Parallel Structures for Joint Channel Estimation and Data Detection over Fading Channels

Mohammad Javad Omidi, P. Glenn Gulak, *Senior Member, IEEE,* and S. Pasupathy, *Fellow, IEEE*

***Abstract*— Joint data and channel estimation for mobile communication receivers can be realized by employing a Viterbi detector along with channel estimators which estimate the channel impulse response. The behavior of the channel estimator has a strong impact on the overall error rate performance of the receiver.**

Kalman filtering is an optimum channel estimation technique which can lead to significant improvement in the receiver bit error rate (BER) performance. However, a Kalman filter is a complex algorithm and is sensitive to roundoff errors. Square-root implementation methods are required for robustness against numerical errors. Real-time computation of the Kalman estimator in a mobile communication receiver calls for parallel and pipelined structures to take advantage of the inherent parallelism in the algorithm.

In this paper different implementation methods are considered for measurement update and time update equations of the Kalman filter. The unit-lower-triangular-diagonal (LD) correction algorithm is used for the time update equations, and systolic array structures are proposed for its implementation. For the overall implementation of joint data and channel estimation, parallel structures are proposed to perform both the Viterbi algorithm and channel estimation. Simulation results show the numerical stability of different implementation techniques and the number of bits required in the digital computations with different estimators.

***Index Terms*— Estimation, fading channels, Kalman filtering, maximum likelihood detection, parallel architectures, systolic arrays, Viterbi detection.**

I. Introduction

FOR data transmission over Rayleigh fading channels in a mobile communication system, advanced equalization techniques are often required. Maximum likelihood sequence detection (MLSD) is a well-known detection method for data signals received over a frequency-selective multipath fading channel. MLSD can be implemented using the Viterbi algorithm. Optimum detection of the transmitted data through channels with intersymbol interference (ISI) requires the knowledge of the channel impulse response (CIR). If the CIR is fed to the Viterbi detector, the digital data can be detected in the MLSD sense. The fading channel, however, is a time varying system and hence the CIR has to be estimated with tracking algorithms. Often, the channel is rapidly time varying and fast tracking methods should be applied for channel identification.

The two most widely used channel estimation methods are: the least mean square (LMS) and the recursive least square (RLS) algorithms. The performance and tracking behavior of the channel estimator directly affects the overall bit error rate (BER) of the receiver [1]. Employing the Kalman filter for channel estimation gives rise to very good tracking performance. The BER obtained by the Kalman filter is lower compared to other estimation methods and, in addition, the Kalman filter can efficiently follow rapid changes of the CIR in fast fading environments.

The Kalman filter is computationally demanding, and this limits its use in real-time applications. The conventional Kalman filter algorithm is also very sensitive to roundoff errors. In order to obtain a numerically accurate and stable algorithm, square-root solutions have been proposed for implementation of the Kalman filter. With recent advances in very large scale integration (VLSI) technology parallel information processing has become more and more feasible, allowing for the implementation of dedicated systolic structures for square-root Kalman filtering. An overview of some algorithms for the implementation of Kalman filter is given in [2].

The implementation of the Kalman filter can be divided into two parts: implementation of the measurement update equations and implementation of time update equations. The first part is basically the same as the RLS algorithm. Jover and Kailath have proposed an algorithm and a parallel structure for the measurement update equations [3]. This algorithm has been adopted in [4] with some modifications for the application to wireless mobile communications, and it is shown that the VLSI structure can be drastically simplified if it is used for the realization of a RLS estimator. To implement the time update equations of the Kalman filter, a weighted Gram–Schmidt (WGS) orthogonalization method is widely used. In a study of target-tracking methods, Raghavan *et al*. [5] proposed the application of a unit-lower-triangular-diagonal correction (LDC) method for time update measurements of the Kalman filter. This algorithm requires less computation compared to the WGS orthogonalization method.

In the literature, the implementation of estimation algorithms are usually considered only generally and not for a specific application. In this paper we study the implementation of fading channel estimators along with the Viterbi detector.

Manuscript received August 15, 1997; revised March 15, 1998. This work was supported in part by the Information Technology Research Center of Ontario and the Natural Sciences and Engineering Research Council of Canada. The work of M. J. Omidi was supported in part by the Ministry of Culture and Higher Education of Iran.

The authors are with the Department of Electrical and Computer Engineering at the University of Toronto, Toronto, Canada.

Publisher Item Identifier S 0733-8716(98)08642-9.

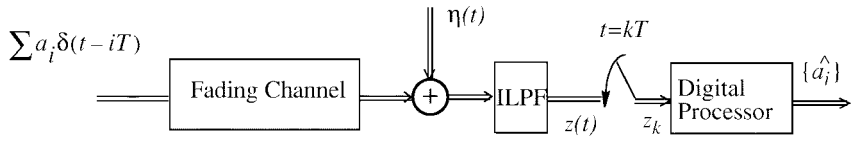


Fig. 1. The signal model for the baseband communication system.

The LDC algorithm of [5] is adopted here, for the first time, for application to mobile communication receivers, and systolic structures are developed and studied for VLSI implementation of the LDC algorithm. Utilization of the estimators is considered in the structure of the Viterbi based receiver, implementing joint data detection and channel estimation. Parallel structures are proposed for the implementation of the Viterbi detector in a per-survivor processing (PSP) [6]–[11] fashion that offers an improved and robust detection technique. Finally, the robustness of the receiver structure is studied with regard to the wordlength required in a digital implementation. Studies show that the WGS orthogonalization method and the correction algorithm need the same number of bits in implementation while the latter requires less computation.

This paper is organized as follows: Section II is a short overview of the mobile communication system under consideration and the proposed receiver algorithm for joint data detection and channel estimation. In Section III, we investigate different methods for implementation of the channel estimator. Parallel structures for joint Viterbi data detection and channel estimation are introduced in Section IV. In Section V we study the issue of choosing the wordlength in a hardware implementation of different channel estimators. Finally, concluding remarks are given in Section VI.

II. The Communication System

We will consider the differentially coded quadrature phase-shift keying (DQPSK) signaling scheme for simplicity. This is close in performance to the $\pi/4$-shifted DQPSK modulation technique of the north American narrowband time division multiple access (TDMA) standard (IS-136). The baseband signal model for the communication system is shown in Fig. 1. The complex data sequence $\{a_i\}$ with the symbol period T is input to the fading channel. The fading channel includes the shaping filter in the transmitter, such as a raised-cosine filter. The additive noise $\eta(t)$ is a complex circularly symmetric [12] Gaussian process with power density N_o'. The signal $z(t)$ is sampled at symbol rate T. The bandwidth of the ideal lowpass filter (ILPF) is B. The noise samples $\eta(kT)$ are complex uncorrelated Gaussian random variables with variance $N_o = 2BN_o'$.

The fading channel can be modeled as a linear time varying system. A model for a two-ray Rayleigh fading channel is shown in Fig. 2. One ray is delayed with respect to the other one and both rays are multiplied by filtered Gaussian noise. Both $x(k)$ and $y(k)$ are zero-mean circularly symmetric Gaussian complex random signals and are shaped by the fading filters according to the maximum Doppler frequency shift to produce the multiplicative coefficients. The fading filter can

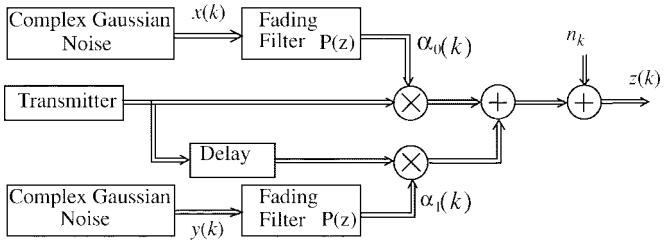


Fig. 2. The fading channel model.

be approximated as a third-order filter [1] with the transfer function

$$P(z) = \frac{D}{1 - Az^{-1} - Bz^{-2} - Cz^{-3}}. \tag{1}$$

At sampling time k the CIR, $\mathbf{h}_k$, a complex Gaussian random vector, is

$$\mathbf{h}_k = (h_{k,0}, h_{k,1}, \ldots, h_{k,\beta})^t \tag{2}$$

where it is truncated to a finite length of $(\beta + 1)$ and $\mathbf{h}_k^t$ is the transpose of $\mathbf{h}_k$. The element $h_{k,i}$ is the CIR at time k due to an impulse applied at time $k - i$. It is shown in [1] that by considering the third-order approximation of (1) an autoregressive (AR) representation for the CIR can be introduced as

$$\mathbf{h}_k = A\mathbf{I}\mathbf{h}_{k-1} + B\mathbf{I}\mathbf{h}_{k-2} + C\mathbf{I}\mathbf{h}_{k-3} + D\mathbf{I}\mathbf{w}_k \tag{3}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{w}_k$ is a zero-mean white complex circularly symmetric Gaussian process with the covariance matrix defined as $E(\mathbf{w}_k\mathbf{w}_l^T) = \mathbf{Q}\delta_{kl}$, and $\mathbf{w}_l^T$ is the conjugate transpose of $\mathbf{w}_l$.

It is clear from (3) that the CIR at time k depends on its three consecutive previous values. Hence, it is possible to derive a state space model for the fading channel [10], [11]. The state of such a system is a vector composed of three consecutive channel impulse responses as

$$\mathbf{x}_k = \left(\mathbf{h}_k^t, \mathbf{h}_{k-1}^t, \mathbf{h}_{k-2}^t\right)^t. \tag{4}$$

Using (4) and (3) we can write

$$\mathbf{x}_{k+1} = \begin{bmatrix} A\mathbf{I} & B\mathbf{I} & C\mathbf{I} \\ \mathbf{I} & 0 & 0 \\ 0 & \mathbf{I} & 0 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} D\mathbf{I} \\ 0 \\ 0 \end{bmatrix} \mathbf{w}_k \tag{5}$$

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{w}_k \tag{6}$$

where $\mathbf{F}$ and $\mathbf{G}$ are $3(\beta+1)\times 3(\beta+1)$ and $3(\beta+1)\times(\beta+1)$ matrices given in (5), respectively. $\mathbf{F}$ is called the state transition matrix and $\mathbf{G}$ is the process noise coupling matrix.

Also, by defining the $1 \times (\beta+1)$ vector $\mathbf{H}_k$ as

$$\mathbf{H}_k = (a_k, a_{k-1}, a_{k-2}, \ldots, a_{k-\beta}, 0, \ldots, 0). \tag{7}$$

where a_k is the transmitted data sequence, we can write the received signal z_k (Fig. 1) as

$$z_k = \mathbf{H}_k \mathbf{x}_k + n_k. \tag{8}$$

This represents the convolution sum when $\mathbf{H}_k$ is the input to the fading channel with impulse response $\mathbf{x}_k$, and n_k is the additive white complex circularly symmetric Gaussian noise with the covariance of $E(n_k n_l^*) = N_o \delta_{kl}$.

Equations (6) and (8) describe a linear time varying system. The state of this system (4) is based on the impulse response of the channel, and an estimation method has to be employed for channel estimation. There are different estimation methods. Among them, the Kalman filter is optimum for minimizing the mean square estimation error [13]. The Kalman filter, however, is a complex algorithm, and, in practice, suboptimal methods are more advantageous due to their implementation simplicity.

To avoid the decision delay in data detection, the PSP method [6] can be employed. In this method, there is a channel estimate for every possible sequence, and to overcome the problem of uncertainty in the transmitted data $\mathbf{H}_k$ (7) a separate estimation is required for any of the possible hypothesized $\mathbf{H}_k$ vectors on the surviving paths. In this way, each estimator uses its own hypothesized data vector for $\mathbf{H}_k$ and, based on that, it gives an estimation of the channel impulse response. The estimated channel impulse response will be used to compute the branch metrics in the trellis diagram of the Viterbi algorithm. The number of required estimators is limited to the number of survivor branches (or the number of states) in the Viterbi algorithm trellis diagram. In PSP each surviving path keeps and updates its own channel estimate. This method eliminates the problem of decision delay, and in order to employ the best available information for data detection the data sequence of the shortest path is used for channel estimation along the same path.

The Simulated System: To study the various implementation alternatives, a data communication system based on the IS-136 standard is considered. The modulation is QPSK with four possible symbols $(\pm 1 \pm j)$ and a symbol rate of 25 ksymbol/s. As in the IS-136 standard, the differentially encoded data sequence is arranged into 162 symbol frames. The first 14 symbols of each frame is a training preamble sequence to help the adaptation of the channel estimator. For the shaping filter at the transmitter, we implement a finite impulse response (FIR) filter which approximates a raised cosine frequency response with an excess bandwidth of 25% (slightly different from the 35% selected in IS-136 as in [15]).

In order to keep the simulation simple we consider a two ray fading channel model as described in Fig. 2, where one ray has a fixed delay equal to one symbol period. The multiplicative coefficients of α_0 and α_1 are produced at the output of two fading filters, where the inputs are two independent zero mean complex Gaussian process with equal variances. The length of the discrete impulse response of the shaping filter is set equal to the symbol interval so that the ISI at the receiver is only due to the multipath nature of the channel. Since one ray is delayed by an amount equal to one symbol interval, the total length of the CIR is two symbol intervals, i.e., $\beta + 1 = 2$ if there is one sample per symbol interval. Therefore, there is ISI between two neighboring symbols and there are four possible states in the trellis diagram.

The LMS algorithm, RLS algorithm, or the Kalman filter can be used to estimate the channel impulse response. In the following section we will consider different algorithms and structures for VLSI implementation of the channel estimator.

III. Implementing the Estimator

To estimate the states of the system described by (6) and (8) the Kalman filter and the RLS algorithm can be employed. The following are the Kalman filter and the RLS algorithm equations.

The Kalman Filter Algorithm:

Measurement update equations:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_k + \mathbf{K}_k(z_k - \mathbf{H}_k \hat{\mathbf{x}}_k) \tag{9}$$
$$\mathbf{K}_k = \mathbf{P}_k \mathbf{H}_k^T R_k^{-1} \tag{10}$$
$$R_k = \mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T + N_o \tag{11}$$
$$\mathbf{P}_{k|k} = \mathbf{P}_k - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_k. \tag{12}$$

Time update equations:

$$\hat{\mathbf{x}}_{k+1} = \mathbf{F} \hat{\mathbf{x}}_{k|k} \tag{13}$$
$$\mathbf{P}_{k+1} = \mathbf{F} \mathbf{P}_{k|k} \mathbf{F}^T + \mathbf{G} \mathbf{Q} \mathbf{G}^T. \tag{14}$$

The RLS Algorithm:

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \mathbf{K}_k(z_k - \mathbf{H}_k \hat{\mathbf{x}}_k) \tag{15}$$
$$\mathbf{K}_k = \mathbf{P}_k \mathbf{H}_k^T R_k^{-1} \tag{16}$$
$$R_k = \mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T + \lambda \tag{17}$$
$$\mathbf{P}_{k+1} = \lambda^{-1}(\mathbf{P}_k - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_k). \tag{18}$$

The measurement updated estimate $\hat{\mathbf{x}}_{k|k}$ is the linear least-squares estimate of $\mathbf{x}_k$ given observations $\{z_0, z_1, \ldots, z_k\}$, and $\hat{\mathbf{x}}_{k+1}$ is the time updated estimate of $\mathbf{x}_k$ given observations $\{z_0, z_1, \ldots, z_k\}$. The corresponding error covariance matrices of these estimations are

$$\mathbf{P}_{k|k} = E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})^T\right] \tag{19}$$
$$\mathbf{P}_k = E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T\right] \tag{20}$$

In the RLS algorithm λ is called the forgetting factor.

The Kalman filter consists of two parts: *measurement update equations*, and *time update equations*. As shown in [16], the RLS algorithm is essentially identical to the measurement update equations of the Kalman filter. The Kalman filter can be used for channel estimation when some *a priori* information about the channel is available at the receiver (i.e., the $\mathbf{F}$ and $\mathbf{G}$ matrices). The RLS algorithm, which is a suboptimal method, does not require this *a priori* information and its computational complexity is less compared to the Kalman filter.

The main reason for the differences between theory and practice of implementing these algorithms can be found in the error analysis of the respective numerical methods. At the same precision, mathematically equivalent implementations can have different numerical stabilities, and some methods of implementation are more robust against roundoff errors. In the Kalman filter and the RLS algorithm the estimation depends on the correct computation of the error covariance matrix. In an ill-conditioned problem the solution will not be equal to the covariance matrix of the actual estimation uncertainty. There are some factors contributing to this problem including large ranges of the actual values of matrix parameters, large matrix dimensions and growing number of arithmetic operations, and poor machine precision. These factors are causes for concern and as a solution to combating with these problems, factorization methods and square-root filtering are widely employed in implementation [3], [4], [17]–[19].

A. Square-Root Filtering

Studies show that some implementations are more robust against roundoff errors and ill-conditioned problems. The so-called square-root filter implementations have generally better error propagation bounds than the conventional Kalman filter equations [20]. In the square-root forms of the Kalman filter matrices are factorized, and triangular square-roots are propagated in the recursive algorithm, to preserve the symmetry of the covariance (information) matrices in the presence of roundoff errors.

There are different factorization methods within which different techniques are used for changing the dependent variable of the recursive estimation algorithm to factors of the covariance matrix. A Cholesky factor of a symmetric nonnegative definite matrix $\mathbf{M}$ is a matrix $\mathbf{C}$ such that $\mathbf{CC}^T = \mathbf{M}$. Cholesky decomposition algorithms solve for $\mathbf{C}$ that is either upper triangular or lower triangular. The modified Cholesky decomposition algorithms solve for a diagonal factor and either a lower triangular factor $\mathbf{L}$ or an upper triangular factor $\mathbf{U}$ such that $\mathbf{M} = \mathbf{UD}_U\mathbf{U}^T = \mathbf{LD}_L\mathbf{L}^T$, where $\mathbf{D}_L$ and $\mathbf{D}_U$ are diagonal factors with nonnegative diagonal elements.

The square-root methods propagate the L–D or U–D factors of the covariance matrix rather than the covariance matrix. The propagation of square-root matrices implicitly preserves the Hermitian symmetry and nonnegative definiteness of the computed covariance matrix. The condition number $\kappa(\mathbf{P}) = [\text{eigenvalue}_{\max}(\mathbf{P})/\text{eigenvalue}_{\min}(\mathbf{P})]$ of the covariance matrix $\mathbf{P}$ can be written as $\kappa(\mathbf{P}) = \kappa(\mathbf{LDL}^T) = \kappa(\mathbf{BB}^T) = [\kappa(\mathbf{B})]^2$, where $\mathbf{B} = \mathbf{LD}^{1/2}$. Therefore, the condition number of $\mathbf{B}$ used in the square-root method is much smaller than the condition number of the $\mathbf{P}$ and this leads to improved numerical robustness of the algorithm. Moreover, in the square-root method the dynamic range of the numbers entering into computations will be reduced. Loosely speaking, we can say that the computations which involve numbers ranging between 2^{-N} to 2^{+N} will be reduced to ranges between $2^{-N/2}$ to $2^{+N/2}$. All of these will directly affect the accuracy of computer computations.

There are also other factorization methods employed for increasing the numerical stability, such as triangularization (QR decomposition) and WGS orthonormalization used for factoring matrices as products of triangular and orthonormal matrices. The block matrix factorization of a matrix expression is a general approach that uses two different factorizations to represent the two sides of an equation such as

$$\mathbf{CC}^T = \mathbf{AA}^T + \mathbf{BB}^T = [\mathbf{A} \quad \mathbf{B}] + \begin{bmatrix}\mathbf{A}^T \\ \mathbf{B}^T\end{bmatrix}. \tag{21}$$

The alternative Cholesky factor C and $[A \ B]$ can be related by orthogonal transformation [20].

B. Implementation Algorithms for Measurement Update Equations

To compute the measurement update equations of the Kalman filter for real numbers, a square-root method is proposed in [3] by Jover and Kailath. Although this algorithm is not complete for implementing the Kalman filter, in the following we will extend it to complex numbers and then we will add some procedures for computing the time update equations.

The measurement update equation for the covariance matrix can be written from (12) and (14) as

$$\mathbf{P}_{k|k} = \mathbf{P}_k - \mathbf{P}_k\mathbf{H}_k^T R_k^{-1}\mathbf{H}_k\mathbf{P}_k. \tag{22}$$

Our implementation algorithm is based on working with unit lower triangular, diagonal, unit upper triangular (LDU) factorizations of $\mathbf{P}_k$ and $\mathbf{P}_{k|k}$ and, since $\mathbf{P}_k$ and $\mathbf{P}_{k|k}$ are Hermitian symmetric, $\mathbf{U} = \mathbf{L}^T$. It can be shown that by choosing a suitable orthogonal transformation matrix Θ, where $\Theta\Theta^T = \mathbf{I}$, we can have

$$\begin{bmatrix} N_o^{1/2} & \mathbf{H}_k\mathbf{P}_k^{1/2} \\ 0 & \mathbf{P}_k^{1/2}\end{bmatrix}\Theta = \begin{bmatrix} R_k^{1/2} & 0 \\ \mathbf{P}_k\mathbf{H}_k^T R_k^{-1/2} & \mathbf{P}_{k|k}^{1/2}\end{bmatrix} \tag{23}$$

and this can be immediately verified by squaring both sides of (23). Generally, computing the triangular factor $\mathbf{P}_{k|k}^{1/2}$ requires taking arithmetic square roots, which are computationally more expensive than multiplication or division. This can be avoided by using $\mathbf{LDU}$ factorizations

$$\mathbf{P}_k = \mathbf{LDL}^T \quad \text{and} \quad \mathbf{P}_{k|k} = \mathbf{L}_p\mathbf{D}_p\mathbf{L}_p^T. \tag{24}$$

Using (24) in (23) and dropping all time-index subscripts yields

$$\begin{bmatrix}1 & \mathbf{HL} \\ 0 & \mathbf{L}\end{bmatrix}\begin{bmatrix}N_o & 0 \\ 0 & \mathbf{D}\end{bmatrix}^{1/2}\Theta = \begin{bmatrix}1 & 0 \\ \mathbf{K} & \mathbf{L}_p\end{bmatrix}\begin{bmatrix}R & 0 \\ 0 & \mathbf{D}_p\end{bmatrix}^{1/2}. \tag{25}$$

Therefore, to compute the measurement update equation for the covariance matrix we start with the left-hand side of (25) and, by applying an orthogonal transformation, the left side of (25) can be converted to the $\mathbf{LD}$ form on the right side. This is possible by application of the fast givens transformation, as we have modified from [3]. We can express (25) in terms of the components of the matrices, as shown below, where the

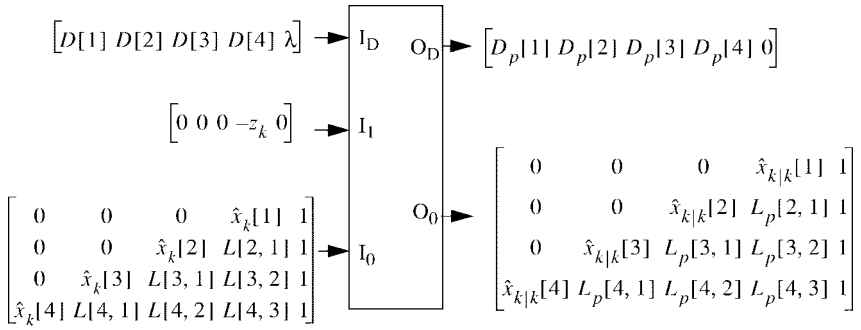


Fig. 3. The data structure for the inputs and outputs of the Jover–Kailath algorithm [3].

size of $\mathbf{L}$ and $\mathbf{D}$ is considered to be $n \times n$

$$\begin{bmatrix} 1 & p_1 & p_2 & \cdots & p_n \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & L_{2,1} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & L_{n,1} & L_{n,2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} N_o & 0 & 0 & \cdots & 0 \\ 0 & D_1 & 0 & \cdots & 0 \\ 0 & 0 & D_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & D_n \end{bmatrix}^{1/2} \Theta$$
$$= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ K_1 & 1 & 0 & \cdots & 0 \\ K_2 & \bar{L}_{2,1} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ K_n & \bar{L}_{n,1} & \bar{L}_{n,2} & \cdots & 1 \end{bmatrix} \cdot \begin{bmatrix} R & 0 & 0 & \cdots & 0 \\ 0 & \bar{D}_1 & 0 & \cdots & 0 \\ 0 & 0 & \bar{D}_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & \bar{D}_n \end{bmatrix}^{1/2} . \quad (26)$$

The goal is to obtain a lower triangular and a diagonal matrix on the right hand side of (26) by applying an orthogonal transformation Θ and setting the p_j components to zero. We can perform this by considering a series of orthogonal transformations as

$$\Theta = \Theta_{1,n}\Theta_{1,n-1}\cdots\Theta_{1,2}\Theta_{1,1} \quad (27)$$

where the matrix $\Theta_{1,j}$ is a $(n+1)\times(n+1)$ identity matrix with only four elements changed on $(1,1)$, $(1,j+1)$, $(j+1,1)$, and $(j+1,j+1)$ positions. Therefore, the $\Theta_{1,j}$ transformation only affects the first and $(j+1)$th columns of the right hand side matrices in (26). If we just consider these two columns the pair $[1 \;\; p_j]$ will be on top and $\Theta_{1,j}$ can be found so that the vector $[1 \;\; p_j]$ will be transformed to $[1 \;\; 0]$ in order to triangularize the matrix.

If we only consider the top pair of the two columns, for instance $[1 \;\; p_2]$ for first and third columns, with a proper orthogonal transformation matrix we will have

$$[1 \quad p_2]\begin{bmatrix} d_{p1} & 0 \\ 0 & d_{p2} \end{bmatrix}^{1/2} \Theta_{1,2} = [1 \quad 0]\begin{bmatrix} d_{q1} & 0 \\ 0 & d_{q2} \end{bmatrix}^{1/2} \quad (28)$$

where d_{p1} and d_{p2} are the components in (26) that affect $[1 \;\; p_2]$ and d_{q1} and d_{q2} have to be found based on the applied orthogonal transformation.

It is shown in [3] and [4] that, by using the parameters in the left hand side of (28), we can choose

$$d_{q1} = d_{p1} + |p_2|^2 d_{p2} \quad (29)$$
$$d_{q2} = \frac{d_{p1}d_{p2}}{d_{q1}} \quad (30)$$

to obtain the proper orthogonal transformation. The $\Theta_{1,2}$ transformation is found to rotate the vector $[1 \;\; p_2]$ to lie along the vector [1 0], keeping the equality of weighted norms [3]. It is also necessary to apply this transformation to other pairs of the first and third columns and find the new transforms of these vectors. By applying the transformation to an arbitrary vector $[\bar{p}_1 \;\; \bar{p}_2]$ to lie along $[\bar{q}_1 \;\; \bar{q}_2]$, we obtain

$$[\bar{p}_1 \quad \bar{p}_2]\begin{bmatrix} d_{p1} & 0 \\ 0 & d_{p2} \end{bmatrix}^{1/2} \Theta_{1,2} = [\bar{q}_1 \quad \bar{q}_2]\begin{bmatrix} d_{q1} & 0 \\ 0 & d_{q2} \end{bmatrix}^{1/2} \quad (31)$$

where

$$\bar{q}_2 = -p_2\bar{p}_1 + \bar{p}_2 \quad (32)$$
$$\bar{q}_1 = \bar{p}_1 + \left(p_2^* \frac{d_{p2}}{d_{q1}}\right)\bar{q}_2. \quad (33)$$

The complex conjugate of p_2 is denoted by p_2^*.

The algorithm to implement the above triangularization is to consider the first and the other columns of the matrix in the right side of (26) one-by-one and apply all $\Theta_{1,j}$ transforms to obtain a lower triangular and a diagonal matrix. The algorithm presented in Appendix A is based on the above method to compute the measurement update equations of the Kalman filter.

The covariance update algorithm that we explained in this section computes (22) or equivalently (12)–(14), however, we need to compute (11) to update the state estimates as well. It can be shown [3] that with an appropriate arrangement for the input data structure, the covariance update algorithm can also be used for updating the state estimates. Fig. 3 shows the arrangement for the inputs and outputs of the algorithm given in Appendix A. The components of the state estimate vector are fed to the algorithm along with the components of $\mathbf{L}$, and the algorithm yields the update, as in Fig. 3. A parallel architecture is proposed in [3] to implement this algorithm for real numbers. This architecture, however, can also be used for complex numbers, with some modifications [4]. This will be useful for implementing both the measurement update equations of the Kalman filter and the RLS algorithm.

Note that the matrix $\mathbf{F}$ is not used in the equations of the RLS algorithm [i.e., (17)–(20)]. It is also possible to assume that the matrix $\mathbf{F}$ is equal to the identity matrix for the RLS algorithm. In this case, from (4)–(6) we can deduce that the CIR of the system at time $k+1$ is only obtained from the CIR of the system at time k. This means that we are not considering the AR representation of (3) for RLS and hence the state vector of the system, as defined in (4), only consists of the CIR at the present time $\mathbf{h}_k$. This will reduce the dimensionality of the state vector and covariance matrices in the RLS algorithm by a factor of three, which is the order of the AR representation in this case, compared to the Kalman filter. Obviously, a smaller matrix size produces less complexity and more robustness against roundoff errors. In [4] a variety of parallel and pipelined structures are proposed for the realization of the RLS algorithm.

C. Implementation Algorithms for Time Update Equations

The Jover–Kailath algorithm can be used to implement the measurement update equations of the Kalman filter. Since the RLS algorithm is basically the same as the measurement update equations of the Kalman filter this method could also be used for implementing the RLS algorithm. For implementation of the Kalman filter, however, we need to calculate (15) and (16) and another algorithm is required to perform this part. Since in the Jover–Kailath algorithm, instead of $\mathbf{P}_{k|k}$, its factors $\mathbf{L}_p$ and $\mathbf{D}_p$ are computed as in (24), we need to employ an algorithm that uses these factors. The propagation of $\mathbf{LD}$ factors implicitly preserves symmetry and nonnegative definiteness of the computed covariance matrix. In the following we will present and compare three different methods for implementing the time update equations.

1) Direct Computation of the Covariance Matrix: One simple approach to carry out the computation in (16) is the direct computation of

$$\mathbf{P}_{k+1} = \mathbf{F}\mathbf{L}_p\mathbf{D}_p\mathbf{L}_p^T\mathbf{F}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T. \tag{34}$$

When the noise process is time invariant $\mathbf{G}\mathbf{Q}\mathbf{G}^T$ needs to be computed only once and (34) requires four matrix multiplications and one addition. Then, since we need to propagate the $\mathbf{LD}$ factors of $\mathbf{P}_{k+1}$, it can be factorized in the $\mathbf{LDL}^T$ form using the factorization algorithms of [22]. The direct method is not very robust against roundoff errors, and we will use it for comparison to show the advantages of using square-root techniques in the implementation of time update equations. The following methods are based on the direct computation of $\mathbf{LD}$ factors for the covariance matrix and result in better numerical stability.

2) The WGS Orthogonalization Algorithm: In this method the covariance update equation implementation is based on a block matrix factorization. Equation (16) can be rewritten in the following matrix form:

$$\mathbf{P}_{k+1} = \begin{bmatrix}\mathbf{F}\mathbf{P}_{k|k}^{1/2} & \mathbf{G}\mathbf{Q}^{1/2}\end{bmatrix}\begin{bmatrix}\mathbf{P}_{k|k}^{T/2}\mathbf{F}^T \\ \mathbf{Q}^{T/2}\mathbf{G}^T\end{bmatrix}. \tag{35}$$

Again, if we use the $\mathbf{LDU}$ factorization for the covariance matrix as in (24) and indicate the diagonal matrix of $\mathbf{Q}$ with $\mathbf{D}_Q$, after dropping all time index subscripts, (35) becomes

$$\mathbf{P} = \mathbf{L}\mathbf{D}\mathbf{L}^T = \begin{bmatrix}\mathbf{F}\mathbf{L}_p & \mathbf{G}\end{bmatrix}\begin{bmatrix}\mathbf{D}_p & 0 \\ 0 & \mathbf{D}_q\end{bmatrix}\begin{bmatrix}\mathbf{L}_p^T\mathbf{F}^T \\ \mathbf{G}^T\end{bmatrix}. \tag{36}$$

$\mathbf{L}_p$ and $\mathbf{D}_p$ in the right side of the equation are known from the measurement update procedure and $\mathbf{L}$ and $\mathbf{D}$ have to be computed.

WGS orthogonalization [20] can be employed here. It is an algorithm for finding a set of n mutually orthogonal vectors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \ldots, \mathbf{b}_n$ that are a linear combination of a set of n linearly independent vectors $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \ldots, \mathbf{a}_n$. For a given matrix $\mathbf{A}$, the Gram–Schmidt algorithm defines a unit upper triangular matrix $\mathbf{U}$ such that

$$\mathbf{A} = \mathbf{B}\mathbf{U}, \quad \text{or}$$

$$\mathbf{A} = \begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \cdots & \mathbf{a}_n\end{bmatrix} = \mathbf{B}\mathbf{U} \tag{37}$$

$$\mathbf{A} = \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 & \cdots & \mathbf{b}_n\end{bmatrix}\begin{bmatrix}1 & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & 1 & u_{23} & \cdots & u_{2n} \\ 0 & 0 & 1 & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1\end{bmatrix}. \tag{38}$$

The Gram–Schmidt orthogonalization is called weighted if the vectors $\mathbf{b}_i$ are orthogonal with respect to the weights $w_1, w_2, w_3, \ldots, w_n$. The vectors $\mathbf{x}$ and $\mathbf{y}$ are said to be orthogonal with respect to the weights w_i if

$$\sum_{i=1}^{n}\mathbf{x}_i w_i \mathbf{y}_i = \mathbf{x}^T\mathbf{D}_w\mathbf{y} = 0 \tag{39}$$

where

$$\mathbf{D}_w = \mathrm{diag}_{1\le i\le n}\{w_i\}. \tag{40}$$

Hence the weighted norms of the mutually orthogonal vectors $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \ldots, \mathbf{b}_n$ appear as the diagonal elements of the diagonal matrix

$$\mathbf{D} = \mathbf{B}^T\mathbf{D}_w\mathbf{B}. \tag{41}$$

To apply the WGS method let

$$\mathbf{A} = \begin{bmatrix}\mathbf{L}_p^T\mathbf{F}^T \\ \mathbf{G}^T\end{bmatrix} \tag{42}$$

$$\mathbf{D}_w = \begin{bmatrix}\mathbf{D}_p & 0 \\ 0 & \mathbf{D}_q\end{bmatrix} \tag{43}$$

then, from (36), we have

$$\mathbf{P}_{k+1} = \mathbf{A}^T\mathbf{D}_w\mathbf{A}. \tag{44}$$

The Gram–Schmidt algorithm will produce a unit upper triangular matrix $\mathbf{U}$ (37) and a diagonal matrix $\mathbf{D}$ (41) such that

$$\mathbf{P}_{k+1} = \mathbf{A}^T\mathbf{D}_w\mathbf{A} \tag{45}$$

$$= (\mathbf{B}\mathbf{U})^T\mathbf{D}_w(\mathbf{B}\mathbf{U}) \tag{46}$$

$$= \mathbf{U}^T\mathbf{B}^T\mathbf{D}_w\mathbf{B}\mathbf{U} \tag{47}$$

and from (41)

$$\mathbf{P}_{k+1} = \mathbf{U}^T\mathbf{D}\mathbf{U} = \mathbf{L}\mathbf{D}\mathbf{L}^T. \tag{48}$$

Therefore, the inputs of the WGS algorithm are the information on the right side of (36) in the form of $\mathbf{A}$ and $\mathbf{D}_w$, and the output of the algorithm is the lower triangular matrix $\mathbf{U}^T = \mathbf{L}$ and the diagonal matrix $\mathbf{D}$.

Before applying the above method, a matrix multiplication is required to compute $\mathbf{F}\mathbf{L}_p$ [see (36)]. Also, for computing the temporal update of state estimations in (15), the $\hat{\mathbf{x}}_{k|k}$ vector obtained at the output of the measurement update procedure must be premultiplied by $\mathbf{F}$. Both of these multiplications can be carried out together by appending the vector $\hat{\mathbf{x}}_{k|k}$ to the matrix $\mathbf{L}$ and multiplying the combination by $\mathbf{F}$. If the multiplication is carried out using array processors, in this way, the same structure can perform both of these multiplications, without any change in the hardware.

3) The LDC Algorithm: In the WGS algorithm, as we can see from (16) and (35) the computation of $\mathbf{G}\mathbf{Q}\mathbf{G}^T$ will be repeated in every iteration. When the process noise is time invariant and the matrix $\mathbf{Q}$ is not changing over time, there is no need to recompute this term in every iteration. This idea leads to the introduction of a less complex algorithm.

The LDC algorithm is used in linear algebra [21], [22] to update the $\mathbf{LD}$ factorization of matrix $\mathbf{A}$ to the $\mathbf{LD}$ factorization of $\mathbf{A}+\mathbf{v}\mathbf{v}^T$, where $\mathbf{A}$ is symmetric and positive definite, and $\mathbf{v}$ is an arbitrary vector with appropriate size. When the process noise covariance is time invariant, this algorithm can be used to implement the time update equations of the Kalman filter and it is shown [5] to have substantial computational saving when compared to the WGS algorithm. To implement this method let

$$\mathbf{P}_{k|k} = \sum_{i=1}^{n} d_i\mathbf{L}_i\mathbf{L}_i^T. \tag{49}$$

The covariance prediction in (16) can be written using the $\mathbf{L}$ and $\mathbf{D}$ factors of $\mathbf{P}$ and $\mathbf{G}\mathbf{Q}\mathbf{G}^T$ as

$$\mathbf{P}_{k+1} = \sum_{i=1}^{n} d_i(\mathbf{F}\mathbf{L}_i)(\mathbf{F}\mathbf{L}_i)^T + \sum_{i=1}^{p} d_{qi}\mathbf{L}_{qi}\mathbf{L}_{qi}^T. \tag{50}$$

The LDC algorithm can be employed to compute the $\mathbf{L}\mathbf{D}\mathbf{L}^T$ factorization of the sum

$$\sum_i d_{qi}\mathbf{L}_{qi}\mathbf{L}_{qi}^T + s\mathbf{v}\mathbf{v}^T. \tag{51}$$

In (50) the LDC algorithm can be applied n times, and each time, one of the components of the first sum is considered to be the $s\mathbf{v}\mathbf{v}^T$ vector. The result will be a $\mathbf{L}\mathbf{D}\mathbf{L}^T$ factorization for $\mathbf{P}_{k+1}$.

The complete algorithm to implement the LDC method is given in Appendix B. This algorithm requires $O(6n^3 + 16n^2 - 2n)$ multiply–add operations, while the WGS method requires $O(10.7n^3 + 11.8n^2 - 2.5n)$ operations. Thus, the LDC algorithm requires less computation compared to the WGS method since the process noise is time invariant and the term $\mathbf{G}\mathbf{Q}\mathbf{G}^T$ needs to be factorized only once during the initialization stage, and these factors will be used repeatedly during each filter iteration.

The performance of the above three methods are compared in Section V. The direct method is not a square-root method and is very sensitive to numerical errors. The performance of the WGS and LDC algorithms are very close in terms of numerical accuracy, while the LDC algorithm requires fewer computations. In the next section we will introduce a systolic structure for the implementation of the LDC algorithm.

D. A Systolic VLSI Structure for the LDC Algorithm

To employ the Kalman filter as a channel estimator in a mobile communication receiver, it is important to carry out all of the required computations in real time. The Kalman estimator is computationally intensive and, to speed up the estimation process, parallel VLSI structures have to be sought for implementation. It is also imperative to utilize the inherent parallelism of the proposed algorithm to be mapped on the parallel VLSI structure.

The LDC algorithm is more appropriate than other methods for implementation of the time update measurement equations of the Kalman channel estimator. The fading channel model, and hence the process noise, can be reasonably assumed to be time invariant in a short period of time (e.g., one frame interval). This allows us to employ the LDC algorithm, which results in a considerable saving in computations compared to the WGS method.

A systolic VLSI structure is proposed in Fig. 4 for implementation of the LDC algorithm. This structure is used for implementing (51) and is based on the *ldltup* function of Appendix B. Two types of processors are employed and the function of each is described in the figure. The size of the state vector is assumed to be $n = 6$ in this example and the size of the $\mathbf{L}$ matrix is 6×6. The $\mathbf{LD}$ factors get updated in place and there is no need to transfer these values during the computation. The inputs to this structure are different columns of the matrix $\mathbf{F}\mathbf{L}$ in the form of the $\mathbf{v}$ vectors scaled by the elements of $\mathbf{D}$. The function *ldltup* is called n times for each column of $\mathbf{F}\mathbf{L}$, and this can be carried out by applying n input vectors to the systolic structure. Once the computation in one row is finished, the next column of $\mathbf{F}\mathbf{L}$ can be applied as the new input vector. This allows for pipelining and results in higher processor utilization and, hence, higher speed. If the computation in each row takes one time step, for the time update of the covariance matrix, $2n - 1$ time steps will be required.

It is also possible to map the above algorithm to a smaller number of processors. Mapping can be performed along different directions, as shown in Fig. 5. By using different mapping vectors we obtain structures with different performance and capabilities. The function of the processing units and the details of communication between units is not shown in the figure. Links only represent the direction of data transfer between processors. Table I summarizes and compares the features for different mapped structures when n is the size of the state vector. It is possible to employ the mapping along d_3 in a pipeline mode, but the other mappings cannot be pipelined.

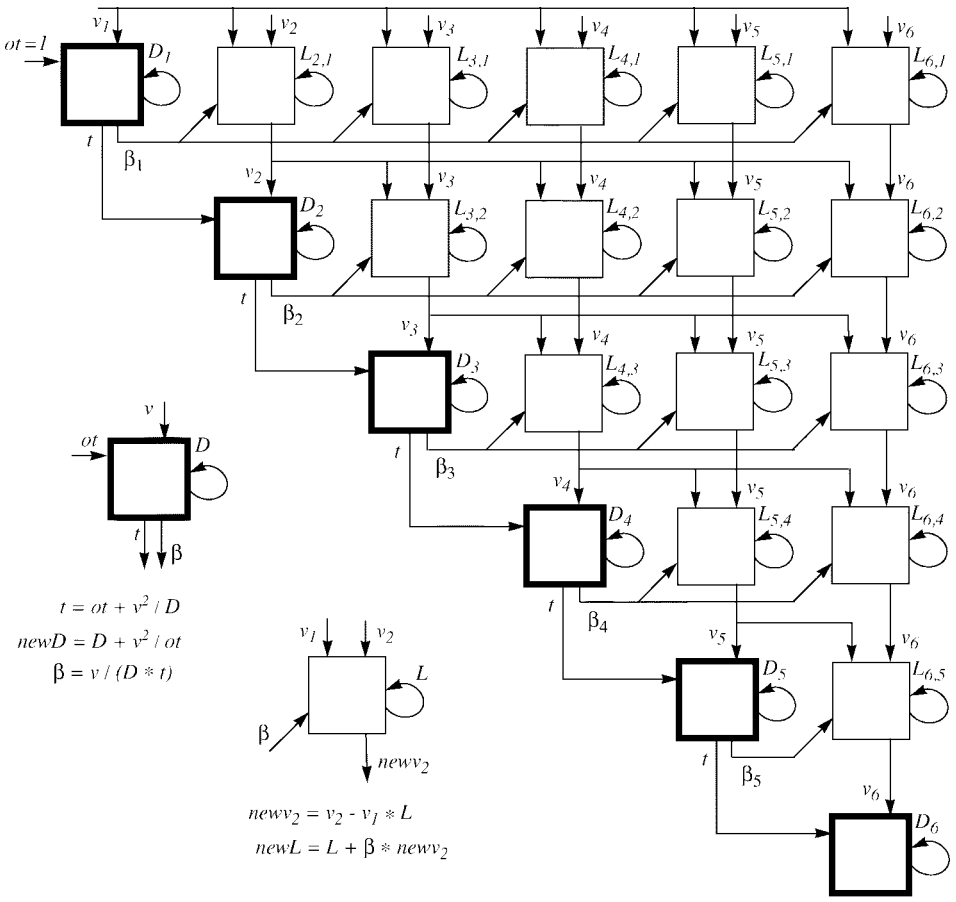


Fig. 4. A systolic VLSI structure for implementation of the LDC algorithm.

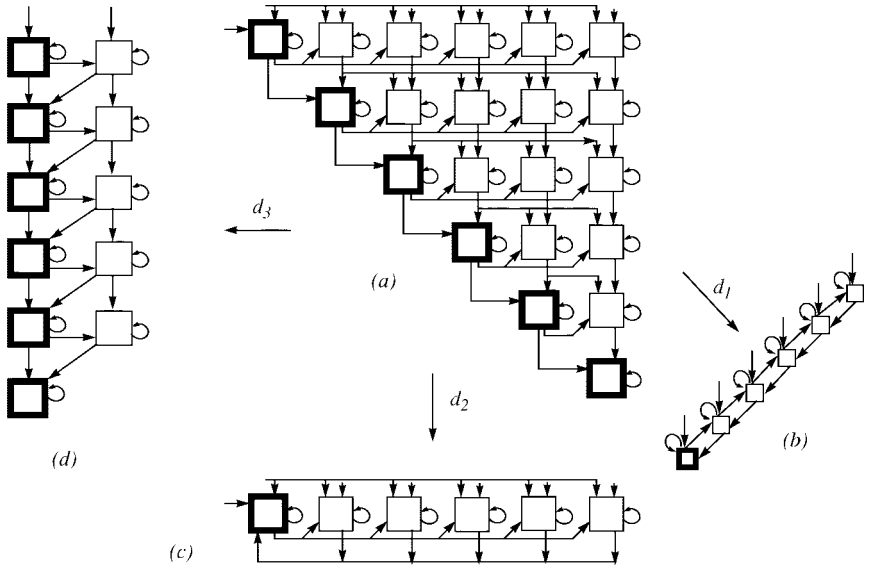


Fig. 5. Mapping the systolic structure of the correction algorithm to a smaller number of processors.

There are two kinds of processors used in each structure. If we assume that the maximum time required for the computations in each processor is T, then the total latency for one application of the correction algorithm will be $11T$. This should be obvious from the data dependency in the two-dimensional structure of Fig. 5(a). By using the structure of Fig. 5(a) in pipeline mode, the throughput, or the time interval between any two applications of the algorithm, will reduce to T. In this case processor utilization will be 100%. It can be shown that the structure of Fig. 5(d) can also be pipelined. There are eleven processors used here and a throughput of $5T$ is attainable in this case. The other two mappings use six processors and cannot be pipelined, resulting in a latency and throughput of $11T$.

TABLE I
COMPARISON BETWEEN DIFFERENT MAPPINGS OF THE SYSTOLIC STRUCTURE (n = SIZE OF THE STATE VECTOR.

	Two dimensional array	Mapping along d_1	Mapping along d_2	Mapping along d_3
Number of Processors	$n(n+1)/2$	n	n	$2n-1$
Pipelineable	Yes	No	No	Yes
Latency	$(2n-1)T$	$(2n-1)T$	$(2n-1)T$	$(2n-1)T$
Throughput	T	$(2n-1)T$	$(2n-1)T$	$(n-1)T$
Processor Utilization	1.0	$\frac{n+1}{4n-2}$	$\frac{n+1}{4n-2}$	$\frac{n+1}{4n-2}$

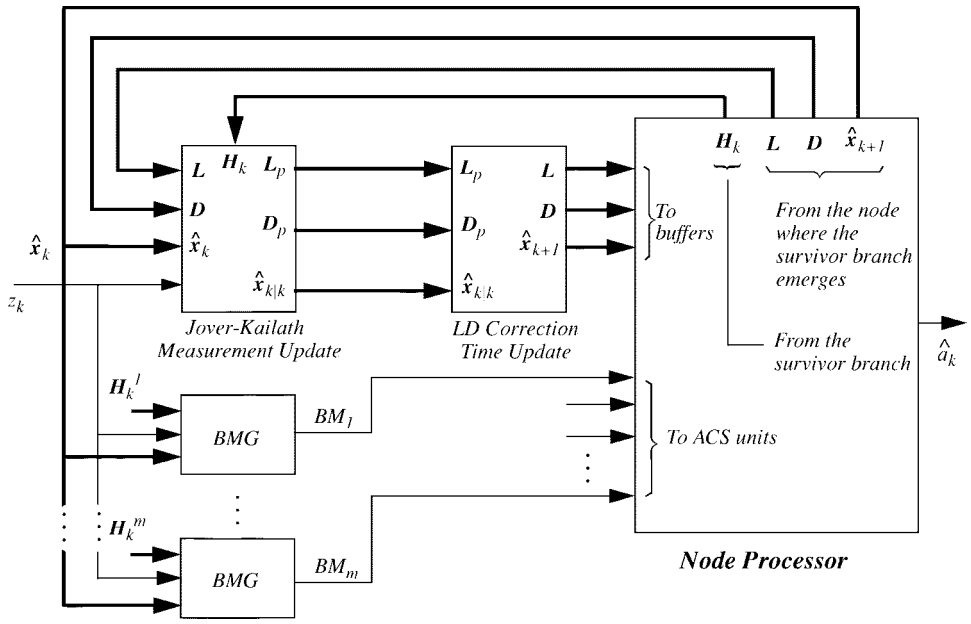


Fig. 6. The required PSP computation for one node (state) of the trellis diagram. First, the branch metrics are generated and the survivor path is found, then the estimator will receive the survivor path information to estimate the channel.

IV. STRUCTURES FOR VITERBI DETECTION AND CHANNEL ESTIMATION

In the Viterbi equalization of a mobile fading channel, a channel estimator has to be used in combination with the Viterbi detector. The impulse response of the channel has to be estimated by the estimation algorithm, and this impulse response is used to generate the branch metrics in the Viterbi algorithm. In the PSP equalization method there is one channel estimation associated with each state of the Viterbi trellis. All of the channel estimations associated with different states, depend on the received signal and the hypothesized data sequence in the Viterbi algorithm and can be performed in parallel.

Fig. 6 shows how to employ the algorithms introduced in Section III to estimate the CIR and calculate the branch metrics for one node of the trellis using the PSP method. The estimator consists of the Jover–Kailath and LDC algorithms. The inputs and outputs are shown based on the parameters introduced in the above algorithms. There is one branch metric generator (BMG) for each of the m branches leading to the considered node. Each branch has its own hypothesized data sequence $\mathbf{H}_k^i$ corresponding to the state transition on that branch. The node processor unit receives all the branch metrics and determines the survivor path using the add–compare–select (ACS) operations. After the survivor path is known, the appropriate values for $\mathbf{H}_k$, $\mathbf{L}$, $\mathbf{D}$, and $\hat{\mathbf{x}}_k$ have to be sent to the estimator to update the channel estimates. For $\mathbf{H}_k$, the hypothesized data sequence of the survivor path has to be chosen, and for $\mathbf{L}$, $\mathbf{D}$, and $\hat{\mathbf{x}}_k$, the output of the channel estimator on the node from which the survivor path is originating has to be considered.

The per-branch processing (PBP) method of [23] is a generalized form for PSP and can be used when there is more than one sample and more than one estimation per symbol interval. In PBP there is one channel estimation on each branch of the Viterbi trellis to generate the branch metrics. In a parallel implementation, several estimators can be employed to generate the branch metrics. The number of estimators required is equal to the number of states in PSP and equal to the number of branches in PBP. In this section we propose parallel structures for implementation of PSP and PBP.

In a parallel structure for the Viterbi algorithm, different estimators must communicate with the Viterbi processing units so that independent computations can be handled in parallel and also pipelined, if possible. To obtain a systolic structure for the Viterbi algorithm we realize that it can be formulated as a form of matrix multiplication. If the node value at $t = k$ is shown by $\Gamma^{[k]}$ and the branch metric from node i to node j is b_{ij}, then to obtain the node values at $t = k + 1$ the Viterbi algorithm computes

$$\begin{aligned}&[\Gamma_1 \quad \Gamma_2 \quad \Gamma_3 \quad \Gamma_4]^{[k+1]} \\ &\quad = [\Gamma_1 \quad \Gamma_2 \quad \Gamma_3 \quad \Gamma_4]^{[k]} * \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{14} \\ \vdots & \vdots & \ldots & \vdots \\ b_{41} & b_{42} & \cdots & b_{44} \end{bmatrix}. \end{aligned} \quad (52)$$

The operation $*$ is not an ordinary multiplication and the elements of the left side vector can be written as

$$\Gamma_j^{[k+1]} = \left(\Gamma_1^{[k]} + b_{1j}\right) \otimes \left(\Gamma_2^{[k]} + b_{2j}\right) \otimes \cdots \otimes \left(\Gamma_4^{[k]} + b_{4j}\right). \quad (53)$$

The $+$ operator is conventional addition and the operation $\otimes$ denotes taking minimum. Hence, (53) becomes

$$\Gamma_j^{[k+1]} = \mathrm{Min}_{1 \le i \le 4}\left(\Gamma_i^{[k]} + b_{ij}\right). \quad (54)$$

Therefore, it is possible to consider the systolic architectures for the vector-matrix multiplications proposed in the literature to implement the Viterbi algorithm. The following structures are based on two different systolic designs proposed for matrix-vector multiplication in [24].

The parallel structure of Fig. 8 is proposed for the joint implementation of the estimator and the Viterbi algorithm. This structure is performing *parallel state-computations* since the computations for all of the states are performed in parallel by considering the incoming branches sequentially. There is one channel estimation on every branch of the trellis and hence the equalization is performed in a PBP fashion. The buffer on top contains four sets of data for different trellis nodes. X is the estimated state, L and D are covariance factors, Γ is the node value in the Viterbi algorithm, and P contains the path information (survivor sequence) obtained in the Viterbi algorithm. All of the data sets are propagated to the four estimators sequentially. Each estimator also receives the hypothesized transmitted data H, related to the branches that end in the same node, and the received signal z_k. In four subsequent pipeline stages the channel estimates and the branch metrics are computed. Each estimator computes the values related to the incoming branches to one node. The output of the BMG is passed to the ACS unit. The ACS unit adds the original node value and the computed branch metric, and selects the smallest obtained value. This determines the survivor path at the end of four pipeline stages. The information of each surviving path, including the states of the channel estimator are passed to the top buffer to be used at the next symbol interval. The new values will be written into memory after all of the states are updated and all of the computations for the current symbol interval are finished. After processing a number of received symbols there will be

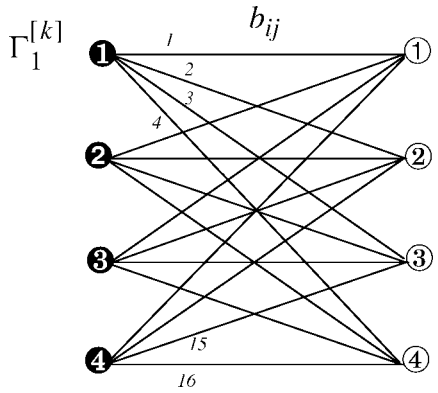


Fig. 7. The trellis diagram for the Viterbi algorithm.

an agreement on the shortest path up to a certain point which depends on the depth of the algorithm and SNR. This part of the path information can be written into the survivor path memory, and it determines the detected sequence.

In the PSP method the number of channel estimations is reduced to the number of states and, in the above example (Fig. 7), only four estimations are required in each symbol interval. In this case, for all of the incoming branches to a node, first the survivor branch will be determined and then only the channel estimation associated with the survivor branch will be carried out. The parallel structure of Fig. 8 can be modified for the implementation of PSP. In any of the four parallel branches, the estimator has to be moved after the BMG and ACS units. First, the ACS unit determines the survivor branch and then the estimator computes the CIR based on the information corresponding to that branch.

Fig. 9 is another approach to the above problem. It is also derived from one of the systolic designs for matrix-vector multiplication in [24]. This structure is performing *sequential state-computations* since the computations for all of the states are performed sequentially by considering the incoming branches in parallel. The computations associated with all incoming branches to a node will be done simultaneously in four different estimators. After all four branch metrics are added to the corresponding node values, the compare–select operations will be performed in a tree structure to find the minimum value. The updated states and covariance factors corresponding to the survivor branch will be directed to the buffer on top. Since, in this structure, the updated states and new values for the nodes are computed in a sequential way, they need to be buffered until all of the new state values are computed and old values can be overwritten by the new values. This might be done by doubling the size of the top buffer. When different states agree on a common ancestral path, the information related to that path can be dumped into the survivor path memory.

The sequential state-computation structure can be simplified significantly for the PSP method. As described before, for PSP there is only one channel estimation for the survivor path and, hence, the estimator has to be moved after the add–compare–select operations. In the structure of Fig. 9, four estimators will be removed and one estimator will be used after the final compare–select unit. The channel estimator usually

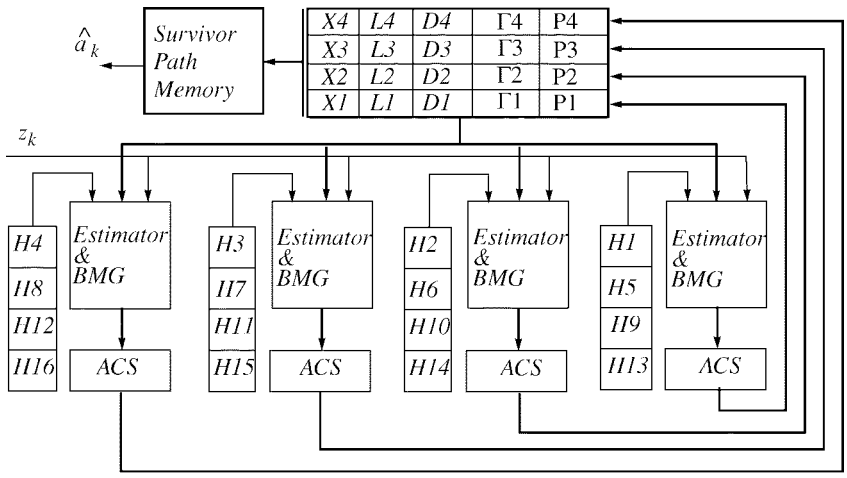


Fig. 8. A parallel architecture for joint data detection and channel estimation (parallel state-computation structure).

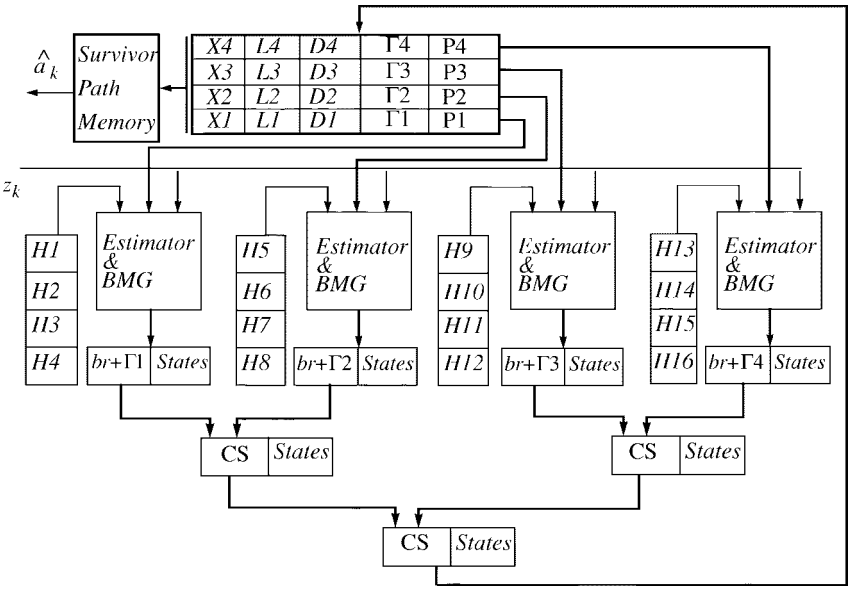


Fig. 9. A parallel architecture for joint data detection and channel estimation (sequential state-computation structure).

requires a considerable amount of computation power and area in a VLSI design. Therefore, reducing the number of estimators to only one is an effective step in the simplification of the VLSI structure. If the structure of the channel estimator is pipelineable, it will speed up the necessary computations for all the states.

V. Hardware Implementation of the Estimator

The estimation algorithms of Section III have to be realized with digital hardware, where state values and coefficients are stored in registers with a finite number of bits. An important issue in the implementation of a filtering algorithm is to consider the problems that arise in dealing with floating-point computation and finite wordlength. The time consumed on the computations and the area used in the VLSI implementation of an algorithm are proportional to the wordlength used in the computations for addition and proportional to its square for multiplication. Therefore, it is always important to use as few a number of bits in the wordlength of the computations as possible.

Some estimation algorithms, like the Kalman filter, are more sensitive to roundoff errors and require a larger number of bits per computation word compared to other estimators. The finite-wordlength effects on the design of the Kalman filter have been addressed somewhat in the literature [25], [26], however, in these analysis the estimation accuracy has been the main concern of the authors. Here, we study the performance of the estimator in a Rayleigh-fading-channel-based communication system and, specifically, in the context of a Viterbi-based receiver. Hence, we will have to note the overall effect of roundoff noise and estimation accuracy on the BER performance in this receiver.

In an adaptive MLSD receiver the CIR is estimated along the surviving paths associated with each state of the trellis. The quality of the channel estimation method has a strong impact on the overall BER performance of the receiver. Particularly, in fast fading conditions, only more advanced

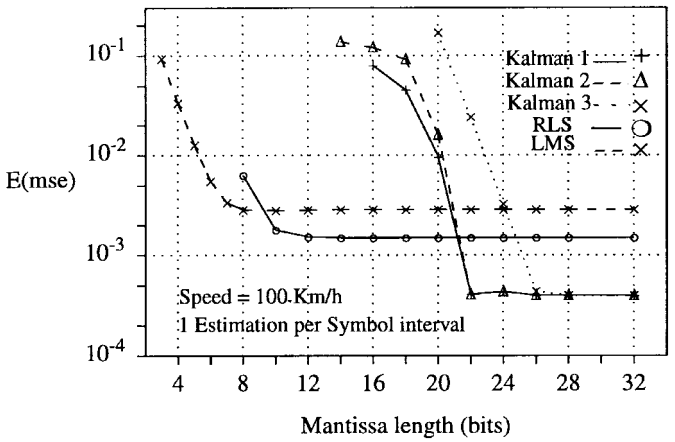


Fig. 10. The effects of changing the wordlength on estimation methods ($E_b/N_o = 15$ dB). Kalman 1 represents the WGS method, Kalman 2 is for the correction method, and Kalman 3 is the direct method. PSP method is employed for detection.

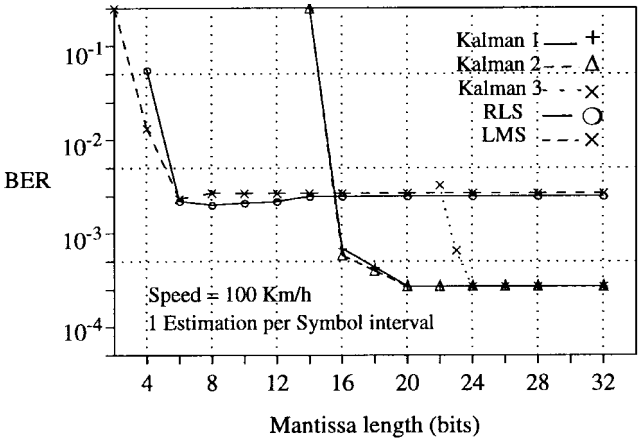


Fig. 11. The effects of changing the wordlength on estimation methods ($E_b/N_o = 15$ dB). See Fig. 10 for details.

and more accurate channel estimators can provide reasonable receiver performances. The estimator precision depends on the employed algorithm, the wordlength, and roundoff noise which is inevitable in the hardware implementation. We will consider both the estimation accuracy and BER performance by studying the simulation results.

In Fig. 10 the expectation of the mean square error (MSE) in estimation of the impulse response of a Rayleigh fading channel is plotted versus the mantissa wordlength in the floating point operations. The simulated system is as described in Section II (i.e., two-ray channel with the third-order AR model). Three different implementation methods of Section III-C for the Kalman estimator are considered, along with the RLS and LMS algorithms, when E_b/N_o is 15 dB (E_b is energy per-transmitted-bit). For the measurement update of the Kalman filter, and also for the RLS estimator, the square-root method of Section III-B has been used. The step size in LMS and the forgetting factor in RLS are chosen to yield the best MSE. The initial values for the states are chosen randomly, also for $\mathbf{L}$ and $\mathbf{D}$ we choose the identity matrix as the initial value. As is clear from Fig. 10, with the Kalman filter the minimum achievable MSE is much lower than that of LMS and RLS algorithms. However, a larger number of bits is required for the Kalman filter. Direct implementation of the Kalman filter requires at least 26 bits per mantissa, while two other methods require 22 bits. The minimum number of bits per mantissa required for channel estimation with the RLS and LMS algorithms are 12 and 8, respectively.

The effect of reducing the number of bits on the overall BER performance is shown in Fig. 11. Joint data detection and channel estimation is performed using the PSP method. Using the Kalman channel estimator leads to a very good BER performance. The BER performance of the Kalman filter is about 10 dB better compared to the RLS and LMS estimators, while it requires a longer wordlength. The required mantissa length, obtained from Fig. 11, is less than what we would expect by observing the MSE of the estimator. Also, we can see that using the WGS method or the LDC algorithm for the time update equations of the Kalman filter result in the same BER performance, while the latter is cheaper and has less computation involved. Both of these methods are much more efficient compared to the direct method, which uses a nonsquare-root algorithm for the time update equations.

```
function [Xk,Lp,Dp,Sigm] = MU(Zk,Hk,Xk,L,D,No,size)
N=size+1;       Beta(1)=0;
Delta(N)=No;
b=Hk*L;
for i= N-1:-1:1,
      Delta(i)=Delta(i+1)+b(i)*b(i)'*D(i);
      Beta(i+1)=b(i)'*(D(i)/Delta(i));
      Dp(i)=Delta(i+1)*(D(i)/Delta(i));
      a(i)=Beta(i+1);
end
% adding one element to the vector [b],
bn=[Hk*Xk-Zk, b];
for i=1:N-1;
      Lin(i,1)=Xk(i);
      Lout(i,1)=0;
      for j=1:N-1;
          Lin(i,j+1)=L(i,j);
          Lout(i,j+1)=0;
      end
      Lout(i,i+1)=1;
end
for i=1:N-1,
      for j=0:N-1-i,
          Lout(i+j,j+1)=Lin(i+j,j+1)-bn(j+1)*a(j+i);
          a(j+i)=a(j+i)+Beta(j+1)*Lout(i+j, j+1);
      end
end
for i=1:N-1,
      Xk(i)=Lout(i,1);
      for j=1:N-1,
          Lp(i,j)=Lout(i,j+1);
      end
end
```

Fig. 12. The Jover–Kailath algorithm for the measurement update equations.

```
function [L,D]=Correction(FL,Dp,GQGL,GQGD)
%GQGL and GQGD are the initial L-D factors of GQGt
L=GQGL; D=GQGD;
n=size(L,1);
for i=1:n,
    v=FL(1:n,i);
    s=sqrt(Dp(i,i));
    [nL,nD]=ldltup(L,D,s*v);
    L=nL; D=nD;
end

function [newL,newD]=ldltup(L,D,v)
n=size(L,1);
newL=L; newD=D;
oldt=1;
for j=1:n,
     p=v(j);
     t=oldt+p^2/D(j,j);
    newD(j,j)=D(j,j)*t/oldt;
    beta=p/(D(j,j)*t);
    if (j < n),
      v(j+1:n)=v(j+1:n)-p*L(j+1:n,j);
      newL(j+1:n,j)=L(j+1:n,j)+beta*v(j+1:n);
    end;
    oldt=t;
end;
```

Fig. 13. The LDC algorithm for the time update equations.

VI. Conclusion

We have studied the implementation of the Kalman filter for channel estimation in a mobile communication receiver. It was shown that by using a proper model for the fading channel, the Kalman filter can be applied for the estimation of CIR, and the matrices required for the computation of the Kalman estimator can be obtained from the channel model. Implementation of the Kalman filter was considered in two stages for measurement update and time update equations and for each stage implementation algorithms were considered. VLSI implementation of the time update equations is more challenging and the LDC algorithm was adopted for this purpose.

A systolic VLSI structure was proposed for the implementation of the LDC algorithm and the performance of different mappings of this structure were compared. For the simultaneous implementation of the Viterbi algorithm and the channel estimator, parallel structures were proposed to utilize the inherent parallelism present in the receiving algorithm.

The accuracy and stability of the hardware implementation was studied by simulations with a different number of bits in the digital wordlength. By comparing the performance of different estimators we conclude that the Kalman filter can improve the BER performance of the receiver by 10 dB compared to other estimators at the expense of a longer wordlength in digital implementation.

Appendix A

The square-root algorithm in Fig. 12, described with MATLAB, is for the measurement update equations based on the method of [3]. This recursive algorithm is described in Section III-A, and the inputs and outputs are defined in Fig. 3. The parameter *size* is equal to $3(\beta+1)$.

Appendix B

The MATLAB algorithm in Fig. 13 is for the temporal update using the LDC algorithm. This algorithm consists of two functions. The main function receives the $n \times n$ matrix product $FL = \mathbf{FL}_p$, Dp, and the $\mathbf{LD}$ factors of $\mathbf{GQG}^T$ as $GQGL$ and $GQGD$. Then the correction algorithm will be applied n times, as in (50), by calling the *ldltup* function. This function updates the Cholesky factorization of A to the Cholesky factorization of $A+v*v'$, i.e., If $A = L*D*L'$ then $A + v * v' = \text{new}L * \text{new}D * \text{new}L'$. It is assumed that A is symmetric and positive definite. The details of this algorithm are given in [21].

References

[1] M. J. Omidi, S. Pasupathy, and P. G. Gulak, "Joint data and Kalman estimation of fading channel using a generalized Viterbi algorithm," in *Proc. Int. Conf. Communications (ICC'96)*, 1996, pp. 1198–1203.

[2] F. M. F. Gaston and G. W. Irwin, "Systolic Kalman filtering: An overview," *Proc. IEE* , vol. 137, pt. D, no. 4, pp. 235–244, July 1990.

[3] M. Jover and T. Kailath, "A parallel architecture for Kalman filter measurement update and parameter estimation," *Automatica*, vol. 22, no. 1, pp. 43–57, 1986.

[4] M. J. Omidi, P. G. Gulak, and S. Pasupathy, "Parallel structures for joint channel estimation and data detection over fading channels," in *Proc. IEEE VLSI Signal Processing IX*, 1996, pp. 325–336.

[5] V. Raghavan, K. R. Pattipati, and Y. Bar-Shalom, "Efficient L-D factorization algorithms for PDA, IMM, and IMMPDA filters," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 29, pp. 1297–1310, Oct. 1993.

[6] R. Raheli, A. Polydoros, and C. Tzou, "Per-survivor processing: A general approach to MLSE in uncertain environments," *IEEE Trans. Commun.*, vol. 43, pp. 354–364, Feb.–Apr. 1995.

[7] R. A. Iltis, "A Bayesian maximum-likelihood sequence estimation algorithm for a priori unknown channels and symbol timing," *IEEE J. Select. Areas Commun.*, vol. 10, pp. 579–588, Apr. 1992.

[8] N. Seshadri, "Joint data and channel estimation using blind trellis search techniques," *IEEE Trans. Commun.*, vol. 42, pp. 1000–1011, Feb.–Apr. 1994.

[9] H. Kubo, K. Murakami, and T. Fujino, "An adaptive maximum-likelihood sequence estimator for fast time-varying intersymbol interference channels," *IEEE Trans. Commun.*, vol. 42, pp. 1872–1880, Feb.–Apr. 1994.

[10] Q. Dai and E. Shwedyk, "Detection of bandlimited signals over frequency selective Rayleigh fading channels" *IEEE Trans. Commun.*, vol. 42, pp. 941–950, Feb.–Apr. 1994.

[11] M. E. Rollins and S. J. Simmons, "Simplified per-survivor Kalman processing in fast frequency-selective fading channels," *IEEE Trans. Commun.*, vol. 45, pp. 544–553, May 1997.

[12] E. A. Lee and D. G. Messerschmitt, *Digital Communication*, 2nd ed. Norwell, MA: Kluwer, 1994.

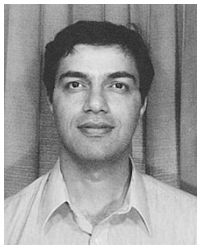
[13] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. Englewood Cliffs, NJ: Prentice–Hall, 1979.

[14] G. M. Vitetta and D. P. Taylor, "Multisampling receivers for uncoded and coded PSK signal sequences transmitted over Rayleigh frequency-flat fading channels," *IEEE Trans. Commun.*, vol. 44, pp. 130–133, Feb. 1996.

[15] J. Lin, F. Ling, and J. G. Proakis, "Joint data and channel estimation for TDMA mobile channels," *Int. J. Wireless Inform. Networks*, vol. 1, no. 4, pp. 229–238, 1994.

[16] A. H. Sayed and T. Kailath, "A state-space approach to adaptive RLS filtering," *IEEE Signal Processing Mag.*, July 1994.

[17] P. Rao and M. A. Bayoumi, "An efficient VLSI implementation of real-time Kalman filter," in *Proc. IEEE Int. Symp. Circuit Syst.*, 1990, vol. 3, pp. 2353–2356.
[18] P. Rao and M. A. Bayoumi, "An algorithm specific VLSI parallel architecture for Kalman filter," in *VLSI Signal Processing IV*. Piscataway, NJ: IEEE, 1991, pp. 264–273.
[19] M. Bayoumi, P. Rao, and B. Alhalabi, "VLSI parallel architecture for Kalman filter an algorithm specific approach," *J. VLSI Signal Processing* vol. 4, nos. 2–3, pp. 147–163, May 1992.
[20] M. S. Grewal and A. P. Andrews, *Kalman Filtering, Theory and Practice*. Englewood Cliffs, NJ: Prentice–Hall, 1993.
[21] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. New York: Academic, 1981.
[22] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 2nd ed. Baltimore, MD: Johns Hopkins Press, 1991, p. 13.
[23] M. J. Omidi, P. G. Gulak, and S. Pasupathy, "Joint data and channel estimation using the per-branch processing method," in *Proc. IEEE Signal Processing Workshop Wireless Communications*, Apr. 1997, pp. 389–392.
[24] S. Y. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice–Hall, 1988.
[25] C. T. Kuo, B. S. Chen, and Z. S. Kuo, "Stability analysis of digital Kalman filters with floating-point computation," *J. Guidance*, vol. 14, no. 3, pp. 636–644, 1991.
[26] F. Ling, D. Manolakis, and J. G. Proakis, "Finite word-length effects in recursive least squares algorithms with application to adaptive equalization," *Ann. Telecommun.*, vol. 41, pp. 1–9, May–June 1986.



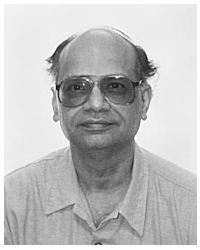
Mohammad Javad Omidi received the B.Eng. degree in electronics and the M.A.Sc. degree in communications from Isfahan University of Technology, Isfahan, Iran, in 1987 and 1989, respectively. Currently, he is a Ph.D. candidate in the Department of Electrical Engineering at the University of Toronto, Toronto, Canada.

His research interests are in the area of VLSI implementation of wireless communication algorithms.

P. Glenn Gulak (S'82–M'83–SM'96) received the Ph.D. degree from the University of Manitoba, Manitoba, Canada, in 1985.

He is currently a Professor in the Department of Electrical and Computer Engineering at the University of Toronto, Toronto, Canada. From 1985 to 1988 he was a Research Associate in the Information Systems Laboratory and the Computer Systems Laboratory at Stanford University, Stanford, CA. His research interests are in the areas of circuits, algorithms, and VLSI architectures for digital communications and signal processing applications.

Dr. Gulak received a Natural Sciences and Engineering Research Council of Canada Postgraduate Scholarship and has received several teaching awards for undergraduate courses taught at the University of Toronto. He has served on the ISSCC Signal Processing Technical Subcommittee since 1990 and currently serves as Program Secretary for ISSCC.



S. Pasupathy (M'73–SM'81–F'91) was born in Chennai, Madras, Tamilnadu, India, on September 21, 1940. He received the B.E. degree in telecommunications from the University of Madras in 1963, the M.Tech. degree in electrical engineering from the Indian Institute of Technology, Madras, in 1966, and the M.Phil. and Ph.D. degrees in engineering and applied science from Yale University, New Haven, CT, in 1970 and 1972, respectively.

He joined the faculty of the University of Toronto, Toronto, Canada, in 1973 and became a Professor of Electrical Engineering in 1983. He has served as the Chairman of the Communications Group and as the Associate Chairman of the Department of Electrical Engineering at the University of Toronto. His research interests are in the areas of communication theory, digital communications, and statistical signal processing. He is a registered Professional Engineer in the province of Ontario.

Dr. Pasupathy has served as Editor for data communications and modulation for the IEEE TRANSACTIONS ON COMMUNICATIONS. From 1979 to 1982 he served as a Technical Associate Editor for the IEEE COMMUNICATIONS MAGAZINE and from 1980 to 1983 as an Associate Editor for the *Canadian Electrical Engineering Journal*. Since 1984, he has been a regular columnist for the IEEE COMMUNICATIONS MAGAZINE.