Hindawi Publishing Corporation
EURASIP Journal on Applied Signal Processing
Volume 2006, Article ID 57134, Pages 1–18
DOI 10.1155/ASP/2006/57134

An Efficient Circulant MIMO Equalizer for CDMA Downlink: Algorithm and VLSI Architecture

Yuanbin Guo,[1] Jianzhong(Charlie) Zhang,[1] Dennis McCain,[1] and Joseph R. Cavallaro[2]

[1] *Nokia Research Center, 6000 Connections Drive, Irving, TX 75039, USA*
[2] *Department of Electrical and Computer Engineering, George R. Brown School of Engineering, Rice University, 6100 Main Street, Houston, TX 77005, USA*

Received 29 November 2004; Revised 5 June 2005; Accepted 14 June 2005

We present an efficient circulant approximation-based MIMO equalizer architecture for the CDMA downlink. This reduces the direct matrix inverse (DMI) of size $(NF \times NF)$ with $\mathcal{O}((NF)^3)$ complexity to some FFT operations with $\mathcal{O}(NF \log_2(F))$ complexity and the inverse of some $(N \times N)$ submatrices. We then propose parallel and pipelined VLSI architectures with Hermitian optimization and reduced-state FFT for further complexity optimization. Generic VLSI architectures are derived for the (4×4) high-order receiver from partitioned (2×2) submatrices. This leads to more parallel VLSI design with $3\times$ further complexity reduction. Comparative study with both the conjugate-gradient and DMI algorithms shows very promising performance/complexity tradeoff. VLSI design space in terms of area/time efficiency is explored extensively for layered parallelism and pipelining with a Catapult C high-level-synthesis methodology.

1. INTRODUCTION

Wireless communication is experiencing radical advancement to support broadband multimedia services and ubiquitous networking via mobile devices. MIMO (multiple-input multiple-output) technology [1–3] using multiple antennas at both the transmitter and receiver has emerged as one of the most significant technical breakthroughs for throughput enhancement. On the other hand, UMTS [4] and CDMA2000 extensions optimized for data services lead to the standardization of multicode CDMA systems such as the high-speed downlink packet access (HSDPA) and its equivalent 1X evolution data and voice/data optimized (EV-DV/DO) standards [5]. This leads to an asymmetric capacity requirement, where the downlink even plays a more essential role than the uplink because of the downloading features. The application of the MIMO technology in CDMA downlink receives increasing interest as a strong candidate for the 3G and beyond wireless communication systems.

Known as D-BLAST [3] and a more realistic strategy as V-BLAST [2] for real-time implementation, the original MIMO spatial multiplexing was proposed for narrowband and flat fading channels. In a multipath fading channel, the orthogonality of the spreading codes is destroyed. This introduces both the multiple-access interference (MAI) and the intersymbol interference (ISI). The conventional Rake receiver [6] could not provide acceptable performance because of the very short spreading gain to support high-rate data services in multicode CDMA downlink. LMMSE (linear-minimum-mean-squared-error)-based chip equalizer is promising to restore the orthogonality of the spreading code and suppress both the ISI and MAI [6] in single-antenna systems. However, this involves the inverse of a large covariance matrix with $\mathcal{O}((NF)^3)$ complexity for MIMO systems, where N is the number of receive antennas and F is the channel length. Traditionally, the implementation of equalizer in hardware has been one of the most complex tasks for receiver designs. The MIMO extension gives even more challenges for real-time hardware implementation [7], especially for the mobile receiver.

To avoid the DMI, adaptive algorithms such as least-mean-square (LMS) algorithm have been studied. However, they suffer from stability problems because the convergence depends on the choice of a good step size [8]. On the other hand, nonadaptive block-based algorithms such as the Levinson and Schur [9, 10] algorithms reduce the complexity to the order of $\mathcal{O}((NF)^2)$. An iterative conjugate gradient (CG) tap solver was proposed in [11, 12] at similar complexity. However, this squared complexity is still very high for efficient real-time implementation. The fact that the downlink receiver must be embedded into a low-cost portable device makes the design of low-complexity equalizer challenging but essential for widespread commercial deployment.

In this paper, we first present an FFT-based fast algorithm for the tap solving by approximating the block Toeplitz structure of the covariance matrix with a block-circulant matrix to avoid the direct matrix inverse. The inverse of the large covariance matrix is reduced to some parallel FFT/IFFT operations and the inverse of some much smaller submatrices. This algorithm reduces the complexity order to $\mathcal{O}(NF\log_2(F))$, which makes the real-time implementation much easier. An algorithmic-level comparative study for different equalizers demonstrates their promising performance/complexity tradeoff.

As real-time implementation is concerned, system-on-chip (SoC) architecture offers more parallelism, more compact size, and lower power consumption than general purpose DSP processors. However, the research for the SoC architectures of MIMO-HSDPA mobile receiver remains a relatively new and hot topic. Recently, Nokia successfully demonstrated a single-antenna HSDPA real-time system in the CTIA'03 wireless trade show [13, 14]. Although MIMO-VLSI implementations have been reported for Lucent's BLAST ASIC chip [15] and some MIMO detection algorithms [16], the VLSI architecture design of MIMO-CDMA equalizers remains a new research topic. To support the MIMO-CDMA downlink in a multipath fading channel, it is necessary to explore the efficient VLSI design architecture [17] for the complex equalizer.

In the second part, we focus on the VLSI-oriented optimizations of the architecture complexity. Hermitian optimization is proposed by utilizing the structures of the correlation coefficients and the FFT algorithm. A reduced-state FFT module is proposed to avoid redundant computation of the symmetric coefficients and the zero coefficients. This reduces both the number and complexity of the conventional FFT module. On the other hand, the matrix inverse of some smaller submatrices of size $(N \times N)$ is inevitable for the MIMO receiver although the $(NF \times NF)$ inverse is avoided. For a high-order MIMO receiver, the complexity still increases dramatically with the number of antennas. Therefore, the Hermitian feature is applied to reduce the submatrix inverse complexity. Of particular interest is the nontrivial (4×4) MIMO configuration. We apply a divide-and-conquer method to partition the (4×4) submatrices into four (2×2) submatrices. The (4×4) matrix inverse is then dramatically simplified by exploring the commonality in a partitioned matrix inverse lemma. Generic VLSI architectures are derived from the special design blocks to eliminate the redundancies in the complex operations. The regulated model facilitates the design of efficient parallel VLSI modules such as "*complex-Hermitian-multiplication*," "*Hermitian inverse*" and "*diagonal transform*." This leads to efficient architectures with 3× further complexity reduction and more parallel and pipelined schematic.

In addition to minimizing the circuit area used, the design needs to work within a time budget. There are many area/time tradeoffs in the VLSI architectures. Extensive architecture tradeoff study provides critical insights into implementation issues that may arise during the product development process. However, this type of SoC design space exploration is extremely time consuming because the standard trial-and-optimize approaches today are usually tied to hand-coded VHDL/Verilog-based methodology [18, 19]. In this paper, we present a Catapult C-based [13] high-level-synthesis (HLS) methodology which integrates several key technologies to explore the VLSI architecture tradeoffs extensively. Extensive design space exploration is enabled by allocating different architecture/resource constraints in a Catapult C architecture scheduler [13]. Synthesizable register-transfer-level (RTL) design is generated from an algorithmic C/C++ fixed-point design, integrated in other downstream flows and validated in a Xilinx FPGA prototyping platform.

The rest of the paper is organized as follows. Section 2 gives the MIMO-CDMA downlink system model. The FFT-based circulant chip equalizer is presented in Section 3. Section 4 presents the system-level partitioning and the VLSI-level complexity optimization. The comparative performance and complexity analysis are presented in Section 5. Finally, Section 6 presents the HLS-based design space exploration and an experimental implementation on FPGA.

2. SYSTEM MODEL FOR MIMO-CDMA DOWNLINK

The system model of the MIMO multicode CDMA downlink with M Tx antennas and N Rx antennas is described in Figure 1. In a multicode CDMA downlink, multiple spreading codes are assigned to a single user to achieve high data rate. By using spatial multiplexing, the high data rate symbols are demultiplexed into KM lower-rate substreams, where K is the number of spreading codes for data transmission. The substreams are divided into M groups, where each substream in the group is spreaded with a spreading code of spreading gain G. Each group is then combined and scrambled with long scrambling codes and transmitted through the mth Tx antenna. The chip-level signal at the mth transmit antenna is given by $d_m(i+j*G) = \sum_{k=1}^{K} s_m^k(j)c_m^k(i) + s_m^P(j)c_m^P(i)$, where j is the symbol index, i is the chip index, and k is the index of the composite spreading code. $s_m^k(j)$ is the jth symbol of the kth code at the mth substream. In the following, we focus on the jth symbol and omit the symbol index for notation simplicity. $c_m^k(i) = c_k(i)c_m^{(s)}(i)$ is the composite spreading code sequence for the kth code at the mth substream, where $c_k(i)$ is the user-specific Hadamard code and $c_m^{(s)}(i)$ is the antenna-specific scrambling long code. $s_m^P(j)$ denotes the pilot symbols at the mth antenna. $c_m^P(i) = c^P(i)c_m^{(s)}(i)$ is the composite spreading code for pilot symbols at the mth antenna. The received chip-level signal at the nth Rx antenna is given by

$$r_n(i) = \sum_{m=1}^{M} \sum_{l=0}^{L_{m,n}} h_{m,n}(l)d_m(i-\tau_l) + z_n(i), \tag{1}$$

where $h_{m,n}(l)$ and $L_{m,n}$ are the lth path channel coefficient and the delay spread between the mth Tx antenna and the nth Rx antenna, respectively. $z_n(i)$ is the additive Gaussian noise at the nth receive antenna.

By packing the received chips from all the receive antennas in a vector $\mathbf{r}(i) = [r_1(i), \ldots, r_n(i), \ldots, r_N(i)]^T$ and collecting the $L_F = 2F + 1$ consecutive chips with center at the ith

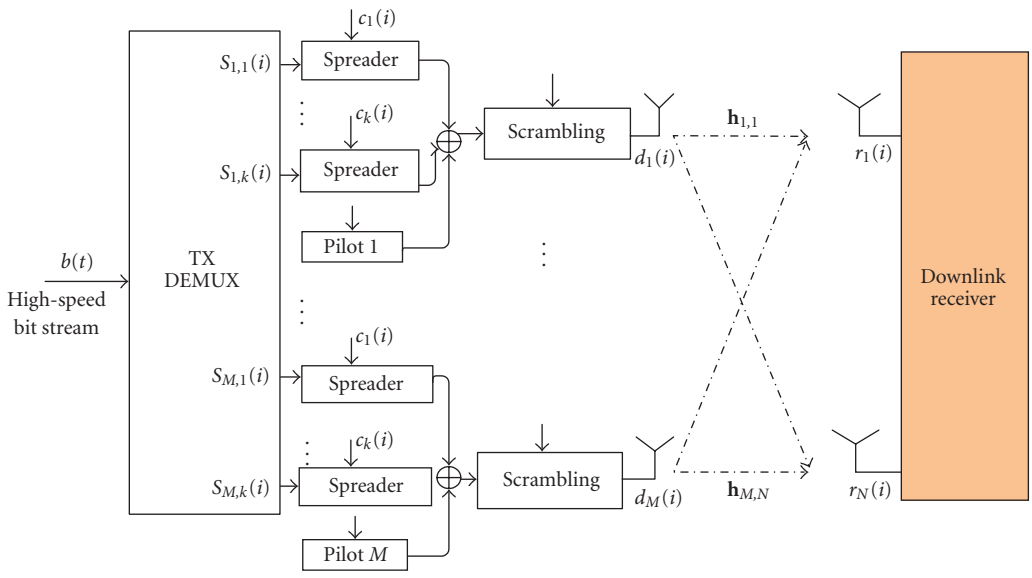


Figure 1: The system model of the MIMO multicode CDMA downlink.

chip from all the N Rx antennas, we form a signal vector as $\mathbf{r}_A = [\mathbf{r}(i+F)^T, \ldots, \mathbf{r}(i)^T, \ldots, \mathbf{r}(i-F)^T]^T$. Here, F is the observation window length corresponding to the channel length. In the vector form, the received signal can be given by

$$\mathbf{r}_A(i) = \sum_{m=1}^{M} \mathbf{H}_m \mathbf{d}_m(i), \tag{2}$$

where $\mathbf{H}_m$ is a block Toeplitz matrix constructed from the channel coefficients as shown in [20]. The multiple receive antennas' channel vector is defined as $\mathbf{h}_m(l) = [h_{m,1}(l), \ldots, h_{m,n}(l), \ldots, h_{m,N}(l)]^T$. The transmitted chip vector for the mth transmit antenna is given by $\mathbf{d}_m(i) = [d_m(i+F), \ldots, d_m(i), \ldots, d_m(i-F-L)]^T$.

3. LMMSE TAP SOLVER WITH CIRCULANT APPROXIMATION

3.1. LMMSE chip equalizer

LMMSE chip-level equalization has been one of the most promising receivers in the single-user CDMA downlink. Chip equalizer estimates the transmitted chip samples by a set of linear FIR filter coefficients $\hat{\mathbf{w}}_m^H(i)$ to restore the code orthogonality as

$$\hat{\mathbf{d}}_m(i) = \hat{\mathbf{w}}_m^H(i)\mathbf{r}_A(i). \tag{3}$$

It is well known that the LMMSE chip equalizer coefficients are given by minimizing the MSE between the transmitted and recovered chip samples as

$$\begin{aligned}\hat{\mathbf{w}}_m^{\text{opt}}(i) &= \arg\min_{\hat{\mathbf{w}}_m(i)} \mathrm{E}\left[\left|\left|\mathbf{d}_m(i) - \hat{\mathbf{w}}_m^H(i)\mathbf{r}_A(i)\right|\right|^2\right] \\ &= \sigma_d^2(i)\hat{\mathbf{R}}_{rr}(i)^{-1}\hat{\mathbf{h}}_m(i),\end{aligned} \tag{4}$$

where $\sigma_d^2(i)$ is the transmitted chip power. $\hat{\mathbf{R}}_{rr}(i)$ and $\hat{\mathbf{h}}_m(i)$ are the covariance estimation and channel estimation, respectively. Here, the covariance matrix is estimated by the time-average with ergodicity assumption as

$$\hat{\mathbf{R}}_{rr}(i) = \mathrm{E}\left[\mathbf{r}_A(i)\mathbf{r}_A^H(i)\right] = \frac{1}{N_B}\sum_{i=0}^{N_B-1} \mathbf{r}_A(i)\mathbf{r}_A^H(i), \tag{5}$$

where N_B is the length for the time average. The channel coefficients are estimated as $\hat{\mathbf{h}}_m(i) = \mathrm{E}[\mathbf{r}_A(i)\mathbf{d}_m^H(i)]$ using the pilot symbols. In the HSDPA standard, about 10% of the total transmit power is dedicated to the common pilot channel (CPICH). This will provide accurate channel estimation. By assuming that the channel is stationary over the observation window length, we can have a block-based operation by omitting the chip index in $\hat{\mathbf{R}}_{rr}(i)$, $\hat{\mathbf{h}}_m(i)$, and $\hat{\mathbf{w}}_m(i)$.

3.2. FFT-based circulant approximation tap solver

Using the stationarity of the channel and the convolution property, it is easy to show that the covariance matrix is a banded block Toeplitz matrix as

$$\mathbf{R}_{rr} = \begin{pmatrix} \mathbf{E}[0] & \cdots & \mathbf{E}^H[L] & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \cdots & \vdots \\ \mathbf{E}[L] & \ddots & \ddots & \cdots & \mathbf{E}^H[L] \\ \vdots & \ddots & \cdots & \vdots & \ddots \\ \mathbf{0} & \cdots & \mathbf{E}[L] & \cdots & \mathbf{E}[0] \end{pmatrix}, \tag{6}$$

where $\mathbf{E}[l]$ is an $(N \times N)$ block matrix with the cross-antenna covariance coefficients. The dimension of the matrix is $(NL_F \times NL_F)$, where L_F is determined by the channel length L. In an outdoor environment, L_F could be up to 32.

The direct inverse of the matrix is very expensive for hardware implementation.

To reduce the computation complexity, an FFT-based fast algorithm is presented in this section. It is known that a circulant matrix $\mathbf{S}$ can be diagonalized by the FFT operation as $\mathbf{S} = \mathbf{D}^H\mathbf{\Lambda}\mathbf{D}$, where $\mathbf{D}$ is the FFT phase coefficient matrix and $\mathbf{\Lambda}$ is a diagonal matrix whose diagonal elements are the FFT result of the first column of the circulant matrix $\mathbf{S}$. This known lemma is applied to simplify the MIMO equalizer computation dramatically. It is shown that the covariance matrix $\mathbf{R}_{rr}$ can be approximated by a block-circulant matrix after we add two corner matrices as

$$\mathbf{C}_{rr} = \mathbf{R}_{rr} + \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{C}_L^H \\ \vdots & \ddots & \mathbf{0} \\ \mathbf{C}_L & \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{7}$$

where

$$\mathbf{C}_L = \begin{pmatrix} \mathbf{E}^H[L] & \mathbf{0} & \mathbf{0} \\ \vdots & \ddots & \mathbf{0} \\ \mathbf{E}^H[1] & \cdots & \mathbf{E}^H[L] \end{pmatrix}. \tag{8}$$

Using the extension of the diagonalization lemma and the features of *Kronecker* product, the block-circulant matrix can be decomposed as

$$\mathbf{C}_{rr} = (\mathbf{D}^H \otimes \mathbf{I})\left(\sum_{i=0}^{L_F-1} \mathbf{W}^i \otimes \mathbf{E}[i]\right)(\mathbf{D} \otimes \mathbf{I}), \tag{9}$$

where $\mathbf{W} = \mathrm{diag}(1, W_{L_F}^{-1} \cdots W_{L_F}^{-(L_F-1)})$ and $W_{L_F} = e^{j(2\pi/L_F)}$ is the phase factor coefficient for the DFT computation. $\otimes$ denotes the *Kronecker* product. By denoting

$$\mathbf{F} = \left(\sum_{i=0}^{L_F-1} \mathbf{W}^i \otimes \mathbf{E}[i]\right), \tag{10}$$

it can be shown that the final MIMO equalizer taps are computed as the following equation:

$$\hat{\mathbf{w}}_m^{\mathrm{opt}} \approx (\mathbf{D}^H \otimes \mathbf{I}) \cdot \mathbf{F}^{-1} \cdot (\mathbf{D} \otimes \mathbf{I})\hat{\mathbf{h}}_m. \tag{11}$$

$\mathbf{F} = \mathrm{diag}(\mathbf{F}[0], \mathbf{F}[1], \ldots, \mathbf{F}[L_F])$ is a block-diagonal matrix with elements taken from the element-wise FFT of the first column of the block-circular matrix $\mathbf{C}_{rr}$. For an $(M \times N)$ MIMO system, this reduces the inverse of an $(NL_F \times NL_F)$ matrix to the inverse of subblock matrices with size $(N \times N)$.

3.3. System-on-chip (SoC) architecture partitioning

To achieve the real-time implementation, either DSP processors or VLSI architectures could be applied. For example, a multiple-processor architecture using TI's DSP processors has been reported in [21] for the 3G base station implementation. However, the requirement for low power consumption and compact size makes it difficult to use multiple DSPs in a mobile handset to achieve the real-time processing power for the chip-level physical layer design, especially for the MIMO systems. SoC architecture is a major revolution for integrated circuits due to the unprecedented levels of integration and many advantages on the power consumption and compact size. However, the straightforward implementation of the proposed equalizer has many redundancies in computation. Many optimizations are needed to make it more suitable for real-time implementation. We emphasize the interaction between architecture, system partitioning, and pipelining in this section with these objectives: (1) propose VLSI-oriented optimizations to further reduce the computation complexity; (2) implement the equalizer with the minimum hardware resource to meet the real-time requirement; (3) obtain an efficient architecture with optimal parallelism and pipelining for the critical computation parts. To explore the efficient architectures, we elaborate the tasks as the following procedure.

(1) Compute the independent correlation elements $[\mathbf{E}[0], \ldots, \mathbf{E}[L]]$, and form the first block column of circulant $\mathbf{C}_{rr}^{(1)}$ by adding the corner elements as $\mathbf{C}_{rr}^{(1)} = [\mathbf{E}[0], \ldots, \mathbf{E}[L], \mathbf{0}, \ldots, \mathbf{0}, \mathbf{E}^H[L], \ldots, \mathbf{E}^H[1]]^T$. Each element is an $(N \times N)$ subblock matrix.

(2) Take the element-wise FFT of $\mathbf{C}_{rr}^{(1)}$, where the element vectors $\mathbf{F}_{n_1,n_2} = \mathrm{FFT}\{\mathbf{E}_{n_1,n_2}^{(c)}\}$ and $\mathbf{E}_{n_1,n_2}^{(c)}(i) = \mathbf{C}_{rr}^{(1)}[(n_1 - i - 1) * N + n_2 - 1]$ for $i \in [0, L_F]$, $n_1, n_2 \in [1, N]$.

(3) For $m \in [1, M]$ and $n \in [1, N]$, compute the dimension-wise FFT of the channel estimation as $\mathbf{\Phi}_m = (\mathbf{D} \otimes \mathbf{I})\hat{\mathbf{h}}_m = \mathrm{FFT}([0, \ldots, 0, h_{m,n}(L), \ldots, h_{m,n}(0), 0, \ldots, 0])$.

(4) Compute the inverse of the block-diagonal matrix $\mathbf{F}$, where $\mathbf{F}^{-1} = \mathrm{diag}(\mathbf{F}[0]^{-1}, \ldots, \mathbf{F}[L_F - 1]^{-1})$ and $\mathbf{F}[i]$ is an $(N \times N)$ submatrix formed from the ith subcarrier of $\mathbf{F}_{n_1,n_2}$.

(5) Compute the matrix multiplication of the submatrices inverse with the FFT output of channel estimation coefficients $\mathbf{\Psi}_m = \mathbf{F}^{-1}\mathbf{\Phi}_m$.

(6) Compute the dimension-wise IFFT of the multiplication results $\hat{\mathbf{w}}_m^{\mathrm{opt}} \approx (\mathbf{D}^H \otimes \mathbf{I})\mathbf{\Psi}_m$.

With a timing- and data-dependency analysis, the top-level block diagram for the MIMO equalizer is shown in Figure 2. The system-level pipeline is designed for better modularity. In the front end, a correlation estimation block takes the multiple-input samples for each chip to compute the covariance coefficients of the first column of $\mathbf{R}_{rr}$. It is made circulant by adding corner to form the matrix $[\mathbf{E}[0], \ldots, \mathbf{E}[L], \mathbf{0}, \ldots, \mathbf{0}, \mathbf{E}[L]^H, \ldots, \mathbf{E}[1]^H]$. The complete coefficients are written to DPRAMs and the $(N \times N)$ element-wise FFT module computes $[\mathbf{F}[0], \ldots, \mathbf{F}[L_F]] = \mathrm{FFT}\{\mathbf{E}[0], \ldots, \mathbf{E}[L], \mathbf{0}, \ldots, \mathbf{0}, \mathbf{E}[L]^H, \ldots, \mathbf{E}[1]^H\}$.

Another parallel data path is for the channel estimation and the $(M \times N)$ dimension-wise FFTs on the channel coefficient vectors as in $(\mathbf{D} \otimes \mathbf{I})\hat{\mathbf{h}}_m$. A submatrix inverse and multiplication block takes the FFT coefficients of both the channel estimation and correlation estimation coefficients from DPRAMs and carries out the computation as in $\mathbf{F}^{-1}\mathbf{\Phi}_m$. Finally, an $(M \times N)$ dimension-wise IFFT module generates the results for the equalizer taps $\hat{\mathbf{w}}_m^{\mathrm{opt}}$ and sends them to the

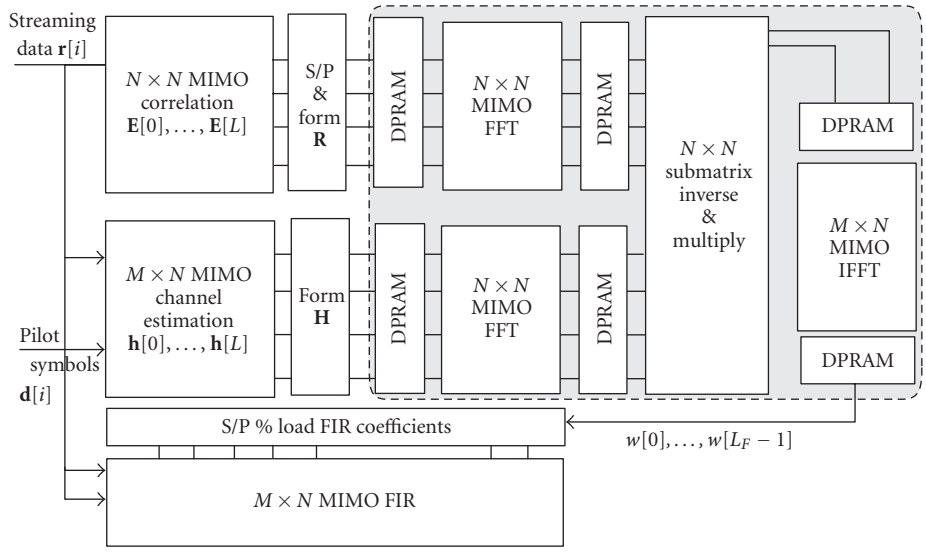


Figure 2: The block diagram of the VLSI architecture of the FFT-based MIMO equalizer.

$(M \times N)$ MIMO-FIR block for filtering. To reflect the correct timing, the correlation and channel estimation modules at the front end will work in a throughput mode on the streaming input samples. The FFT-inverse-IFFT modules in the dotted line block construct the postprocessing of the tap solver. They are suitable to work in a block mode using dual-port RAM blocks as interface between blocks. The MIMO-FIR filtering will also work in throughput mode on the buffered streaming input data.

4. VLSI-LEVEL COMPLEXITY OPTIMIZATION

4.1. Hermitian optimization

In this section, more emphasis is given to the VLSI-oriented implementation aspects. For QPSK and QAM modulation schemes, all the numerical computations in the algorithm are associated with complex numbers. However, the complexity in the hardware is reflected by the number of real multiplications, additions, and divisions, and so forth. It is more accurate to clarify the complexity for different types of computations. For example, a general "*complex* $(a) \times$ *complex* (b)" numerical computation has 4 real multiplications and 2 real additions, but a "*complex* $(a) \times$ *conjugate* (a)" reduces to only 2 real multiplications and 1 real addition. By defining $\mathbf{F}_{n_1,n_2}[0 : L_F - 1]$ as the element-wise FFT vector of the covariance block-vector for $n_1, n_2 \in [1, N]$, we show that the element-wise FFT of the circulant covariance vectors admits a Hermitian structure. This leads to the following lemmas for complexity reduction.

Lemma 1 (Hermitian). $\mathbf{F}_{n_1,n_2} = \mathrm{conj}(\mathbf{F}_{n_2,n_1})$, *where the vector is formed from the covariance element vector between antennas* n_1 *and* n_2. $\mathbf{F}_{n_2,n_1}$ *is redundant for* $n_2 < n_1$.

Lemma 2 (Hermitian complexity). *The computation of* $\mathbf{F}_{n_1,n_1}$ *can be reduced to only* L/L_F *of the full DFT module.*

Proof. For the Rx antennas n_1, n_2, it can be shown that the elements in the circulant column have the following relations, where N_B is the covariance time-average window length:

$$\begin{aligned}
\mathbf{E}^{(c)}_{n_1,n_2}(0) &= \left(\mathbf{E}^{(c)}_{n_2,n_1}(0)\right)^* = \sum_{i=0}^{N_B-1} r_{n_1}(i) r_{n_2}(i)^*, \\
\mathbf{E}^{(c)}_{n_1,n_2}(l) &= \left(\mathbf{E}^{(c)}_{n_2,n_1}(L_F - l)\right)^* = \sum_{i=0}^{N_B-1} r_{n_1}(i) r_{n_2}(i+l)^*, \\
\mathbf{E}^{(c)}_{n_1,n_2}(L_F - l) &= \left(\mathbf{E}^{(c)}_{n_2,n_1}(l)\right)^* = \sum_{i=0}^{N_B-1} r_{n_2}(i) r_{n_1}(i+l)^*, \\
\mathbf{E}^{(c)}_{n_1,n_2}(l) &= \left(\mathbf{E}^{(c)}_{n_2,n_1}(L_F - l)\right)^* = 0 \quad \text{otherwise.}
\end{aligned} \tag{12}$$

Using the features of the FFT, it can be proven that the element-wise FFT results have the relation that $\mathbf{F}_{n_1,n_2} = (\mathbf{F}_{n_2,n_1})^*$. The submatrix formed by the ith entry of $\mathbf{F}_{n_1,n_2}$ is an $(N \times N)$ Hermitian symmetric matrix as $\mathbf{F}(i) = (\mathbf{F}_{n_1,n_2}(i))_{N\times N} = \mathbf{F}(i)^H$.

Thus, instead of having $N \times N$ complex FFT computations, we only need to compute the element-wise FFT for the lower triangle matrix. The number of FFTs in the element-wise FFT is reduced from N^2 to $(N^2 + N)/2$. Moreover, the element-wise FFT coefficients of the diagonal elements are all real numbers. This leads to the design of the reduced-state MIMO-FFT blocks. □

4.2. Reduced-state FFT

Because the FFT algorithm applies the features of the rotation coefficients, the application of the Hermitian feature to FFT is not straightforward. Here, we derive the VLSI-level optimization for the reduced-state FFT with pruning operations based on the standard radix-2 decimation-in-time (DIT) FFT algorithm. Notice that in the standard butterfly

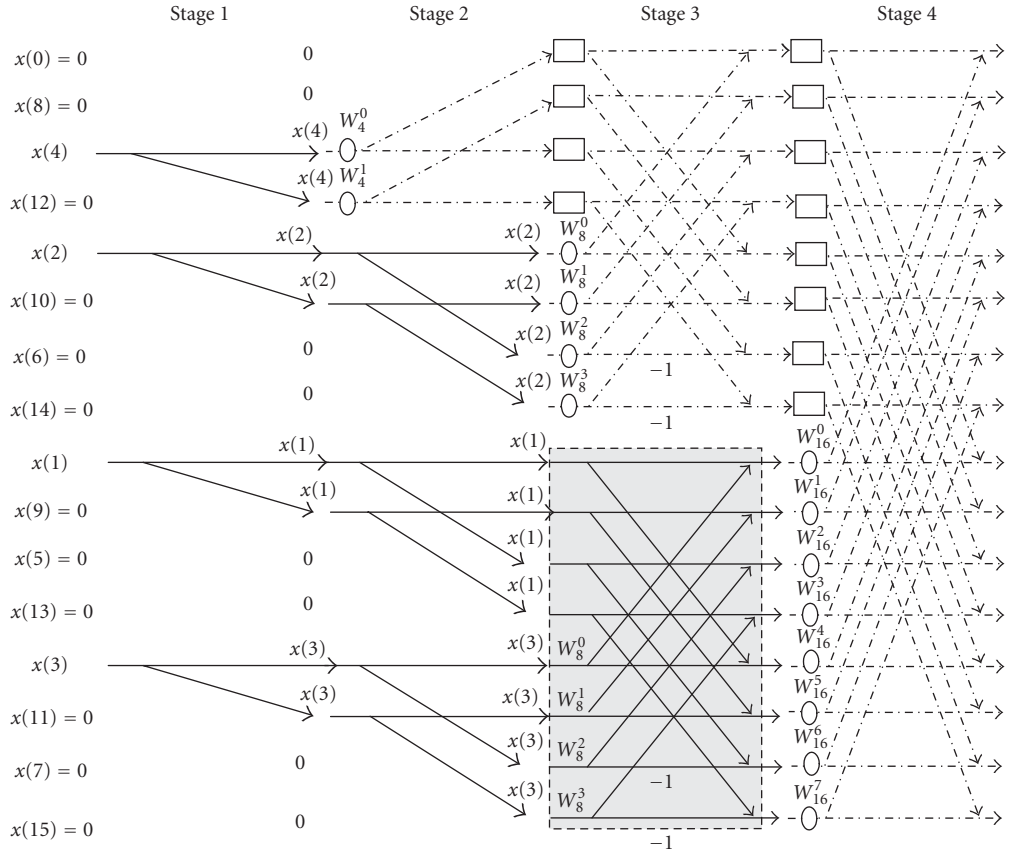


Figure 3: Reduced-state FFT butterfly tree.

unit, each operation involves a full complex multiplication, which has 4 real multiplications and 2 real additions. Since the kth subcarrier of the $\mathbf{F}_{mm}$ vector is

$$f_{mm}(k) = e_{mm}(0) + 2\Re\left(\sum_{i=1}^{L} e_{mm}(i) W_{L_F}^{-ki}\right), \tag{13}$$

by defining the input sequence to the FFT module as $\{x(i)\} = [0, e_{mm}(1), \ldots, e_{m,m}(L), 0, \ldots, 0]$, we only need to compute the real part FFT of the $x(i)$ to get $f_{mm}(k)$. From the butterfly decomposition, we have the recursion for the real-part FFT computation as

$$\begin{aligned} \Re(X(k)) &= \Re(X_1(k)) + \Re(W_{L_F}^{k} X_2(k)), \\ \Re\left(X\left(k + \frac{L_F}{2}\right)\right) &= \Re(X_1(k)) - \Re(W_{L_F}^{k} X_2(k)) \end{aligned} \tag{14}$$

for $k = 0, 1, \ldots, L_F/2 - 1$. This reduces the complex multiplication and addition to only real multiplication and addition for one stage. The butterfly unit becomes a reduced-state partial-butterfly-unit (PBFU) as the dotted line units shown in Figure 3 for an example of 16-point FFT.

From the recursion, it can be shown that we can prune the redundant computations by replacing the complex multiplication in the butterfly units for some portion of the FFT BFU tree. Before considering the many zeros in the input coefficients, the total number of PBFU is $L_F - 1$. Since the total number of BFU is $(L_F/2)\log_2 L_F$, the total number of full-BFU (FBFU) is given by $(L_F/2)\log_2 L_F - L_F + 1$. Considering that $x(i) \neq 0$ only for $i \in [1, L]$, $L < L_F/2$, we can further truncate the computations related to the zero values. After pruning all the unnecessary BFU branches, the FBFUs and PBFUs only take effects from stage 3. The number of FBFU is reduced to $(L_F/2) * \log_2 L_F - 2L_F + 6$. This also reduces the number of memory access and register files for stage 1 and stage 2 as well as in the partial BFUs. The final data flow is shown as the BFU tree in Figure 3. In the figure, only the shaded portion has full-BFUs. Table 1 summarizes the required operations in terms of the real multiplications/additions and memory read/write. In the table, RS-FFT indicates the reduced-state FFT- and ZP-means zero pruning. Although the saving diminishes when the length of FFT increases to a very large number, the RS-FFT with ZP

Table 1: Complexity comparison for different FFT schemes.

	Real mult	Real add
Full FFT	$2L_F \log_2 L_F$	$L_F \log_2 L_F$
RS-FFT w/o ZP	$2N \log_2 L_F - 2L_F + 2$	$L_F \log_2 L_F - 2L_F + 2$
RS-FFT with ZP	$2L_F \log_2 L_F - 6L_F + 12$	$L_F \log_2 L_F - 4L_F + 12$

saves roughly 50% of the real multiplications because the FFT length within 64 points will suffice for most realistic equalizer applications.

4.3. Hermitian matrix inverse architectures

In this section, we utilize the Hermitian feature and focus on the optimization of the submatrix inverse and multiplication module following the element-wise FFT modules in the tap solver. Although the FFT-based tap solver avoids the direct matrix inverse of the original covariance matrix with the dimension of $(NF \times NF)$, the inverse of the diagonal matrix $\mathbf{F}$ is inevitable. For a MIMO receiver with high receive dimension, the matrix inverse and multiplication in $\mathbf{F}^{-1}\hat{\mathbf{h}}_m$ is not trivial. Because $\mathbf{F}$ is a block-diagonal matrix, its inverse can be decomposed to the inverse of L_F submatrices of size $(N \times N)$ as in

$$\mathbf{F}^{-1} = \operatorname{diag}\left(\mathbf{F}[0]^{-1}, \mathbf{F}[1]^{-1}, \ldots, \mathbf{F}[L_F - 1]^{-1}\right). \tag{15}$$

A traditional $(N \times N)$ matrix inverse using Gaussian elimination has the complexity at $\mathcal{O}(N^3)$ complex operations. Cholesky decomposition can be applied to facilitate the inverse of these matrices. However, this method requires arithmetic square root operation, which is expensive for hardware implementation. Considering the fact that it is unlikely to have more than four Rx antennas in a mobile terminal, we consider the two special cases individually, that is, 2 and 4 Rx antennas. We propose complexity-reduction schemes and efficient architectures suitable for VLSI implementation based on the exploration of block partitioning. The commonality of the partitioned block matrix inverse is extracted to design generic RTL modules for reusable modularity. We then build the (4×4) receiver by reusing the (2×2) block partitioning.

4.3.1. Dual-antenna MIMO receiver

From (11), a straightforward partitioning is at the matrix inversion of $\mathbf{F}$ and then the matrix multiplication of the dimension-wise FFT of the channel coefficients as $\mathbf{F}^{-1}(\mathbf{D} \otimes \mathbf{I})\hat{\mathbf{h}}_m$. In this partitioning, we would first compute the inverse of the entire subblock matrix in $\mathbf{F}$ and then carry out a matrix multiplication. However, this partitioning involves two separate loop structures. In the VLSI circuit design, this will introduce some overhead for memory access and finite-state machine logic. Since the two steps have the same loop structure, it is more desirable to merge the two steps and reduce the overhead shown as follows. The inverse of a (2×2) submatrix is given by

$$\begin{aligned}\mathbf{F}[k]^{-1} &= \begin{pmatrix} f_{00}(k) & f_{01}(k) \\ f_{10}(k) & f_{11}(k) \end{pmatrix}^{-1} \\ &= \frac{1}{f_{00}(k)f_{11}(k) - f_{01}(k)f_{10}(k)} \begin{pmatrix} f_{11}(k) & -f_{01}(k) \\ -f_{10}(k) & f_{00}(k) \end{pmatrix}.\end{aligned} \tag{16}$$

Let $\boldsymbol{\Gamma} = (\mathbf{D} \otimes \mathbf{I})\hat{\mathbf{h}}_m = [\boldsymbol{\Gamma}[0], \boldsymbol{\Gamma}[1], \ldots, \boldsymbol{\Gamma}[L_F - 1]]$, where $\boldsymbol{\Gamma}[k] = [e_1(k)\ e_2(k)]$ is the combination of the kth elements of the dimension-wise FFT coefficients, then a merged computation of the matrix inverse and multiplication is given by

$$\begin{aligned}\mathbb{W} &= \mathbf{F}^{-1} \cdot (\mathbf{D} \otimes \mathbf{I})\hat{\mathbf{h}}_m \\ &= \operatorname{diag}\left(\mathbf{F}[0]^{-1}, \mathbf{F}[1]^{-1}, \ldots, \mathbf{F}[L_F - 1]^{-1}\right)\boldsymbol{\Gamma} \\ &= \left[\mathbf{F}[0]^{-1}\boldsymbol{\Gamma}[0]^T, \mathbf{F}[1]^{-1}\boldsymbol{\Gamma}[1]^T, \ldots, \mathbf{F}[L_F - 1]^{-1}\boldsymbol{\Gamma}[L_F - 1]^T\right].\end{aligned} \tag{17}$$

Thus, we can use a single merged loop to compute the final result of $\mathbb{W}$ instead of using separate loops. Moreover, with the *Hermitian* features of $\mathbf{F}_{00}$ and $\mathbf{F}_{11}$, we can reduce the number of real operations in the matrix inverse and multiplication module. This leads to a simplified equation for the kth element of the matrix $\mathbb{W}$ as

$$\begin{aligned}\mathbb{W}(k) = &\frac{1}{f_{00}(k) \cdot f_{11}(k) - |f_{01}(k)|^2} \\ &\cdot \begin{pmatrix} f_{11}(k) \circ e_1(k) - f_{01}(k) * e_2(k) \\ -f_{10}(k) \circ e_2(k) - f_{01}(k) * e_1(k) \end{pmatrix},\end{aligned} \tag{18}$$

where "$a \cdot b$," "$a \circ b$," and "$a * b$" indicate a "real $\times$ real," "real $\times$ complex" and "complex $\times$ complex" multiplication, respectively. The complex division is replaced by a real division. From this, we derived the simplified data path with the Hermitian optimization as in Figure 4. In this figure, $f_{00}(k)$ and $f_{11}(k)$ are real numbers. The single multiplier means a real multiplication. The multiplier with a circle means the "real $\times$ complex" multiplication and the multiplier with a rectangle is a "complex $\times$ complex" multiplication. The simplified data path facilitates the scaling, and thus increases the stability in the fixed-point implementation.

4.3.2. Receiver with 4 Rx antennas

The principle operation of interest is the inverse of the (4×4) submatrices. To achieve a scalable design, we first partition the (4×4) submatrices in $\mathbf{F}[i]$ into four (2×2) block sub matrices as

$$\begin{aligned}\mathbf{F}(i)_{4\times4} &= \begin{pmatrix} f_{11}(i) & f_{12}(i) & f_{13}(i) & f_{14}(i) \\ f_{21}(i) & f_{22}(i) & f_{23}(i) & f_{24}(i) \\ f_{31}(i) & f_{32}(i) & f_{33}(i) & f_{34}(i) \\ f_{41}(i) & f_{42}(i) & f_{43}(i) & f_{44}(i) \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{B}_{11}(i) & \mathbf{B}_{12}(i) \\ \mathbf{B}_{21}(i) & \mathbf{B}_{22}(i) \end{pmatrix}.\end{aligned} \tag{19}$$

The inverse of the (4×4) matrix can be carried out by a sequential inverse of four (2×2) submatrices. We also partition the inverse of the (4×4) element matrix as

$$\mathbf{F}(i)^{-1} = \begin{pmatrix} \mathbf{C}_{11}(i) & \mathbf{C}_{12}(i) \\ \mathbf{C}_{21}(i) & \mathbf{C}_{22}(i) \end{pmatrix}. \tag{20}$$

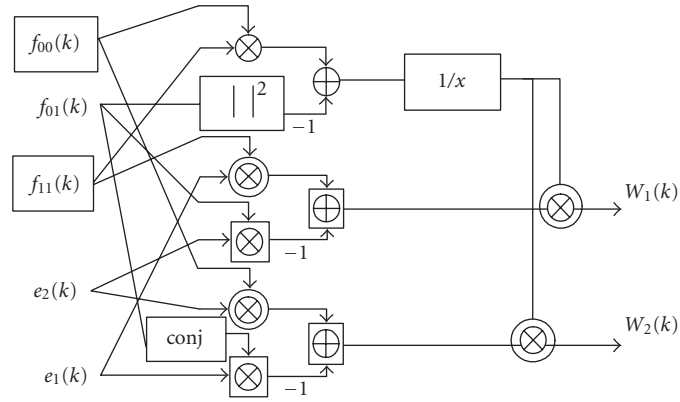


Figure 4: The merged 2×2 inverse and multiplication.

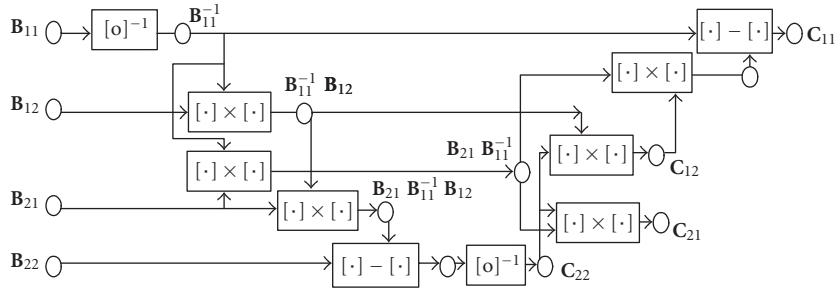


Figure 5: The data path of the partitioned 4×4 matrix inverse for each subcarrier.

It can be shown that the subblocks are given by the following equations from the matrix inverse lemma [22]:

$$\begin{aligned}
\mathbf{C}_{22}(i) &= \left[\mathbf{B}_{22}(i) - \mathbf{B}_{21}(i)\mathbf{B}_{11}(i)^{-1}\mathbf{B}_{12}(i)\right]^{-1},\\
\mathbf{C}_{12}(i) &= -\mathbf{B}_{11}(i)^{-1}\mathbf{B}_{12}(i)\mathbf{C}_{22}(i),\\
\mathbf{C}_{21}(i) &= -\mathbf{C}_{22}(i)\mathbf{B}_{21}(i)\mathbf{B}_{11}(i)^{-1},\\
\mathbf{C}_{11}(i) &= \mathbf{B}_{11}(i)^{-1} - \mathbf{C}_{12}(i)\mathbf{B}_{21}(i)\mathbf{B}_{11}(i)^{-1}.
\end{aligned} \tag{21}$$

Without looking into the data dependency, a straightforward computation will have 8 complex matrix multiplications, 2 complex matrix inverses, and 2 complex matrix subtractions, all of the size (2×2). By examining the data dependency, we will find some duplicate operations in the data path. For a general case before considering the Hermitian structure of the $\mathbf{F}[i]$ matrix, a sequential computation has the data-dependency path given by Figure 5. The raw complexity is given by 6 matrix mult, 2 inverses, and 2 substractions. From the data path flow, the critical path can be identified.

Now we utilize the Hermitian feature of the $\mathbf{F}$ matrix to derive more parallel computing architecture. Because the inverse of a Hermitian matrix is Hermitian, that is, $\mathbf{F}^{-1} = [\mathbf{F}^{-1}]^H$, it can be shown that

$$\begin{aligned}
\mathbf{B}_{11}^{-1}(i) = \left[\mathbf{B}_{11}^{-1}(i)\right]^H &\Longrightarrow \mathbf{C}_{11}(i) = \left[\mathbf{C}_{11}(i)\right]^H,\\
\mathbf{B}_{12}(i) = \left[\mathbf{B}_{21}(i)\right]^H &\Longrightarrow \mathbf{C}_{12}(i) = \left[\mathbf{C}_{21}(i)\right]^H,\\
\mathbf{B}_{22}(i) = \left[\mathbf{B}_{22}(i)\right]^H &\Longrightarrow \mathbf{C}_{22}(i) = \left[\mathbf{C}_{22}(i)\right]^H.
\end{aligned} \tag{22}$$

This leads to the data path by removing the duplicate computation blocks that has the Hermitian relationship. However, this straightforward treatment still does not lead to the most efficient computing architecture. The data path is still constructed with a very long dependency path. To fully extract the commonality and regulate the design blocks in VLSI, we define the following special operators on the (2×2) matrices for the different complex operations. These special operators are mapped to VLSI processing units to deal with the special Hermitian matrix.

Definition 1 (pseudo-power). $\mathrm{pPow}(a, b) = \Re(a) \cdot \Re(b) + \Im(a) \cdot \Im(b)$ is defined as the *pseudo-power* function of two complex numbers and $\Re(a, b) = \Re(a) \cdot \Re(b) - \Im(a) \cdot \Im(b)$ is defined as the real part of a complex multiplication.

Definition 2 (complex-hermitian-mult). For a general (2×2) matrix $\mathbf{A}$ and a Hermitian (2×2) matrix $\mathbf{B} = \mathbf{B}^H$, we define the operator CHM (*Complex-Hermitian-mult*) as

$$M(\mathbf{A}, \mathbf{B}) = \mathbf{AB} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{21}^* \\ b_{21} & b_{22} \end{pmatrix}. \tag{23}$$

Note that all the numbers are complex except $\{b_{11}, b_{22}\} \in \mathbb{R}$.

Definition 3 (Hermitian inverse). For a (2×2) Hermitian matrix $\mathbf{B} = \mathbf{B}^H$, the *Hermitian inverse* (HInv) operator is

defined as

$$\mathrm{HInv}(\mathbf{B}) = \frac{1}{b_{11}b_{22} - |b_{21}|^2}\begin{pmatrix} b_{22} & -b_{21}^* \\ -b_{21} & b_{11}\end{pmatrix}, \tag{24}$$

where there are only real multiplications and divisions.

Definition 4 (diagonal transform). Given the (4×4) Hermitian matrix $\mathbf{A}$ which is partitioned into four subblocks as $\mathbf{A} = \begin{pmatrix}\mathbf{A}_{11} & \mathbf{A}_{21}^H \\ \mathbf{A}_{21} & \mathbf{A}_{22}\end{pmatrix} = \mathbf{A}^H$, the DT *(diagonal transform)* of $\mathbf{A}$ is defined as

$$\begin{aligned} T(\mathbf{A}) &= T(\mathbf{A}_{11}, \mathbf{A}_{21}, \mathbf{A}_{22}) \\ &= \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}\mathbf{A}21^H \\ &= \mathbf{A}_{22} - M(\mathbf{A}_{21}, \mathbf{A}_{11})\mathbf{A}_{21}^H. \end{aligned} \tag{25}$$

With these definitions, we regulate the inverse of the (4×4) Hermitian matrix $\mathbf{F} = \mathbf{F}^H$ into simplified operations on (2×2) matrices. After some manipulation, the partitioned subblock computation equations can be mapped to the following procedure using the defined operators:

$$\begin{aligned} \mathbf{B}_{\mathrm{inv}} &= \mathrm{HInv}\,(\mathbf{B}_{11}) = \mathbf{B}_{\mathrm{inv}}^H; \\ \mathbf{D} &= M(\mathbf{B}_{21}, \mathbf{B}_{\mathrm{inv}}); \\ \mathbf{C}_{22} &= \mathrm{HInv}\,(T(\mathbf{B}_{\mathrm{inv}}, \mathbf{B}_{21}, \mathbf{B}_{22})); \\ \mathbf{C}_{12} &= -M(\mathbf{D}^H, \mathbf{C}_{22}); \\ \mathbf{C}_{11} &= \mathbf{B}_{\mathrm{inv}} + \mathbf{D}^H\mathbf{C}_{22}\mathbf{D} = T(-\mathbf{C}_{22}, \mathbf{D}^H, \mathbf{B}_{\mathrm{inv}}). \end{aligned} \tag{26}$$

The overall computation complexity is reduced to 2 HInv operations, 2 DTs, 1 extra CHM block. Because the sign inverter and the Hermitian formatter $[\cdot]^H$ have no hardware resource at all, the computation complexity is determined by these three generic blocks. The data path of the computation shows the timing relationship between different design modules. This regulated procedure facilitates the design of efficient parallel VLSI modules, whose details are given in the following.

4.3.3. Parallel architecture modules

Now we derive the efficient VLSI modules for the generic M and T operations. Because the operation M is also embedded in the T transform, we need to design the interface so that the computing architecture is reused efficiently. The grouping of computations and the smart usage of interim registers will eliminate the redundancy and give simple and generic interface to the design modules. For a single $M(\mathbf{A}, \mathbf{B})$ module, we define

$$\begin{aligned} \widetilde{\mathbf{D}} &= \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22}\end{pmatrix} = M(\mathbf{A}, \mathbf{B}) \\ &= \begin{pmatrix} a_{11}\circ b_{11} + a_{12} * b_{21} & a_{11} * b_{21}^* + a_{12}\circ b_{22} \\ a_{21}\circ b_{11} + a_{22} * b_{21} & a_{21} * b_{21}^* + a_{22}\circ b_{22}\end{pmatrix}. \end{aligned} \tag{27}$$

To extract the commonality in the M and T operations, we have the following lemma for Hermitian matrix.

Lemma 3 (inverse 4×4). *If $\mathbf{B} = \mathbf{B}^H$ is a (2×2)* Hermitian *matrix, then $\mathbf{ABA}^H$ is also a* Hermitian *matrix, where $\mathbf{A}$ in this lemma is a general (2×2) matrix. The associated computation is given by* 6 complex multiplications(CM)s, 4 complex-real multiplication (CRM)s, 4 $\mathrm{pPow}(a, b)$, *and* $2\Re(a, b)$.

Proof. We extend the computation of $\mathbf{G} = \mathbf{ABA}^H$ as

$$\begin{aligned} \mathbf{G} &= \begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22}\end{pmatrix} = \mathbf{ABA}^H = M(\mathbf{A}, \mathbf{B})\mathbf{A}^H \\ &= \begin{pmatrix} a_{11}\circ b_{11} + a_{12} * b_{21} & a_{11} * b_{21}^* + a12\circ b_{22} \\ a_{21}\circ b_{11} + a_{22} * b_{21} & a_{21} * b_{21}^* + a_{22}\circ b_{22}\end{pmatrix}\cdot\begin{pmatrix} a_{11}^* & a_{21}^* \\ a_{12}^* & a_{22}^*\end{pmatrix}. \end{aligned} \tag{28}$$

We then group the operations for each element as

$$\begin{aligned} g_{11} &= (a_{11}\circ b_{11}) * a_{11}^* + [a_{12} * b_{21} * a_{11}^* + a_{11} * b_{21}^* * a_{12}^*] \\ &\quad + (a_{12}\circ b_{22}) * a_{12}^*, \\ g_{21} &= d_{21} * a_{11}^* + d_{22} * a_{12}^*, \\ g_{12} &= d_{21} * a_{11}^* + d_{22} * a_{12}^*, \\ g_{22} &= (a_{21}\circ b_{11}) * a_{21}^* + [a_{22} * b_{21} * a_{21}^* + a_{21} * b_{21}^* * a_{22}^*] \\ &\quad + (a_{22}\circ b_{22}) * a_{22}^*. \end{aligned} \tag{29}$$

We define the interim registers $\mathrm{tmp}_1 = (a_{11}\circ b_{11})$, $\mathrm{tmp}_2 = (a_{12} * b_{21})$, $\mathrm{tmp}_3 = (a_{12}\circ b_{22})$, $\mathrm{tmp}_5 = (a_{21}\circ b_{11})$, $\mathrm{tmp}_6 = (a_{22} * b_{11})$, $\mathrm{tmp}_7 = (a_{22}\circ b_{22})$. These interim values are added to generate d_{11}, d_{12}, d_{21}, d_{22}. Moreover, instead of having a general complex multiplication, we can employ the special functional components. For example, it is easy to verify that $(a_{11}\circ b_{11}) * a_{11}^* = \mathrm{pPow}(\mathrm{tmp}_1, a_{11})$. By changing the computation order and combining common computations, we can finally show that $\mathbf{G}$ is a Hermitian matrix with the elements given by

$$\begin{aligned} g_{11} &= \mathrm{pPow}\,(\mathrm{tmp}_1, a_{11}^*) + 2\Re\,(\mathrm{tmp}_2, a_{11}^*) + \mathrm{pPow}\,(\mathrm{tmp}_3, a_{12}^*), \\ g_{21} &= d_{21} * a_{11}^* + d_{22} * a_{12}*, \\ g_{12} &= g_{21}^*, \\ g_{22} &= \mathrm{pPow}\,(\mathrm{tmp}_5, a_{21}^*) + 2\Re\,(\mathrm{tmp}_6, a_{21}^*) + \mathrm{pPow}\,(\mathrm{tmp}_7, a_{22}^*). \end{aligned} \tag{30}$$

□

Thus the simplified $M(\mathbf{A}, \mathbf{B})$ RTL module can be designed as in Figure 6, with the input of both real and imaginary parts of $\mathbf{A}$ as $\{a_{11}(r/i), a_{12}(r/i), a_{21}(r/i), a_{22}(r/i)\}$ and only the necessary elements of the *Hermitian* matrix $\mathbf{B}$ as in $\{b_{11}(r), b_{21}(r/i), b_{22}(r)\}$. The output ports include $\{\mathrm{tmp}_1, \mathrm{tmp}_2, \mathrm{tmp}_3, \mathrm{tmp}_5, \mathrm{tmp}_6, \mathrm{tmp}_7\}$. We only need to compute d_{21} and d_{22} to get the $\mathbf{G}$ elements. Built from the simplified $M(\mathbf{A}, \mathbf{B})$ module, the data path RTL module of the

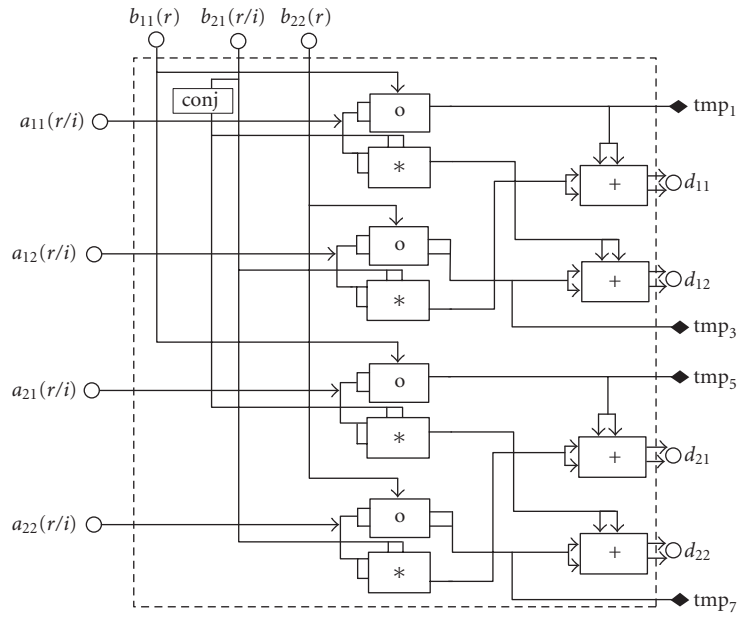


Figure 6: The simplified parallel VLSI RTL layout of the $M(\mathbf{A}, \mathbf{B})$ processing unit.

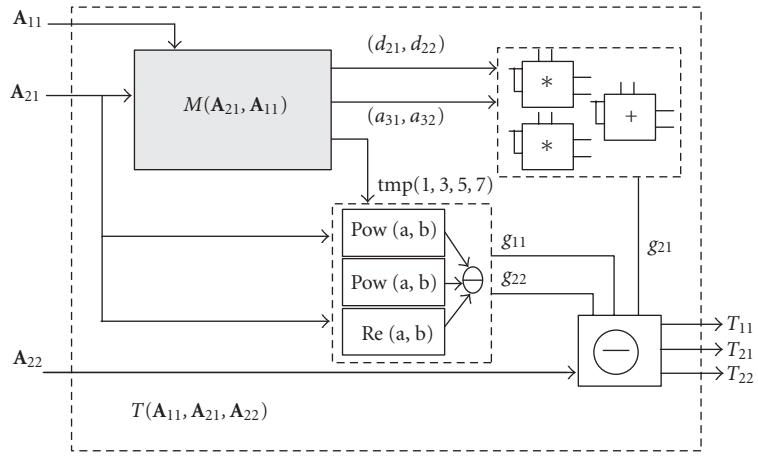


Figure 7: The VLSI RTL architecture layout of the $T(\mathbf{A}_{11}, \mathbf{A}_{21}, \mathbf{A}_{22})$ block.

transform $T(\mathbf{A}_{11}, \mathbf{A}_{21}, \mathbf{A}_{22})$ of the (4×4) Hermitian matrix is given by Figure 7. The output ports of the $T(\mathbf{A}_{11}, \mathbf{A}_{21}, \mathbf{A}_{22})$ include the independent elements $\{t_{11}, t_{21}, t_{22}\}$.

We can further simplify the top-level RTL schematic by extracting the commonality of the M and T module designs as in Figure 8 to eliminate the extra individual M module. Thus, the results of $\mathbf{C}_{11}, \mathbf{C}_{12}$, and $\mathbf{C}_{21}$ are generated together from the second T module. Compared with the design in Figure 5, the architecture demonstrates better parallelism and reduced redundancy. The data path is much better balanced and facilitates the pipelining in multiple subcarriers for high-speed design.

If we use a standard computing architecture of the partitioned (4×4) matrix inverse, we need 308 real multiplications before dependency optimization (DO). With a straightforward DO, the complexity is still 244 real multiplications. Traditionally, a complex multiplication is given by "$c = c_r + jc_i = (a_r + ja_i) * (b_r + jb_i) = (a_r b_r - a_i b_i) + j(a_r b_i + a_i b_r)$." This has 4 real multiplications (RM) and 2 real additions (RA). By rearranging the computation order, we can reduce the number of real multiplications as (1) $p_1 = a_r b_r$, $p_2 = a_i b_i$, $s_1 = a_r + a_i$, $s_2 = b_r + b_i$; (2) $c_r = p_1 - p_2$, $d = (p_1 + p_2)$, $s = s_1 s_2$; (3) $c_i = s - d$. This requires 3 real multiplications and 5 real additions in three steps. A single T transform needs only 38 RMs for a (4×4) Hermitian matrix. Thus, there are 90 RMs to compute the $\mathbf{F}(i)^{-1}$ with the optimized Hermitian architecture. This is only less than 1/3 of the real multiplications for a traditional architecture as shown in Table 2. Note that the critical data path is also dramatically shortened with better modularity and pipelining.

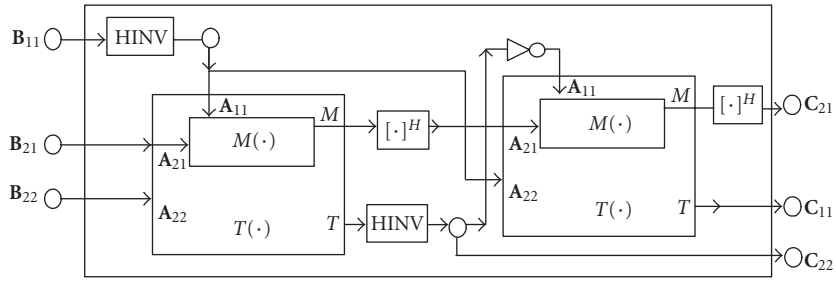


Figure 8: The commonality extracted VLSI design architecture based on T, M, and HINV.

5. COMPARATIVE PERFORMANCE AND COMPLEXITY ANALYSIS

5.1. BER performance

The performance is evaluated in a MIMO-HSDPA simulation chain for different antenna configurations. We compare the performance of four different schemes: the LMS adaptive algorithm, the CG algorithm, the FFT-based algorithm, and the DMI using Cholesky decomposition. We simulated the Pedestrian-A and Pedestrian-B channels following the I-METRA channel model [23], which are typical for high-speed downlink application. The chip rate for the transmit signal is 3.84 Mcps, which is in compliance with the 3GPP standard. The channel state information is estimated from the CPICH at the receiver. Ten percent of the total transmit power is dedicated to the pilot training symbols.

We provide the simulation results for QPSK modulation with antenna configuration in the form of $(M \times N)$. In the figures, L_h is the channel delay spread. Figures 9 and 10 show the fully loaded system for Pedestrian-A and Pedestrian-B channels with (2×2) configuration, while Figure 11 shows a highly loaded system with 10 codes for (2×2) Pedestrian-B channel. Figure 12 shows the simulation results for Pedestrian-A with (4×4) configuration. It can be seen that for Figure 9, the FFT-based algorithm overlaps with both the DMI and the CG at 5 iterations very closely. In a (2×2) case for Pedestrian-B channel, both the CG and FFT-based algorithms show very small divergence from the DMI at the very high SNR range in Figure 10. For a fully loaded system, CG with 5 iterations seems to be slightly better than FFT-based algorithm. But in a case with 10 codes, FFT-based algorithm outperforms the CG for both 3 iterations and 5 iterations. In the (4×4) case as shown in Figure 12, the FFT-based algorithm also outperforms the CG with 5 iterations. However, because the realistic system is most unlikely to work in the very high SNR range, the small difference in the BER performance is negligible. In all cases, the DMI, CG, and FFT-based algorithms significantly outperform the LMS adaptive algorithm.

It should be pointed out that the performance of the LMMSE-based chip equalizer is limited for the fast fading channel because of its block-based feature could not track the fast fading channel environments very well. To deal with this, a Kalman filter-based equalizer has been proposed in one of the authors' papers [24] with much higher complexity. The discussion of the related architecture is out of the scope of this paper.

Table 2: Complexity reduction for submatrix inverse in $\mathbf{F}^{-1}$.

Architecture	RM
Traditional w/o DO(4×4)	$308L_F$
Traditional w/ DO(4×4)	$244L_F$
Hermitian opt(4×4)	$90L_F$

5.2. Complexity

The complexity is a very important consideration for real-time implementation. Although the complete equalizer system consists of the correlation/channel estimation, the tap solver, and the FIR filtering, we focus on the three-tap-solver complexity with similar performance, that is, the DMI, the CG, and the FFT-based algorithm. The other two parts are common for the algorithms presented here. Cholesky decomposition is assumed for the DMI. The complexity is compared in terms of number of equivalent complex multiplications and additions.

For the DMI, the complexity is at the order of $\mathcal{O}((N(F+1))^3)$ for the inverse of $\mathbf{R}_{rr}$ and $\mathcal{O}((N(F+1))^2M)$ for the matrix multiplication in $(\mathbf{R}_{rr})^{-1}\mathbf{h}_m$. For the conjugate gradient algorithm, there are $\mathcal{O}\{MJ[N(F+1)]^2 + M(5J+1)N(F+1)\}$ complex multiplications and $\mathcal{O}\{MJ[N(F+1)]^2 + 8MJN(F+1)\}$ complex additions. Usually, $J = 5$ iterations for the CG algorithm will suffice for convergence near the DMI solution. For the FFT-based algorithm, the overall complexity before Hermitian optimization is $\mathcal{O}\{(N^2 + 2MN)L_F(\log_2 L_F)/2 + (N^3 + MN^2)L_F\}$. With the Hermitian optimization, the complexity reduces to $\mathcal{O}\{(N^2/2 + 2MN)L_F(\log_2 L_F)/2 + (N^3 + MN^2)L_F/2\}$. For the FFT-based algorithm, we usually require $L_F \geq 2F + 1$. The complexity is summarized in Table 3. For simplicity, we only list the most significant part of equivalent number of complex multiplications. An example is given for the (4×4) case with $F = 10$, $J = 5$. The length of FFT $L_F = 32$ will suffice for both Pedestrian-A and Pedestrian-B channels.

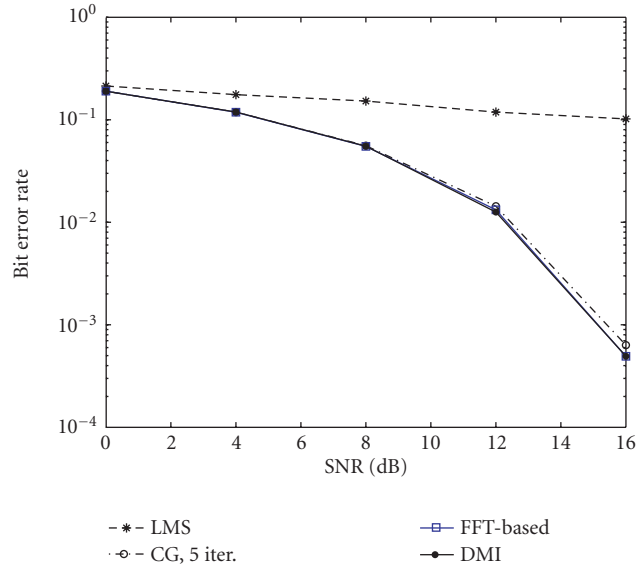


Figure 9: BER performance of 2×2 in Pedestrian-A channel; $K = 14$, $G = 16$, $L_h = 3$, $T = 2$, $M = 2$, and $F = 10$.

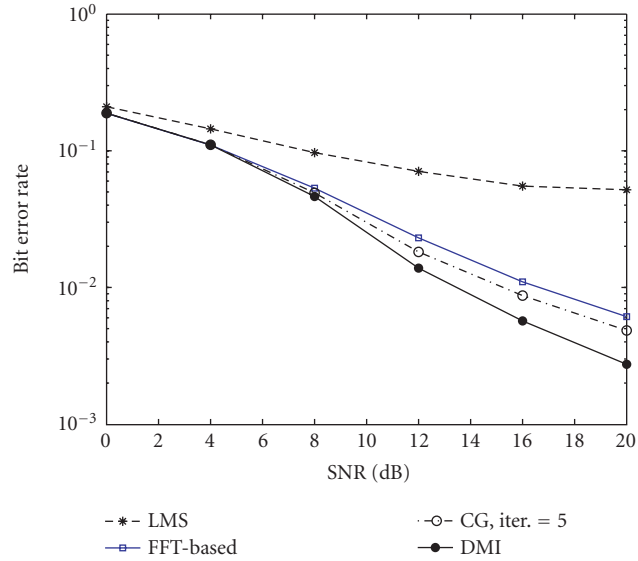


Figure 10: BER performance of 2×2 in Pedestrian-B channel; $K = 14$, $G = 16$, $L_h = 6$, $T = 2$, $M = 2$, and $F = 10$.

In Figure 13, we show the complexity trend for different J and different L_F versus the channel length for a (4×4) system. Although the conjugate gradient algorithm has reduced complexity compared with the DMI, the complexity reduction in the FFT-based algorithm is much more significant.

6. VLSI DESIGN ARCHITECTURE EXPLORATION

6.1. High-level-synthesis architecture scheduling

As a major revolution for the design of integrated circuits, SoC architecture leads to a demand in new methodologies and tools to address design, verification, and test problems in this rapidly evolving area. There are many area/time/power tradeoffs in the VLSI architectures. Extensive study of the different architecture tradeoffs provides critical insights into implementation issues and allows designers to identify the critical performance bottlenecks in meeting real-time requirements. Field-programmable gate array (FPGA) can behave like a number of different ASICs with hardware programmability to study architecture area/time tradeoffs. This makes FPGA a good platform to build, verify, and prototype SoC designs quickly. It has been well accepted as a powerful rapid prototyping platform for the SoC architectures in the literature [13, 25]. A detailed discussion on the tradeoffs using FPGA and DSP for real-time implementation

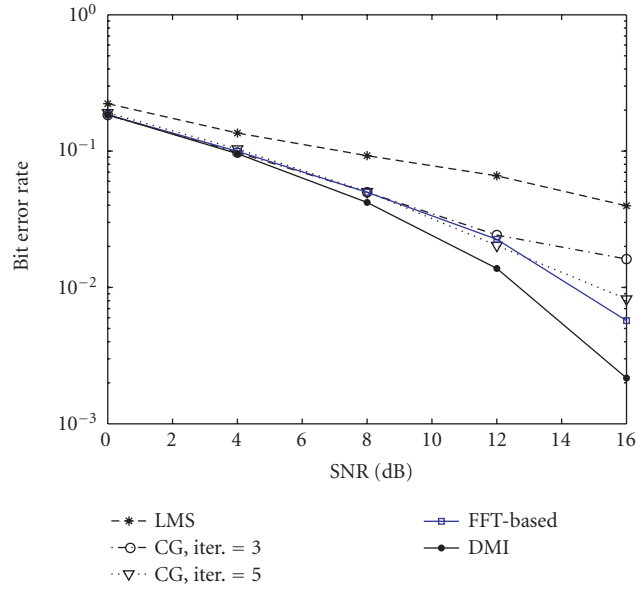


Figure 11: BER performance of 2×2 in Pedestrian-B channel case 2: $K = 10$ codes; $K = 10$, $G = 16$, $L_h = 6$, $T = 2$, $M = 2$, and $F = 10$.

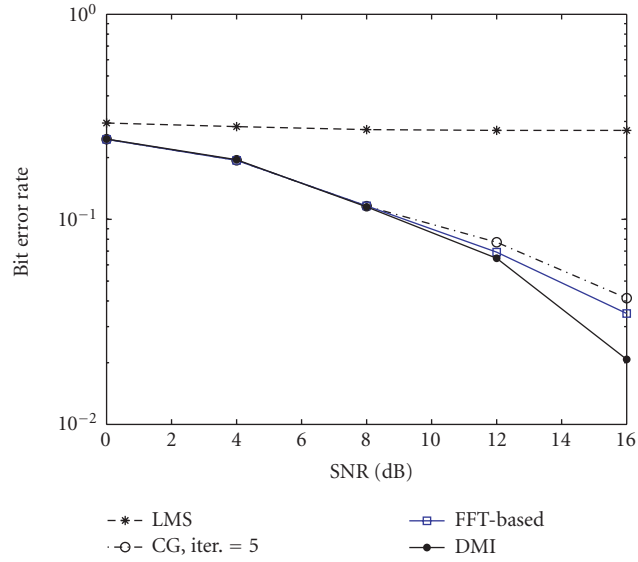


Figure 12: BER performance of 4×4 in Pedestrian-A channel; $K = 12$, $G = 16$, $L_h = 3$, $T = 4$, $M = 4$, and $F = 10$.

is presented in [13].

However, this type of SoC design space exploration is very time consuming because the current standard trial-and-optimize approaches apply hand-coded VHDL/Verilog or graphical schematic tools. In this section, we present a Catapult C-based HLS methodology [26] to explore the VLSI architecture space extensively in terms of the area/time tradeoff. This is enabled with high-level architecture and resource constraints. Synthesizable RTL is generated from a fixed-point C/C++ level design and imported to the graphical tools for module binding. The proposed procedure for implementing an algorithm to the SoC hardware is shown in Figure 14. The number of FUs is assigned according to the time/area constraints. Software resources such as registers and arrays are mapped to hardware components and required finite-state machines (FSMs) necessary for accessing these resources are generated. In this way, we can study several architecture solutions efficiently. In the next step of the design flow, the generated RTL is imported into the HDL environment and integrated with other modules of the system,

Table 3: The overall tap-solver complexity comparison.

	Equivalent complex multiplication	Example
DMI	$\mathcal{O}\{(M+NF)(NF)^2\}$	92928
CG	$\mathcal{O}\{JM[(NF)^2+5NF]\}$	43120
FFT-based	$\mathcal{O}\{[(N^2/2+2MN)\log_2 L_F+(N^3+MN^2)]L_F/2\}$	5248

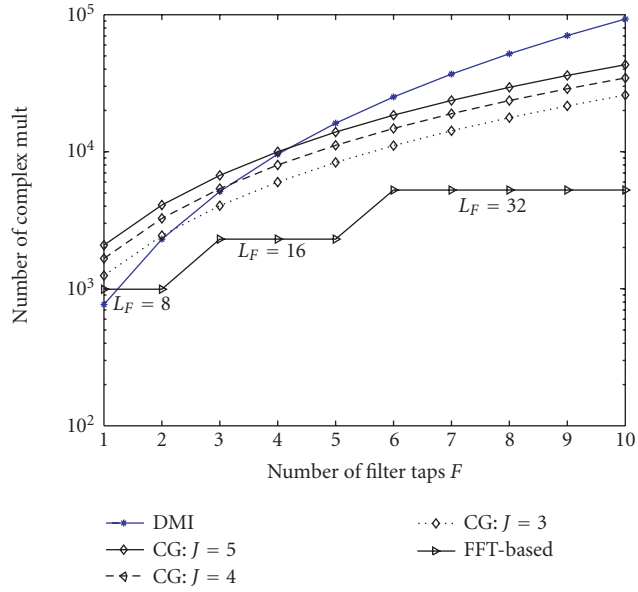


Figure 13: Overall tap-solver complexity comparison; algorithm complexity comparison for $M = 4$, $N = 4$ tap solver.

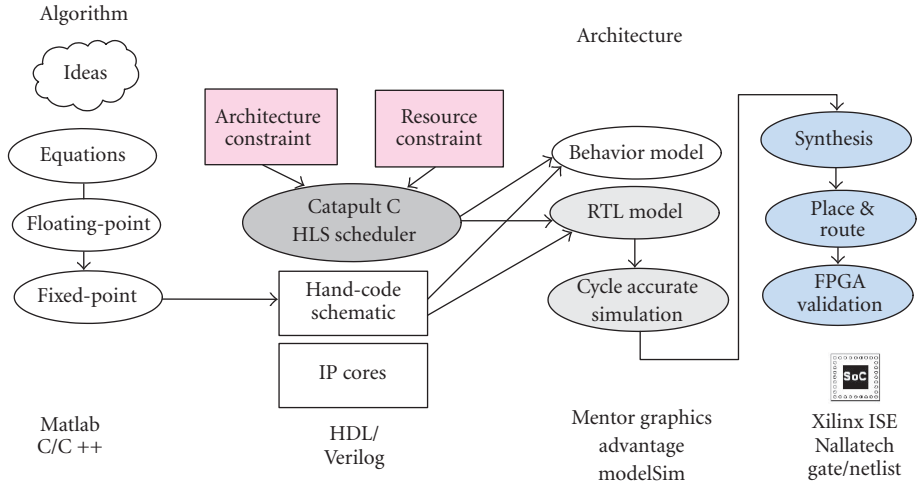


Figure 14: Integrated Catapult C high-level-synthesis design methodology.

which are either another Catapult C design or a legacy IP core. Leonardo spectrum is invoked for gate-level synthesis. Xilinx ISE place & route tool is used to generate gate-level bit-stream file. Raising the language level may lead to concerns about the architecture efficiency, which highly depends on the design tool's capability. To address these concerns, we have compared both the architecture area/time efficiency and the achieved productivity in [13] with the conventional

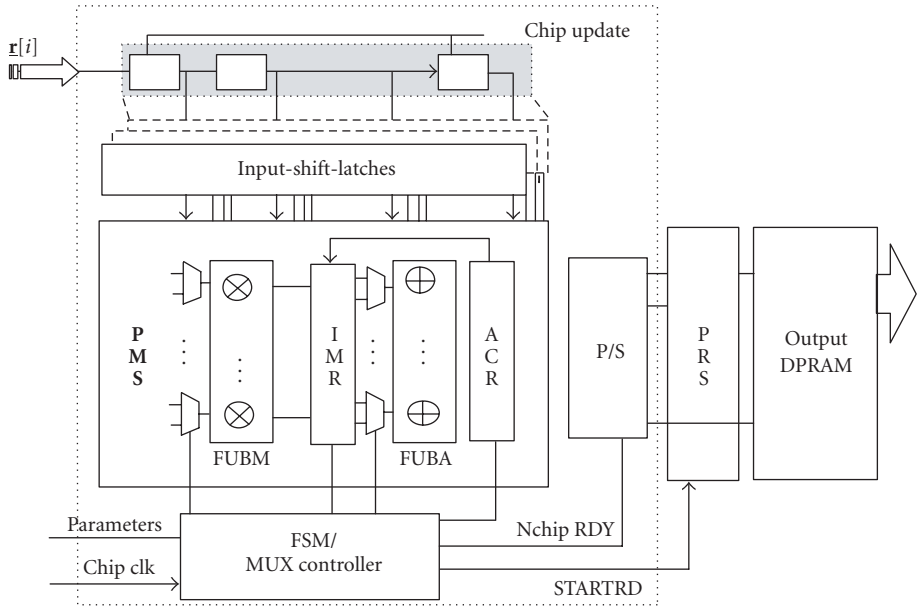


Figure 15: Throughput mode correlation update module using PMS.

design flow. In most cases, the manual tradeoff study of a complex design with hundreds of multipliers could be extremely time consuming and difficult. However, we can almost achieve the most efficient design architecture for a given specification using the architecture scheduling in Catapult C, especially for the computation-intensive algorithms. Compared with the conventional hand-code and schematic-based design methodologies, the Catapult C-based methodology demonstrates not only improved productivity, but also a capability to study the architecture tradeoffs extensively in a short design cycle.

6.2. Real-time VLSI architecture exploration

The complete equalizer includes two major steps: the computation of the equalizer coefficients $\hat{\mathbf{w}}$ and the actual FIR filtering using the updated equalizer taps as in $\hat{\mathbf{w}}^H \mathbf{r}_A(i)$. The update of the equalizer coefficients is a block-based operation depending on the channel varying speed. The FIR filtering depends on the chip rate. Thus, we need to compute the L-tap convolution for each input chip from the N receive antennas for the FIR filtering within $f_{\text{clk}}/f_{\text{chip}}$ cycles, where f_{clk} and f_{chip} are the system clock rate and chip rate, respectively. The WCDMA chip rate is 3.84 MHz. We applied a clock rate of 38.4 MHz for the *Xilinx* Virtex-II V6000-4 FPGA. There will be 10 cycles time constraint per input chip. For the tap solver, the experiment shows that 2 updates per slot are sufficient to provide acceptable performance for slow and median fading channels. Since there are 1920 chips per slot, the latency requirement for each update is 250 microseconds.

We schedule architectures in two basic modes according to the real-time behavior of the subsystem in Catapult C: the throughput mode or the block mode. Throughput mode assumes that there is a top-level main loop for each incoming sample, which is processed immediately in the computation period. The module processes for each input sample periodically, so there is a strict limit for the processing time. Block-mode processes once after a block data is ready. Because the finite-state machine (FSM) usually depends on complex logic and extensive memory access, the computation patten is more like a processor architecture in loading data to the functional units. In the following, we use two typical design modules to demonstrate these different working modes.

6.2.1. Scalable pipelined-multiplexing scheduler

The covariance estimation is computed as

$$\mathbf{R}_{rr} = \left(\frac{1}{N_B}\right) \sum_{i=0}^{N_B-1} \mathbf{r}_A(i)\mathbf{r}_A^H(i) \tag{31}$$

assuming ergodicity. Theoretically, the front-end covariance estimation module can also be designed in block mode similar to a processor implementation. However, this architecture causes a large processing latency and requires big ping pong buffers to store the input samples. For $N_B = 960$ chips per block, the fastest RTL takes more than 6 millisecond latency because the heavy memory access stalls the pipelining and does not provide sufficient parallelism. To meet the real-time requirement, a scalable architecture is designed with throughput mode as in Figure 15. L input-shift-latches (ISLs) shift the new samples and the delayed samples in one cycle. The core is the pipelined-multiplexing scheduler (PMS) with a set of functional-unit banks (FUB) for both multipliers and adders. The temporary values are stored in intermediate-multiplication registers (IMRs) and accumulation-register

Table 4: Architecture tradeoff exploration for covariance estimation module.

Cycles	1	2	3	4–8	9	10
MU(a)	0	176	0	0	0	0
AD(a)	0	0	136	0	0	0
MU(b)	0	22	22	22	22	0
AD(b)	0	0	17	17	17	17

(ACR). After the word length is adjusted by shifting, a separate parallel-read-shuffle (PRS) module designed by Catapult C reads the registers in parallel for $[\mathbf{E}_0, \ldots, \mathbf{E}_L]$ and writes the memory and shuffles the Hermitian part $[\mathbf{E}_L^H, \ldots, \mathbf{E}_1^H]$. Memory stalls are avoided and scalability is achieved because it can stop at any chip to adjust to different update rates.

In the PMS, the number of FUs is assigned according to the time/area constraints. As an example for a (2×2) case with $L = 10$, the VLSI area/time tradeoff is shown in Table 4. The complexity is 176 multiplications and 136 additions in each computation period. A typical manual design will layout 176 multipliers and 136 adders all in parallel. This will take 4 cycles to complete the computation. However, the multipliers are in IDLE state for 9 cycles and wasted. On the other extreme, an area-constraint solution will reuse one multiplier and one adder, but has to take more than 176 cycles. The most area/time efficient architecture in 10 cycles is to reuse 22 multipliers and 15 adders as the pipelined operations. The multiplexing of so many multipliers in manual RTL layout could be very difficult and time consuming. Moreover, for a changed specification such as the chip rate or clock rate, we can rapidly reschedule the design to meet the real-time requirement by using the minimum hardware resource. The similar design method is applied for the FIR and channel estimation.

6.2.2. Block-based MIMO-FFT IP cores

For the multiple FFTs in the tap solver, the keys for optimization of the area/speed are loop unrolling, pipelining, and resource multiplexing. Although Xilinx provides FFT IP cores, they are considerably large and much faster than required. For example, a single v32FFT core in Xilinx CoreGen library utilizes 12 multipliers and 2066 slices. Moreover, it is not easy to apply the commonality by using the IP core for the MIMO-FFTs. To achieve the best area/time tradeoff in different situations, we design the customized MIMO-FFT modules to utilize the commonality in control logic and phase coefficient loading. Parallelism/pipelining in the parallel FFTs are studied extensively in multilevels, for example, the BFU level, the stage level, and the FFT-processor level. Catapult C scheduled RTLs for 32-point FFTs with 16 bits are compared with Xilinx v32FFT Core in Table 5 for a single FFT. Catapult C design demonstrates much smaller size for different solutions, for example, from solution 1 with 8 multipliers and 535 slices to solution 3 with only one multiplier and 551 slices. Overall, solution 3 represents the smallest design with slower but acceptable speed for a single FFT. For the MIMO-FFT/IFFT modules, we can reuse the control logic inside the FFT module and schedule the number of FUs more efficiently in the merged mode.

Table 5: Architecture efficiency comparison for Catapult C versus Xilinx IP core.

Architecture	mult	Cycles	Slices
Xilinx core	12	128	2066
Catapult C Sol1	8	570	535
Catapult C Sol2	2	625	543
Catapult C Sol3	1	810	551

Table 6: The area/time specification of the major FPGA design cores.

Architecture	Latency	CLB	ASICMult
Correlator	1 chip	22399	80
16-FFT32	43.1 μs	2530	4
32 MatInvMult(4×4)	37.6 μs	4526	6
16-IFFT32	43.1 μs	2530	4
Overall tap solver	123.8 μs	7109.3	14

6.3. Prototyping implementation

Based on the above algorithmic and architectural optimizations, we have prototyped the VLSI architecture of a (4×4) MIMO equalizer on the *Aptix* FPGA platform [27]. The correlation window is set to 10 chips for all 4 receive antennas. Fixed-point simulation shows that 8-bit input chip could provide negligible performance loss. To give a safe range, the input chip samples to both the corelator and the channel estimator have 10-bit precision. The 32-point MIMO-FFT module has 16-bit input word length for both the covariance and channel coefficients. To support even faster fading speed, we design the prototyping system for up to 4 updates per slot with an overall tap-solving latency requirement of 125 microseconds. In Table 6, we give the specification of the major design blocks. Overall, we utilize only 4 multipliers to achieve area/time efficient design for 16 merged FFT/IFFT modules. For the L_F inverse of the (4×4) Hermitian submatrices, the latency is 38 microseconds with 6 multipliers. It is also noticed that the different modules have very similar latency, which provides a very balanced pipelining in multiple stages. The overall 124 microseconds meet the real-time requirement very closely and give area efficiency. This efficiency not only benefits from the afore-mentioned algorithmic and architectural optimization, but also from the extensive design space exploration to find the most compact design by meeting the real-time requirement. The integration of the MIMO equalizer into the complete HSDPA transceiver system following the same methodology as in [13] is also being considered.

7. CONCLUSION

In this paper, we propose an efficient circulant MIMO chip equalizer for multicode CDMA downlink by using FFT-based operations to avoid the direct matrix inverse. A comparative study demonstrates very promising performance/complexity tradeoff. VLSI-oriented optimizations are proposed to reduce the number and complexity of FFTs. The inverse of (4×4) submatrices is solved by partitioned (2×2) submatrices, which leads to dramatically simplified VLSI modules. The VLSI design space is explored extensively for area/time efficiency by a Catapult C-based HLS methodology. The VLSI design is validated in a real-time FPGA prototyping system.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Behnaam Aazhang and the anonymous reviewers for their instructive comments. Joseph R. Cavallaro was supported in part by NSF under Grants ANI-9979465, EIA-0224458, and EIA-0321266.

REFERENCES

[1] D. Gesbert, M. Shafi, D. Shiu, P. J. Smith, and A. Naguib, "From theory to practice: an overview of MIMO space-time coded wireless systems," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 281–302, 2003.

[2] G. D. Golden, C. J. Foschini, R. A. Valenzuela, and P. W. Wolniansky, "Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture," *Electronics Letters*, vol. 35, no. 1, pp. 14–16, 1999.

[3] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Technical Journal*, vol. 1, no. 2, pp. 41–59, 1996.

[4] H. Holma and A. Toskala, *Wideband CDMA for UMTS*, John Wiley & Sons, New York, NY, USA, 2000.

[5] A. Wiesel, L. García, J. Vidal, A. Pagès, and J. R. Fonollosa, "Turbo linear dispersion space time coding for MIMO HSDPA systems," in *Proceedings of 12th IST Summit on Mobile and Wireless Communications*, Aveiro, Portugal, June 2003.

[6] K. Hooli, M. Juntti, M. J. Heikkilä, P. Komulainen, M. Latva-aho, and J. Lilleberg, "Chip-level channel equalization in WCDMA downlink," *EURASIP Journal on Applied Signal Processing*, vol. 2002, no. 8, pp. 757–770, 2002.

[7] S. Das, C. Sengupta, and J. R. Cavallaro, "Hardware design issues for a mobile unit for next-generation CDMA systems," in *Advanced Signal Processing Algorithms, Architectures, and Implementations VIII*, vol. 3461 of *Proceedings of SPIE*, pp. 476–487, San Diego, Calif, USA, July 1998.

[8] L. L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*, Addison-Wesley, New York, NY, USA, 1990.

[9] T. Kailath and J. Chun, "Generalized displacement structure for block-Toeplitz, Toeplitz-block, and Toeplitz-derived matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 15, no. 1, pp. 114–128, 1994.

[10] S. Chandrasekarnan and A. H. Sayed, "Stablizing the generalized schur algorithm," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 4, pp. 950–983, 1996.

[11] M. J. Heikkila, K. Ruotsalainen, and J. Lilleberg, "Space-time equalization using conjugate-gradient algorithm in WCDMA downlink," in *Proceedings of 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '02)*, vol. 2, pp. 673–677, Lisbon, Portugal, September 2002.

[12] F. R. Jevic, J. R. Cavallaro, and A. de Baynast, "ASIP architecture implementation of channel equalization algorithms for MIMO systems in WCDMA downlink," in *Proceedings of 60th IEEE Vehicular Technology Conference (VTC '04)*, vol. 3, pp. 1735–1739, Los Angeles, Calif, USA, September 2004.

[13] Y. Guo, G. Xu, D. McCain, and J. R. Cavallaro, "Rapid scheduling of efficient VLSI architectures for next-generation HSDPA wireless system using Precision C synthesizer," in *Proceedings of 14th IEEE International Workshop on Rapid Systems Prototyping (RSP '03)*, pp. 179–185, San Diego, Calif, USA, June 2003.

[14] http://www.nokia.com/nokia/0,,53713,00.html.

[15] J. Wrolstad, *Bell Labs BLASTs New High-Speed Wireless Chips*, Wireless NewsFactor, Los Angeles, Calif, USA, 2002.

[16] Z. Guo, F. Edman, P. Nilsson, and V. Ovall, "On VLSI implementations of MIMO detectors for future wireless communications," in *Proceedings of 1st IST-MAGNET Workshop*, Shanghai, China, November 2004.

[17] Y. Guo, D. McCain, J. Zhang, and J. R. Cavallaro, "Scalable FPGA architectures for LMMSE-based SIMO chip equalizer in HSDPA downlink," in *Proceedings of 37th Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 2171–2175, Monterey, Calif, USA, November 2003.

[18] A. Evans, A. Silburt, G. Vrckovnik, et al., "Functional verification of large ASICs," in *Proceedings of 35th ACM/IEEE Design Automation Conference (DAC '98)*, pp. 650–655, San Francisco, Calif, USA, June 1998.

[19] P. Bellows and B. Hutchings, "JHDL-An HDL for reconfigurable systems," in *Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 175–184, IEEE Computer Society Press, Napa Valley, Calif, USA, April 1998.

[20] Y. Guo, J. Zhang, D. McCain, and J. R. Cavallaro, "Efficient MIMO equalization for downlink multi-code CDMA: complexity optimization and comparative study," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '04)*, vol. 4, pp. 2513–2519, Dallas, Tex, USA, November 2004.

[21] S. Rajagopal, B. A. Jones, and J. R. Cavallaro, "Task partitioning wireless base-station receiver algorithms on multiple DSPs and FPGAs," in *Proceedings of International Conference on Signal Processing Applications and Technology (ICSPAT '00)*, Dallas, Tex, USA, October 2000.

[22] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, New York, NY, USA, 1985.

[23] J. P. Kermoal, L. Schumacher, K. I. Pedersen, P. E. Mogensen, and F. Frederiksen, "A stochastic MIMO radio channel model with experimental validation," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 6, pp. 1211–1226, 2002.

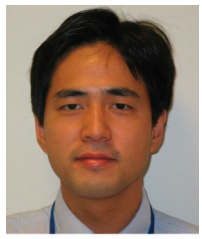
[24] H. Nguyen, J. Zhang, and B. Raghothaman, "A Kalman-filter approach to equalization of CDMA downlink channels," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 5, pp. 611–625, 2005.

[25] A. Burg, M. Rupp, M. Guillaud, et al., "FPGA implementation of a MIMO receiver front-end for the UMTS downlink," in *Proceedings of International Zurich Seminar on Broadband Communications (IZS '02)*, pp. 8-1–8-6, Zurich, Switzerland, February 2002.

[26] Mentor Graphics, *Catapult C Manual and C/C++ style guide*, 2004, Wilsonville, Ala, USA.

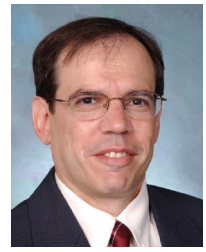
[27] U. Knippin, "Early design evaluation in hardware and system prototyping for concurrent hardware/software validation in one environment," in *Proceedings of 13th IEEE International Workshop on Rapid System Prototyping (RSP '02)*, Darmstadt, Germany, July 2002.

Yuanbin Guo received the B.S. degree from Peking University, and the M.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 1996 and 1999, respectively, and the Ph.D. degree from Rice University, Houston, Tex, in May 2005, all in electrical engineering. He was a winner of the Presidential Fellowship in Rice University in 2000. From 1999 to 2000, he was with Lucent Bell Laboratories, Beijing, where he conducted R&D in the Intelligent Network Department. He joined Nokia Research Center, Irving, Tex, in 2002 as a Research Engineer. His current research interests include equalization and detection for multiple-antenna systems, VLSI design and prototyping, and DSP and VLSI architectures for wireless systems. He is a Member of the IEEE. He has 6 patents pending in wireless communications field.



Jianzhong(Charlie) Zhang received the B.S. degrees in both electrical engineering and applied physics from Tsinghua University, Beijing, China, in 1995, the M.S. degree in electrical engineering from Clemson University in 1998, and the Ph.D. degree in electrical engineering from the University of Wisconsin-Madison in May 2003. He has been with Nokia Research Center in Irving, Tex, since June 2001, where he is currently a Senior Research Engineer. His Research has focused on the application of statistical signal processing methods to wireless communication problems. From 2001 to 2004, he worked on the transceiver designs for both EDGE and CDMA2000/WCDMA cellular systems. Since July 2004, he has participated in Nokia's contributions to the IEEE 802.16e Standard, especially in the PHY layer topics such as LDPC codes, space-time-frequency coding and limited-feedback-based MIMO precoding.

Dennis M. McCain received his B.S. degree in electrical engineering from Lousiana State University in 1990 and his M.S. degree in electrical engineering from Texas A&M University in 1992. From 1992 to 1996, he served in the USA Army as a Signal Officer responsible for deploying communication networks in tactical environments. From 1996 to 1998, he worked at Texas Instruments and Raytheon Systems as a Digital Design Engineer. In 1999, he joined Nokia Research Center, Dallas, Tex, to develop a prototype WLAN system. He is currently a Research Manager in Nokia Research Center leading a team responsible for implementing novel physical-layer algorithms for new cellular and noncellular wireless systems. His interests are in the areas of hardware architecture research, digital baseband design, and rapid-prototype design flows.



Joseph R. Cavallaro received the B.S. degree from the University of Pennsylvania, Philadelphia, Pa, in 1981, the M.S. degree from Princeton University, Princeton, NJ, in 1982, and the Ph.D. degree from Cornell University, Ithaca, NY, in 1988, all in electrical engineering. From 1981 to 1983, he was with AT&T Bell Laboratories, Holmdel, NJ. In 1988, he joined the faculty of Rice University, Houston, Tex, where he is currently a Professor of electrical and computer engineering. His research interests include computer arithmetic, VLSI design and microlithography, and DSP and VLSI architectures for applications in wireless communications. During the 1996–1997 academic year, he served at the USA National Science Foundation as Director of the Prototyping Tools and Methodology Program. During 2005, he was a Nokia Foundation Fellow and a Visiting Professor at the University of Oulu, Finland. He is currently the Associate Director of the Center for Multimedia Communication at Rice University. He is a Senior Member of the IEEE. He was Cochair of the 2004 Signal Processing for Communications Symposium at the IEEE Global Communications Conference and General Cochair of the 2004 IEEE 15th International Conference on Application-Specific Systems, Architectures, and Processors (ASAP).