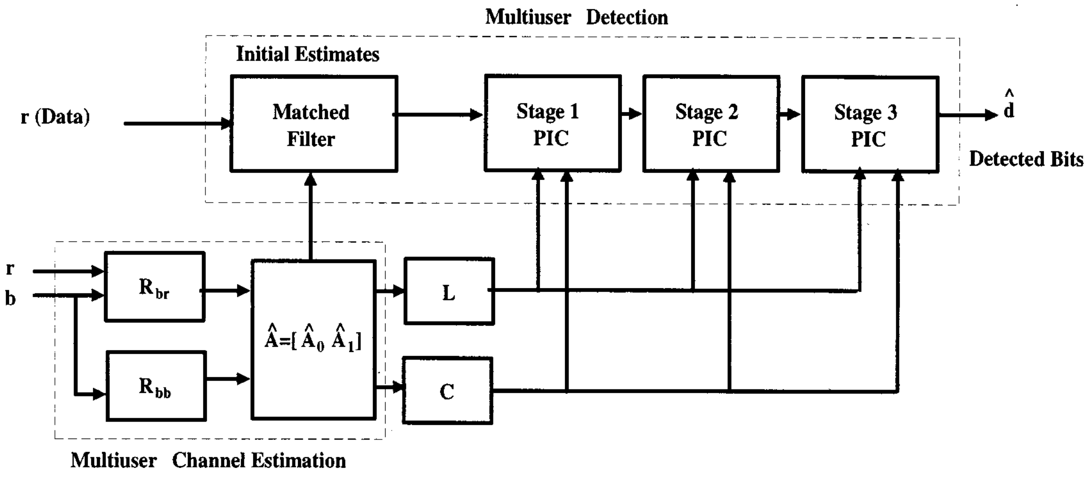


Fig. 7. Task decomposition of joint multiuser channel estimation and detection. This figure illustrates how multiuser channel estimation and detection can be split into different tasks and can be pipelined. Vertically aligned blocks can be computed in parallel while horizontally aligned blocks are pipelined.

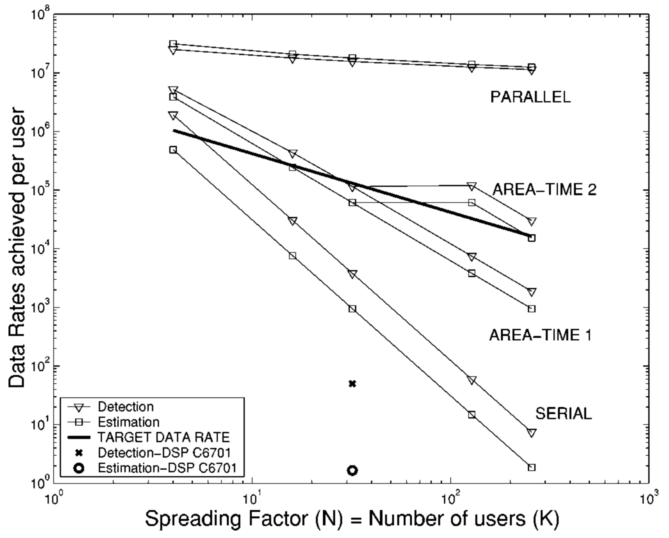


Fig. 8. Data rates achieved by different VLSI architecture tradeoffs with varying spreading gains. The figure shows the data rates achieved for a serial, a parallel and two data rate targeted area-time tradeoff architectures.

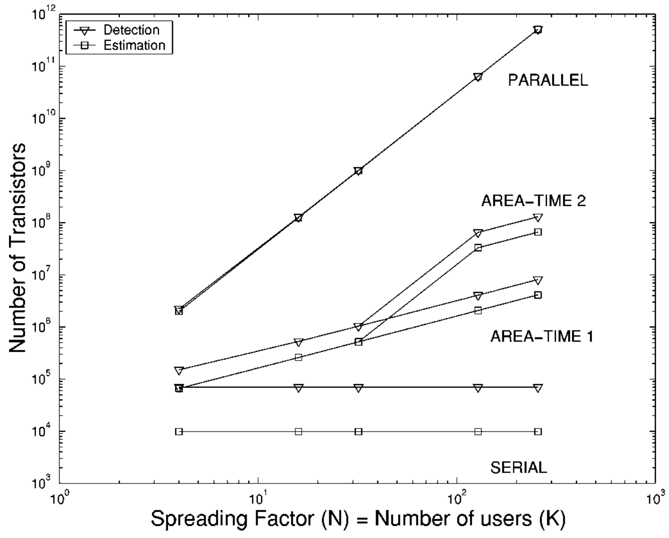


Fig. 9. Area requirements for different VLSI architecture tradeoffs with varying spreading gains. The figure shows the number of transistors required for a serial, a parallel and two data rate targeted area-time tradeoff architectures with 16-bit precision.

our paper was on the computational complexity and area-time tradeoffs needed to meet real-time requirements. We have done an analysis for the memory requirements in previous work for channel estimation [36]. Fig. 8 shows the achievable data rates and Fig. 9 shows the transistor count for the architectures discussed below. We assume 28 transistors per 1-bit standard FA cell as in [34].

B. Area-Time Tradeoffs for Channel Estimation Architectures

1) Time-Constrained Architecture: The block diagram of a time-constrained architecture is as shown in Fig. 10. In this architecture, the available parallelism in the algorithm is exploited to the maximum extent. Hence, all the elements needed to perform a parallel matrix multiplication are computed simultaneously. The entire matrices $\mathbf{R}_{bb}$ and $\hat{\mathbf{A}}$ are multiplied using an array of multipliers. The entire product matrix is subtracted by the autocorrelation matrix, $\mathbf{R}_{br}$, shifted and a new channel estimate is formed. Thus, as the time taken by the other computations is pipelined with the time for the multiplication, the output matrix can be formed in parallel every $\log_2(2K) + 1$ using $4K^2N$ multipliers. This is because each element of an $N * N$ product matrix can be computed in $\log_2(N) + 1$ time using N^3 multipliers and using a tree structure to compute the inner products [37], in a time-constrained architecture.

We also exploit the bit-level arithmetic and parallel structure of the correlation matrices to form the correlation matrices simultaneously within a cycle. Since the autocorrelation matrix update is a symmetric matrix and all the diagonal elements are 1s ($\overline{a \oplus a} = 1$), we need to compute only the strictly upper triangular (or lower triangular) part of the autocorrelation matrix. Also, as the updates are all $+1$s or -1s, this can be obtained from a simple single-bit XNOR gate structure. As the autocorrelation matrix is always updated and down-dated by ±1s, increment/decrement counters can be used in place of general adders in our design. Also, the elements in the cross-correlation update are $+\mathbf{r}$ or $-\mathbf{r}$ and hence, the vector $\mathbf{r}$ could be directly added or subtracted with every column of the cross-correlation matrix based on the sign of the bit vector $\mathbf{b}$.

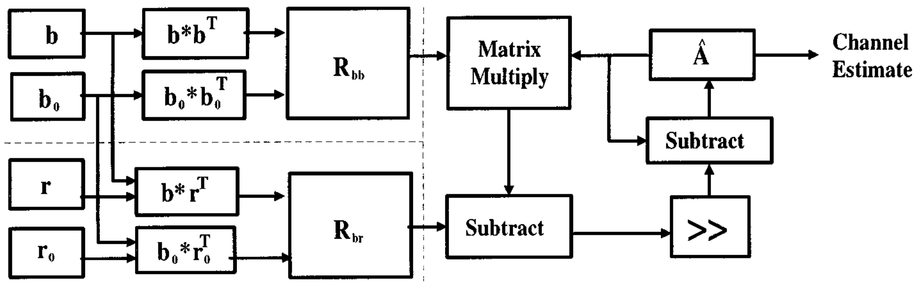


Fig. 10. Time-constrained VLSI architecture block diagram. The dotted lines show the parts corresponding to (13)–(15). All operations within a block are being computed by exploiting the maximum parallelism available in that block. The parallel matrix multiplication is the bottleneck.

The area requirements for the time-constrained architecture are as shown in Fig. 9. The area requirements vary from 10^6 to 10^{12} transistors. This is a highly aggressive solution with today's technology and it is not feasible to devote so many FA cells just for channel estimation, which is only a part of the complete receiver. However, this states the theoretical minimum time requirements by exploiting the available parallelism as $\log_2(2K) + 1$, which is the time required to do the parallel multiplication and pipelined integration with the other blocks. We require $2KN(2K - 1)$ adders for doing the recursive doubling [37] in $\log_2(2K)$ time [adding $2K$ elements in $\log_2(2K)$ time requires $(2K - 1)$ adders] and $2KN$ adders for the subtraction following the multiplication. The data rates achieved by this fully parallel architecture is shown in Fig. 8. We can see that we are able to get one to two orders of magnitude performance more than necessary using the amount of parallelism in the algorithms. Therefore, we propose better area-time tradeoffs more closely matched to the target data rates in Section IV-B3.

2) Area-Constrained Architecture: For an area-constrained architecture, we assume that only a single multiplier and adder are available. Thus, the matrix–matrix multiplication serially takes $4K^2N$ cycles. The data rates achieved and area requirements for this architecture are shown in Figs. 8 and 9. We see that though the serial architecture uses very little area, it falls below real-time requirements by one to two orders of magnitude.

3) Data Rate Targeted Area-Time Tradeoffs: In this section, we use part of the available parallelism to achieve real-time performance with minimum area overhead. We use a vector multiplier calculating each row of the multiplication in parallel. This is shown in Figs. 8 and 9 as *AREA-TIME1*. Thus, the multiplication now takes $2KN$ cycles at an $2K$ increase in the number of multipliers. This seems to meet real-time requirements up to $N = 32$ as seen in Fig. 8. However, for $N > 32$, it can be seen that greater amounts of parallelism need to be used to meet real-time. For $N > 32$, we found that additionally 16 columns of the matrix need to be computed in parallel. This implies that the matrix multiplication is done in $KN/8$ cycles and at a further 16× increase in the number of multipliers. This is shown in Figs. 8 and 9 as *AREA-TIME2*.

C. Area-Time Tradeoffs for Multiuser Detection Architectures

1) Time-Constrained Architecture: A detailed task partition of the blocks for multiuser detection are as shown in Fig. 11. The blocks consist of a MF detector which provides the initial hard

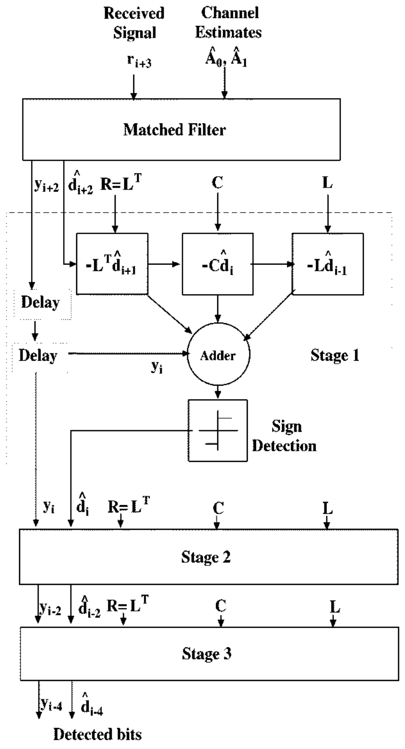


Fig. 11. Time-constrained pipelined detector architecture. The figure shows the matched filter which provides the initial estimates for parallel interference cancellation. A three-stage parallel interference cancellation detector is shown. The first-stage of the detector is expanded in detail.

$(\hat{\mathbf{d}})$ and soft estimates $(\mathbf{y})$ to the parallel interference cancellation stages. A three-stage detector is chosen for implementation as it provides sufficient convergence [23].

An array of parallel multipliers is used for computing the entire matched filter estimate $\hat{\mathbf{A}}_0^H\mathbf{r}$ and $\hat{\mathbf{A}}_1^H\mathbf{r}$ vectors in parallel. As the imaginary parts of the products need not be computed, this requires $4KN$ multipliers. To form the inner product addition in parallel for every row of $\hat{\mathbf{A}}_0^H$ and $\hat{\mathbf{A}}_1^H$, we use an adder tree utilizing $K(2N - 1)$ adders. The matched filter estimate can be computed in $\log_2(N) + 2$ time.

The glue matrices, $\mathbf{L}$, $\mathbf{C}$, between the channel estimation and detection schemes require a significant amount of computation. Since $\hat{\mathbf{A}}_0^H\hat{\mathbf{A}}_0$ and $\hat{\mathbf{A}}_1^H\hat{\mathbf{A}}_1$ are symmetric and their diagonal elements and imaginary parts need not be computed to get the matrix products in a time-constrained architecture, we require $2K(K-1)N$ multipliers and $K(K-1)(4N-1)/2$ adders to find the dot products in a tree fashion. This requires $\log_2(K) +$

3 time. Similarly, for the computation of $\hat{\mathbf{A}}_1^H\hat{\mathbf{A}}_0$, we require $2K^2N$ multipliers and $K^2(2N-1)$ adders in $\log_2(N)+2$ time.

Each stage of the multiuser detector uses only adders as multiplication by single bits can be reduced to addition and subtraction. In order to form the various vectors such as $\mathbf{C}\hat{\mathbf{d}}_i$ in (20), an adder tree of $2K-1$ adders. Thus, for computing $\mathbf{L}\hat{\mathbf{d}}_{i-1}$, $\mathbf{C}\hat{\mathbf{d}}_i$ and $\mathbf{L}^H\hat{\mathbf{d}}_{i+1}$, we need $3(2K-1)$ adders followed by $3K$ more adders (for a four-operand tree addition) to get the final soft decisions $\mathbf{y}$. Each stage of the multistage detector can be computed in $\log_2(K)+3$ time, assuming two cycles for the final four-operand addition and a single cycle for the multiplication. The achieved data rates and area requirements for detection are also shown in Figs. 8 and 9. The detector architecture also takes 10^6–10^{12} transistors, which is not an efficient solution with today's technology but serves to reveal the parallelism and pipelining in the algorithm and determine the maximum data rate.

2) Area-Constrained Architecture: For an area-constrained architecture, we use a single multiplier for the $\mathbf{L}$, one for $\mathbf{C}$ and one for the matched filter. The latency time depends on the matrix–matrix multiplications for the $\mathbf{L}$ matrix, which takes K^2N cycles. The data rates achieved and area requirements for the area constrained architectures are shown in Figs. 8 and 9.

3) Data Rate Targeted Area-Time Tradeoffs: The area-time complexity for multiuser detection is found to be similar to that for channel estimation and hence, we use the same type of area-time tradeoffs as before. This is shown in Figs. 8 and 9 as *AREA-TIME1*. Thus, the multiplication now takes KN cycles at a K increase in the number of multipliers. This potentially can meet real-time requirements up to $N=32$ as observed in Fig. 8. However, for $N>32$, it can be seen that greater amounts of parallelism need to be used to meet real-time. Hence, for $N>32$, we found that 16 columns of the matrix also needs to be computed in parallel. This implies that the multiplication is done in $KN/16$ cycles and at a further 16 times increase in the number of multipliers. This is shown in Figs. 8 and 9 as *AREA-TIME2*.

V. Results and Comparisons

A. Computational Savings

The computational advantages of the newly proposed schemes over the previous schemes are shown in Table I. The original algorithm for channel estimation required a matrix inversion and a matrix multiplication requiring $O(6K^3+4K^2N)$ cycles on a sequential uni-processor machine such as a DSP while estimation using the iterative method requires only a matrix multiplication $O(4K^2N)$ on a sequential machine. As N and K are of the same order, this only implies a savings of the order of two times. However, a fully parallel VLSI solution for implementation can accelerate the time requirements to $O(\log_2(2K)+1)$. Similarly, for comparing the detection schemes, we assume that a window of D bits need to be detected. For every window, we save $O(2MK^2)$ computations, assuming an M-stage detector as the edge bits do not need to be calculated. A fully pipelined time-constrained detector can reduce the time requirements to $O(\log_2(N)+3)$ by exploiting available parallelism. Note that the enhanced algorithms, as seen from Table I do not have inherent computational savings but are designed to benefit from exploiting parallelism and pipelining in an architecture. Thus, significant benefits in performance can be achieved by enhancing the existing schemes for channel estimation and detection with schemes having an efficient hardware implementation and exploiting the available parallelism.

TABLE I
COMPARISONS OF COMPUTATIONAL TIME SAVINGS. THIS TABLE SHOWS THE COMPUTATIONAL SAVINGS ACHIEVED BY THE ENHANCED SCHEMES FOR MULTIUSER ESTIMATION AND DETECTION OVER THE PREVIOUS SCHEMES. K—NUMBER OF USERS, N—SPREADING GAIN, D—DETECTION WINDOW, M—NUMBER OF STAGES

Blocks	Architecture	Original (Cycles)	Enhanced (Cycles)
Channel Estimation	Uni-processor	$O(6K^3+4K^2N)$	$O(4K^2N)$
	Parallel	-	$O(\log_2(2K)+1)$
Multiuser Detection	Uni-processor	$O(DNK+M(D+2)K^2)$	$O(DNK+MDK^2)$
	Parallel	-	$O(\log_2(N)+3)$

B. Comparisons With DSPs

Though DSPs and general purpose processors with MMX-enhanced instruction sets exploit byte-wide parallelism, they are inefficient for processing on bits. Storage of bits as bytes on such processors is inefficient as there is a large overhead involved in packing and unpacking these bits. Also, the compiler may not replace bit-level multiplications with additions and subtractions. Using a control structure instead, also limits the utilization of available parallelism. Formation of bit-level matrix updates is much more effective and simpler to build in parallel with XNOR gates than as sequential multiplications on DSPs.

Fig. 8 also compares the VLSI architectures at 500 MHz with the single processor DSP implementation of the multiuser channel estimation and detection algorithms on a TI C6701 floating-point DSP at 167 MHz. We did the DSP analysis in an earlier work [38] and hence, have comparison points only for the $N=K=32$ case. The channel estimation DSP implementation takes 600 ms for all 32 users. This poor performance is due to the computation of a matrix multiplication per received bit on the DSP. The frequency of updates to the channel estimates can be reduced for slow fading channels for better time performance. Similarly, detection takes 20 ms for all 32 users. The low data rate performance of the detector is because we consider a more realistic and complete system with continuous updating of channel estimates to the detector as compared with a static channel assumption and neglecting effects of channel estimation in other detector DSP implementations [6], [23].

VI. Conclusion

We first present computationally efficient algorithms to meet real-time requirements of multiuser channel estimation and detection in future wireless base stations. Existing algorithms for multiuser channel estimation and detection are redesigned from an implementation perspective for a reduced complexity solution. The ML based channel estimation algorithm requiring matrix inversions, block-based computations and floating point accuracy is redesigned for an iterative scheme, which has a simpler fixed point VLSI architecture and reduced complexity. Multiuser detection is also redesigned for a pipelined structure, that reduces the memory requirements by a factor of D^2 and worst case latency by $D/2$. The edge bit computations in the block scheme are eliminated and a $2/D$ improvement in computational complexity per detection stage is achieved.

We then present fixed point, real-time VLSI architectures for multiuser channel estimation and detection. The proposed VLSI architecture schemes can be integrated with DSP architectures as a coprocessor support [39] to build single DSP base-station solutions. Bit-level extensions [40] can also be similarly developed to utilize bit-level parallelism on DSPs and accelerate wireless communication algorithms.

REFERENCES

[1] F. Adachi, M. Sawahashi, and H. Suda, “Wideband DS-CDMA for next-generation mobile communication systems,” *IEEE Commun. Mag.*, vol. 36, pp. 56–69, Sept. 1998.

[2] T. Ojanpera and R. Prasad, Eds., *Wideband CDMA for Third Generation Mobile Communications*. Norwood, MA: Artech House, 1998.

[3] Third Generation Partnership Project.. [Online]. Available: http://www.3gpp.org

[4] S. Verdú, *Multiuser Detection*. Cambridge, U.K.: Cambridge University Press, 1998.

[5] S. Moshavi, “Multi-user detection for DS-CDMA communications,” *IEEE Commun. Mag.*, pp. 124–136, Oct. 1996.

[6] I. Seskar and N. B. Mandayam, “A software radio architecture for linear multiuser detection,” *IEEE J. Select. Areas Commun.*, vol. 17, pp. 814–823, May 1999.

[7] S. Das, S. Rajagopal, C. Sengupta, and J. R. Cavallaro, “Arithmetic acceleration techniques for wireless communication receivers,” in *33rd Asilomar Conf. Signals, Systems, and Computers*, Pacific Grove, CA, Oct. 1999, pp. 1469–1474.

[8] R. L. Pickholtz, D. L. Schilling, and L. B. Milstein, “Theory of spread-spectrum communications—A tutorial,” *IEEE Trans. Communs.*, vol. 30, pp. 855–884, May 1982.

[9] A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communication*. Reading, MA: Addison-Wesley, 1995.

[10] N. S. Correal, R. M. Buehrer, and B. D. Woerner, “A DSP-based DS-CDMA multiuser receiver employing partial parallel interference cancellation,” *IEEE J. Select. Areas Commun.*, vol. 17, pp. 613–630, Apr. 1999.

[11] T. Long and N. R. Shanbhag, “Low-power CDMA multiuser receiver architectures,” in *Proc. IEEE Workshop on Signal Processing Systems*, Taipei, Taiwan, Oct. 1999, pp. 493–502.

[12] C. Teuscher, D. Lee, N. Zhang, and R. Brodersen, “Design of a wideband spread spectrum radio using adaptive multiuser detection,” in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 4, May 1998, pp. 593–599.

[13] C. Sengupta, J. R. Cavallaro, and B. Aazhang, “On multipath channel estimation for CDMA using multiple sensors,” *IEEE Trans. Commun.*, vol. 49, pp. 543–553, Mar. 2001.

[14] M. K. Varanasi and B. Aazhang, “Multistage detection in asynchronous code-division multiple-access communications,” *IEEE Trans. Commun.*, vol. 38, pp. 509–519, Apr. 1990.

[15] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996, Information and System Sciences Series.

[16] G. H. Golub and C. F. VanLoan, *Matrix Computations*, third ed. Baltimore, MD: The John Hopkins University Press, 1996, ch. 10, pp. 520–521.

[17] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*: Society of Industrial and Applied Mathematics, 1996.

[18] M. J. Omidi, P. G. Gulak, and S. Pasupathy, “Parallel structures for joint channel estimation and data detection over fading channels,” *IEEE J. Select. Areas Commun.*, vol. 16, pp. 1616–1629, Dec. 1998.

[19] S. Rajagopal and J. R. Cavallaro, “A bit-streaming pipelined multiuser detector for wireless communications,” in *IEEE Int. Symp. Circuits and Systems (ISCAS)*, vol. 4, Sydney, Australia, May 2001, pp. 128–131.

[20] S. Verdú, “Minimum probability of error for asynchronous Gaussian multiple-access channels,” *IEEE Trans. Inform. Theory*, vol. 32, pp. 85–96, 1986.

[21] E. Ertin, U. Mitra, and S. Siwamogsatham, “Iterative techniques for DS/CDMA multipath channel estimation,” in *Proc. Allerton Conf. Communication, Control and Computing*, Monticello, IL, Sept. 1998, pp. 772–781.

[22] H. L. Van Trees, *Detection, Estimation, and Modulation Theory—Part 1: Detection, Estimation, and Linear Modulation Theory*, 1st ed. New York: Wiley, 1968.

[23] G. Xu and J. R. Cavallaro, “Real-time implementation of multistage algorithm for next generation wideband CDMA systems,” in *Proc/ Advanced Signal Processing Algorithms, Architectures, and Implementations IX, SPIE*, vol. 3807, Denver, CO, July 1999, pp. 62–73.

[24] S. Bhashyam and B. Aazhang, “Multiuser channel estimation for long code CDMA systems,” in *Proc. IEEE Wireless Communication and Networking Conf. (WCNC)*, Chicago, IL, Sept. 2000, pp. 664–669.

[25] S. U. H. Qureshi, “Adaptive equalization,” *Proc. IEEE*, vol. 73, pp. 1349–1387, Sept. 1985.

[26] P. B. Rapajic and B. S. Vucetic, “Adaptive receiver structures for asynchronous CDMA systems,” *IEEE J. Select. Areas Commun.*, vol. 12, pp. 685–697, May 1994.

[27] M. Honig, U. Madhow, and S. Verdú, “Blind adaptive multiuser detection,” *IEEE Trans. Inform. Theory*, vol. 41, pp. 944–960, July 1995.

[28] R. F. Smith and S. L. Miller, “Acquisition performance of an adaptive receiver for DS-CDMA,” *IEEE Trans. Commun.*, vol. 47, pp. 1416–1424, Sept. 1999.

[29] G. Woodward, P. B. Rapajic, and B. S. Vucetic, “Adaptive algorithms for asynchronous DS-CDMA receivers,” in *Proc. IEEE Int. Symp. Personal, Indoor and Mobile Radio Communications (PIMRC)*, vol. 2, Taipei, Taiwan, Oct. 1996, pp. 583–587.

[30] P. S. R. Diniz and M. G. Siqueira, “Fixed-point error analysis of the QR-recursive least square algorithm,” *IEEE Trans. Circuits Syst. II: Analog and Digital Signal Processing*, vol. 42, pp. 334–348, May 1995.

[31] K. J. Raghunath and K. K. Parhi, “Finite-precision error analysis of QRD-RLS and STAR-RLS adaptive filters,” *IEEE Trans. Signal Processing*, vol. 45, pp. 1193–1209, May 1997.

[32] M. J. Juntti and B. Aazhang, “Finite memory-length multiuser detection for asynchronous CDMA communications,” *IEEE Trans. Commun.*, vol. 45, pp. 611–622, May 1997.

[33] R. D. Gitlin and S. B. Weinstein, “On the required tap-weight precision for digitally implemented, adaptive, mean-squared equalizers,” *Bell Syst. Tech. J.*, vol. 58, pp. 301–321, Feb. 1979.

[34] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*, 2nd ed. Reading, MA: Addison-Wesley, 1993, ch. 8.

[35] M. J. Schulte and E. E. Swartzlander, “Truncated multiplication with correction constant,” in *Workshop on VLSI Signal Processing*, vol. VI, Veldhoven, Netherlands, Oct. 1993, pp. 388–396.

[36] S. Rajagopal, S. Bhashyam, J. R. Cavallaro, and B. Aazhang, “Efficient VLSI architectures for baseband signal processing in wireless base-station receivers,” in *Proc. 12th IEEE Int. Conf. Application-Specific Systems, Architectures and Processors (ASAP)*, Boston, MA, July 2000, pp. 173–184.

[37] J. J. Modi, *Parallel Algorithms and Matrix Computation*. London, U.K.: Oxford Univ. Press, 1988.

[38] S. Rajagopal, B. A. Jones, and J. R. Cavallaro, “Task partitioning base-station receiver algorithms on multiple DSPs and FPGAs,” in *Proc. Int. Conf. Signal Processing, Applications and Technology*, Dallas, TX, Aug. 2000.

[39] A. Gatherer, T. Stetzler, M. McMahan, and E. Auslander, “DSP-based architectures for mobile communications: Past, present and future,” *IEEE Commun. Mag.*, vol. 38, pp. 84–90, Jan. 2000.

[40] S. Rajagopal, “Baseband architecture design for future wireless base-stations,” M.S. thesis, Rice University, May 2000.

Sridhar Rajagopal (S’98) received the B.E degree (honors with distinction) in electronics engineering from VJTI, Bombay University, India, in 1998, and the M.S. degree in electrical and computer engineering from Rice University, Houston, TX, in 2000. He is currently a doctoral candidate at Rice University.

His research interests are in wireless communications, VLSI signal processing, computer arithmetic, and computer architecture.

Srikrishna Bhashyam (S'96–M'01) received the B.Tech. degree in electronics and communication engineering from the Indian Institute of Technology, Madras, India, in 1996, and the M.S. and Ph. D. degrees in electrical engineering from Rice University, Houston, TX, in 1998 and 2001, respectively.

He is currently a Senior Engineer at Qualcomm, Inc., Campbell, CA. His interests include wireless multimedia communications, multiple access communications, and information theory.

Joseph R. Cavallaro (M'82) was born in Philadelphia, PA on August 22, 1959. He received the B.S. degree from the University of Pennsylvania, Philadelphia, PA, in 1981, the M.S. degree from Princeton University, Princeton, NJ, in 1982, and the Ph.D. degree from Cornell University, Ithaca, NY, in 1988, all in electrical engineering.

From 1981 to 1983, he was with AT&T Bell Laboratories, Holmdel, NJ. In 1988, he joined the faculty of Rice University, Houston, TX, where he is currently an Associate Professor of Electrical and Computer Engineering, having received tenure in 1994. During the 1996–1997 academic year, he served at the U.S. National Science Foundation as Director of the Prototyping Tools and Methodology Program in the Computer (CISE) Directorate. He is currently the Associate Director of the Center for Multimedia Communication at Rice University. His research interests include computer arithmetic, fault tolerance, VLSI design and microlithography, and DSP and VLSI architectures and algorithms for applications in wireless communications and robotics.

Dr. Cavallaro is a recipient of the NSF Research Initiation Award 1989–1992, the IBM Graduate Fellowship 1987–1988, and is a member of Tau Beta Pi and Eta Kappa Nu.

Behnaam Aazhang (S'82–M'82–SM'91–F'99) received the B.S. (with highest honors), M.S., and Ph.D. degrees in electrical and computer engineering from University of Illinois at Urbana-Champaign, in 1981, 1983 and 1986, respectively.

From 1981 to 1985, he was a Research Assistant in the Coordinated Science Laboratory, University of Illinois. In August 1985, he joined the faculty of Rice University, Houston, Texas, where he is now the J.S. Abercrombie Professor in the Department of Electrical and Computer Engineering and the Director of Center for Multimedia Communications. He has been a Visiting Professor at IBM Federal Systems Company, Houston, Texas, the Laboratory for Communication Technology at Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, the Telecommunications Laboratory at University of Oulu, Oulu, Finland, and at the U.S. Air Force Phillips Laboratory, Albuquerque, New Mexico. His research interests are in the areas of communication theory, information theory, and their applications with emphasis on multiple access communications, cellular mobile radio communications, and optical communication networks.

Dr. Aazhang is a recipient of the Alcoa Foundation Award 1993, the NSF Engineering Initiation Award 1987–1989, and the IBM Graduate Fellowship 1984–1985, and is a member of Tau Beta Pi and Eta Kappa Nu. He is currently serving on Houston Mayor's Commission on Cellular Towers. He has served as the Editor for Spread Spectrum Networks of IEEE Transactions on Communications 1993–1998, as the Treasurer of IEEE Information Theory Society 1995–1998, the Technical Area Chair of 1997 Asilomar Conference, Monterey, California, the Secretary of the Information Theory Society 1990–1993, the Publications Chairman of the 1993 IEEE International Symposium on Information Theory, San Antonio, TX, and as the cochair of the Technical Program Committee of 2001 Multidimensional and Mobile Communication (MDMC) Conference in Pori, Finland.