Regularized Frequency Domain Equalization Algorithm and its VLSI Implementation

A. Burg[†]*, S. Haene*[†]*, W. Fichtner*[†]*, and M. Rupp*[‡]

[†] Integrated Systems Laboratory, ETH Zurich
8092 Zurich, Switzerland, email: {apburg, haene, fw}@iis.ee.ethz.ch

[‡] Institute of Communications and Radio-Frequency Engineering, Vienna University of Technology
1040 Wien, Austria, email: mrupp@nt.tuwien.ac.at

***Abstract*— Approximation of Toeplitz matrices with circulant matrices is a well-known approach to reduce the computational complexity of linear equalizers. This paper presents a novel technique to compute linear equalizer coefficients in the frequency domain. It is shown how a regularization term can help to reduce the error caused by the frequency domain approximation. A corresponding VLSI implementation provides reference for the true silicon complexity and for the complexity increase associated with the proposed algorithm.**

I. Introduction

Many wireless communication systems (such as UMTS or CDMA2000) suffer from inter-symbol and multiple-access interference due to multipath propagation. In practice, this interference often exceeds the thermal noise in the system and becomes the performance-limiting factor. Hence, techniques must be found to mitigate this negative impact of multipath channels on error rate performance. An efficient (in terms of performance and computational complexity) approach for emerging standards is the use of orthogonal frequency division multiplexing (OFDM) modulation. However, solving the problem for established standards requires leaving the existing modulation schemes intact so that only signal processing at the receiver is a viable option to suppress interference. Linear equalization allows to partially restore the transmitted signal by inverting the transfer function of the multipath channel with a properly designed finite impulse response filter at the receiver. The corresponding filter coefficients can be obtained using different approaches: One possibility is to employ low-complexity adaptive algorithms such as LMS or RLS to adjust the filter coefficients directly based on a received training or pilot sequence. The drawback of this adaptive approach is that it often suffers from slow convergence and requires continuous tracking which may entail a considerable computational effort. Another possibility is to first estimate and track the channel's impulse response followed by the computation of the equalizer coefficients using an algorithm for direct matrix inversion (DMI). The main difficulty associated with this second approach is that a straightforward implementation requires the inversion of a large Toeplitz matrix which is a costly operation in terms of computational complexity and memory consumption.

Hence, for the practical application of equalization algorithms which are based on explicit estimation of the channel's impulse response, we are interested in techniques that avoid or simplify the associated matrix inversion. Viable solutions to this problem have been described for example in [1], [2], and [3]. The basic idea in these publications is to start from the DMI-based approach and to approximate the Toeplitz matrices with circulant matrices. The latter are easy to invert in the frequency domain, resulting in a complexity that is comparable to that of an OFDM receiver. However, it is also mentioned in [1] and [2] that achieving good performance with the cyclic approximation requires a regularization (or conditioning) of the circulant matrices. Unfortunately, none of the two papers gives guidelines for the computation of this term.

Contribution: In this paper, equalizer coefficients are computed using a circulant approximation of the convolution in the channel. The corresponding approach deviates slightly from [1] and [2] since our derivation is designed to yield an expression for the regularization term required to partially mitigate the error caused by the cyclic approximation. The approach is described for single-input single-output system, but can also be adopted for multiple-input multiple-output systems. The paper also describes a corresponding VLSI architecture and provides reference implementation results.

Outline: The next section introduces the system model and briefly repeats the computation of a straightforward time-domain (TD) equalizer. In Section III the low-complexity frequency-domain (FD) algorithm is introduced. Section IV then describes our *regularized frequency-domain* (RFD) equalizer algorithm together with an analysis of the performance and of the complexity scaling behavior with the length of the equalizer. A VLSI architecture and corresponding implementation results are presented in Section V.

II. System Model and Linear Equalizer

Consider a single-carrier communication system in which the delay-spread of the frequency-selective channel exceeds the symbol period causing inter-symbol interference (ISI). The sampled impulse response of this channel with L_c significant taps is given by the vector $\mathbf{h} = [h_0, h_1, \ldots, h_{L_c-1}]^T$ and the vector $\mathbf{x} = [x_{N-1}, x_{N-2}, \ldots, x_0]^T$ denotes a collection of N subsequent transmitted data symbols in time-reverse order. The linear convolution in the channel that yields a vector $\mathbf{y}$ of L_e consecutive samples, where L_e is the length of the subsequent equalizer, can be described in matrix-notation according to

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}, \tag{1}$$

where the $L_e \times (L_c + L_e - 1)$-dimensional Toeplitz matrix $\mathbf{H}$ is given by

$$\mathbf{H} = \begin{bmatrix} h_0 & h_1 & \ldots & h_{L_c-1} & 0 & 0 & \ldots & 0 \\ 0 & h_0 & \ldots & h_{L_c-2} & h_{L_c-1} & 0 & \ldots & 0 \\ \vdots & & \ddots & & & & \ddots & \vdots \\ 0 & & \ldots & & 0 & h_0 & \ldots & h_{L_c-1} \end{bmatrix}. \tag{2}$$

The vector $\mathbf{n}$ denotes the additive (assumed i.i.d. Gaussian) noise with zero mean and variance σ^2 per complex dimension.

The output of a linear length L_e equalizer at the receiver is obtained by convolving the received signal with the equalizers length L_e impulse response $\mathbf{w}^H$. In matrix notation, a single result $\hat{x}$ of this convolution is obtained by premultiplying the received vector $\mathbf{y}$ with $\mathbf{w}^H$:

$$\hat{x}_D = \mathbf{w}^H \mathbf{y}. \tag{3}$$

The vector $\mathbf{w}$, is obtained from the Wiener-Hopf equations according to

$$\mathbf{w} = (\mathbf{H}^H \mathbf{H})^{-1} \mathbf{H} \mathbf{e}_D, \tag{4}$$

where $\mathbf{e}_D$ is a unit-vector of appropriate length with a one in the Dth row. The design parameter D thereby determines the combined delay of the channel and the equalizer and is set to $D = L_e/2$ in the following.

The bottleneck at the receiver is in the costly computation of (4) since the effort required for a straightforward matrix-inverse grows according to $\mathcal{O}(L_e^3)$. Hence, if L_e is large (typically $L_e \geq 16$), the computational complexity becomes quickly prohibitive.

III. Frequency-Domain Equalization

To reduce this complexity, we start by defining an $\tilde{L}_e \times \tilde{L}_e$-dimensional circulant matrix $\tilde{\mathbf{H}}$ whose first row $\tilde{\mathbf{h}}$ is given by the vector obtained from zero-extending $\mathbf{h}$ to a length of $\tilde{L}_e$. Note that as opposed to [2] we allow for $\tilde{L}_e \geq L_e$. Since $\tilde{\mathbf{H}}$ is circulant, the corresponding inverse is given by $\tilde{\mathbf{H}}^{-1} = \mathbf{F} \mathbf{\Lambda}^{-1} \mathbf{F}^H$, where $\mathbf{\Lambda} = \mathrm{diag}(\mathrm{DFT}\{\tilde{\mathbf{h}}\})$ is a diagonal matrix composed of the elements of the discrete Fourier transformation (DFT[1]) of $\tilde{\mathbf{h}}$, and where $\mathbf{F}$ denotes the Fourier transformation matrix of appropriate dimension ($\mathbf{F}\mathbf{F}^H = \mathbf{I}$). The approximate frequency-domain equalizer of length L_e under the cyclic assumption is now given by

$$\tilde{\mathbf{w}}^H = \mathbf{e}_D^H \mathbf{F} \mathbf{\Lambda}^{-1} \mathbf{F}^H \mathbf{Z}^H, \tag{5}$$

where $\mathbf{e}_D$ is again a unit-vector of appropriate length with a one in the Dth row. The matrix $\mathbf{Z} = [\mathbf{I}_{L_e \times L_e}, \mathbf{0}_{L_e \times (\tilde{L}_e - L_c)}]$ in (5) serves to truncate the impulse response from $\tilde{L}_e$ to the desired length L_e.

For a performance comparison between the FD equalizer computed according to (5) and the TD equalizer given by (4), consider the cumulative distribution function of the squared error (SE) at the receiver, after equalization. Corresponding simulation results for a channel with eight sample-spaced, equal-power taps and for an equalizer length of $L_e = 32$ and $\tilde{L}_e = 64$ are shown in Fig. 1. As expected, the FD equalizer suffers from a loss in terms of the average SE (taken in [dB]). However, it can also be seen that in many cases the FD equalizer completely fails to suppress the ISI causing excessive levels of interference.

IV. Regularized Frequency-Domain Equalizer

The goal of the subsequent derivation is to reduce the MSE caused by the cyclic-approximation and to avoid complete equalization failures. To this end, it is proposed to employ an additive noise model for taking the approximation error into account when computing the coefficients $\tilde{\mathbf{w}}$ of the regularized FD equalizer. For clarity of exposition, the corresponding algorithm is obtained for the case of high-SNR, where the influence of the thermal noise is neglected ($\mathcal{E}\{\|\mathbf{n}\|^2\} = 0$).

[1]The DFT operator is normalized so that $\mathrm{trace}(\Lambda^H \Lambda) = \tilde{L}_e \|\mathbf{h}\|^2$.

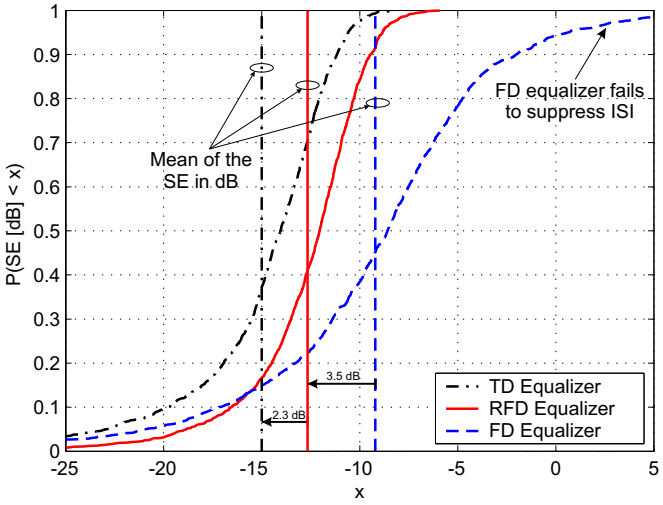

Fig. 1. CDF of the MSE after equalization. The PDP of channel has $L_c = 8$ taps with identical power. The parameters of the equalizer are $L_e = 32$ and $\tilde{L}_e = 64$.

A. RFD Algorithm

The derivation starts by writing a received sample $\tilde{x}_D$ after equalization according to

$$\tilde{x}_D = \tilde{\mathbf{w}}^H \mathbf{H} \mathbf{x} \quad \text{with} \quad \tilde{\mathbf{w}}^H = \mathbf{e}_D^H \mathbf{F} \tilde{\mathbf{\Lambda}}^{-1} \mathbf{F}^H \mathbf{Z}^H. \tag{6}$$

The next step is to to express the convolution of the transmitted signal with the channel in (6) based on the circulant matrix $\tilde{\mathbf{H}}$. To this end, substitute $\mathbf{H} = \mathbf{Z}\tilde{\mathbf{H}}\mathbf{K}$ into (6), where $\mathbf{K} = \begin{bmatrix} \mathbf{I}_N & \mathbf{0}_{(\tilde{L}_e - N) \times N} \end{bmatrix}^T$ with $N = L_e + L_c - 1$ to obtain

$$\tilde{x}_D = \mathbf{e}_D^H \mathbf{F} \tilde{\mathbf{\Lambda}}^{-1} \mathbf{F}^H \mathbf{Z}^H \left(\mathbf{Z} \tilde{\mathbf{H}} \right) \mathbf{K} \mathbf{x}. \tag{7}$$

Further substituting $\mathbf{Z}^H \mathbf{Z} \tilde{\mathbf{H}} = \tilde{\mathbf{H}} - \mathbf{P}$ with $\mathbf{P} = \tilde{\mathbf{H}} - \mathbf{Z}^H \mathbf{Z} \tilde{\mathbf{H}}$ and subsequently $\tilde{\mathbf{H}} = \mathbf{F} \mathbf{\Lambda} \mathbf{F}^H$ yields

$$\tilde{x}_D = \mathbf{e}_D^H \mathbf{F} \tilde{\mathbf{\Lambda}}^{-1} \left(\mathbf{\Lambda} \mathbf{F}^H \mathbf{K} \mathbf{x} - \tilde{\mathbf{n}} \right), \tag{8}$$

where $\tilde{\mathbf{n}} = \mathbf{F}^H \mathbf{P} \mathbf{K} \mathbf{x}$ subsumes the source of residual interference from the cyclic approximation[2]. In our model, the components of $\tilde{\mathbf{n}}$ are described by zero-mean, independent identically distributed (i.i.d.) random variables with variance $\sigma_{\tilde{\mathbf{n}}}^2$ per complex dimension. To compute this variance, consider first the expected overall interference-noise power which is given by $\mathcal{E}\{\|\tilde{\mathbf{n}}\|^2\} = \|\mathbf{P}\mathbf{K}\|_F^2$, where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. Since this interference-noise is modeled as i.i.d., $\sigma_{\tilde{\mathbf{n}}}^2 = \mathcal{E}\{\|\tilde{\mathbf{n}}\|^2\}/\tilde{L}_e$, which can be written as

$$\sigma_{\tilde{\mathbf{n}}}^2 = \frac{(L_c - 1)}{\tilde{L}_e} \sum_{i=1}^{L_c} |h_i|^2 \tag{9}$$

by exploiting the structure of the matrix $\mathbf{P}\mathbf{K}$.

[2]Note that $\tilde{\mathbf{n}}$ also includes a component that depends on the desired signal x so that our estimator will be biased.

To obtain the RFD equalizer coefficients from this additive-noise model, substitute $\tilde{\mathbf{z}} = \mathbf{F}^H \mathbf{K} \mathbf{x}$ into (8) and consider now the problem of estimating $\tilde{\mathbf{z}}$ from

$$\tilde{\mathbf{y}} = \mathbf{\Lambda} \tilde{\mathbf{z}} + \tilde{\mathbf{n}}. \tag{10}$$

The entries of $\tilde{\mathbf{z}}$ are treated as if they were zero-mean, i.i.d. random variables with variance $\sigma_{\tilde{\mathbf{z}}} = N/\tilde{L}_e$ per complex dimension. The corresponding minimum mean squared error estimator, is given by

$$\tilde{\Lambda}^{-1} = \left(\Lambda^H \Lambda + \frac{\sigma_{\tilde{n}}}{\sigma_{\tilde{z}}} \mathbf{I} \right)^{-1} \Lambda^H, \tag{11}$$

which can then be used in (6) to obtain $\tilde{\mathbf{w}}$.

B. Efficient Implementation and Complexity

A block diagram summarizing the procedure for computing the RFD equalizer is shown in Fig. 2. To properly exploit the complexity savings offered by the (R)FD algorithm, $\tilde{L}_e$ must be chosen as a power of two. This choice enables the use of the fast Fourier transform (FFT) to obtain the diagonal entries λ_i of Λ according to $[\lambda_1, \lambda_2, \cdots, \lambda_{\tilde{L}_e}] = \mathrm{FFT}\{\tilde{\mathbf{h}}\}$ which has a complexity scaling behavior of $\mathcal{O}(\tilde{L}_e \log_2 \tilde{L}_e)$. The reduced computational complexity compared to the TD algorithm results from the fact that the (regularized) matrix inversion in the frequency domain in (11) is trivial, with a complexity scaling behavior given by $\mathcal{O}(L_e)$. The final conversion of $\tilde{\Lambda}^{-1}$ back into the time domain is performed by an IFFT, which can be truncated to yield only $\tilde{w}_i$ for $i = 1, \ldots, L_e$.

The only overhead in the computation of the RFD algorithm, compared to the FD algorithm results from (9). While the associated complexity scaling behavior is given by $\mathcal{O}(L_e)$, it is noted that the corresponding number of operations is extremely low when compared to the other required operations and is therefore usually negligible.

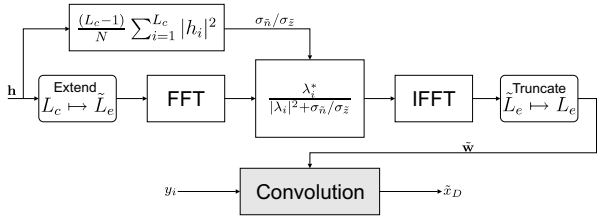

Fig. 2. Simplified block diagram of the RFD equalizer.

C. Performance Analysis

The MSE performance of the RFD algorithm is illustrated by the simulation results shown in Fig. 1. In the example, the use of the regularization term reduces the average of the SE (taken in [dB]) by approximately 3.5 dB compared to the FD algorithm without regularization. In addition to that, a significant reduction of the probability that the FD equalization leads to excessive interference is observed as desired.

Fig. 3 shows a comparison of the residual MSE after TD, FD, and RFD equalization for different channel- and equalizer lengths. Over the entire range of the simulation, the RFD algorithm provides a significant improvement over the FD approximation without regularization. It can also be seen that increasing the length of the equalizer improves the MSE performance since the circulant approximation becomes more

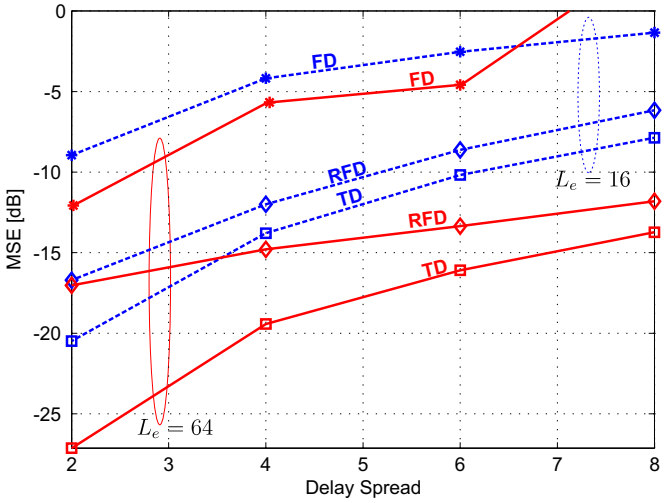

Fig. 3. MSE after TD, FD, and RFD equalizer for channels with uniform PDP of different length. For the RFD equalizer, $\tilde{L}_e = 2L_e$ and $D = L_e/2$.

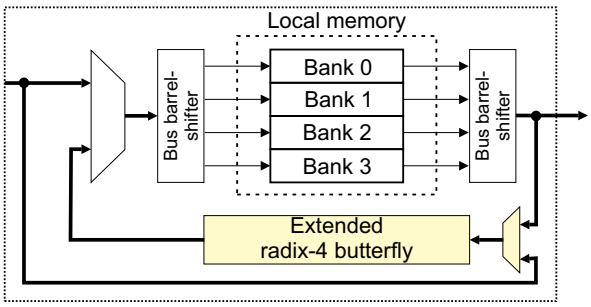

Fig. 4. Radix-4 FFT/IFFT processor based high-level architecture.

accurate. However, it is also evident that the TD equalizer still outperforms both low-complexity frequency domain schemes.

V. VLSI Implementation

In order to assess the true silicon complexity of the proposed algorithm, we shall briefly describe a corresponding VLSI implementation. The circuit under consideration is designed for a maximum channel length of $L_c = 16$, an equalizer length of $L_e = 32$, and an FFT/IFFT length of $\tilde{L}_e = 64$.

A. VLSI Architecture

The basic idea behind the chosen high-level VLSI architecture is to extend the arithmetic unit (*butterfly unit*) of an FFT/IFFT processor for the computation of (9) and (11). A suitable starting point for the 64-point FFT/IFFT operations required for $\tilde{L}_e = 64$ is a radix-4 architecture [4] implemented with a single time-shared radix-4 butterfly unit (BFU) as shown in Fig. 4. The memory of the FFT/IFFT processor is divided into four independent dual ported banks (each holding 16 data words). The connection between these banks and the BFU is made through two *bus barrelshifters* which can cyclically shift the connections between the four memories and the four inputs/outputs of the BFU.

The extended radix-4 BFU is comprised of three pipeline stages which corresponds to the maximum degree of pipelining that does not cause any data hazards in this configuration.

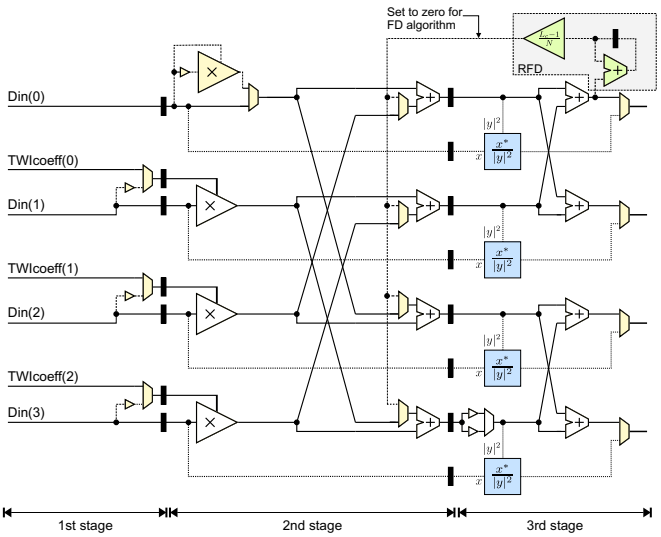

Fig. 5. Extended radix-4 butterfly unit.

The additional components added for the computation of (9) and (11) are highlighted in the schematic in Fig. 5. The fast computation of (9) adds almost no overhead, since three out of four branches of the standard radix-4 BFU already contain multipliers that can easily be reconfigured for the required norm computations. A squarer is added to the originally multiplier-free first branch of the radix-4 BFU to treat always four values in parallel. The norms are added and scaled after the accumulator circuit shown in the top-right corner of Fig. 5. This accumulator and the subsequent constant-coefficient multiplier are the only components that can be removed when implementing the FD instead of the RFD algorithm.

The computation of (11) is carried out in the second and third pipeline stage of the BFU. To this end, the multipliers and adders in the second stage are reused to obtain $\left|\lambda_i^2\right|$ and to add the regularization term $\sigma_{\tilde{n}}/\sigma_{\tilde{z}}$. For the FD algorithm this term is simply set to zero. The subsequent division must be carried out on dedicated dividers which have to be added to the standard radix-4 BFU. These dividers are responsible for most of the overhead (in terms of area and delay) compared to a standard radix-4 FFT/IFFT processor.

B. Operation

The operation sequence of the FD/RFD circuit is illustrated in Fig. 6. First, the 16-entries of channel impulse response vector $\mathbf{h}$ are read. At the same time, the RFD implementation computes (9) on the extended BFU (the corresponding connection, shown in Fig. 4, is not needed for the FD algorithm). Next, a zero-extended 64-point FFT is computed, requiring 32 plus three cycles to flush the BFU pipeline since the first butterfly stage of a zero-extended FFT can be skipped. Next, (11) is computed in 16+3 cycles, followed by a 64-point IFFT in 48+3 cycles. Unloading the 32 data words of interest requires another 8 cycles since four words can be accessed per cycle. In total, a single equalizer update requires 120 clock cycles, for both the FD and the RFD algorithm.

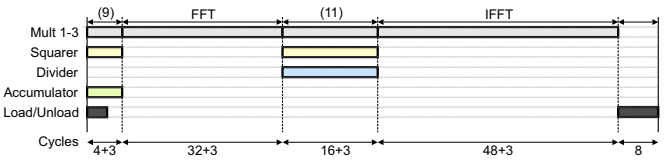

Fig. 6. Timing diagram illustrating the operation of the RFD equalizer computation.

C. Implementation Results

Tbl. I shows the VLSI implementation results for the original FFT/IFFT processor and for the derived FD and RFD circuits. Compared to the FFT/IFFT processor, the FD and RFD circuits suffer from a 75% increase in silicon area and achieve only one third of the original clock frequency. The main reason for this speed penalty are the slow 12-bit radix-2 dividers. Nevertheless, the achievable update rate of 2.15μs is already sufficient for many applications with mobile speeds in excess of 200 km/h at carrier frequencies up to 5 GHz. Hence, the area overhead for additional pipeline registers in the dividers would not be justified.

TABLE I
VLSI IMPLEMENTATION RESULTS

	FFT/IFFT	FD Alg.	RFD Alg.
Techn.	0.25μm, 1P/5M		
Area	43.2K GE	72.4K GE	73.7K GE
Max. clock freq.	166 MHz	56 MHz	

VI. Conclusions

Computing the coefficients of a linear equalizer in the frequency domain (via FFTs) using a circulant approximation of the Toeplitz matrix describing the convolution in the channel is a viable approach to reduce computational complexity. The associated loss in mean squared error performance from the circulant approximation can be partially mitigated by including a regularization term derived from the channels impulse response. The incorporation of this term entails only a marginal increase in computational complexity and the hardware overhead in a VLSI implementation is below 2%.

Acknowledgement

This work is supported by the STREP project No. IST-026905 (MASCOT) within the sixth framework programme of the European Commission.

References

[1] A. Burg, M. Rupp, N. Felber, and W. Fichtner, "Practical low complexity linear equalization for MIMO-CDMA systems," in *Proc. 37th IEEE Asilomar Conf. Signals Syst. Comput.*, vol. 2, Nov. 2003, pp. 1266–1272.

[2] J. Zhang, T. Bhatt, and G. Mandyam, "Efficient linear equalization for high data rate downlink CDMA signaling," in *Proc. 37th IEEE Asilomar Conf. Signals Syst. Comput.*, vol. 1, Nov. 2003, pp. 141–145.

[3] Y. Guo, D. McCain, and J. R. Cavallaro, "Hermitian optimization and scalable VLSI architecture for circulant approximated MIMO equalizer in CDMA downlink," in *Proc. 62nd IEEE Vehicular Techn. Conf. (VTC-Fall)*, vol. 4, Sept. 2005, pp. 2096–2101.

[4] L. G. Johnson, "Conflict free memory addressing for dedicated FFT hardware," *IEEE Trans. Circuits Syst. II*, vol. 39, pp. 312–316, 1992.