DISPLACEMENT MIMO KALMAN EQUALIZER ARCHITECTURE FOR CDMA DOWNLINK IN FAST FADING CHANNELS

Yuanbin Guo, Jianzhong Zhang, Dennis McCain [†] Joseph R. Cavallaro

Rice University
Department of Electrical and Computer Engineering
Houston TX 77005.
{Yuanbin.Guo, Charlie.zhang,Dennis.Mccain@nokia.com, cavallar@rice.edu}

Abstract

In this paper, we explore the displacement structure in a Kalman equalizer for MIMO-CDMA downlink. A streamlined MIMO Kalman equalizer architecture is proposed to extract the commonality in the data path by exploiting the displacement structure of the transition matrix and the block-Toeplitz structure of the channel matrix. Numerical matrix multiplications with $\mathcal{O}(F^3)$ complexity are eliminated by simple data loading process. Utilizing the block Toeplitz structure of the channel matrix, an FFT-based acceleration is proposed to avoid direct matrix multiplications in the time domain. Finally, an iterative Conjugate-Gradient based algorithm is proposed to avoid the inversion of the innovation correlation matrix in Kalman gain calculation. The proposed architecture not only reduces the numerical complexity to $\mathcal{O}(F \log_2 F)$ per chip, but also facilitates the parallel and pipelined VLSI implementation for real-time processing.

Keywords : MIMO, Kalman filter, displacement structure, FFT, Conjugate Gradient.
Part of this paper is to be presented in the IEEE GlobeCom'05 in St. Louis, MO, in Nov., 2005.

[†]Nokia Research Center, Irving, Tx, 75039

I. Introduction

MIMO (Multiple-Input-Multiple-Output) technology using multiple antennas at both the transmitter and receiver sides has recently emerged as a significant breakthrough to increase the spectral efficiency dramatically. The original invention is known as D-BLAST [1] and a more realistic strategy is V-BLAST [2] with reasonable tradeoff between performance and complexity. On the other hand, UMTS and CDMA2000 extensions optimized for data services lead to the standardization of Multi-Code CDMA systems such as the High-Speed-Downlink-Packet-Access (HSDPA) and its equivalent 1X EV-DV (Evolution Data and Voice) [3] to support multimedia services. Recently, MIMO extensions for the 3G wireless systems have received increasing attentions from the research community [3]. One of the challenges in applying MIMO to 3G systems is that, the orthogonality of the spreading codes is destroyed in a multipath-fading channel, and the Multiple-Access-Interference (MAI) is introduced along with the Inter-Symbol-Interference (ISI).

Linear-Minimum-Mean-Square-Error (LMMSE)-based chip equalizers have been proposed to restore the orthogonality of the spreading code, so as to suppress both the ISI and MAI [4]. Although the LMMSE equalizers involve the inversion of a large correlation matrix with the $O((N_r L)^3)$ complexity, where N_r is the number of Rx antenna and L is the channel length, many sub optimal algorithms have been proposed to avoid the Direct-Matrix-Inverse (DMI) and reduce the complexity. These include two major flavors of algorithms: the adaptive stochastic gradient algorithms such as LMS [4], and non-adaptive block-based algorithms such as the Conjugate Gradient and the Levinson algorithms [5][9]. Despite of the low complexity, the adaptive LMS algorithm suffers from stability problems because the convergence depends on the choice of good step size. On the other hand, several algorithms have been recently proposed to further reduce the $O((N_r L)^2)$ complexity in the block-based algorithms. For example, the FFT-based solution applies a block circulant matrix to approximate the Toeplitz structure in [6] [7] and simplifies the decomposition of the correlation matrix, and the FFT-accelerated Conjugate Gradient algorithm is proposed in [8]. These algorithms achieve $O(N_r L \log_2(L))$ complexity. With the complexity reduction efforts in these algorithms, the LMMSE equalization problem becomes more tractable for real-time hardware implementations. The relevant VLSI architectures are reported in [9] [10].

LMMSE equalizers have demonstrated fairly good performance in slow-fading channels. However, they lack the tracking capability in fast-fading channels because they are based on the assumption of quasi stationarity within a block data. Kalman filter is known to provide the Best-Linear-Unbiased-Estimator (BLUE) of a linear system [13] [14]. It has been applied in numerous applications to solve different system problems [13]-[17]. To mention a few, the work of Iltis et al [15] [16] developed the state-space model for estimation of the path gains and delays of multipath channels. In [17], the Kalman filter is applied to the MIMO CDMA downlink system and a proper state-space model is proposed for the channel equalization problem. It is preferable in the fast-fading environments for the following reasons: 1). it has the capability to track the non-stationarity in the time-variant channel, noise process etc; 2). it is an optimal estimator, i.e. a BLUE estimator in the sense of MMSE.

However, the conventional Kalman equalizer proposed in [17] has prohibitively high complexity for real-time hardware implementation. The Kalman filter involves an recursive computing structure to compute the Kalman gain and predict the state of the system. The complexity is dominated by numerous matrix-matrix multiplications of large size and the inversion of the innovation correlation matrix in Kalman gain and new state estimation. The fact that the receiver must be embedded in a portable device limits its applicability in the downlink receiver. For the purpose of real-time implementation, significant complexity reduction and the exploration for efficient computing architectures are of essential importance.

In this paper, a parallel VLSI-oriented recursive architecture for MIMO Kalman equalizer is proposed by examining the timing relationship to extract the commonalities. We then explore the block-displacement structure in the state transition and Kalman gain to dramatically reduce the redundant multiplications dramatically. Numerical matrix multiplications with $\mathcal{O}(F^3)$ complexity are eliminated by simple data loading process. A divide-and-conquer methodology is applied to partition the MIMO displacement structure into more tractable sub block architectures in the Kalman recursion. By utilizing the block Toeplitz structure of the channel matrix, an FFT-based acceleration is proposed to avoid direct matrix multiplications in the time domain for the predicted state-error correlation matrix and the Kalman gain computations. Finally, an iterative Conjugate-Gradient based algorithm is proposed to avoid the inversion of the Hermitian innovation correlation

matrix in Kalman gain computer. The data path is streamlined by combining the displacement and block-Toeplitz structure. The proposed architecture not only reduces the numerical complexity to $O(F \log_2 F)$ per chip, but also facilitates the parallel and pipelined real-time VLSI implementation. Simulation demonstrates significant complexity reduction and memory storage saving in the MIMO Kalman filter while keeping the performance gain over the conventional LMMSE-based chip equalizer.

The paper is organized as follows. In section II, we present the system model of MIMO CDMA downlink and a brief recap of Kalman filter based on the state-space model. In section III, we derive a commonality extracted computing architecture and explore the displacement structure in the MIMO state-transition equation to eliminate many matrix multiplication. In section IV, an "over-lap save" FFT-based acceleration scheme for the Kalman gain computer is proposed by exploring the block Toeplitz structure in the MIMO channel matrix. In section V, an inner iterative algorithm based on conjugate gradient method is applied to avoid the inverse of the innovation correlation matrix. The performance and complexity is discussed in section VI and conclusion is given in section VII.

II. MIMO CDMA System And State-Space Model

We consider the system model of the MIMO CDMA downlink based on spatial multiplexing with M Tx antennas and N_r Rx antennas. First, the high data rate symbols are demultiplexed into $U \times M$ lower rate substreams, where U is the number of spreading codes used in the system for data transmission. The substreams are broken into M groups, where each substream in the group is spread with a spreading code of spreading factor F. Each group of substreams are then combined and scrambled with long scrambling codes and transmitted through the t^{th} transmit antenna. The chip level signal at the t^{th} transmit antenna is given by $x_t(i) = \sum_{u=1}^{U} s_t^u(j)c_t^u(i) + s_t^P(j)c_t^P(i)$, where j is the symbol index, i is chip index and u is the index of the composite spreading code. $s_t^u(j)$ is the j^{th} symbol of the u^{th} code at the t^{th} substream. In the following, we focus on the j^{th} symbol index and omit the index for simplicity. $c_t^u(i) = c^u(i)c_t^{(s)}(i)$ is the composite spreading

code sequence for the u^{th} code at the t^{th} substream where $c^u(i)$ is the user specific Hadamard spreading code and $c_t^{(s)}(i)$ is the antenna specific scrambling long code. $s_t^P(j)$ denotes the pilot symbols at the t^{th} antenna. $c_t^P(i) = c^P(i)c_t^{(s)}(i)$ is the composite spreading code for pilot symbols at the t^{th} antenna. The received chip level signal at the r^{th} Rx antenna is given by

$$y_r(i) = \sum_{t=1}^{M} \sum_{l=0}^{L_{t,r}} h_{t,r}(l) x_t(i - \tau_l) + \nu_t(i). \tag{1}$$

Here $L_{t,r}$ is the channel delay spread between the t^{th} transmit and the r^{th} receive antennas. The channel is characterized by a channel matrix between the t^{th} Tx and the r^{th} Rx antenna as $h_{t,r}(t) = \sum_{l=0}^{L_{t,r}} h_{t,r}(l)\delta(t-\tau_{t,r,l})$ where $\delta(t)$ is the Kronecker Delta function. By collecting the F consecutive chips at the k^{th} symbol from each of the N_r receive antennas in a signal vector $\mathbf{y}_r(k) = [y_r(kF + F-1) \ \cdots \ y_r(kF)]^T$ and packing the signal vectors from each receive antenna, we form the signal vector as $\mathbf{y}(k) = [\mathbf{y}_1(k)^T \ \cdots \ \mathbf{y}_r(k)^T \ \cdots \ \mathbf{y}_{N_r}(k)^T]^T$. Here F is the spreading gain. In vector form, the received signal is given by

$$\mathbf{y}_r(k) = \sum_{t=1}^{M} \mathbf{H}_{t,r}(k)\mathbf{x}_t(k) + \mathbf{v}_r(k) = \mathbf{H}_r(k)\mathbf{x}(k) + \mathbf{v}_r(k) \tag{2}$$

where $\mathbf{v}_r(k)$ is the additive Gaussian noise. The transmitted chip vector for the t^{th} transmit antenna is given by $\mathbf{x}_t(k) = [x_t(kF + F - 1) \ \cdots \ x_t(kF) \ \cdots \ x_t(kF - D)]^T$ and the overall transmitted signal vector is given by stacking up the substreams for multiple transmit antennas as $\mathbf{x}(k) = [\mathbf{x}_1(k)^T \ \cdots \ \mathbf{x}_t(k)^T \cdots \ \mathbf{x}_M(k)^T]^T$. D is the channel delay spread. The channel matrix from multiple transmit is defined as $\mathbf{H}_r(k) = [\mathbf{H}_{1,r}(k) \ \cdots \ \mathbf{H}_{t,r}(k) \ \cdots \ \mathbf{H}_{M,r}(k)]$, where $\mathbf{H}_{t,r}(k)$ is the channel matrix between the t^{th} transmit antenna and r^{th} receive antenna.

The Kalman filter is a recursive method to provide the BLUE estimate of the state $\mathbf{x}(k)$ given the entire observed data $\mathbf{y}(1), \ \cdots, \ \mathbf{y}(k)$. The Kalman filter is derived from a state-space model consisting of a measurement equation and a process equation. The measurement equation describes the generation model of the observation $\mathbf{y}(k)$ from the state $\mathbf{x}(k)$ in a stochastic noise process. The process equation describes the state transition of the new estimate $\mathbf{x}(k)$ at time k from the estimate $\mathbf{x}(k-1)$ at time $k-1$. It is assumed that the transition matrix satisfies the product rule and the inverse rule [14]. By defining the transition matrix as $\mathbf{\Theta}(k)$, it is natural to have the measurement

equation as the received signal model and the process equation as an excitation of some process noise:

$$\mathbf{y}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k); \quad (3)$$

$$\mathbf{x}(k) = \mathbf{\Theta}(k)\mathbf{x}(k-1) + \mathbf{w}(k); \quad (4)$$

where the measure matrix is the overall MIMO channel matrix $\mathbf{H}(k)$ given by $\mathbf{H}(k) = [\mathbf{H}_1(k)^T, \cdots, \mathbf{H}_r(k)^T, \cdots, \mathbf{H}_{N_r}(k)^T]$. Moreover, $\mathbf{v}(k)$ denotes the measurement noise and $\mathbf{w}(k)$ denotes the process noise.

III. Displacement MIMO Kalman Equalizer

The conventional Kalman procedure given in [14] [17] contains many redundant computations with prohibitively high complexity for real-time implementation. The main purpose of this work is to explore architecture suitable for VLSI implementation. We observe that the complexity is dominated by matrix-matrix multiplications and matrix inversion, which has a complexity of $\mathcal{O}(F^3)$. However, we explore a variety of properties inherent to the MIMO Kalman filter, especially the block-displacement structure in the state transition matrix, the block-Toeplitz structure in the channel matrix and the Hermitian-symmetric structure in the correlation matrix. In the following sections, we will apply these structures step-by-step to reduce the complexity dramatically.

In this section, we first propose the parallel and pipelined architecture suitable for VLSI implementation by exacting the extract the commonality in a conventional Kalman procedure from the physical meaning of the process. We then apply the displacement structure to reduce the complexity in each step.

A. Commonality Extracted Architecture

The Kalman filter solves the process and the measurement equations jointly for the unknown states in an optimal manner. An innovation process and the correlation matrix of the innovation

process are defined by

$$\boldsymbol{\alpha}(k) = \mathbf{y}(k) - \hat{\mathbf{y}}(k|k-1), \quad (5)$$

$$\mathbf{R}(k) = E[\boldsymbol{\alpha}(k)\boldsymbol{\alpha}^H(k)]. \quad (6)$$

whose physical meaning represents the new information in the observation data $\mathbf{y}(k)$. $\hat{\mathbf{y}}(k|k-1)$ denotes the MMSE of the observed data at time k, given all the past observed data from time 1 to $k-1$. It is shown that $\mathbf{R}(k) = \mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}^H(k) + \mathbf{Q}_v(k)$ where the $\mathbf{P}(k|k-1)$ matrix is the predicted state error $\mathbf{P}(k|k-1) = E[\boldsymbol{\varepsilon}(k|k-1)\boldsymbol{\varepsilon}^H(k|k-1)]$. Here $\boldsymbol{\varepsilon}(k|k-1) = \hat{\mathbf{x}}(k) - \hat{\mathbf{x}}(k|k-1)$ is the predicted state error vector at time k using data up to time $k-1$. By defining a *Kalman gain* as $\mathbf{G}(k) = E[\mathbf{x}(k)\boldsymbol{\alpha}^H(k)]\mathbf{R}^{-1}(k)$, the new state estimate can be given by

$$\hat{\mathbf{x}}(k|k) = \boldsymbol{\Theta}(k)\mathbf{x}(k-1|k-1) + \mathbf{G}(k)\boldsymbol{\alpha}(k). \quad (7)$$

The physical meaning is that we may compute the MMSE of the state $\hat{\mathbf{x}}(k|k)$of a linear dynamical system by adding a correction item $\mathbf{G}(k)\boldsymbol{\alpha}(k)$ to the previous estimate, which is pre-multiplied by the transition matrix $\boldsymbol{\Theta}(k)$. The Riccati equation provides a recursive computation procedure of the predicted state error correlation matrix $\mathbf{P}(k|k-1)$ and the Kalman gain. By analyzing the data dependency and the timing relationship, the streamlined procedure is given in Table 1.

Note that in most Kalman filter notations, the innovation correlation matrix is generated by first pre-multiplying the measurement matrix and then post-multiplying the Hermitian transpose of the measurement matrix as $\mathbf{R}(k) = \mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}^H(k) + \mathbf{Q}_v(k)$. However, to facilitate the complexity reduction as shown in later sections, we introduce an intermediate matrix $\boldsymbol{\Omega}(k) = \mathbf{H}(k)\mathbf{P}(k|k-1)$ by only pre-multiplying the channel matrix. It is easy to show that $\mathbf{P}(k|k-1)$ is Hermitian symmetric. Thus, the generation of the $\mathbf{R}(k)$ is formed in a generic pre-multiplication by $\mathbf{H}$ form as $\mathbf{R}(k) = \mathbf{H}(k)\boldsymbol{\Omega}^H(k) + \mathbf{Q}_v(k)$.

The logic block diagram of the VLSI oriented architecture is shown in Fig. 1. The architecture is constructed with two parallel feedback loop structures that are associated with a common Kalman gain $\mathbf{G}(k)$. On the top is the one step predictor of the state $\hat{\mathbf{x}}(k|k)$ using the input observation $\mathbf{y}(k)$. A MUX first selects either the initial state or the delayed feedback state estimate for $\hat{\mathbf{x}}(k-1|k-1)$, where $z^{-1}\mathbf{I}$ in Fig. 1 denotes the z-transform of a sequence vector or matrix.

$\hat{\mathbf{x}}(k-1|k-1)$ is first pre-multiplied (PRM) by the transition matrix to generate $\hat{\mathbf{x}}(k|k-1)$ and then pre-multiplied by the channel matrix $\mathbf{H}(k)$. The result is subtracted from the input observation $\mathbf{y}(k)$ to generate the innovation process $\boldsymbol{\alpha}(k)$. The innovation is then multiplied by the Kalman gain $\mathbf{G}(k)$ and added to the $\hat{\mathbf{x}}(k|k-1)$ to finally generate the filtered state estimate $\hat{\mathbf{x}}(k|k)$. The dynamical transition is repeated for each time k to get the state sequence estimate.

On the bottom of Fig. 1 is the feedback loop of the correlation $\mathbf{P}(k|k)$ and the Kalman gain processor. Similarly, a MUX first selects from the initial value $\mathbf{P}(0|0)$ or the delayed feed $\mathbf{P}(k|k)$ for $\mathbf{P}(k-1|k-1)$. It is pre-multiplied and then post-multiplied by the transition matrix. The correlation of the process noise $\mathbf{Q}_w(k)$ is then added to form an intermediate correlation $\mathbf{P}(k|k-1)$. This is pre-multiplied by $\mathbf{H}(k)$ to generate $\boldsymbol{\Omega}(k)$, whose result is then Hermitian transposed. Note that the Hermitian transpose is a virtual operation with no time/memory resource usage because the subsequent operations can use the structure of $\boldsymbol{\Omega}^H(k)$ explicitly. $\boldsymbol{\Omega}^H(k)$ is pre-multiplied by $\mathbf{H}(k)$, and the result is added to the measurement noise correlation matrix $\mathbf{Q}_v(k)$ to form the innovation correlation $\mathbf{R}(k)$. The Kalman gain is produced as the result of the pre-multiplication of $\boldsymbol{\Omega}^H(k)$ with the inverse of $\mathbf{R}(k)$. The $\mathbf{P}(k|k)$ is then updated in the Riccati processor accordingly. In this streamlined data path, the commonality for the Riccati processor and Kalman gain processor is extracted as the dotted gray area compared with the conventional procedure in [17]. The timing and dependency relationship between each block is clear. Fundamentally, the recursive structure is reduced to several pre/post-multiplications by the transition matrix, the pre-multiplications by the channel matrix and one inverse.

B. MIMO Displacement Structure and Its Application

Despite the streamlining to reduce redundancy, the computation complexity still remains the same order. Both the matrix inversion and the matrix-matrix multiplication have $\mathcal{O}(N^3)$ complexity for an $N \times N$ matrix. The key challenge in implementing the proposed Kalman procedure in real-time is the dominant complexity in computing the matrix-matrix multiplications and inversion in Table. 1, which include $\boldsymbol{\Theta}(k)\hat{\mathbf{x}}(k-1|k-1)$, $\boldsymbol{\Theta}(k)\mathbf{P}(k-1|k-1)\boldsymbol{\Theta}(k)^H$, $\mathbf{H}(k)\hat{\mathbf{x}}(k|k-1)$, $\mathbf{H}(k)\mathbf{P}(k|k-1)$, $\mathbf{H}(k)\boldsymbol{\Omega}^H(k)$ and $\boldsymbol{\Omega}^H(k)\mathbf{R}^{-1}(k)$ etc. To reduce the complexity order dramatically, we need to apply the matrix structures for each critical data path. In this section, we utilize

the first structure, namely the displacement structure to eliminate some explicit matrix multiplications by simple data-loading of a small portion of the full matrix. This effort will be described step by step according to the procedure in Table. 1.

It is shown that the transition matrix can be designed as following

$$\begin{aligned} \mathbf{\Theta}(k) &= \mathbf{I}_M \otimes \tilde{\mathbf{\Theta}}(k) && (8) \\ \tilde{\mathbf{\Theta}}(k) &= \begin{pmatrix} \mathbf{0}_{F\times D} & \mathbf{0}_{F\times F} \\ \mathbf{I}_{D\times D} & \mathbf{0}_{D\times F} \end{pmatrix} && (9) \end{aligned}$$

where $\otimes$ denotes the Kronecker product. It is assumed that $D < F$ in most situations. The process noise is then given by $\mathbf{w}(k) = [\mathbf{w}_1(k)^T \ \cdots \ \mathbf{w}_t(k)^T \ \cdots \ \mathbf{w}_M(k)^T]^T$ where the process noise for the t^{th} transmit antenna is given by $\mathbf{w}_t(k) = [x_t(kF+F-1) \ \cdots \ x_t(kF) \ 0 \ \cdots \ 0]^T$. It is easy to verify that to pre-multiply a matrix with $\tilde{\mathbf{\Theta}}(k)$ is equivalent to shifting the first D rows of the matrix to the bottom and adding F rows of zeros to the upper portion. To post-multiply a matrix with $\tilde{\mathbf{\Theta}}^H(k)$ is equivalent to shifting the first D columns of the matrix to the right and adding F rows of zeros to the left portion. This is demonstrated by the following equations:

$$\tilde{\mathbf{\Theta}}(k) \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,F+D} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,F+D} \\ \cdots & & & \\ a_{F+D,1} & a_{F+D,2} & \cdots & a_{F+D,F+D} \end{pmatrix} = \begin{pmatrix} \mathbf{0}_{F\times 1} & \mathbf{0}_{F\times 1} & \cdots & \mathbf{0}_{F\times 1} \\ a_{1,1} & a_{1,2} & \cdots & a_{1,F+D} \\ \cdots & & & \\ a_{D,1} & a_{D,2} & \cdots & a_{D,F+D} \end{pmatrix}, \quad (10)$$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,F+D} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,F+D} \\ \cdots & & & \\ a_{F+D,1} & a_{F+D,2} & \cdots & a_{F+D,F+D} \end{pmatrix} \tilde{\mathbf{\Theta}}^H(k) = \begin{pmatrix} \mathbf{0}_{1\times F} & a_{1,1} & \cdots & a_{1,D} \\ \mathbf{0}_{1\times F} & a_{2,1} & \cdots & a_{2,D} \\ \cdots & & & \\ \mathbf{0}_{1\times F} & a_{F+D,1} & \cdots & a_{F+D,D} \end{pmatrix}. \quad (11)$$

For the MIMO case, the feature forms a block-displacement structure and will be applied to the following related computations step-by-step to reduce the computational complexity dramatically.

1). Displacement State Transition and Estimation: It is shown that the state transition equation can be partitioned into M transmit antennas as $\hat{\mathbf{x}}(k|k-1) = [\hat{\mathbf{x}}_1(k|k-1)^T \ \cdots \ \hat{\mathbf{x}}_t(k|k-$

$1)^T \cdots \hat{\mathbf{x}}_M(k|k-1)^T]^T$. Thus, the t^{th} sub block of the transition is given by

$$\hat{\mathbf{x}}_t(k|k-1) = \tilde{\mathbf{\Theta}}(k)\hat{\mathbf{x}}_t(k-1|k-1) = [\mathbf{0}_{1\times F},\ \hat{\mathbf{x}}_t^U(k-1|k-1)^T]^T, \tag{12}$$

where $\hat{\mathbf{x}}_t^U(k-1|k-1) \equiv [\hat{x}_t(k-1|k-1,0)\ \cdots\ \hat{x}_t(k-1|k-1,D-1)]^T$ is the upper D rows of the previous state. This process is shown in the Fig. 2.

Jointly considering the steps (1) and (4) in Table. 1, this displacement structure can be further applied in the filtered state estimation and feedback process. This is because $\mathbf{x}(k|k-1)$ contains many zero elements and we only need to compute a small portion of $\mathbf{x}(k|k)$ to form the feedback loop for the next recursion in computing the state transition equation. To show this, we partition the update equation $\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + \mathbf{G}(k)\boldsymbol{\alpha}(k)$ into $\hat{\mathbf{x}}_t(k|k) = \hat{\mathbf{x}}_t(k|k-1) + \mathbf{G}_t(k)\boldsymbol{\alpha}(k)$ where $\hat{\mathbf{x}}(k|k) = [\cdots\ \hat{\mathbf{x}}_t(k|k)\ \cdots]$ and $\mathbf{G}(k) = [\cdots\ \mathbf{G}_t(k)^T\ \cdots]$. We further partition the element-wise state estimate and the Kalman gain into three sub-blocks, the upper D rows, the lower D rows and the rest rows in the center as $\hat{\mathbf{x}}_t(k|k) = [\hat{\mathbf{x}}_t^U(k|k)^T\ \hat{\mathbf{x}}_t^C(k|k)^T\ \hat{\mathbf{x}}_t^L(k|k)^T]^T$ and $\mathbf{G}_t(k)^T = [\mathbf{G}_t^U(k)^T\ \mathbf{G}_t^C(k)^T\ \mathbf{G}_t^L(k)^T]^T$. We define the effective transition state vector as the lower D rows of the state at time $(k-1)$. It can be shown from the transition that the upper and center portions of the new states do not need to add the previous states. Only the lower portion is updated from the previous state with the Kalman gain. Then the new effective transition state vector is simply a copy of the new upper portion of the state. In the real-time implementation, only this portion is stored and feeded back to form the state transition. This is shown as the following procedure: 1). $\hat{\mathbf{x}}_t^U(k|k) = \mathbf{G}_t^U(k)\boldsymbol{\alpha}(k)$; 2). $\hat{\mathbf{x}}_t^C(k|k) = \mathbf{G}_t^C(k)\boldsymbol{\alpha}(k)$; 3). $\hat{\mathbf{x}}_t^L(k|k) = \boldsymbol{\chi}_t^L(k-1) + \mathbf{G}_t^L(k)\boldsymbol{\alpha}(k)$; 4).$\boldsymbol{\chi}_t^L(k) = \hat{\mathbf{x}}_t^U(k|k)$. This new transition accelerates the feedback before the whole vector is ready. The transition matrix-vector multiplication and part of the vector addition are eliminated. The storage of the transition vector is also reduced.

2). Predicted State Error Correlation Matrix: Another process involved with the transition matrix is the computation of the predicted state error correlation matrix $\mathbf{P}(k|k-1) = \mathbf{\Theta}(k)\mathbf{P}(k-1|k-1)\mathbf{\Theta}(k)^H + \mathbf{Q}_w(k)$. It is shown that the process noise correlation is given by

$$\mathbf{Q}_w(k) = E\{\mathbf{w}(k)\mathbf{w}(k)^H\} = \mathbf{I}_M \otimes \mathbf{Q}(k), \qquad \mathbf{Q}(k) = \begin{pmatrix} \tilde{\mathbf{Q}}_w(k) & \mathbf{0}_{F\times D} \\ \mathbf{0}_{D\times F} & \mathbf{0}_{D\times D} \end{pmatrix}. \tag{13}$$

Thus if we partition the MIMO correlation matrix to sub blocks as $\mathbf{P}(k|k-1) = \{\mathbf{P}_{t_1,t_2}(k|k-1)\}$ and $\mathbf{P}(k-1|k-1) = \{\mathbf{P}_{t_1,t_2}(k-1|k-1)\}$ for t_1 and $t_2 \in [1, M]$, we can get the partitioned sub blocks given by $\mathbf{P}_{t_1,t_2}(k|k-1) = \tilde{\mathbf{\Theta}}(k)\mathbf{P}_{t_1,t_2}(k-1|k-1)\tilde{\mathbf{\Theta}}(k)^H + \mathbf{Q}_{t_1,t_2}(k)$ where $\mathbf{Q}_{t_1,t_2}(k) = \mathbf{Q}(k)\delta(t_1 - t_2)$. Using the feature of the pre-multiplication and post-multiplication with the displacement transition matrix, we can show that the new state error correlation matrix is given by the following partitioning,

$$\begin{cases} \mathbf{P}_{t_1,t_2}(k|k-1, F:F+D-1, F:F+D-1) = \boldsymbol{\rho}_{t_1,t_2}(k-1) \\ \mathbf{P}_{t,t}(k|k-1, 0:F-1, 0:F-1) = \tilde{\mathbf{Q}}_w(k) \\ \mathbf{P}_{t_1,t_2}(k|k-1, i, j) = 0, \qquad o.w. \end{cases} \tag{14}$$

where the sub block matrix $\boldsymbol{\rho}_{t_1,t_2}(k-1)$ is a $D \times D$ left-upper corner of the partitioned correlation matrix defined by

$$\boldsymbol{\rho}_{t_1,t_2}(k-1) = \mathbf{P}_{t_1,t_2}(k-1|k-1, 0:D-1, 0:D-1). \tag{15}$$

Thus, the matrix multiplications and additions in computing $\mathbf{P}(k|k-1)$ from $\mathbf{P}(k-1|k-1)$ are all eliminated. Logically we only need to copy some small sub-blocks of $\mathbf{P}(k-1|k-1)$ to $\mathbf{Q}_w(k)$ following the special pattern. Actually, the storage of the full matrix is not necessary since the matrix is sparse with many zero entries. This displacement procedure is demonstrated by the data loading process in Fig. 3.

3). Updated State Error Correlation Matrix: Jointly considering the feedback data path of $\mathbf{P}(k|k)$ and the displacement structure in $\mathbf{P}(k|k-1)$, it is clear that only the upper left corner $\boldsymbol{\rho}_{t_1,t_2}(k-1)$ are utilized for the element matrix $\mathbf{P}_{t_1,t_2}(k-1|k-1)$. The other elements are redundant information that will be dropped during the displacement procedure. Thus, there is no need to compute and keep these components. Because there is matrix multiplication of the Kalman gain with $\mathbf{\Omega}(k)$ as in $\mathbf{P}(k|k) = \mathbf{P}(k|k-1) - \mathbf{G}(k)\mathbf{\Omega}(k)$, we define an intermediate variable $\mathbf{\Psi}(k)$ for the multiplication and partition it to MIMO sub-blocks as $\mathbf{\Psi}(k) = \mathbf{G}(k)\mathbf{\Omega}(k) = \{\mathbf{\Psi}_{t_1,t_2}(k)\}$ for $t_1, t_2 \in [1, M]$. Instead of computing the full matrix of $\mathbf{P}(k|k)$, we only need to compute the relevant submatrices given by

$$\boldsymbol{\rho}_{t_1,t_2}(k) = \mathbf{P}_{t_1,t_2}(k|k-1, 0:D-1, 0:D-1) + \mathbf{\Psi}_{t_1,t_2}(k, 0:D-1, 0:D-1). \tag{16}$$

We also partition the Kalman Gain $\mathbf{G}(k)$ and the $\mathbf{\Omega}(k)$ matrices into the MIMO sub blocks as $\mathbf{G}(k) = \{\mathbf{G}_{t,r}(k)\}$ and $\mathbf{\Omega}(k) = \{\mathbf{\Omega}_{t,r}(k)\}$ for $t \in [1, M]$ and $r \in [1, N_r]$. Moreover, $\mathbf{G}_{t,r}(k) = [\mathbf{G}^U_{t,r}(k)^T \quad \mathbf{G}^L_{t,r}(k)^T]^T$ is further partitioned into the upper and lower sub-matrices of sizes $(D \times F)$ and $(F \times F)$, respectively. $\mathbf{\Omega}_{r,t}(k) = [\mathbf{\Omega}^L_{r,t}(k) \quad \mathbf{\Omega}^R_{r,t}(k)]$ is partitioned into the left and right sub-matrices of sizes $(F \times D)$ and $(F \times F)$, respectively. It is clear that the element block in the $\mathbf{\Psi}(k)$ is given by

$$\mathbf{\Psi}_{t_1,t_2}(k) = \sum_{r=1}^{N_r} \mathbf{G}_{t_1,r}(k)\mathbf{\Omega}_{r,t_2}(k) = \begin{pmatrix} \sum_{r=1}^{N_r} \mathbf{G}^U_{t_1,r}(k)\mathbf{\Omega}^L_{r,t_2}(k) & \sum_{r=1}^{N_r} \mathbf{G}^U_{t_1,r}(k)\mathbf{\Omega}^R_{r,t_2}(k) \\ \\ \sum_{r=1}^{N_r} \mathbf{G}^L_{t_1,r}(k)\mathbf{\Omega}^L_{r,t_2}(k) & \sum_{r=1}^{N_r} \mathbf{G}^L_{t_1,r}(k)\mathbf{\Omega}^R_{r,t_2}(k) \end{pmatrix}. \quad (17)$$

Comparing the displacement structure, only the left-upper corner of size $D \times D$ is necessary, which is given by $\widetilde{\mathbf{\Psi}}_{t_1,t_2}(k) = \mathbf{\Psi}_{t_1,t_2}(k, 0 : D-1, 0 : D-1) = \sum_{r=1}^{N_r} \mathbf{G}^U_{t_1,r}(k)\mathbf{\Omega}^L_{r,t_2}(k)$. This is only associated with the upper part of $\mathbf{G}_{t_1,r}(k)$ and left part of $\mathbf{\Omega}_{r,t_2}(k)$. As a summary, the updated effective state error correlation is simplified by adding the correction item to the $D \times D$ corner of $\tilde{\mathbf{Q}}_w(k)$ which is constant to the transmit antenna elements t_1 and t_2 as $\boldsymbol{\rho}_{t_1,t_2}(k) = \tilde{\mathbf{Q}}_w(k, 0 : D-1, 0 : D-1)\delta(t_1 - t_2) + \widetilde{\mathbf{\Psi}}_{t_1,t_2}(k)$. This optimization not only reduces computation and storage complexity but also fastens the update and feedback time.

IV. FFT-Acceleration Using Block Toeplitz Structure

In the innovation and the omega matrix generation as shown in the step (2) of Table. 1, there are some pre-multiplications by the channel matrix $\mathbf{H}(k)$ as in $\boldsymbol{\alpha}(k) = \mathbf{y}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1)$ and $\mathbf{\Omega} = \mathbf{H}(k)\mathbf{P}(k|k-1)$. To reduce the complexity order, the direct matrix multiplications with $\mathcal{O}(N^3)$ need to be avoided. In this section, we apply the block-Toeplitz structure of the channel matrix to reduce the complexity related to the multiplication with the channel matrix. It can be shown that the MIMO channel matrix has the form

$$\mathbf{H}(k) = \begin{pmatrix} \mathbf{H}_{1,1}(k) & \cdots & \mathbf{H}_{M,1}(k) \\ \vdots & & \\ \mathbf{H}_{1,N_r}(k) & & \mathbf{H}_{M,N_r}(k) \end{pmatrix}. \quad (18)$$

We define the estimated observation and partition it into the sub-vectors for the multiple receive antennas as $\hat{\mathbf{y}}(k) = \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1) = \left[\sum_{t=1}^{M} \mathbf{H}_{t,1}(k)\hat{\mathbf{x}}_t(k|k-1), \cdots, \sum_{t=1}^{M} \mathbf{H}_{t,N_r}(k)\hat{\mathbf{x}}_t(k|k-1) \right]^T$, we can show that the channel matrix from the t^{th} transmit antenna and r^{th} receive antenna $\mathbf{H}_{t,r}(n)$ assumes the Toeplitz structure as

$$\mathbf{H}_{t,r}(n) = \begin{pmatrix} h_{t,0}^r & \cdots & h_{t,D}^r & & 0 \\ & \ddots & & \ddots & \\ 0 & & h_{t,0}^r & \cdots & h_{t,D}^r \end{pmatrix}. \tag{19}$$

Thus, the matrix-vector multiplication can be viewed as a FIR filter with the channel impulse response $[h_{t,D}^r, \cdots, h_{t,0}^r]$. This can be implemented in the time domain by delayed tap line architecture as a conventional FIR. It is well known that the time-domain FIR filtering can also be implemented by FFT-based circular convolution in the frequency domain. In [8], the "overlap-save" based matrix-vector multiplication architecture is proposed to accelerate the computation. The similar architecture can be applied directly to the Kalman filtering problem in this paper. This achieves $\mathcal{O}((F+D)\log_2(F+D))$ complexity algorithm versus $\mathcal{O}((F+D)^2)$ for the matrix-vector multiplication and $\mathcal{O}((F+D)^2\log_2(F+D))$ versus $\mathcal{O}((F+D)^3)$ for the matrix-matrix multiplications in the innovation estimation and the Kalman Gain computer. The procedure is described briefly as following:

1. Take the FFT of the zero-padded channel impulse response $[h_{t,D}^r, \cdots, h_{t,0}^r, 0, \cdots, 0]$;
2. Take the FFT of the right-product vector $\hat{\mathbf{x}}_t(k|k-1)$;
3. Compute the dot product of the frequency-domain coefficients;
4. Take the IFFT of the product;
5. Truncate the result to get the valid coefficients as the matrix-vector multiplication result.

The FFT-based architecture for the MIMO observation estimate $\hat{\mathbf{y}}(k) = \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1)$ is depicted in Fig. 4. First, the element-wise FFT bank computes the frequency coefficients of each zero-padded MIMO channel impulse response. Simultaneously, another FFT bank computes the

dimension-wise coefficients of the estimated state. The dot product of the two groups of coefficients is computed according to the transmit antenna index t. Then the results are grouped by the receive antenna index r by summing the result for all the transmit antennas. A dimension-wise FFT-bank with N_r IFFTs computes the dot products correspondingly and truncates the result according to the "over-lap save" architecture to generate the estimated observation. For a matrix-matrix multiplication involved with the block-Toeplitz channel matrix, we can extend the matrix-vector multiplication architecture to multiple vectors in a straightforward way. Note that we only need to take the FFT once for each channel impulse response. For the multiple vectors to be filtered, we can form a pipelined FFT computation to use the hardware resource efficiently. The computation procedure is summarized with the different computation rate in Table 2.

The computation of the correlation matrix of innovation as $\mathbf{R}(k) = \mathbf{H}(k)\mathbf{\Omega}^H(k) + \mathbf{Q}_v(k)$ is also accelerated by the FFT-based computing architecture in the frequency domain after the change of order with the Hermitian feature of $\mathbf{P}(k|k-1)$ and $\mathbf{Q}_v(k)$. The procedure is similar to Table. 2 for the computation of $\mathbf{\Omega}^H(k)$. Thus, the direct matrix computation involving the channel matrix $\mathbf{H}(k)$ is replaced by the FFT-based procedure. This reduces the complexity and facilitates the parallel processing in VLSI architectures. We will analyze the compute update rate in section VI.

V. Iterative Inverse Solver in Kalman Gain Processor

With the afore-mentioned optimizations, the complexity has been reduced dramatically. However, there is one last hard work in computing the Kalman gain as in $\mathbf{G}(k) = \mathbf{\Omega}^H\mathbf{R}^{-1}(k)$. It is known that a Gaussian elimination can be applied to solve the matrix inverse with complexity at the order of $\mathcal{O}[(NF)^3]$. Moreover, Cholesky decomposition can also be applied to accelerate the speed by reducing the hidden constant factor in the order of complexity. However, since these two solutions do not use the structure of the matrix, the complexity is at the same order as to solve the inverse of a general matrix.

We made the following observations: (1). $\mathbf{R}$ is a $(NF \times NF)$ Hermitian symmetric matrix. This can be easily verified as $\mathbf{R}^H(k) = \mathbf{\Omega}(k)\mathbf{H}(k)^H + \mathbf{Q}_v^H(k) = \mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}(k)^H + \mathbf{Q}_v(k) =$

$\mathbf{R}(k)$ because $\mathbf{P}(k|k-1) = \mathbf{P}(k|k-1)^H$ and $\mathbf{Q}_v(k) = \mathbf{Q}_v^H(k)$ are also Hermitian symmetric. It is known that the iterative Conjugate Gradient algorithm can solve the inverse of this type of matrix more efficiently; (2). The full matrix of the $\mathbf{G}$ is not necessary from the displacement structure of the state transition matrix. Only the lower $D \times NF$ ($\mathbf{G}_t^L$) and the left upper $D \times D$ ($\mathbf{G}_{t,r}^U$) corner are required. Thus, the matrix inversion and multiplication in the Kalman Gain computation can be simplified dramatically by jointly exploiting these two structures. To avoid the direct inversion of $\mathbf{R}$ by using the iterative CG algorithm, the Kalman gain computation and the state update is re-partitioned to generate the following new problem.

$$\boldsymbol{\chi}(k) = \mathbf{G}(k)\boldsymbol{\alpha}(k) = \boldsymbol{\Omega}^H(k)[\mathbf{R}^{-1}(k)\boldsymbol{\alpha}(k)] = \boldsymbol{\Omega}^H(k)\boldsymbol{\Phi}(k) \quad (20)$$

$$\boldsymbol{\Psi}(k) = \mathbf{G}(k)\boldsymbol{\Omega}(k) = \boldsymbol{\Omega}^H(k)[\mathbf{R}^{-1}(k)\boldsymbol{\Omega}(k)] = \boldsymbol{\Omega}^H(k)\boldsymbol{\Pi}(k) \quad (21)$$

where $\boldsymbol{\Phi}(k) = \mathbf{R}^{-1}(k)\boldsymbol{\alpha}(k)$ and $\boldsymbol{\Pi}(k) = \mathbf{R}^{-1}(k)\boldsymbol{\Omega}(k)$ respectively. With this changed order of computation, the iterative procedure of the CG-based algorithm is shown as the following two tasks.

A. Computation of $\boldsymbol{\Phi}(k) = \mathbf{R}^{-1}(k)\boldsymbol{\alpha}(k)$

The computation of $\boldsymbol{\Phi}(k)$ is a direct application of the iterative Conjugate-Gradient algorithm. The procedure is shown as in Table 3. Thus, the inversion of the $\mathbf{R}$ matrix is reduced to performing matrix-vector multiplication in the recursive structure. The Kalman gain is not computed explicitly. Note that the vector $\boldsymbol{\chi}(k) = \boldsymbol{\Omega}^H(k)\boldsymbol{\Phi}(k)$ can also be partitioned into the $\boldsymbol{\chi}(k) = [\cdots, \boldsymbol{\chi}_t(k)^T, \cdots]^T$. Using the displacement structure for the filtered state estimate discussed in section III, the element vector $\boldsymbol{\chi}_t(k)$ can still be partitioned into the upper, center and lower portion as $\boldsymbol{\chi}_t(k) = [\boldsymbol{\chi}_t^U(k)\ \boldsymbol{\chi}_t^C(k)\ \boldsymbol{\chi}_t^L(k)]$, where $\boldsymbol{\chi}_t^U(k) = \boldsymbol{\Omega}^U(k)^H\boldsymbol{\Phi}(k),\ \boldsymbol{\chi}_t^C(k) = \boldsymbol{\Omega}^C(k)^H\boldsymbol{\Phi}(k)$ and $\boldsymbol{\chi}_t^L(k) = \boldsymbol{\Omega}^L(k)^H\boldsymbol{\Phi}(k)$. Using the displacement feedback, we can feedback the upper portion once the result is ready to speedup the iteration pipelining. The complexity of this portion is reduced dramatically.

B. Update of Predicted State Error Correlation

Another major task related to the Kalman gain is the update of the predicted state error correlation $\mathbf{P}(k|k)$. With the definition of $\mathbf{\Pi}(k) = \mathbf{R}^{-1}(k)\mathbf{\Omega}(k)$, the CG procedure will need to be applied to the column vectors of $\mathbf{\Pi}(k)$ and $\mathbf{\Omega}(k)$. Similar to (17), it can be shown that $\mathbf{\Psi}(k) = \mathbf{\Omega}^H(k)\mathbf{\Pi}(k)$ can also be partitioned into sub-block matrices for the MIMO configuration. The element is given by $\mathbf{\Psi}_{t_1,t_2}(k) = \sum_{r=1}^{N_r}[\mathbf{\Omega}_{r,t_1}(k)]^H\mathbf{\Pi}_{r,t_2}(k)$ where $\mathbf{\Omega}_{r,t}(k)$ is the element of the $\mathbf{\Omega}$ matrix and $\mathbf{\Pi}(k)$ is partitioned to $\left[\mathbf{\Pi}_{r,t}(k)\right]$ for $r \in N_r$ and $t \in M$. Since only the left upper corner in $\mathbf{\Psi}_{t_1,t_2}(k)$ is of interest as shown in

$$\mathbf{\Psi}_{t_1,t_2}(k) = \begin{pmatrix} \sum_{r=1}^{N_r}[\mathbf{\Omega}_{r,t_1}^L(k)]^H\mathbf{\Pi}_{r,t_2}^L(k) & \cdots \\ \cdots & \cdots \end{pmatrix}, \tag{22}$$

the full matrix of $\mathbf{\Pi}(k)$ is not necessary and the whole matrix multiplication by $\mathbf{\Omega}^H(k)$ is redundant. Thus, if the $\mathbf{\Pi}(k)$ is defined by column sub-matrices as $\mathbf{\Pi}(k)=[\mathbf{\Pi}_1(k)\cdots\mathbf{\Pi}_t(k)\cdots\mathbf{\Pi}_M(k)]$, and each $\mathbf{\Pi}_t(k)$ is further partitioned into the left portion and right portion as $\mathbf{\Pi}_t(k) = [\mathbf{\Pi}_t^L(k)\,\mathbf{\Pi}_t^R(k)]$, we only need to calculate the left portion from the CG iterative algorithm. Because the iterative algorithm finally reduces to matrix-vector multiplications in a loop, the interesting columns can be easily identified and picked up by simply ignoring the right portions. The effective data for both the $\mathbf{\Omega}^H(k)$ and $\mathbf{\Pi}(k)$ are shown in Fig. 5 as the shaded portion. The iterative procedure to compute the matrix inverse and multiplication $\mathbf{\Pi}(k) = \mathbf{R}^{-1}(k)\mathbf{\Omega}(k)$ is only necessary for the effective data as in Table. 4.

Note that the D columns in the t^{th} sub-column matrix can be computed independently, as well as the M sub-columns. The computation of $\kappa_{t,0,l} = \boldsymbol{\eta}_{t,0}^H(:,l)\boldsymbol{\eta}_{t,0}(:,l)$ is actually a norm computation for the l^{th} column vector of $\boldsymbol{\eta}_{t,0}$, i.e. $\boldsymbol{\eta}_{t,0}(:,l)$. Similarly, $\boldsymbol{\xi}_{t,j}(:,l)$ is the l^{th} column vector of $\boldsymbol{\xi}_{t,j}$ and $\boldsymbol{\lambda}_{t,j-1}(:,l)$ the l^{th} column vector of $\boldsymbol{\lambda}_{t,j-1}$. The computation of $\kappa_{t,j+1,l}$ and $\rho_{t,l}$ only involves dot-product of two vectors. The matrix multiplication of "$\boldsymbol{\lambda}_{t,j-1} * \text{diag}(\rho_{t,1},\ \cdots,\ \rho_{t,l},\ \cdots, \rho_{t,D})$", "$\boldsymbol{\xi}_{t,j} * \text{diag}(\rho_{t,1},\ \cdots,\ \rho_{t,l},\ \cdots,\ \rho_{t,D})$" and "$\boldsymbol{\lambda}_{t,j-1} * \text{diag}(\sigma_{t,1},\ \cdots,\ \sigma_{t,l},\ \cdots,\ \sigma_{t,D})$" are actually implemented by independent scaling of the column vectors of the left-side matrix. The complexity is dominated by the matrix-submatrix multiplication "$\boldsymbol{\xi}_{t,j} = \mathbf{R}\boldsymbol{\lambda}_{t,j-1}$" multiplication which is D independent matrix-vector multiplications.

Thus, the direct-matrix inversion of $\mathbf{R}$ is avoided and the "inversion + multiplication" is reduced to a small portion of the matrix-vector multiplications in an iteration loop. Combining the complexity reduction in calculating $\mathbf{\Psi}_{t_1,t_2}(k)$, the complexity order is reduced significantly.

VI. Performance & Complexity

A. *Performance*

Since it is well known that the chip equalizer has much better performance than the conventional Rake receiver, we only compare the performance of the LMMSE based chip equalizer using the direct-matrix inverse with the Kalman based receiver. The performance is evaluated in a CDMA2000 1X EV-DV simulation chain. The focus is to optimize the complexity without sacrificing the performance. Both the Bit-Error-Rate (BER) and the Block-Error-Rate (BLER) are compared from a link level simulation. The ITU Veh-A channel model is applied. Forward Error Correcting (FEC) code of rate 0.7083 and 0.5156 is applied for the QPSK and 16-QAM respectively. In the simulation, the spreading gain is 32 and U=25 codes are applied for data transmission. Fig. 6 and 7 show the performance for $M = 2, N_r = 1$ with vehicular speed of $30km/h$ and $50km/h$ respectively. Fig. 8 shows the performance for the 16-QAM for 2×2 MIMO configuration with speed of 50 km/h. The superiority of the Kalman filter over the LMMSE chip equalizer is obvious.

B. *Numerical Complexity*

The complexity is a major consideration besides the performance. In this section, we briefly summarize the complexity reduction achieved from the displacement structure, the FFT-based acceleration and the effective Conjugate Gradient iterative solver. It is clear that the original Kalman procedure has the complexity of $\mathcal{O}(F^3)$ for each the matrix-matrix multiplication and the inversion of the Kalman gain. The procedure is applied for symbol duration. Thus, the complexity of the original procedure is $\mathcal{O}(F^2)$ per chip. After using the displacement structure, many matrix multiplications involving the transition matrix are replaced by simple data loading procedures.

Moreover, the FFT acceleration of the matrix multiplication for the channel matrix reduces the complexity to several FFT operations. Finally, the conjugate gradient procedure with only the effective sub blocks avoids the direct matrix inverse and reduces the complexity to some matrix vector multiplications. Overall, the complexity per chip becomes $\mathcal{O}(F \log_2 F)$. Moreover, the proposed architecture has more parallel structure, which is more suitable for VLSI real-time implementation.

C. Computation Update Rate for FFT-acceleration

We discuss the computation rate of the FFT coefficients here. It is clear that for all the $\mathbf{H}(k)$ pre-multiplications in each of the k^{th} iteration, we only need to compute the element-wise FFT of $\mathbf{H}(k)$ once. We only need to compute the FFTs of the right-hand multiply factor and the dot products individually. Specifically, if we define

$$\hat{\mathbf{y}}(k) = \sum_{t=1}^{M} \mathbf{H}_{t,r}(k)\hat{\mathbf{x}}_t(k|k-1) \qquad for \quad r \in [1, N_r], t \in [1, M]; \tag{23}$$

$$\mathbf{\Omega}_{r,t}(k) = \sum_{t_1=1}^{M} \mathbf{H}_{t_1,t}\mathbf{P}_{t_1,t}(k|k-1) \qquad for \quad r \in [1, N_r], t \in [1, M], \tag{24}$$

we only need to compute the element-wise FFT of $\mathbf{H}_{t,r}(k)$ once for both multiplications. Moreover, even in a fast fading environment, it is most likely that we can assume the channel impulse response is quasi-stationary for many symbols in a frame. Thus, for the coherence time that the channel coefficients are assumed to be quasi-static, i.e $\mathbf{H}(k) = \mathbf{H}$, we only need to compute the FFTs once. For each symbol, there will be one dimension-wise (in the domain of transmit antenna) FFT bank for the estimated state $\hat{\mathbf{x}}(k|k-1)$ and one dimension-wise (in the domain of receive antenna) IFFT bank for the observation estimate . For the computation of the $\mathbf{\Omega}$ matrix, there will be element-wise FFT banks for the $\mathbf{P}(k|k-1)$. However, after we examine the structure of the $\mathbf{P}(k|k-1)$, it is clear that the first F column of each of the $\mathbf{P}_{t_1,t}(k|k-1)$ is constant matrix if the $\mathbf{Q}_w(k)$ is assumed to be constant for the observation frame. Only the right-bottom $(D \times D)$ corner of $\mathbf{P}_{t_1,t}(k|k-1)$ is variable for each k. This is very likely even in a fast-fading environment as this is an input for a frame. Thus, only D columns need to be recomputed for each k. If we further assume that $\mathbf{P}_{t_1,t}(k|k-1) = \mathbf{P}_{t_1,t}(k|k-1)\delta(t_1 - t)$, i.e. only the diagonal block of $\mathbf{P}(k|k-1)$

will be effective, the computation of omega is simplified as $\mathbf{\Omega}_{r,t}(k) = \mathbf{H}_{t,r}\mathbf{P}_{t,t}(k|k-1)$.

VII. Conclusion

In this paper, we propose a displacement MIMO Kalman equalizer which is more suitable for real-time VLSI implementation. The computation complexity is significantly reduced by exploiting a variety of special features inherent to the MIMO Kalman filter, especially the block-displacement structure of the transition matrix, the block Toeplitz structure of the channel matrix and the Hermitian symmetric structure in the Kalman gain computation. Numerical matrix-matrix multiplications with $\mathcal{O}(F^3)$ complexity are eliminated by simple data loading process and FFT-based accelerator. Furthermore, the MIMO displacement structure is partitioned into more tractable sub block architectures and an iterative Conjugate-Gradient algorithm is proposed to compute the matrix inverse more efficiently. The proposed architecture only reduces the numerical complexity to $\mathcal{O}(F \log_2 F)$ per chip, but also provides a parallel and pipelined hardware architecture, which are of high significance in real-time VLSI implementation of practical MIMO systems.

Acknowledgments

The authors would like to appreciate Hoang Nguyen for providing the simulation chain and helpful discussions. Dr. Cavallaro was supported in part by Nokia Corporation, Texas Instruments Inc., the Texas Advanced Technology Program under grant 1999-003604-080, and by NSF under grant ANI-9979465.

References

[1] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas", *Bell Labs Tech. J.*, pp. 41-59, 1996.

[2] G. D. Golden, J. G. Foschini, R. A. Valenzuela and P. W. Wolniansky, "Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture", *Electron. Lett.*, vol. 35, pp.14-15, Jan. 1999.

[3] A. Wiesel, L. Garca, J. Vidal, A. Pags and Javier R. Fonollosa, "Turbo linear dispersion space time coding for MIMO HSDPA systems", *12th IST Summit on Mobile and Wireless Communications*, Aveiro, Portugal, June 15-18, 2003.

[4] K. Hooli, M. Juntti, M. J. Heikkila, P. Komulainen, M. Latva-aho and J. Lilleberg, "Chip-level channel equalization in WCDMA downlink", *EURASIP Journal on Applied Signal Processing*, pp. 757-770, Aug.2002.

[5] M. J. Heikkila, K. Ruotsalainen and J. Lilleberg, "Space-time equalization using conjugate-gradient algorithm in WCDMA downlink", *IEEE Proceeding in PIMRC*, pp. 673-677, 2002.

[6] J. Zhang, T. Bhatt and G. Mandyam, "Efficient linear equalization for high data rate downlink CDMA signaling", *37th IEEE Asilomar Conference on Signals, Systems and Computers, pp. 2171 - 2175, Volume 2, Monterey, CA*, 2003.

[7] Y. Guo, J. Zhang, D. McCain and J. R. Cavallaro, "Efficient MIMO equalization for downlink multi-code CDMA: complexity optimization and comparative study", *pp. 2513 - 2519, Volume 4, No. 7, Dallas, Tx, November 2004.*.

[8] Y. Guo, D. McCain and J. Cavallaro, "FFT-accelerated superfast iterative MIMO chip equalizer architecture for downlink CDMA receiver", *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Philadelphia, PI, March, 2005.*

[9] P. Radosavljevic, J. R. Cavallaro and A. D. Baynast, "ASIP Architecture Implementation of Channel Equalization Algorithms for MIMO Systems in WCDMA Downlink", *proceeding of VTC 2004, Sept. 2004*.

[10] Y. Guo, J. Zhang, D. McCain and J. R. Cavallaro, "Scalable FPGA architectures for LMMSE-based SIMO chip equalizer in HSDPA downlink", *37th IEEE Asilomar Conference on Signals, Systems and Computers*, Monterey, CA, 2003.

[11] V. Y. Pan, A. Zheng, "Superfast algorithms for Cauchy-like matrix computations and extensions", *Linear algebra and its applications*, 310, 83-108, 2000.

[12] V. Y. Pan, "Structured matrices and polynomials: unified superfast algorithms", *springer*, 2001.

[13] M. H. Hayes, "Statistical Digital Signal Processing and Modeling", *John Wiley & Sons*, Inc.: New York, NY, 1996.

[14] S. Haykin, "Adaptive Filter Theory: fourth edition", *Pearson Education Inc*, 2002.

[15] K. J. Kim and R. A. Iltis, "Joint detection and channel estimation algorithms for QS-CDMA signals over time-varying channels", *IEEE Trans. on Com.*, vol. 50, pp. 845-55, May 2002.

[16] T. J. Lim and Y. Ma, "The Kalman filter as the optimal linear minimum mean-squared error multiuser CDMA detector", *IEEE Trans. Info. Theory*, vol. 46, pp. 2561-6, Nov. 2000.

[17] H. Nguyen, J. Zhang and B. Raghothaman, "Linear Minimum MSE Equalization of SISO and MIMO CDMA Downlink Channels Via The Kalman and FIR Filters", *37th IEEE Asilomar Conference on Signals, Systems and Computers*, 2003.

Table 1: Summary of the commonality extracted Kalman procedure

Init :

$\hat{\mathbf{x}}(0|0) = E[\mathbf{x}(0)]$;

$\mathbf{P}(0|0) = E\{[\mathbf{x}(0) - \hat{\mathbf{x}}(0|0)][\mathbf{x}(0) - \hat{\mathbf{x}}(0|0)]^H\}$

Input vector : $\mathbf{y}(k)$; Output vector : $\hat{\mathbf{x}}(k|k)$;

Predefined parameters :

Transition matrix = $\mathbf{\Theta}(k)$; Measure matrix = $\mathbf{H}(k)$;

Correlation matrix of the process noise : $\mathbf{Q}_w(k) = E[\mathbf{w}(k)\mathbf{w}^H(k)]$;

Correlation matrix of the measure noise : $\mathbf{Q}_v(k) = E[\mathbf{v}(k)\mathbf{v}^H(k)]$.

Recursion for $k = 1, 2, \cdots$

(1). State transition equations :

$\hat{\mathbf{x}}(k|k-1) = \mathbf{\Theta}(k)\hat{\mathbf{x}}(k-1|k-1)$;

$\mathbf{P}(k|k-1) = \mathbf{\Theta}(k)\mathbf{P}(k-1|k-1)\mathbf{\Theta}(k)^H + \mathbf{Q}_w(k)$;

(2). Innovation generation :

$\boldsymbol{\alpha}(k) = \mathbf{y}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1)$;

$\mathbf{\Omega}(k) = \mathbf{H}(k)\mathbf{P}(k|k-1)$;

(3). Kalman gain computation :

$\mathbf{R}(k) = \mathbf{H}(k)\mathbf{\Omega}^H(k) + \mathbf{Q}_v(k)$;

$\mathbf{G}(k) = \mathbf{\Omega}^H(k)\mathbf{R}^{-1}(k)$;

(4). State estimate & Predicted state error correlation update :

$\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + \mathbf{G}(k)\boldsymbol{\alpha}(k)$;

$\mathbf{P}(k|k) = \mathbf{P}(k|k-1) - \mathbf{G}(k)\mathbf{\Omega}(k)$.

Table 2: Summary of the FFT acceleration of Matrix-Vector Multiplication.

Update/frame :

(1). The FFT bank of the channel and process error correlation coefficients :

$$\mathbf{\Phi}_{t,r} = \text{FFT}(\widetilde{\mathbf{h}}_{t,r}); \qquad \widetilde{\mathbf{h}}_{t,r} = [\mathbf{h}_{t,r} \quad \mathbf{0}], \qquad \text{for } t \in [1, M]; r \in [1, N_r]$$
$$\widetilde{\mathbf{\Lambda}}(\omega) = \text{FFT}(\widehat{\mathbf{Q}}_w); \qquad \widehat{\mathbf{Q}}_w = [\tilde{\mathbf{Q}}_w(k)^T \quad \mathbf{0}_{F\times D}]^T;$$

(2). The IFFT-truncation of dot product in the frequency domain coefficients :

$$\mathbf{\Omega}^U_{r,t}(col) = truncate\Big\langle \text{IFFT}\Big\{ \mathbf{\Phi}_{t,r} \circ \widetilde{\mathbf{\Lambda}}(\omega, col) \Big\} \Big\rangle, \qquad col \in [0, F-1];$$

Update/iteration k :

(1). FFT bank of the state estimate and effective state error correlation :

$$\widehat{\mathbf{X}}_t(\omega, k) = \text{FFT}\big\{ \hat{\mathbf{x}}_t(k|k-1) \big\} \qquad t \in [1, M]$$
$$\widehat{\mathbf{\Gamma}}_{t_1,t_2}(\omega, k) = \text{FFT}\Big\{ \big[\mathbf{0}_{D\times F} \quad \boldsymbol{\rho}_{t_1,t_2}(k-1)^T \big]^T \Big\}$$

(2). The IFFT-truncation of the dot product in the frequency domain:

$$\hat{\mathbf{y}}_r(k) = truncate\Big\langle \text{IFFT}\Big\{ \textstyle\sum_{t=1}^{M} \mathbf{\Phi}_{t,r} \circ \widetilde{\mathbf{X}}_t(\omega, k) \Big\} \Big\rangle, \qquad r \in [1, N_r];$$
$$\mathbf{\Omega}_{r,t}(k, col) = \begin{cases} \mathbf{\Omega}^U_{r,t}(col), & col = 0 : F-1; \\ truncate\Big\langle \text{IFFT}\big\{ \mathbf{\Phi}_{t,r} \circ \widehat{\mathbf{\Gamma}}_{t,t}(\omega, k, col - F) \big\} \Big\rangle, & col \in [F, F+D]; \end{cases}$$

Table 3: Summary of the CG procedure for the $\mathbf{\Phi}(k) = \mathbf{R}^{-1}\boldsymbol{\alpha}(k)$

(1). Initialization

$$\mathbf{\Phi}_0 = \mathbf{0};$$
$$\boldsymbol{\gamma}_0 = \boldsymbol{\alpha}(k); \qquad \mathbf{\Delta}_0 = \boldsymbol{\alpha}(k);$$
$$\delta_0 = \boldsymbol{\gamma}_0^H \cdot \boldsymbol{\gamma}_0; \qquad \delta_1 = \delta_0;$$

(2). For an iteration from $j = 1 : J$ until convergence:

$$\mathbf{\Gamma}_j = \mathbf{R} \cdot \mathbf{\Delta}_{j-1}; \qquad \mu = \delta_j / \mathbf{\Delta}_{j-1}^H \mathbf{\Gamma}_j;$$
$$\mathbf{\Phi}_j = \mathbf{\Phi}_{j-1} + \mu \mathbf{\Delta}_{j-1}; \qquad \boldsymbol{\gamma}_j = \boldsymbol{\gamma}_{j-1} - \mu \mathbf{\Gamma}_j;$$
$$\delta_{j+1} = \boldsymbol{\gamma}_j^H \boldsymbol{\gamma}_j; \qquad \nu = \delta_{j+1} / \delta_j$$
$$\mathbf{\Delta}_j = \boldsymbol{\gamma}_j + \nu \mathbf{\Delta}_{j-1}.$$

Table 4: Summary of the CG procedure for partial $\mathbf{\Pi}(k) = \mathbf{R}^{-1}\mathbf{\Omega}(k)$

(1). Initialization: for $t = 1 : M$

$\mathbf{\Pi}_{t,0} = \mathbf{0}$;

$\boldsymbol{\eta}_{t,0} = \mathbf{\Omega}_t^L(k)$; $\quad \boldsymbol{\lambda}_{t,0} = \mathbf{\Omega}_t^L(k)$;

$\kappa_{t,0,l} = [\boldsymbol{\eta}_{t,0}(:,l)]^H \boldsymbol{\eta}_{t,0}(:,l)$; $\quad \kappa_{t,1,l} = \kappa_{t,1,l}$; for $l \in [1, D]$.

(2). for $t \in [1, M]$, form an iteration from $j = 1 : J$ until convergence:

$\boldsymbol{\xi}_{t,j} = \mathbf{R}\boldsymbol{\lambda}_{t,j-1}$;

for $l = 1 : D$ {

$\quad \rho_{t,l} = \kappa_{t,j,l} / \big\{ [\boldsymbol{\lambda}_{t,j-1}(:,l)]^H * \boldsymbol{\xi}_{t,i}(:,l) \big\}$;

}

$\mathbf{\Pi}_{t,j} = \mathbf{\Pi}_{t,j-1} + \boldsymbol{\lambda}_{t,j-1} * \text{diag}(\rho_{t,1}, \ \cdots, \rho_{t,l}, \ \cdots, \ \rho_{t,D})$;

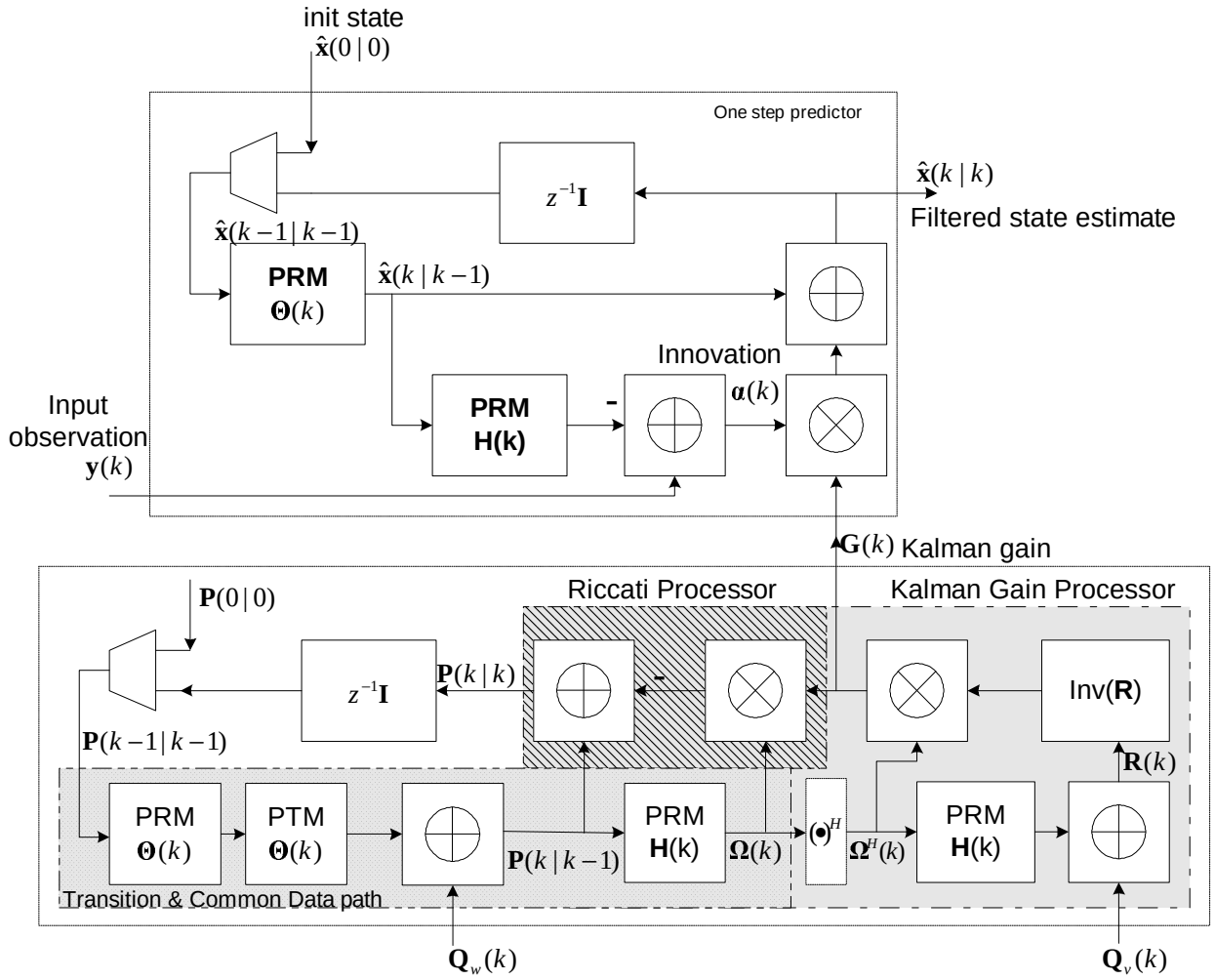
$\boldsymbol{\eta}_{t,j} = \boldsymbol{\eta}_{t,j-1} - \boldsymbol{\xi}_{t,j} * \text{diag}(\rho_{t,1}, \ \cdots, \ \rho_{t,l}, \ \cdots, \ \rho_{t,D})$;

for $l = 1 : D$ {

$\quad \kappa_{t,j+1,l} = [\boldsymbol{\eta}_{t,j}(:,l)]^H \boldsymbol{\eta}_{t,j}(:,l)$; $\quad \sigma_{t,l} = \kappa_{t,j+1,l} / \kappa_{t,j,l}$;

}

$\boldsymbol{\lambda}_{t,j} = \boldsymbol{\eta}_{t,j} + \boldsymbol{\lambda}_{t,j-1} * \text{diag}(\sigma_{t,1}, \ \cdots, \sigma_{t,l}, \ \cdots, \ \sigma_{t,D})$.



Commonality extracted data path block diagram of the state-space model of dynamic system

Figure 1: Commonality Extracted Data Path Block Diagram in the Kalman-based State-space Model of Dynamic Systems.

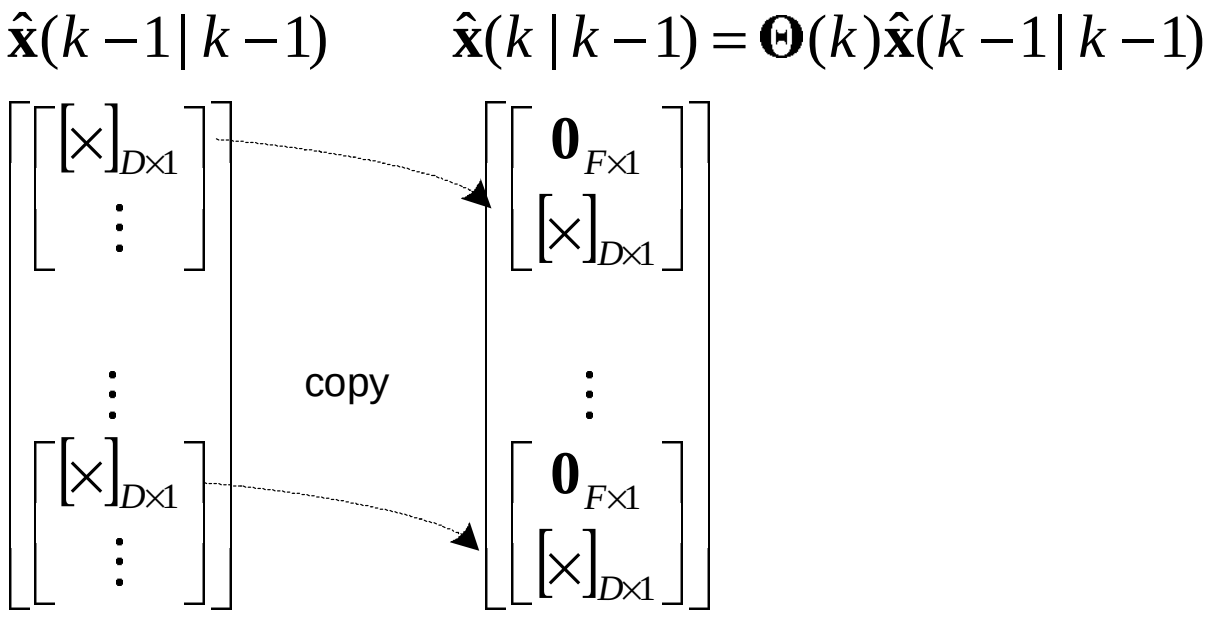


Figure 2: The displacement based architecture for the state transition.

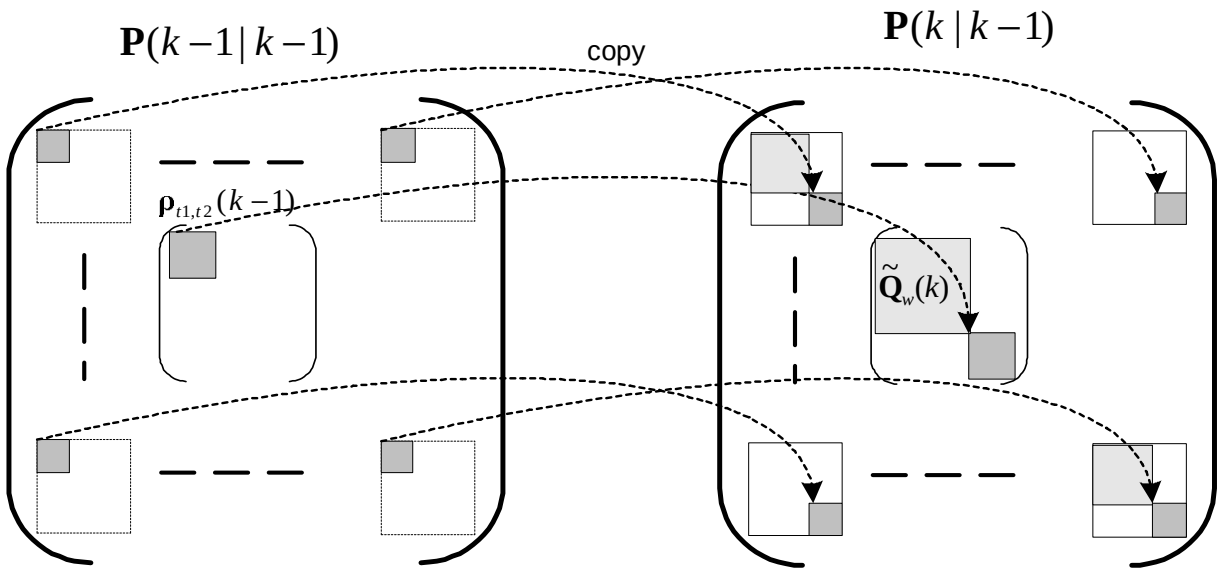


Figure 3: The displacement based architecture for predicted state error correlation matrix.

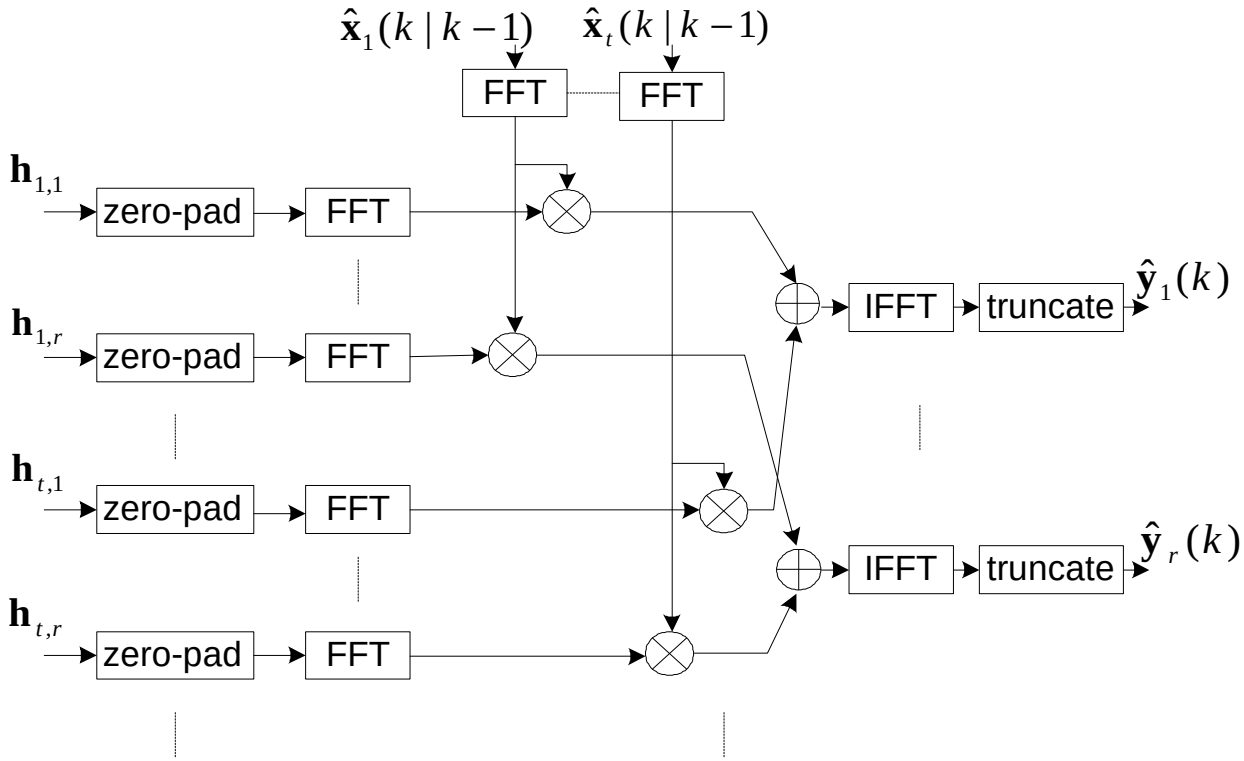


Figure 4: Functional logic block diagram of the FFT acceleration for the observation estimate computing architecture.

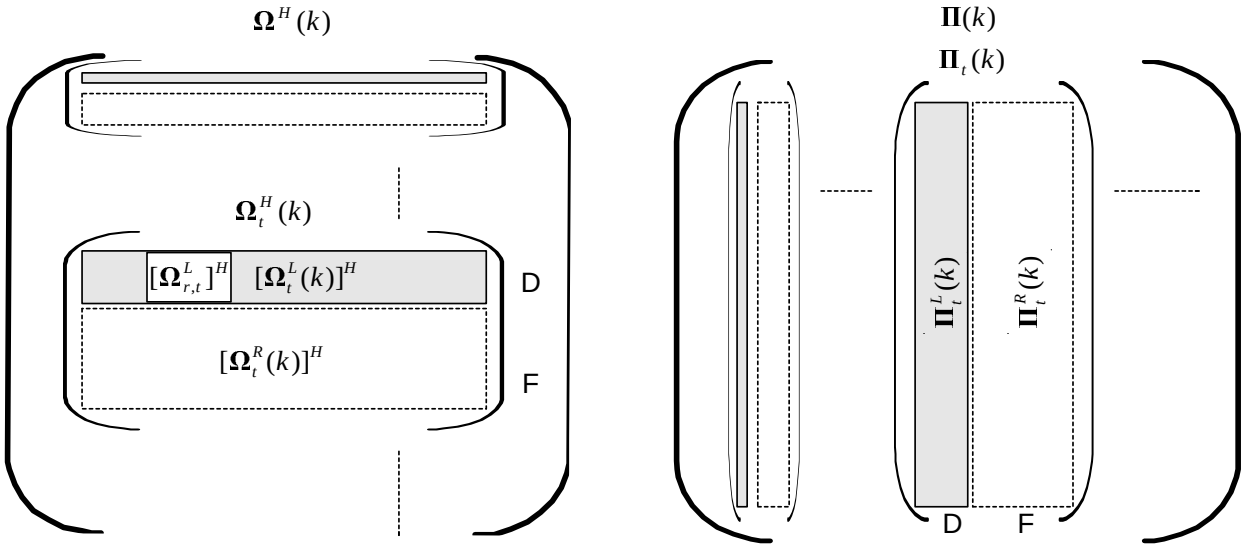


Figure 5: Effective data in the $\mathbf{\Omega}(k)$ and $\mathbf{\Pi}(k)$ matrices.

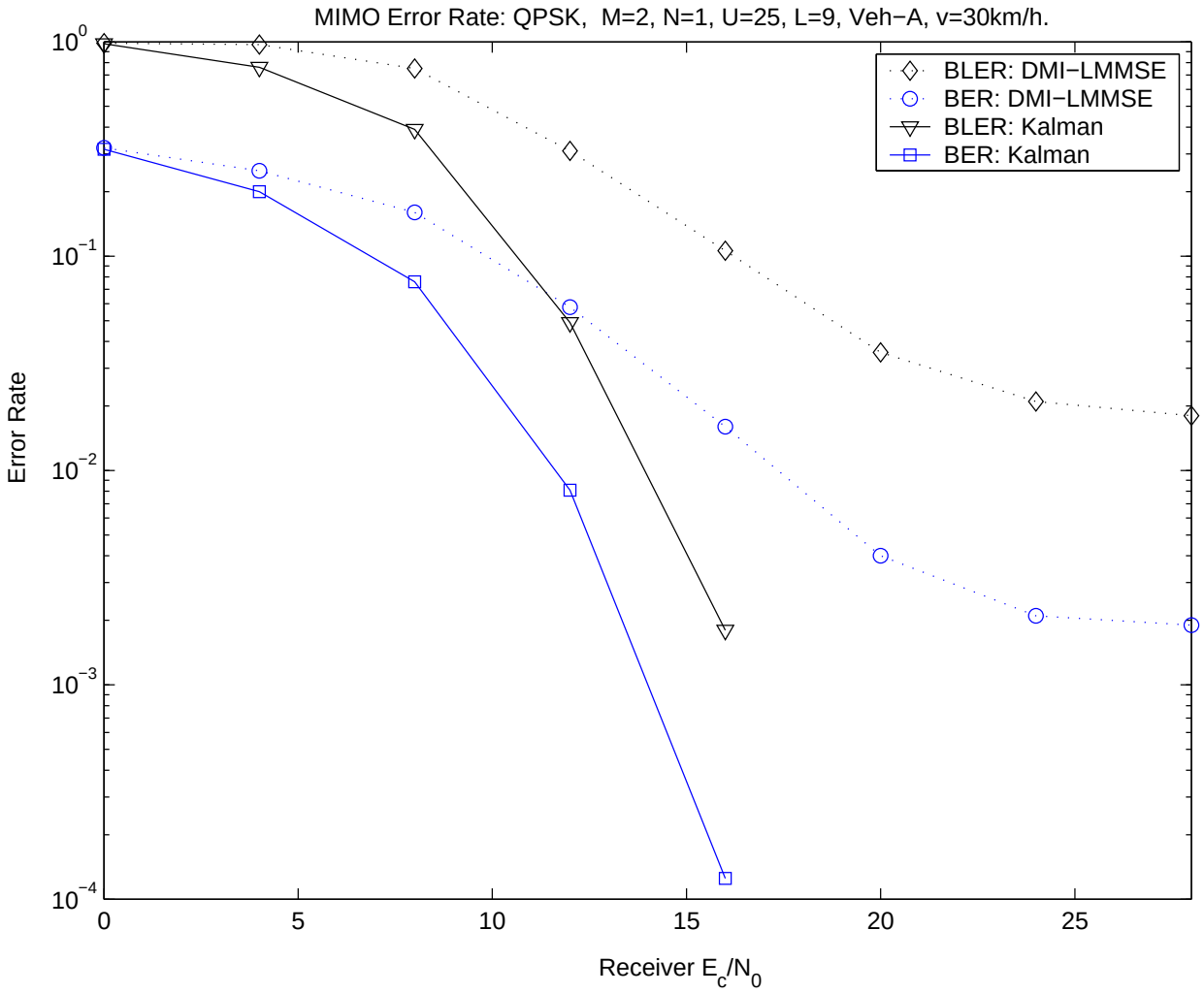


Figure 6: Performance of the QPSK MIMO link with $M = 2, N_r = 1, U = 25, L = 9$, Veh-A at speed of $v = 30km/h$.

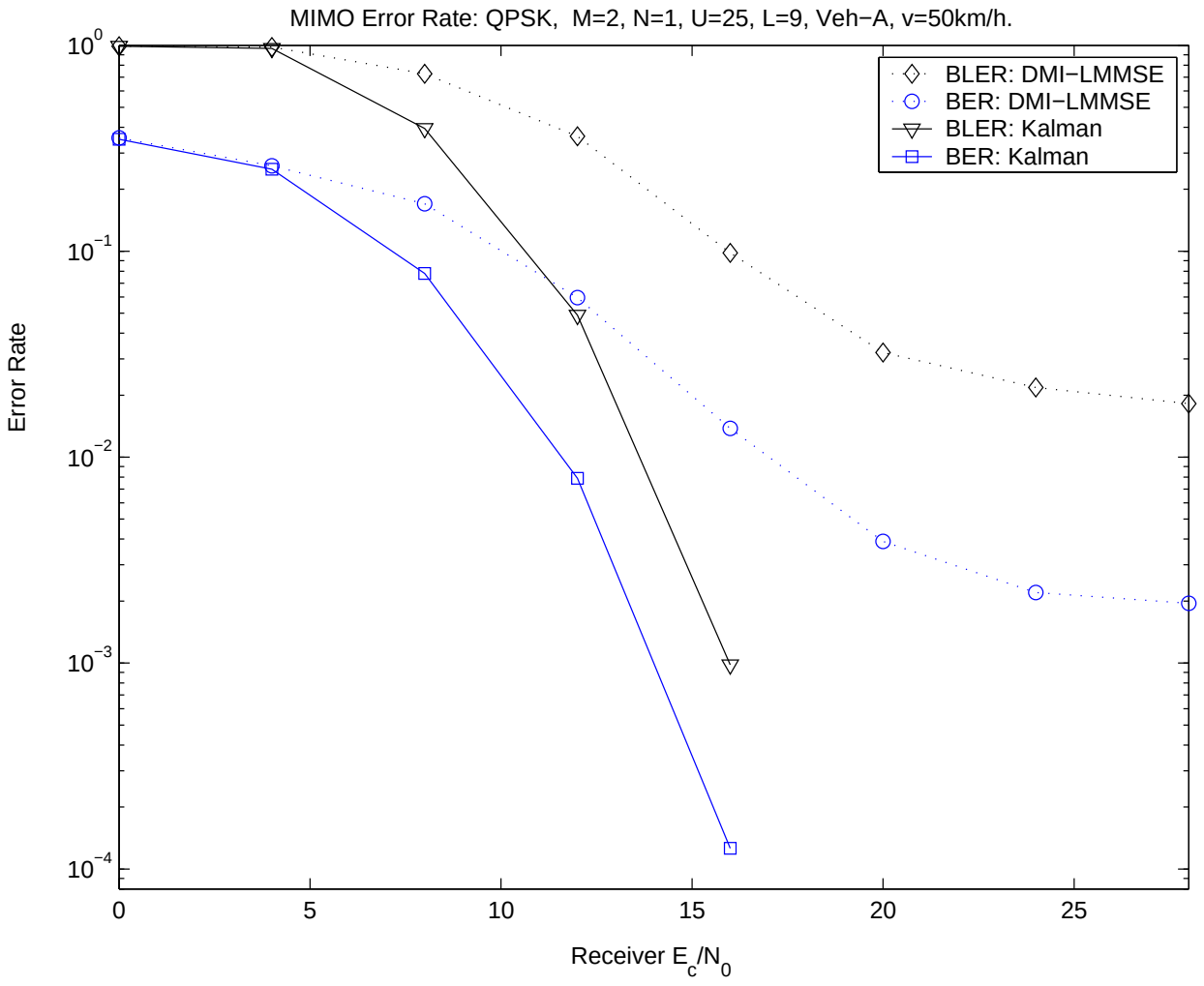


Figure 7: Performance of the QPSK MIMO link with $M = 2, N_r = 1, U = 25, L = 9$, Veh-A at speed of $v = 50km/h$.

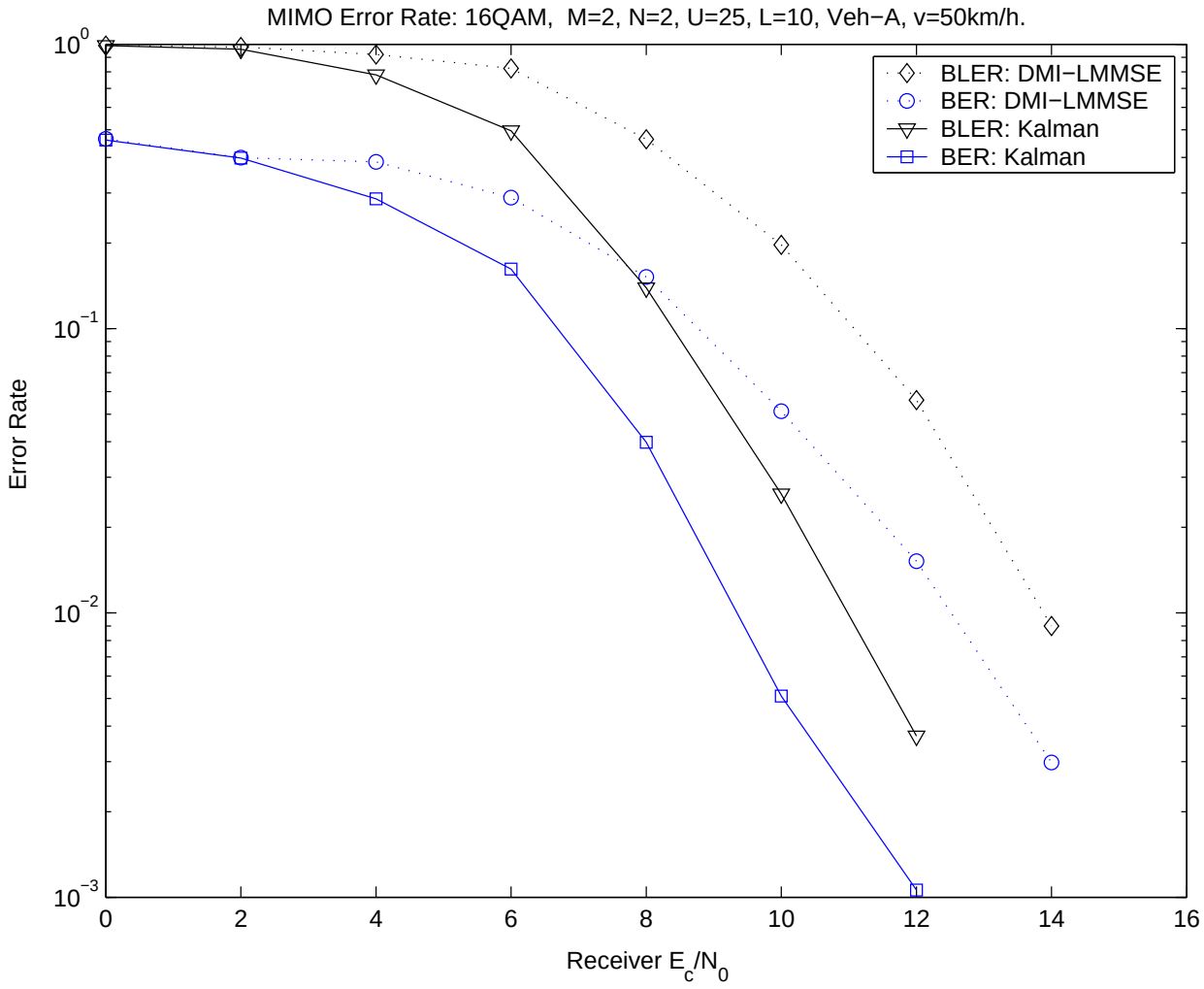


Figure 8: Performance of the 16-QAM MIMO link with $M = 2, N_r = 2, U = 25, L = 10$, Veh-A at speed of $v = 50km/h$.