Incremental Refinement of a Multi-User-Detection Algorithm

Marius Vollmer, Jürgen Götze
University of Dortmund, Information Processing Lab,
http://www-dt.e-technik.uni-dortmund.de,
marius.vollmer@udo.edu, juergen.goetze@udo.edu

Abstract

Multi-user detection is a technique proposed for mobile radio systems based on the CDMA principle, such as the upcoming UMTS. While offering an elegant solution to problems such as intra-cell interference, it demands very significant computational resources.

In this paper, we present a high-level approach for reducing the required resources for performing multi-user detection in an exemplary multi-user system. This approach is based on using a displacement representation for the parameters that describe the transmission system, and a generalized Schur algorithm that works on this representation. The Schur algorithm naturally leads to a highly parallel hardware implementation using CORDIC cells.

It is very beneficial to introduce incremental refinement structures into the solution process, both at the algorithmic level and in the individual cells of the hardware architecture. We detail these approximations and present simulation results that confirm their effectiveness.

1 Introduction

The next generation of mobile radio systems, UMTS, has a very high demand for signal processing hardware, both in the base station as well as in the mobile terminal. This is due to the amount of computations needed to combat the negative effects in a CDMA system. One particularly elegant approach is the employment of multi-user detection at the receiver [14]. With this technique, the data symbols of one user are estimated using knowledge of all other users that are transmitting at the same time but on different codes.

While the basic formulation of a linear multi-user detector is quite simple, the resulting raw amount of computing power required for its real-time realization is formidable [5]. However, for the specific case of the TDD mode of UMTS [4] with its burst-structured transmission and relatively few simultaneously active users, there exist algorithms and implementation strategies that can significantly reduce the computational requirements [17].

In this paper, we concentrate on the Schur algorithm [12, 13]. It leads to a fine-grained, parallel formulation of its computations which should prove beneficial for a dedicated hardware implementation.

One key characteristic of the Schur algorithm is that it exploits the inherent mathematical structure of the specific problem, such as the periodicity of the spreading code and the (assumed) time-invariance of the radio channel during one burst. Another important insight is that limited inter-symbol-interference of the system allows for far reaching approximations. The approximations can be chosen such that they directly lead to less consumed resources (area, clock frequency, power).

In the sequel, we will shortly present the mathematical formulation of our specific linear multi-user detector in section 2. In section 3, we explain its displacement representation and the corresponding Schur algorithm, together with approximations that work on the algorithmic level. Section 4 shows the parallel and pipelined formulation of this algorithm, using approximating hyperbolic CORDIC cells for complex valued signals. Simulation results are shown in section 5.

2 Linear Multi-User Detection in a Burst-Structured System

Figure 1 depicts a simple CDMA system with K users. Each user transmits N symbols per burst, represented as a vector $\boldsymbol{d}^{(k)} \in \mathbb{C}^N$ with $1 \leq k \leq K$. The data symbols are spread with a user-specific code $\boldsymbol{c}^{(k)} \in \mathbb{C}^Q$ and transmitted over a channel that is modeled as an FIR filter with impulse response $\boldsymbol{h}^{(k)} \in \mathbb{C}^W$. To yield the vector of the received chips $\boldsymbol{x} \in \mathbb{C}^L$ where $L = NQ + W - 1$ is the length of the received burst in chips, the distorted chip sequences are superimposed and corrupted by additive white Gaussian noise. The receiver estimates the channel impulse responses and uses this information together with the known spreading codes to arrive at an estimate $\hat{\boldsymbol{d}}^{(k)}$ for the data symbols of each user.

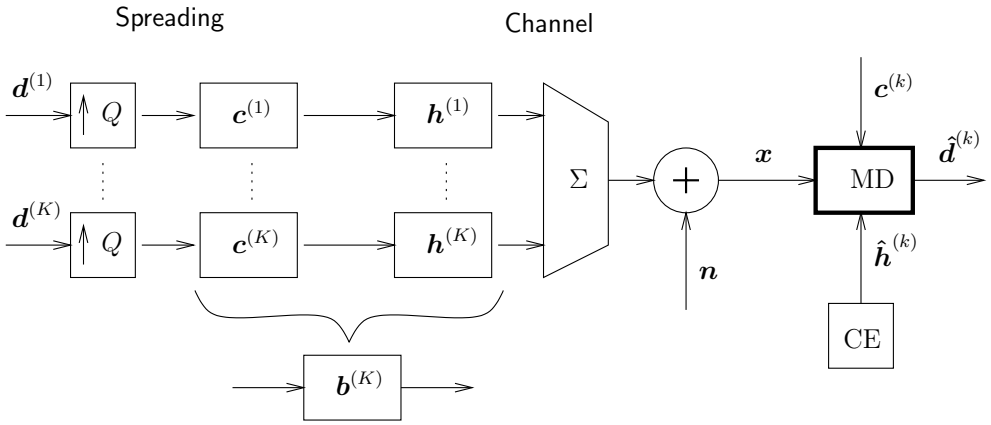

Figure 1. Simple CDMA system.

Since the spreading at the transmitter can be modeled as zero-inserting upsampling with a factor Q followed by the convolution with $\boldsymbol{c}^{(k)}$, we can combine it in our model with the following convolution for the channel into a single convolution with the vector $\boldsymbol{b}^{(k)} \in \mathbb{C}^{(Q+W-1)}$. It is then easy to see that the system in Figure 1 can be expressed as

$$\boldsymbol{x} = \sum_{k=1}^{K} \boldsymbol{B}^{(k)} \boldsymbol{d}^{(k)} + \boldsymbol{n} \tag{1}$$

where $\boldsymbol{B}^{(k)}$ is the matrix describing the upsampling by Q and convolution with $\boldsymbol{b}^{(k)}$:

$$\boldsymbol{B}^{(k)} = \begin{bmatrix} \boldsymbol{b}^{(k)} & & \\ & \boldsymbol{b}^{(k)} & \\ & & \ddots \\ & & & \boldsymbol{b}^{(k)} \end{bmatrix}$$

The sum of matrix vector products in equation (1) can be merged into a single matrix vector product that gives us the final formulation of our data model:

$$\boldsymbol{x} = \boldsymbol{T}\boldsymbol{d} + \boldsymbol{n},$$

where the system matrix $\boldsymbol{T}$ is defined as

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{V} & & \\ & \boldsymbol{V} & \\ & & \ddots \\ & & & \boldsymbol{V} \end{bmatrix}, \quad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{b}^{(1)} & \boldsymbol{b}^{(2)} & \cdots & \boldsymbol{b}^{(K)} \end{bmatrix} \tag{2}$$

and where $\boldsymbol{d} \in \mathbb{C}^{NK}$ collects all transmitted symbols of all users. The internal structure of $\boldsymbol{T}$ is a "Block-Toeplitz" structure. Additionally, $\boldsymbol{T}$ is strongly band structured.

A linear multi-user detection criterion can now be defined by requesting that the 'best' solution $\hat{\boldsymbol{d}}$ is the one that has the *least square error*

$$\|\boldsymbol{T}\hat{\boldsymbol{d}} - \boldsymbol{x}\|.$$

(We can find the user specific estimated data symbols $\hat{\boldsymbol{d}}^{(k)}$ easily in $\hat{\boldsymbol{d}}$.) This is a standard optimization problem that leads to the *pseudo-inverse* $\boldsymbol{T}^{+}$ of $\boldsymbol{T}$ [9] which is used to compute

$$\hat{\boldsymbol{d}} = \boldsymbol{T}^{+}\boldsymbol{x}. \tag{3}$$

In our case, $\boldsymbol{T}^{+}$ can be expressed as

$$\boldsymbol{T}^{+} = (\boldsymbol{T}^{H}\boldsymbol{T})^{-1}\boldsymbol{T}^{H}.$$

Computing $\boldsymbol{T}^{+}$ in general is expensive. However, for the pronounced structure of $\boldsymbol{T}$ as shown in equation (2), there exist quite effective methods to find $\hat{\boldsymbol{d}}$ [17].

3 Displacement Representation and Schur Algorithm

Instead of computing $\hat{\boldsymbol{d}}$ via equation (3), one can also compute the QR factorization of $\boldsymbol{T}$ such that $\boldsymbol{T} = \boldsymbol{Q}\boldsymbol{R}$ and

$$\hat{\boldsymbol{d}} = \boldsymbol{R}^{-1}\boldsymbol{Q}^{H}\boldsymbol{x}. \tag{4}$$

The matrix $\boldsymbol{Q} \in \mathbb{C}^{L \times KN}$ satisfies $\boldsymbol{Q}^{H}\boldsymbol{Q} = \boldsymbol{I}$ and $\boldsymbol{R} \in \mathbb{C}^{KN \times KN}$ is upper triangular. The matrix $\boldsymbol{R}$ is also known as the Cholesky factor of $\boldsymbol{T}^{H}\boldsymbol{T}$.

The Schur algorithm is a way to compute $\boldsymbol{R}$ and $\boldsymbol{z} = \boldsymbol{Q}^{H}\boldsymbol{x}$ while exploiting the Block-Toeplitz structure of $\boldsymbol{T}$. It starts by computing a *displacement representation* for $\boldsymbol{T}$ and $\boldsymbol{x}$ and then continues to gradually transform them into

$\boldsymbol{R}$ and $\boldsymbol{z}$ by applying local unitary and hyperbolic transformations [1, 8].

The displacement representation starts from the Gramian matrix $\boldsymbol{S} = \boldsymbol{T}^H\boldsymbol{T}$. This is a band-structured, hermitian, positive-semidefinite Block-Toeplitz matrix. In the following, we assume that it is actually positive-definite instead of only semidefinite. The first step is to split $\boldsymbol{S}$ into a sum of rank-two matrices $\boldsymbol{\Gamma}_i$, one for each 'hook' of $\boldsymbol{S}$. By 'hook i', we refer to the hook-shaped region of a matrix that consists of all elements below and to the right of the ith diagonal element. See Figure 2. Each hook, in turn, is expressed as the difference of two outer products.

More precisely,

$$\boldsymbol{\Gamma}_i = \boldsymbol{\alpha}_i^H\boldsymbol{\alpha}_i - \boldsymbol{\beta}_i^H\boldsymbol{\beta}_i$$

where $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ are suitable row vectors. They can be computed simply as

$$\begin{aligned}\boldsymbol{\alpha}_i &= \boldsymbol{\Gamma}_i(i,:)\big/\sqrt{\boldsymbol{\Gamma}_i(i,i)},\\ \boldsymbol{\beta}_i(j) &= \boldsymbol{\alpha}_i(j) \quad \text{for } j \neq i,\\ \boldsymbol{\beta}_i(i) &= 0.\end{aligned}$$

To summarize, we now have

$$\boldsymbol{S} = \sum_{i=1}^{NK}\big(\boldsymbol{\alpha}_i^H\boldsymbol{\alpha}_i - \boldsymbol{\beta}_i^H\boldsymbol{\beta}_i\big). \tag{5}$$

The next step is to collect all $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ vectors into two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively,

$$\boldsymbol{A} = \begin{bmatrix}\boldsymbol{\alpha}_1\\ \boldsymbol{\alpha}_2\\ \vdots\\ \boldsymbol{\alpha}_{NK}\end{bmatrix} \qquad \boldsymbol{B} = \begin{bmatrix}\boldsymbol{\beta}_1\\ \boldsymbol{\beta}_2\\ \vdots\\ \boldsymbol{\beta}_{NK}\end{bmatrix},$$

to get rid of the explicit sum in equation (5):

$$\boldsymbol{S} = \boldsymbol{A}^H\boldsymbol{A} - \boldsymbol{B}^H\boldsymbol{B}.$$

We can further reduce this expression to

$$\boldsymbol{S} = \boldsymbol{X}^H\boldsymbol{J}\boldsymbol{X}$$

with

$$\boldsymbol{X} = \begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{B}\end{bmatrix}, \quad \boldsymbol{J} = \begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & -\boldsymbol{I}\end{bmatrix}.$$

The task of the Schur algorithm is now to find a transformation $\boldsymbol{\Theta}$ such that

$$\boldsymbol{\Theta}\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{B}\end{bmatrix} = \begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix}$$

where $\boldsymbol{R}$ is an upper triangular matrix. When $\boldsymbol{\Theta}$ has been chosen to be $\boldsymbol{J}$-orthogonal, that is, when it satisfies

$$\boldsymbol{\Theta}^H\boldsymbol{J}\boldsymbol{\Theta} = \boldsymbol{J},$$

then $\boldsymbol{R}$ is the desired Cholesky factor of $\boldsymbol{S}$.

Analogous to the more well-known QR decomposition with unitary rotations [2], the transformation $\boldsymbol{\Theta}$ itself is composed of individual elementary transformations that each eliminate one element in the $\boldsymbol{B}$-part of $\boldsymbol{X}$. Such a elementary transformation is either an unitary or a hyperbolic rotation, depending on whether it affects only the $\boldsymbol{B}$-part of $\boldsymbol{X}$, or both the $\boldsymbol{A}$- and $\boldsymbol{B}$-part.

An elementary hyperbolic rotation $\boldsymbol{H} \in \mathbb{C}^{2\times 2}$ is defined by the conditions

$$\boldsymbol{H}\begin{bmatrix}a\\ b\end{bmatrix} = \begin{bmatrix}r\\ 0\end{bmatrix} \text{ and}$$
$$\boldsymbol{H}^H\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\boldsymbol{H} = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}.$$

Similar to a unitary rotation, it can be computed (for real valued a and b) as

$$\boldsymbol{H} = \begin{bmatrix}\cosh(\phi) & -\sinh(\phi)\\ -\sinh(\phi) & \cosh(\phi)\end{bmatrix}, \quad \phi = \tanh^{-1}\Big(\frac{b}{a}\Big)$$

or more directly (and also for complex valued a and b) as

$$\boldsymbol{H} = \begin{bmatrix}c^* & s^*\\ s & c\end{bmatrix}, \qquad \begin{aligned}c &= a/r, \quad s = -b/r,\\ r &= \sqrt{a^*a - b^*b}\end{aligned}$$

Hyperbolic rotations are only defined for $|b| < |a|$.

Figure 3 shows a possible sequence of hyperbolic rotations to eliminate the upper-left and lower-right blocks of the $\boldsymbol{B}$-part for $K = 2$ and $N = 2$. It should be easy to see how to eliminate the remaining upper-right block. (The reason why we include two seemingly unnecessary rotations will become apparent shortly.)

This complicated way of arriving at $\boldsymbol{R}$ does not seem to gain anything compared to a more straightforward QR decomposition of $\boldsymbol{T}$. The trick is to observe that $\boldsymbol{A}$ and $\boldsymbol{B}$ inherit the Block-Toeplitz structure of $\boldsymbol{S}$ and that this structure is preserved to a large degree while $\boldsymbol{B}$ is transformed to zero and $\boldsymbol{A}$ to $\boldsymbol{R}$. For example, the lower-right block in Figure 3 can be eliminated with the same sequence of rotations as the upper left block, only placed differently. Therefore, we don't need to explicitely carry out these computations and can just copy their result from their previous applications. Applying this to Figure 3 lets us skip the second batch of transformations below the dotted line.

Looking more closely at the process, as for example done in [16], we can see that $\boldsymbol{B}$ remains Block-Toeplitz throughout, and it therefore suffices to store only the first K rows. Additionally, $\boldsymbol{A}$ can be partitioned into two parts: an upper one which is not Block-Toeplitz, and a lower one, which is. The border between these two parts moves downwards during the transformation and it can be seen that the elements in the upper part will not be touched again. This is depicted in Figure 3 by separating the upper two rows of $\boldsymbol{A}$ during

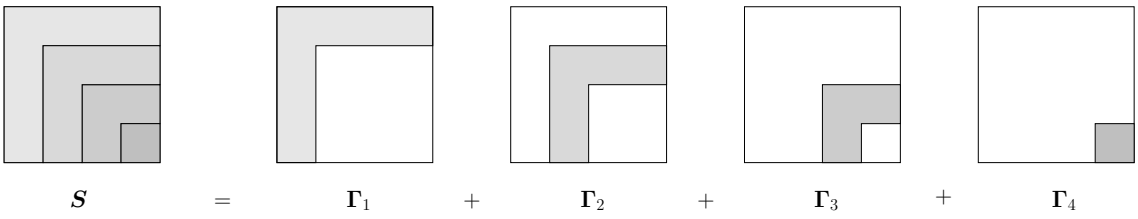

Figure 2. The hooks of S for $S \in \mathbb{C}^{4\times 4}$.

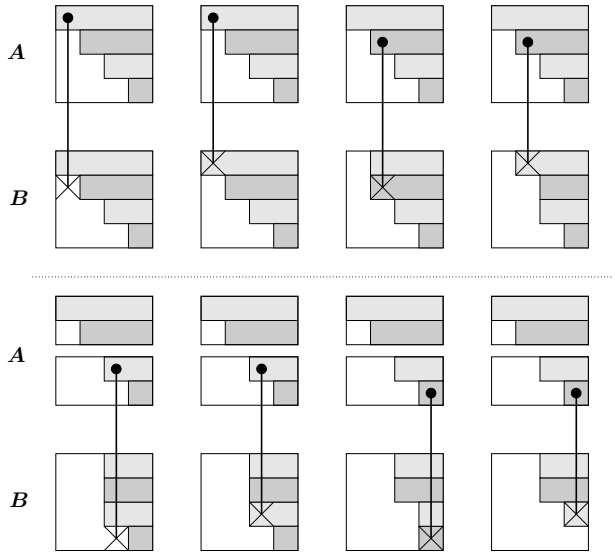

Figure 3. Eliminating the upper-left and lower-right blocks in B for $K = 2, N = 2$.

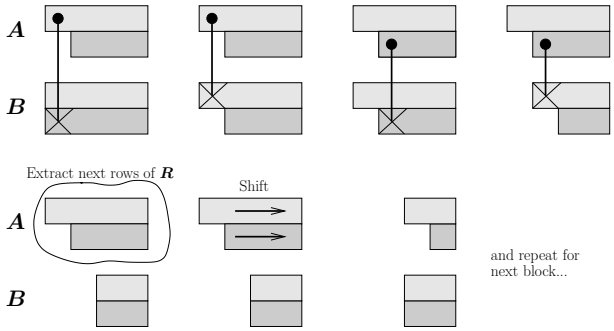

Figure 4. Eliminating one block in B while exploiting the structure.

the second batch of transformations. Effectively, the upper part of $\boldsymbol{A}$ contains the first rows of $\boldsymbol{R}$.

Figure 4 shows how to exploit these insights by only working with non-redundant data.

The right hand side $\boldsymbol{x}$ can be included in this process so that $\boldsymbol{Q}^H \boldsymbol{x}$ becomes available at the same time as $\boldsymbol{R}$, see [16] for details. Also, it is of course possible (and straightforward) to exploit the band structure that is present in $\boldsymbol{T}$ and $\boldsymbol{S}$ by avoiding operations that are known to process only zeros.

In addition to exploiting the structure of $\boldsymbol{T}$, we can also introduce approximations into the solution process. The limited inter-symbol-interference of the system leads to the fact that $\boldsymbol{B}$ converges to zero quite rapidly and thus the transformations can be stopped early. In other words, the later transformations will find $\boldsymbol{B}$ to be already quite close to zero, and can be omitted entirely.

Figure 5 depicts this process for arbitrary K and $N = 6$ when only $2K$ rows of $\boldsymbol{R}$ are computed. When it has been decided that $\boldsymbol{B}$ is close enough to zero, the transformations are stopped and the remaining rows of $\boldsymbol{R}$ are produced by continuing to shift $\boldsymbol{A}$ and copying it into the appropriate parts of $\boldsymbol{R}$. The figure also shows the effects of the band structure of $\boldsymbol{T}$ and $\boldsymbol{S}$.

4 Parallel Processor Array for the Schur Algorithm

The algorithm laid out in the previous section can be implemented on a processor array that is very similar to the familiar QR array [6]. Instead of orthogonal or unitary rotations, however, it uses hyperbolic ones.

A hyperbolic rotation for complex values can be build from three real-valued rotations: two orthogonal ones and one hyperbolic. The orthogonal rotations are used to extinguish the imaginary parts of a and b, the hyperbolic rotation eliminates the remaining real part of b. This can be reduced to just one orthogonal rotation and one hyperbolic one when care is taken to produce only real-valued elements on the diagonal of $\boldsymbol{A}$. Figure 7 shows this graphically, using the cells defined in Figure 6.

The complete array for eliminating one block of $\boldsymbol{B}$ can then be composed from these complex-valued, hyperbolic vector and rotation cells as shown in Figure 8 for $K = 2$ and $N = 2$. It also includes the registers that store $\boldsymbol{A}$ and indicates their initial values. After the K rows of $\boldsymbol{B}$ have been put through this array, their new contents can be re-

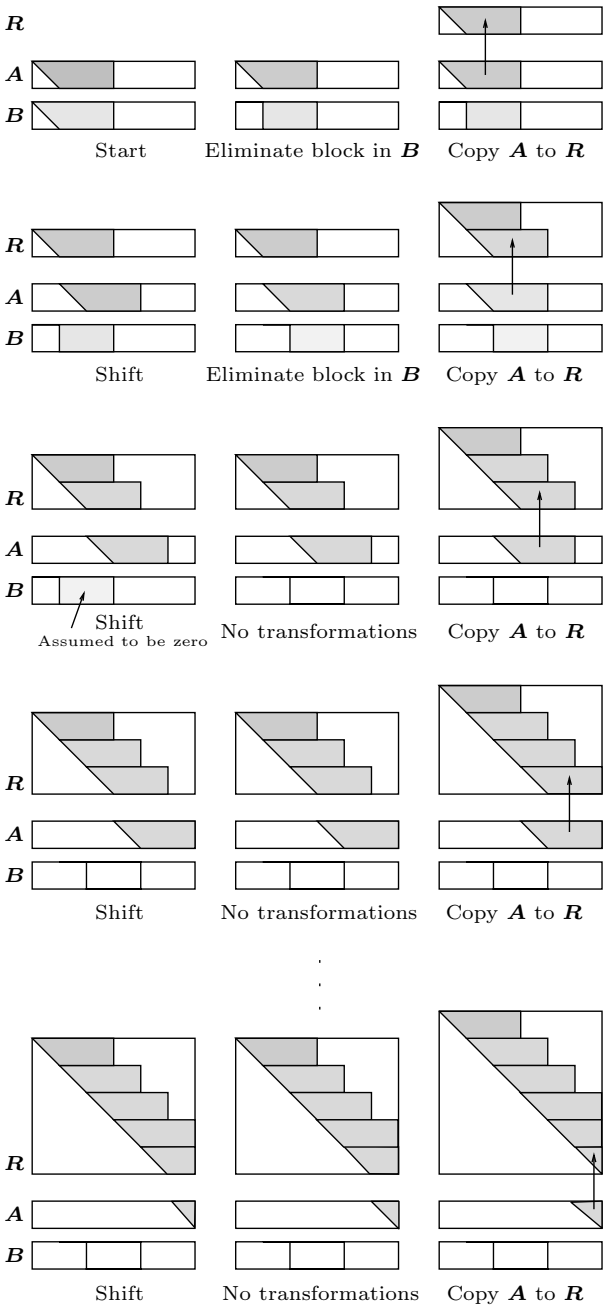

Figure 5. Approximated transformation

trieved at the outputs. The registers will then contain two new rows for $\boldsymbol{R}$. This process is then repeated until $\boldsymbol{B}$ is sufficiently close to zero.

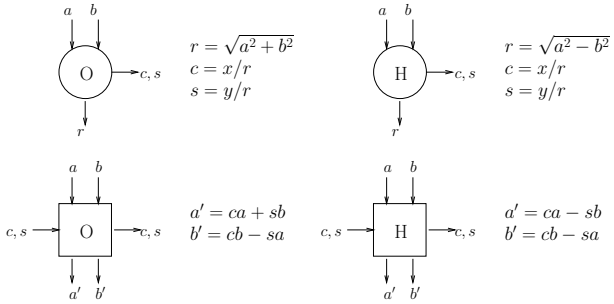

Figure 6. Real-valued orthogonal and hyperbolic cells.

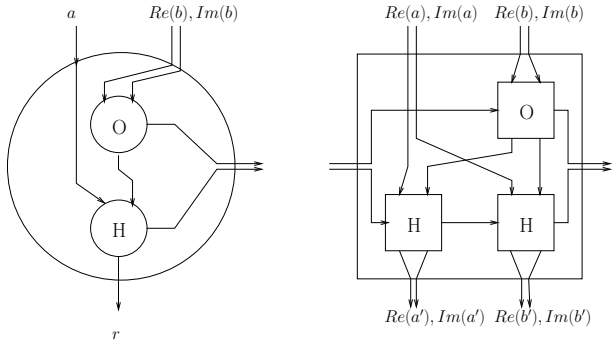

Figure 7. Building complex hyperbolic rotations from real-valued rotations.

The real-valued cells in Figure 6 can be implemented using CORDIC devices [15, 7]. Each CORDIC operation consists of a number of micro rotations [3] that are chosen to be practical for hardware implementations.

Such a CORDIC device implements the complete transformation $\boldsymbol{H}$ by approximating it by a sequence of micro rotations of the form

$$\boldsymbol{M}_i = K_i \begin{bmatrix} 1 & -\mu_i 2^{-s(i)} \\ -\mu_i 2^{-s(i)} & 1 \end{bmatrix},$$
$$\mu_i = \pm 1, \quad K_i = (1 - 2^{-2s(i)})^{-\frac{1}{2}}.$$

The parameter μ_i determines the direction of rotation and i determines the angle. The function $s(i)$ specifies the shift sequence and can, for hyperbolic rotations, be taken as [18]

$$s(i) = \begin{cases} i & \text{for } i \leq 4, \\ i-1 & \text{for } i \leq 14, \\ i-2 & \text{for } i \leq 42, \\ i-3 & \text{else.} \end{cases}$$

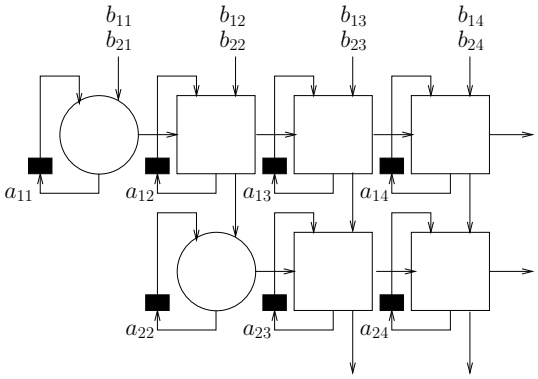

Figure 8. Systolic array for Schur algorithm.

The hardware operations corresponding to $\boldsymbol{M}_i$ (except for the multiplication by K_i) are additions and shifts. Figure 9 shows a schematic for implementing a micro rotation. The scaling factor K_i is independent of μ_i and can be accumulated over the sequence of micro rotations.

The number of micro rotations performed per elementary rotation is an indicator of the amount of hardware resources required to implement the CORDIC device, and of the accuracy attained. The goal is to keep this number as low as possible while still achieving useful results from the multi-user detector.

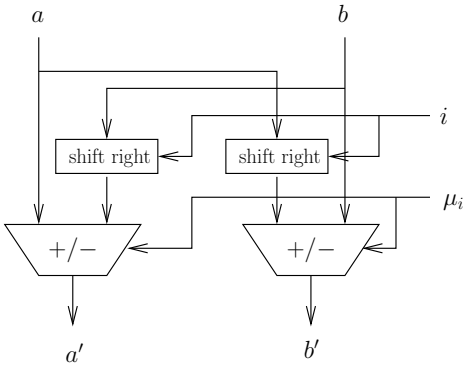

Figure 9. Hardware for computing an unscaled micro rotation.

5 Simulation Results

To verify the effectiveness of the described approximation method, simulations of a linear multi-user detector were performed. The simulated system consisted of a CDMA mobile radio model as depicted in Figure 1 for $K = 2$ users, $N = 60$ symbols per data block, a spreading factor of $Q = 2$ and a channel of length $W = 3$. The amplitude of the channel coefficients were choosen to be

$$|h_i| = \frac{1}{2} + \frac{1}{2}\cos\Big(\frac{2\pi}{M}(i-2)\Big), \quad i = 1, 2, 3.$$

Their phase was varied from sample to sample to be uniformly distributed over time. The parameter M influences the amount of distortion caused by the channel. Larger values of M lead to a worse condition of $\boldsymbol{T}$. The second user was amplified by 20dB compared to the first user to model a severe near/far scenario. The channel coefficients were assumed to be perfectly known at the receiver.

The CORDIC cells in the receiver were simulated with double precision floating point numbers.

Figure 10 shows the bit error ratio of the first (weak) user when no approximations are applied in the Schur algorithm.

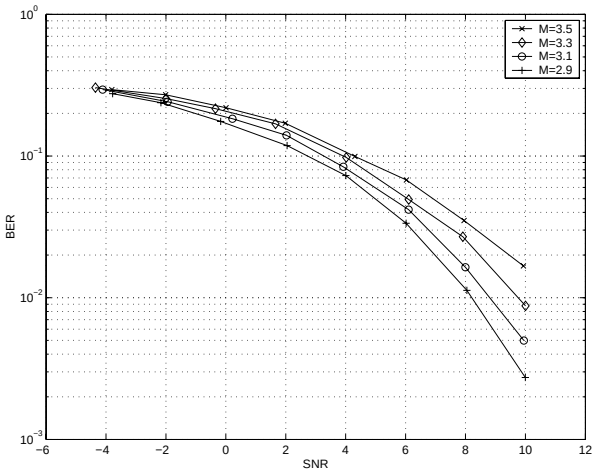

Figure 10. Simulation results without approximation.

The following results use a channel with $M = 2.9$.

Figure 11 shows results for a decreasing number of computed rows in $\boldsymbol{R}$. The legend "depth=5" indicates that $5K$ rows have been computed before stopping the transformation, for example. It can be seen that already $3K$ rows might suffice (out of $60K$) for this simulation scenario.

Figure 12 shows the results for a decreasing number of micro rotations when $4K$ rows of $\boldsymbol{R}$ are computed. It can be seen that 8 iterations already suffice to attain the performance of double precision floating point.

6 Summary

We have presented a hardware oriented, systolic architecture for implementing a complex valued, linear multi-user detector for a burst structured system described by a Block-Toeplitz structured system matrix. The architecture is able to exploit this inherent structure of the matrix.

The presented algorithmic modifications directly lead to less power consumption. As always, a parallel and pipelined implementation is the first step in reducing the power/time consumption [10]. Refinement structures [11]

are then introduced on different levels. First the original algorithm is modified by re-interpreting it as a iterative method and introducing a "depth" parameter that controls the number of iterations. This is justified by the observation that the algorithm converges quickly towards a steady state due to the band structure of the system matrix.

At the architectural level, the complex rotations (realized by two or three real CORDIC devices) are approximated by reducing their number of micro rotations.

Future work will compare the switching activity of the realizations at different incremental refinement steps in order to explicitely show the possible reduction of power consumption by the presented methodology.

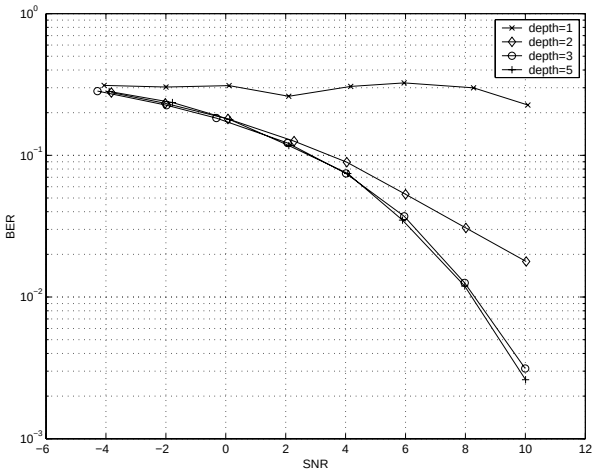

Figure 11. Simulation results for different depths.

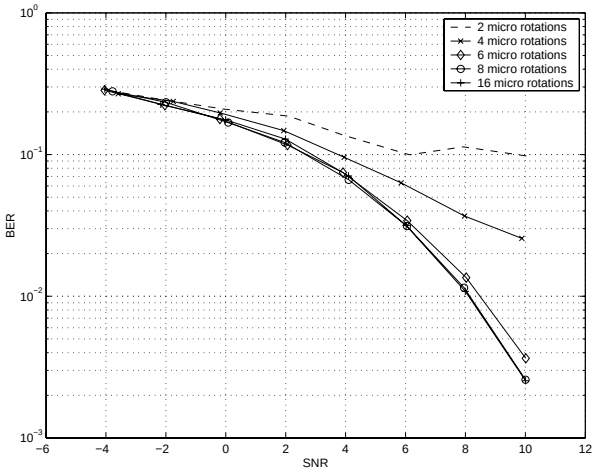

Figure 12. Simulation results for different numbers of micro rotations.

References

[1] J. Chun, T. Kailath, and H. Lev-Ari. Fast Parallel Algorithms for QR and Triangular Factorization. *SIAM J. Sci. Stat. Comput.*, 8(6), November 1987.

[2] G. Golub and C. van Loan. *Matrix Computations*. The John Hopkins University Press, third edition, 1996.

[3] J. Götze and G. Hekstra. An algorithm and architecture based on orthonormal μ–rotations for computing the symmetric evd. *INTEGRATION – The VLSI Journal (Special Issue on Algorithms and Parallel VLSI Architectures)*, 20:21–39, 1995.

[4] M. Haardt, A. Klein, R. Koehn, S. Oestreich, M. Purat, V. Sommer, and T. Ulrich. The TD-CDMA based UTRA TDD mode. *IEEE J. Select. Areas Commun.*, 18:1375–1386, August 2000. special issue on "Wideband CDMA".

[5] M. Haardt and W. Mohr. The complete solution for third generation mobile communications: Two modes on air - One winning strategy. In *Proc. IEEE Int. Conference on Third Generation Wireless Communications*, San Francisco, USA, June 2000.

[6] S. Haykin. *Adaptive Filter Theory*. Prentice Hall, third edition, 1996.

[7] Y. H. Hu. CORDIC-Based VLSI Architectures for Digital Signal Processing. *IEEE Signal Processing Magazine*, July 1992.

[8] T. Kailath and J. Chun. Generalized Displacement Structure for Block-Toeplitz, Toeplitz-Block, and Toeplitz-Derived Matrices. *SIAM J. Matrix Anal. Appl*, 15(1):114–128, January 1994.

[9] T. Lewis and P. Odell. *Estimation in Linear Models*. Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1971.

[10] R. Mehra, D. Lidsky, A. Abnous, P. Landman, and J. Rabaey. Algorithm and architectural level methodologies for low power. *Low Power Design Methodologies, Kluwer Academic Publishers*, 1996.

[11] S. Nawab, A. Oppenheim, A. Chandrakasan, J. Winograd, and J. Ludwig. Approximate signal processing. *J. of VLSI Sig. Proc. Syst.*, 15:177–200, 1997.

[12] I. Schur. Über Potenzreihen, die im Inneren des Einheitskreises beschränkt sind. I. *J. für reine und angewandte Mathematik*, 147:205–232, 1917.

[13] I. Schur. On power series which are bounded in the interior of the unit circle. i. In I. Gohberg, editor, *Operator Theory: Advances and Applications*, pages 31–59. Birkhäuser Verlag, 1986.

[14] S. Verdu. *Multiuser Detection*. Cambridge University Press, 1998.

[15] J. E. Volder. The CORDIC Trigonmetric Computing Technique. *IRE Transactions on Electronic Computers*, EC(8):330–334, 1959.

[16] M. Vollmer, M. Haardt, and J. Götze. Schur algorithms for Joint Detection in TD-CDMA based mobile radio systems. *Annals of Telecommunications (special issue on multi user detection)*, 54(7-8):365–378, July-August 1999.

[17] M. Vollmer, M. Haardt, and J. Götze. Comparative Study of Joint-Detection Techniques for TD-CDMA Based Mobile Radio Systems. *IEEE J. Select. Areas Commun.*, 19:1461–1475, August 2001.

[18] J. Walther. A unified algorithm for elementary functions. *Spring Joint Computer Conf.*, pages 379–385, 1971.