Advances in Radio Science (2003) 1: 211–217
© Copernicus GmbH 2003

Advances in
Radio Science

Incremental refinement of a multi-user-detection algorithm (II)

M. Vollmer and J. Götze

University of Dortmund, Information Processing Lab, Germany

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 a 3GPP TDD multi-user system. This approach is based on a displacement representation of 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 shown that this hardware architecture can also be used to compute the initial displacement representation.

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 third 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 (Verdu, 1998). With this technique, the data symbols of one user are estimated by 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 (Haardt and Mohr, 2000). However, for the specific case of the TDD mode of UMTS (Haardt et al., 2000) with its burst-structured transmission and relatively few simultaneously active users, there exist algorithms and implementation strategies that can significantly reduce the computational requirements (Vollmer et al., 2001).

In this paper, we concentrate on the Schur algorithm (Schur, 1917, 1986). 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).

We will shortly present the mathematical formulation of our specific linear multi-user detector in Sect. 2. In Sect. 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 Sect. 5.

2 Linear multi-user detection in a burst-structured system

As explained in, for example (Vollmer et al., 2001), the transmission in a burst-structured CDMA system can be modeled by the matrix equation

$$\boldsymbol{x} = \mathbf{T}\boldsymbol{d} + \boldsymbol{n}.$$

In this equation, $\boldsymbol{d}$ represents the collection of the data symbols sent by K users, where each user contributes N sym-

Correspondence to: M. Vollmer
(marius.vollmer@udo.edu)

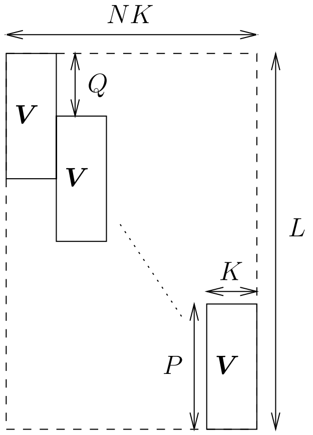

Fig. 1. The structure of $\mathbf{T}$. It is build from copies of the unstructured matrix $\mathbf{V}$, resulting in a Block-Toeplitz-Structure for $\mathbf{T}$. K: number of active CDMA codes, Q: spreading factor, P: $Q + W - 1$ with W the channel length.

bols; $\mathbf{T}$ models the (assumed) linear and time-invariant transmission; n contains noise; and $\boldsymbol{x}$ contains the received samples. The task of the receiver is to compute an estimate $\hat{\boldsymbol{d}}$ of $\boldsymbol{d}$ from the knowledge of $\boldsymbol{x}$ and $\mathbf{T}$. This can be done by using the pseudo-inverse $\mathbf{T}^+$ of $\mathbf{T}$ (Lewis and Odell, 1971) and computing

$$\hat{\boldsymbol{d}} = \mathbf{T}^+\boldsymbol{x} = (\mathbf{T}^H\mathbf{T})^{-1}\mathbf{T}^H\boldsymbol{x}. \tag{1}$$

Computing $\mathbf{T}^+$ in general is expensive. However, the structure of $\mathbf{T}$ as shown in Fig. 1 allows the application of specialized methods that can reduce the computational requirements significantly (Vollmer et al., 2001).

3 Displacement representation and Schur algorithm

Instead of computing $\hat{\boldsymbol{d}}$ via Eq. (1), one can also compute the QR factorization of $\mathbf{T}$ such that $\mathbf{T} = \mathbf{QR}$ and

$$\hat{\boldsymbol{d}} = \mathbf{R}^{-1}\mathbf{Q}^H\boldsymbol{x}. \tag{2}$$

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

The Schur algorithm computes $\mathbf{R}$ and $z = \mathbf{Q}^H\boldsymbol{x}$ while exploiting the Block-Toeplitz structure of $\mathbf{T}$. It starts by computing a *displacement representation* for $\mathbf{T}$ and $\boldsymbol{x}$ and then continues to gradually transform them into $\mathbf{R}$ and z by applying local unitary and hyperbolic transformations (Chun et al., 1987; Kailath and Chun, 1994).

The displacement representation for $\mathbf{T}$ can be written down as

$$\mathbf{T}^H\mathbf{T} = \mathbf{R}^H\mathbf{R} = \mathbf{U}_A^H\mathbf{U}_A - \mathbf{U}_B^H\mathbf{U}_B, \tag{3}$$

where $\mathbf{U}_A$ and $\mathbf{U}_B$ are as depicted in Fig. 2. As can be seen, $\mathbf{U}_A$ is constructed from copies of the matrix $\mathbf{A}$, while $\mathbf{U}_B$ is constructed from copies of $\mathbf{B}$. Moreover, $\mathbf{B}$ can be found in $\mathbf{A}$: the two matrices differ only in the upper triangular part, which is zero in $\mathbf{B}$ and the non-zero matrix $\mathbf{D}$ in $\mathbf{A}$.

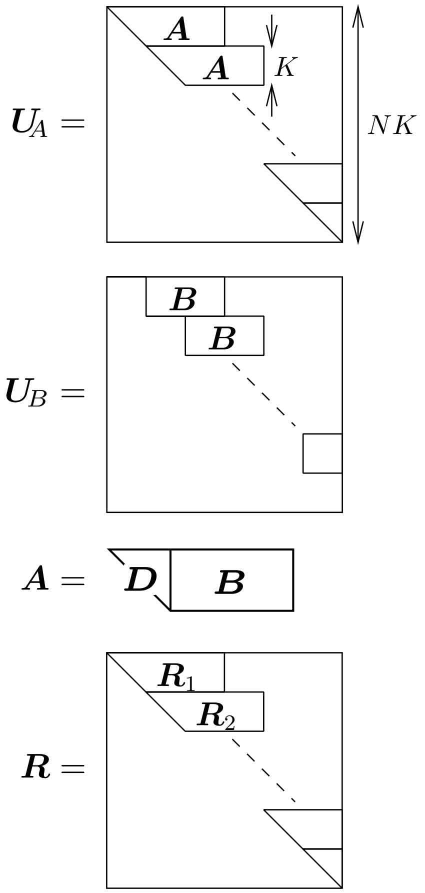

Fig. 2. The structure of $\mathbf{U}_A$, $\mathbf{U}_B$ and $\mathbf{A}$, $\mathbf{B}$. Both $\mathbf{U}_A$ and $\mathbf{U}_B$ inherit the Block-Toeplitz and the band-structure of $\mathbf{T}$.

The matrices $\mathbf{A}$ and $\mathbf{B}$ are called the generators of the displacement representation and need to be computed before the Schur algorithm can start to compute $\mathbf{R}$. By using the matrix partitionings of Fig. 2, one can show that $\mathbf{A} = \mathbf{R}_1$. However, $\mathbf{R}_1$ can also be found in a partial QR-Decomposition of $\mathbf{T}$ that only annihilates the first block-row of $\mathbf{T}$, as shown in Fig. 3. This partial transformation can be computed using a parallel processor array and fortunately, we will find that we can use the same processor array to compute the rest of the Schur algorithm.

As explained in (Vollmer et al., 1999), z can be computed by the Schur algorithm from the right hand side generator $\boldsymbol{y}$ that is chosen such that $\mathbf{T}^H\boldsymbol{x} = \mathbf{U}_D^H\boldsymbol{y}$ where $\mathbf{U}_D = \mathbf{U}_A - \mathbf{U}_B$ is a block-diagonal, Block-Toeplitz matrix with $\mathbf{D}$ on its diagonal. Just as $\mathbf{A}$ can be found in the first rows of $\mathbf{GT}$, it can be shown that the first K elements of $\mathbf{Gx}$ contain the first elements of $\boldsymbol{y}$. The next K elements of $\boldsymbol{y}$ can be found in the first elements of $\mathbf{Gx}'$ where $\boldsymbol{x}'$ is a shifted version of $\boldsymbol{x}$ with $\boldsymbol{x}'_i = \boldsymbol{x}_{i+Q}$. Figure 4 shows graphically how this can be extended to all parts of y.

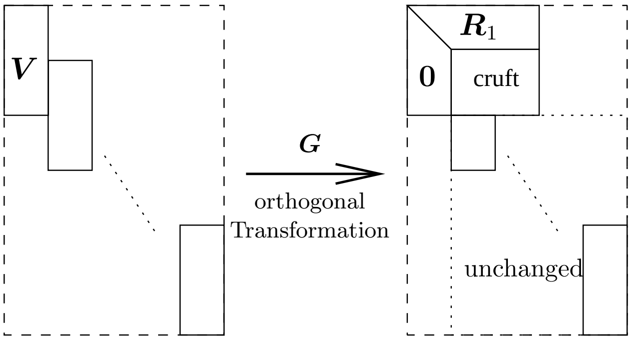

Fig. 3. Partial QR-Decomposition of **T**. After annihilating the indicated part with a orthogonal transformation, the first rows of the result **GT** contain $\mathbf{R}_1$.

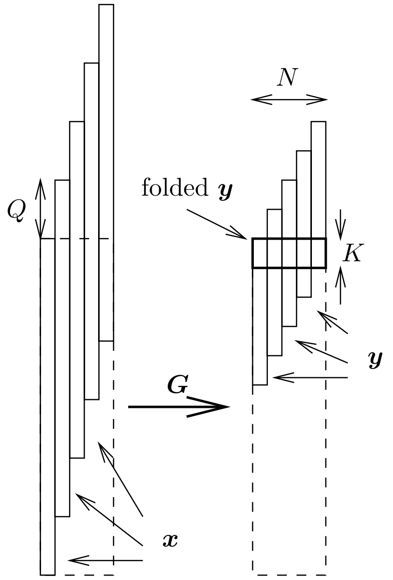

Fig. 4. Transforming $\boldsymbol{x}$ piecewise into $\boldsymbol{y}$, using **G** from Figure 3

Equation (3) can be rewritten as

$$\mathbf{T}^H\mathbf{T} = \begin{bmatrix}\mathbf{U}_A\\ \mathbf{U}_B\end{bmatrix}^H \begin{bmatrix}\mathbf{I} & \\ & -\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{U}_A\\ \mathbf{U}_B\end{bmatrix}.$$

The Schur algorithm then completes the QR-Decomposition by applying a transformation $\boldsymbol{\Theta}$ with the properties:

$$\boldsymbol{\Theta}\begin{bmatrix}\mathbf{U}_A\\ \mathbf{U}_B\end{bmatrix} = \begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix} \quad \text{and} \quad \boldsymbol{\Theta}^H\begin{bmatrix}\mathbf{I} & \\ & -\mathbf{I}\end{bmatrix}\boldsymbol{\Theta} = \begin{bmatrix}\mathbf{I} & \\ & -\mathbf{I}\end{bmatrix}.$$

It can be shown that the result of applying $\boldsymbol{\Theta}$ to two stacked $\boldsymbol{y}$ contains z:

$$\boldsymbol{\Theta}\begin{bmatrix}\boldsymbol{y}\\ \boldsymbol{y}\end{bmatrix} = \begin{bmatrix}z\\ *\end{bmatrix} \quad \text{since} \quad \mathbf{T}^H\boldsymbol{x} = \begin{bmatrix}\mathbf{U}_A\\ \mathbf{U}_B\end{bmatrix}\begin{bmatrix}\mathbf{I} & \\ & -\mathbf{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{y}\\ \boldsymbol{y}\end{bmatrix}. \quad (4)$$

Analogous to the more well-known QR decomposition with unitary rotations (Golub and van Loan, 1996), the transformation $\boldsymbol{\Theta}$ itself is composed of individual elementary transformations that each eliminate one element in $\mathbf{U}_B$. Such an elementary transformation is either a unitary or a hyperbolic rotation, depending on whether it affects only the $\mathbf{U}_B$-part of **x**, or both the $\mathbf{U}_A$- and $\mathbf{U}_B$-part.

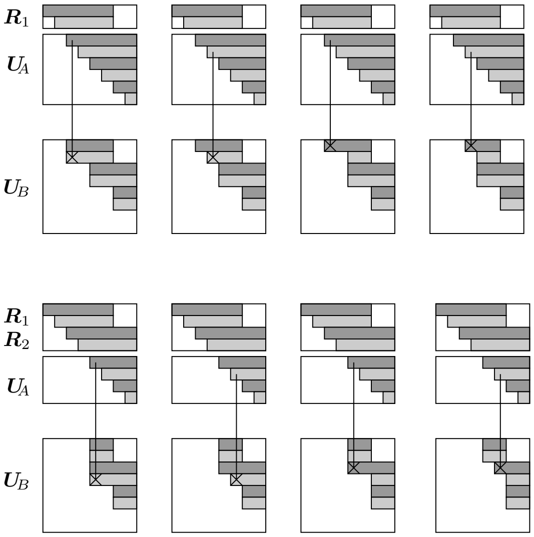

Fig. 5. Eliminating two blocks in $\mathbf{U}_B$ for $K = 2$, $N = 4$.

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

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

Similar to a unitary rotation, it can be computed as

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

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

The first variant is only valid for real valued a and b, while the second variant is also valid for complex values. Hyperbolic rotations are only defined for $|b| < |a|$.

The task of $\boldsymbol{\Theta}$ is to transform $\mathbf{U}_B$ to zero. Figure 5 shows a possible sequence of elementary hyperbolic transformations to eliminate two blocks. It should be easy to see how to eliminate the remaining blocks.

This complicated way of arriving at **R** does not seem to gain anything compared to a more straightforward QR decomposition of **T**. The trick is to observe that $\mathbf{U}_A$ and $\mathbf{U}_B$ inherit the Block-Toeplitz structure of $\mathbf{T}^H\mathbf{T}$ and that this structure is preserved to a large degree while $\mathbf{U}_B$ is transformed to zero and $\mathbf{U}_A$ to **R**. For example, the second block in Fig. 5 can be eliminated with the same sequence of rotations as the first block 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 Fig. 5 allows us to skip the second batch of transformations completely.

Looking more closely at the process, as for example done in (Vollmer et al., 1999), we can see that $\mathbf{U}_B$ remains Block-

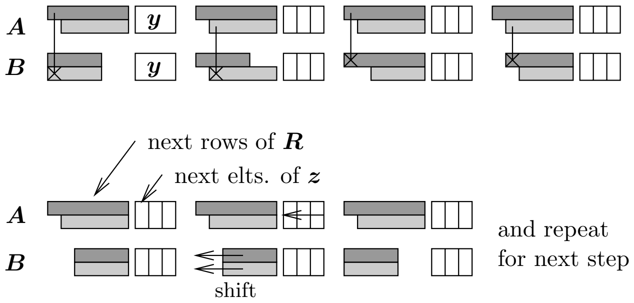

Fig. 6. Eliminating one block in **B** while exploiting the structure.

Toeplitz throughout, and it therefore suffices to store only the first K rows, that is, **B**. Additionally, $\mathbf{U}_A$ 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 Fig. 5 by separating the upper rows of $\mathbf{U}_A$ and labeling them $\mathbf{R}_1$ and $\mathbf{R}_2$, respectively. Thus, the upper part of the gradually transformed $\mathbf{U}_A$ contains more and more rows of the desired result **R**. Figure 6 shows how to exploit these insights by only working with non-redundant data.

The generator for the right hand side $\boldsymbol{y}$ needs to be transformed in the same way as $\mathbf{U}_A$ and $\mathbf{U}_B$. Since the actual computation will only apply the non-redundant parts of $\boldsymbol{\Theta}$, but $\boldsymbol{y}$ is in principle affected by all parts, this requires a small twist. The redundant parts of $\boldsymbol{\Theta}$ can be resurrected by transforming the folded version of $\boldsymbol{y}$ (indicated in Fig. 4) alongside with **A** and **B**. As shown above in (4), two copies of $\boldsymbol{y}$ enter the transformation and thus two copies of the folded $\boldsymbol{y}$ are present in the final algorithm: one next to **A**, and one next to **B**; see Fig. 6.

It should be noted that is is of course possible (and straightforward) to exploit the band structure that is present in **T** (and thus **A** and **B**) by avoiding operations that are known to process only zeros.

In addition to exploiting the structure of **T**, we can also introduce approximations into the solution process. The limited inter-symbol-interference of the system leads to the fact that **B** converges to zero quite rapidly and thus the transformations can be stopped early. In other words, the later transformations will find **B** to be already quite close to zero, and can be omitted entirely. The remaining rows of **R** are produced by continuing to shift **A** and $\boldsymbol{y}$ and copying them into the appropriate parts of **R** and $\boldsymbol{z}$.

4 Parallel processor array for the Schur algorithm

The partial QR-Decomposition for computing the generators can be carried out on a partial QR array, as depicted in Fig. 7. The relevant part of **T** is fed into the array and when the partial decomposition has finished, the feedback registers contain the indicated elements of **A** and $\boldsymbol{y}$. One step of the Schur algorithm (Fig. 6) can then be computed on the same array by replacing the orthogonal rotations with hyperbolic ones. After the K rows of **B** have been put through this array, their new contents can be retrieved at the outputs. The registers will then contain new rows for **R**. This process is repeated until **B** is sufficiently close to zero. At the same time, the generator $\boldsymbol{y}$ for the right hand side is transformed and after each step, new elements for $\boldsymbol{z}$ are produced.

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 **A**. Figure 8 shows this graphically.

The complete array can then be composed from these complex-valued, hyperbolic vector and rotation cells as shown in Fig. 7 for $K = 2$ and $N = 2$.

The real-valued cells in Fig. 8 can be implemented using CORDIC devices (Volder, 1959; Hu, 1992). Each CORDIC operation consists of a number of micro rotations (Götze and Hekstra, 1995) that are chosen to be practical for hardware implementations.

A CORDIC device implements the complete transformation **H** by approximating it by a sequence of micro rotations of the form

$$\mathbf{M}_i = K_i \begin{bmatrix} 1 & -\mu_i 2^{-s(i)} \\ -\mu_i 2^{-s(i)} & 1 \end{bmatrix},$$

$$\mu_i = \pm 1, \qquad K_i = (1 - 2^{-2s(i)})^{-\frac{1}{2}}$$

such that $\mathbf{H} \approx \prod_{i=0}^{n-1} \mathbf{M}_{n-i}$. 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 (Walther, 1971)

$$s(i) = \{1, 1, 2, 3, 4, 4, 5, \ldots, 12, 13, 13, 14, \\ \ldots, 39, 40, 40, 41, 42, \ldots\}.$$

The hardware operations corresponding to $\mathbf{M}_i$ (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 n 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.

5 Simulation results

To verify the effectiveness of the described approximation method, simulations of a linear multi-user detector in a CDMA system as specified in TS 25.102 (3GPP, 2002) were

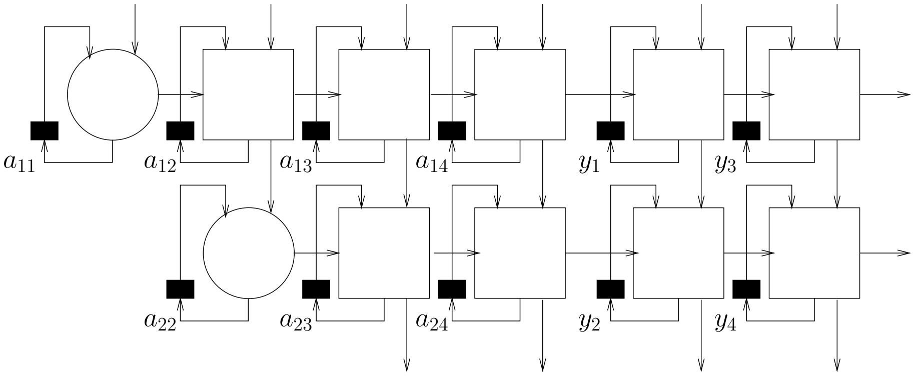

Fig. 7. Systolic array for Schur algorithm.

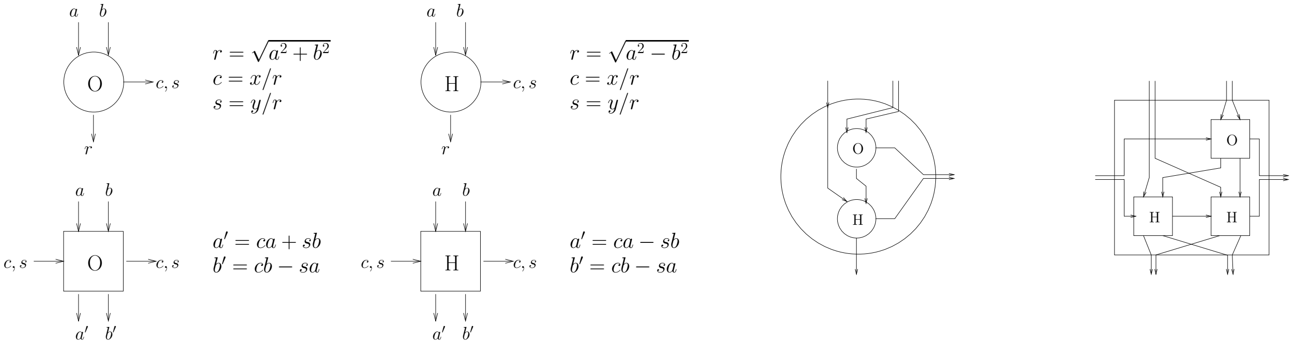

Fig. 8. Real-valued orthogonal and hyperbolic cells and how to build complex hyperbolic cells from them.

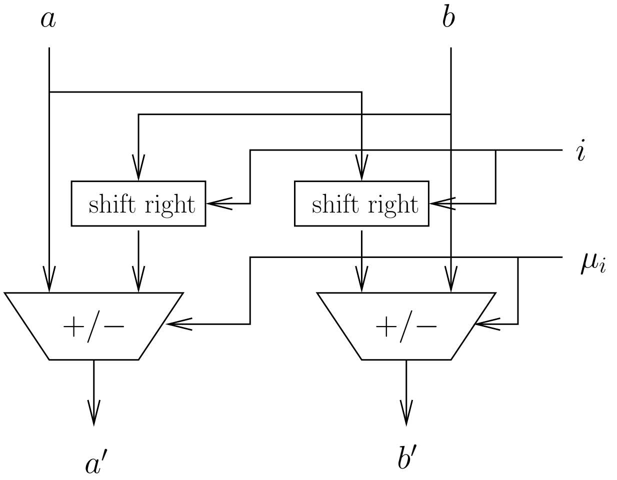

Fig. 9. Hardware for computing an unscaled micro rotation.

Table 1. Simulation parameters

spreading factor	Q	8
number of active codes	K	6
symbols per half-burst	N	122
channel length in chips	W	57
midamble length in chips		512
mobile speed in km/h		120
relative tap delays in ns		0, 260, 521, 781
corresp. rel. power in dB		0, −3, −6, −9
simulated bits per code		10^6

performed. The scenario was chosen according to Annex A.2.1: uplink service with 12.2 kbs using the 3.84 Mcps TDD option. The channel was modeled according to Annex B.2.1, case 3. The resulting parameters are summarized in Table 1. The channel estimation was assumed to be perfect.

The described processor array was used as the receiver in this system. The number of Schur steps and the number of micro rotations per hyperbolic/orthogonal transformation was varied. The quality of the resulting receiver was assessed by measuring the bit error ratio for different signal to noise ratios. The CORDIC cells in the receiver were simulated with IEEE double precision floating point numbers.

Each instantiation of the array received the exact same input data. That is, although the simulation used random numbers to model the data source and the time variance of the radio channel, each BER measurement used the same sequence of random numbers. As can be seen in Fig. 10, the bit error ratios for a user depend heavily on the concrete realization of the radio channel and it is thus important to keep all parameters of a simulation constant (including the random numbers), except for the parameter that is being examined.

Figure 11 shows results for a decreasing number of computed rows in **R**, for two selected users. The legend “depth=5” indicates that $5K$ rows have been computed before stopping the transformation, for example. It can be seen that already $5K$ rows might suffice (out of $122K$) for this simulation scenario. Also, results for a decreasing number of micro rotations when $5K$ rows of **R** are computed are shown. The legend “n,m Micro-Rotations” means that n micro-rotations are used for the orthogonal cells of Fig. 8 and m are used for the hyperbolic ones. It can be seen that 9 orthogonal and 10 hyperbolic micro-rotations suffice to nearly attain the performance of double precision floating point.

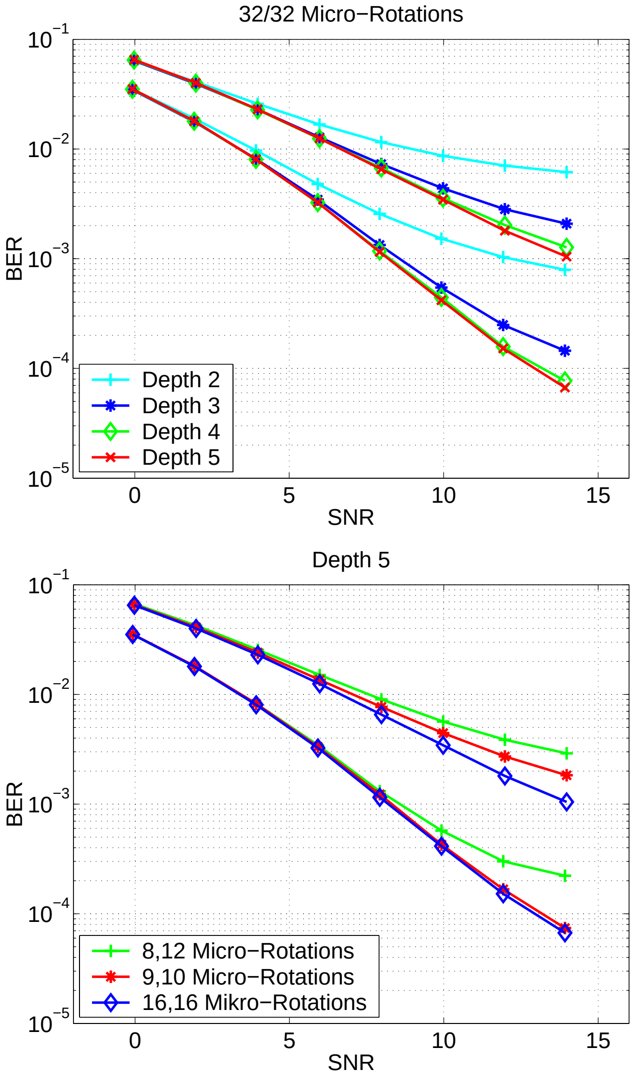

Fig. 11. Simulation results.

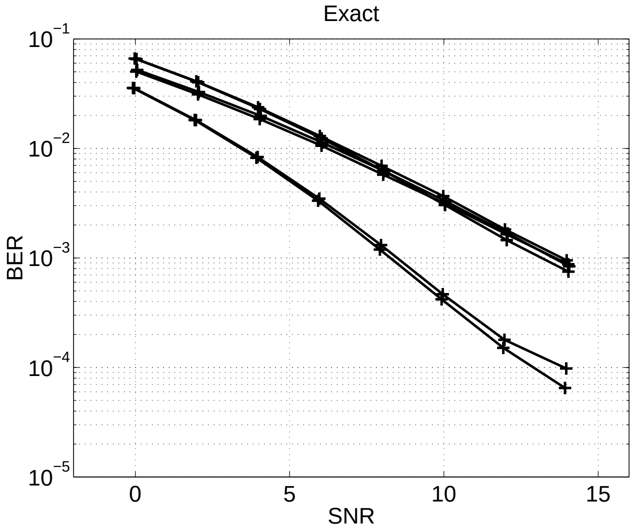

Fig. 10. BER without approximations for each user.

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 (Mehra et al., 1996). Refinement structures (Nawab et al., 1997) are then introduced on different levels. First the original algorithm is modified by re-interpreting it as an 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 explicitly show the possible reduction of power consumption by the presented methodology.

References

3GPP: TS 25.102 UE Radio Transmission and Reception (TDD), 2002.

Chun, J., Kailath, T., and Lev-Ari, H.: Fast Parallel Algorithms for QR and Triangular Factorization, SIAM J. Sci. Stat. Comput., 8, 1987.

Golub, G. and van Loan, C.: Matrix Computations, The John Hopkins University Press, third edn., 1996.

Götze, J. and Hekstra, G.: 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.

Haardt, M. and Mohr, W.: 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, 2000.

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

Hu, Y. H.: CORDIC-Based VLSI Architectures for Digital Signal Processing, IEEE Signal Processing Magazine, 1992.

Kailath, T. and Chun, J.: Generalized Displacement Structure for Block-Toeplitz, Toeplitz-Block, and Toeplitz-Derived Matrices, SIAM J. Matrix Anal. Appl, 15, 114–128, 1994.

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

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

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

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

Schur, I.: On power series which are bounded in the interior of the unit circle. i., in: Operator Theory: Advances and Applications, (Ed) Gohberg, I., pp. 31–59, Birkhäuser Verlag, 1986.

Verdu, S.: Multiuser Detection, Cambridge University Press, 1998.

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

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

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

Walther, J.: A unified algorithm for elementary functions, Spring Joint Computer Conf., pp. 379–385, 1971.