Advanced Receiver Algorithms for MIMO Wireless Communications

A. Burg*, M. Borgmann†, M. Wenk*, C. Studer*, and H. Bölcskei†
* Integrated Systems Laboratory
ETH Zurich
8092 Zurich, Switzerland
Email: apburg@iis.ee.ethz.ch

† Communication Technology Laboratory
ETH Zurich
8092 Zurich, Switzerland
Email: {moriborg|boelcskei}@nari.ee.ethz.ch

Abstract

We describe the VLSI implementation of MIMO detectors that exhibit close-to optimum error-rate performance, but still achieve high throughput at low silicon area. In particular, algorithms and VLSI architectures for sphere decoding (SD) and K-best detection are considered, and the corresponding trade-offs between uncoded error-rate performance, silicon area, and throughput are explored. We show that SD with a per-block run-time constraint is best suited for practical implementations.

1. Introduction

Multiple-input multiple-output (MIMO) technology constitutes the basis for next-generation wireless communication systems, for example in the standards IEEE 802.11n or IEEE 802.16. In particular, *spatial multiplexing* allows to achieve a spectral efficiency that grows linearly with the minimum of the number of transmit and receive antennas [1].

Unfortunately, the considerable gains in spectral efficiency resulting from spatial multiplexing are bought dearly at the expense of additional signal processing complexity at the receiver, most prominently in the MIMO detection unit. Optimum error-rate performance is achieved by the *maximum likelihood* (ML) detector. However, since the computational complexity of ML detection grows exponentially in the transmission rate (measured in bits per channel use), its implementation has so far been considered infeasible for high-rate systems.

Contributions. In this paper, we show how MIMO detection algorithms with close-to-ML performance can be implemented for high-rate systems under reasonable resource constraints, i.e., limits on silicon area and processing delay. The two algorithms considered are sphere decoding (SD) and K-best detection, which both have their origins in [2, 3] and were introduced to wireless communications in [4] and [5], respectively. Besides the implementation complexity per se, an important obstacle to the practical application of the SD algorithm has been its variable throughput. We show that this problem can be alleviated, without a significant degradation of error-rate performance, by block processing. Our comparison of SD (with block processing) and K-best detection demonstrates that SD is more efficient in terms of hardware complexity and provides more flexibility to trade throughput against error-rate performance.

Outline. In the remainder of this section, we introduce the system model and describe the practical problems of implementing MIMO detectors with close-to-ML performance. In Section 2, we explain how tree-search algorithms with resource constraints can achieve close-to-ML performance. Section 3 describes the key VLSI implementation aspects of SD and K-best detection. A comparison between the two algorithms is provided in Section 4.

1.1. System Model

We consider a spatial multiplexing MIMO system with M_T transmit and M_R receive antennas. The transmitter sends M_T *spatial streams*, i.e., it chooses the entries of the transmitted M_T-dimensional vector symbol $\mathbf{s}$ independently from a set of complex-valued constellation points $\mathcal{O}$; we write $\mathbf{s} \in \mathcal{O}^{M_T}$. The transmission rate is given by $R = M_T \log_2|\mathcal{O}|$ bits per channel use (bpcu), where $|\mathcal{O}|$ denotes the cardinality of the set $\mathcal{O}$. The corresponding received vector $\mathbf{y}$ is given by

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

where $\mathbf{H}$ is the $M_R \times M_T$-dimensional channel matrix and $\mathbf{n}$ is an M_R-dimensional noise vector. The noise is i.i.d. proper complex Gaussian distributed.

Throughout the paper, we consider an uncoded 4×4 system with Gray-labeled 16-QAM modulation. SNR denotes the average (with respect to the channel) signal-to-noise ratio at each receive antenna.

1.2. The MIMO ML Detection Problem

We consider *coherent detection*, i.e., we assume that the receiver has perfect knowledge of the channel matrix $\mathbf{H}$, obtained for example through a training phase preceding the

3-9810801-0-6/DATE06 © 2006 EDAA

data transmission phase. The task of the MIMO detector is to recover $\mathbf{s}$ from $\mathbf{y}$. Optimum error-rate performance is achieved by ML detection, which amounts to finding the vector symbol

$$\hat{\mathbf{s}} = \underset{\mathbf{s}\in\mathcal{O}^{M_T}}{\arg\min} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|. \tag{2}$$

A straightforward approach to solving (2) is an exhaustive search over all possible candidate vector symbols $\mathcal{O}^{M_T}$. However, since the number of possible solutions grows exponentially in R, the implementation of an exhaustive search becomes quickly prohibitive as R increases. For example, in our setting of a 4×4 MIMO system with 16-QAM modulation ($R = 16$ bpcu), an exhaustive search would require the consideration of 65 536 candidate vector symbols.

Traditional approaches to MIMO detection resort to linear receivers or successive interference cancellation (SIC) algorithms, both of which exhibit an error-rate performance inferior to that achieved by ML detection. In this paper, we shall focus on the question of how to build VLSI implementations of MIMO detectors that exhibit close-to-ML performance, but are yet feasible under reasonable resource constraints. A viable solution to this problem is to search only a fraction of the vector-symbol alphabet. However, to achieve close-to-ML performance we need to ask the question: *How can we confine our search space while maintaining a high probability of finding the ML solution?* We shall see that, in terms of achievable throughput and error-rate performance, SD and K-best detection represent attractive solutions to this problem.

2. MIMO Detection by Tree Search

In the following, we describe the two most relevant tree-search algorithms for MIMO detection, SD and K-best detection.

2.1. Principles of Tree-Search Algorithms

Triangularization. The first step in mapping the ML detection problem to a tree-search problem is computing the QR decomposition $\mathbf{H} = \mathbf{Q}\mathbf{R}$, where $\mathbf{R}$ is upper-triangular and $\mathbf{Q}$ is unitary. Left-multiplying (1) by $\mathbf{Q}^H$ leads to a modified input-output relation according to

$$\hat{\mathbf{y}} = \mathbf{R}\mathbf{s} + \mathbf{Q}^H\mathbf{n} \qquad \text{with } \hat{\mathbf{y}} = \mathbf{Q}^H\mathbf{y}$$

so that (2) can be written as

$$\hat{\mathbf{s}} = \underset{\mathbf{s}\in\mathcal{O}^{M_T}}{\arg\min}\, d(\mathbf{s}) \qquad \text{with } d(\mathbf{s}) = \|\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}\|. \tag{3}$$

Before we show how this modification can be exploited to reduce detection complexity, we introduce the set of partial candidate vector symbols $\mathbf{s}^{(i)} = \begin{bmatrix} s_i & s_{i+1} & \cdots & s_{M_T} \end{bmatrix}$, $i = 1, 2, \ldots, M_T$, and we note that the $\mathbf{s}^{(i)}$ can be arranged in a tree that has its root on level $i = M_T + 1$ and leaves, which correspond to the set of all possible candidate vector symbols, on level $i = 1$.

Computing Partial Euclidean Distances. The vector norm in (3) can be computed recursively as $d(\mathbf{s}) = d_1$ with the *partial Euclidean distances* (PEDs)

$$d_i = d_{i+1} + |e_i|^2 \tag{4}$$

and the *distance increments*

$$|e_i|^2 = \left| \hat{y}_i - \sum_{j=i+1}^{M_T} R_{ij} s_j - R_{ii} s_i \right|^2 \tag{5}$$

where we set $d_{M_T+1} = 0$ (cf. [6]). Due to the upper triangular structure of $\mathbf{R}$, the PEDs d_i depend only on $\mathbf{s}^{(i)}$ and can, therefore, be associated with the corresponding nodes in the tree. In other words, the computation of $d(\mathbf{s})$ can be interpreted as a traversal of the tree from the root to the leaf corresponding to $\mathbf{s}$. When proceeding from a node on level $i+1$ to one of its children on level i, the detector increments the PED by the nonnegative quantity $|e_i|^2$.

Complexity Reduction with Tree Pruning. The ML solution (2), or equivalently (3), can now be found by an exhaustive tree traversal that searches for the leaf associated with the smallest $d(\mathbf{s})$. Efficient tree-search algorithms reduce the search space by pruning the tree below certain nodes, such that the path leading to the ML solution is preferably not discarded. Pruning criteria are typically based on the PED and resource constraints.

2.2. Sphere Decoding with Run-Time Constraints

The SD algorithm traverses the tree depth-first, i.e., the detector visits the children of a node before visiting its siblings. The rule for pruning the tree follows from the *sphere constraint* (SC), which limits the candidate vector symbols to those points in $\mathcal{O}^{M_T}$ for which $\mathbf{H}\mathbf{s}$ lies within a radius r around the received point $\mathbf{y}$. Since the PEDs are monotonically increasing along a path in the tree, the SC corresponds to a constraint on the PED for each level of the tree, given by $d_i < r^2$.

A considerable problem with the application of the SC is the fact that the search complexity depends critically on the a-priori choice of the radius [7]. If r is chosen too small, no solution is found, and the algorithm must be restarted with a larger radius. If it is chosen too large, many candidate vector symbols lie within the sphere, and the detection effort is high.

A technique known as *radius reduction* allows to avoid the problem of selecting a suitable radius altogether; in addition, this technique improves the pruning efficiency. The basic idea of radius reduction is to start the algorithm with $r = \infty$ and to update the radius according to $r^2 \leftarrow d(\mathbf{s})$ whenever a leaf $\mathbf{s}$ is reached. However, this procedure is efficient only if the depth-first paradigm is used in conjunction with *Schnorr-Euchner (SE) enumeration* [8], which en-

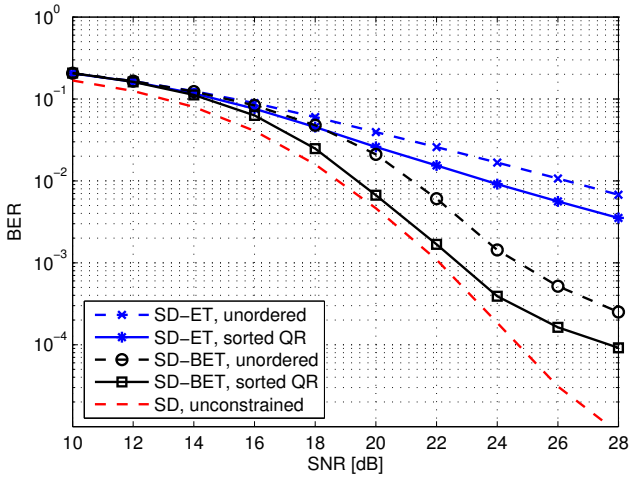

Figure 1. BER performance of SD in a fast-fading scenario with a per–vector symbol (ET) and with a per-block (BET, $N = 64$) run-time constraint of $D_{\max} = D_{\text{avg}} = 7$ visited nodes.

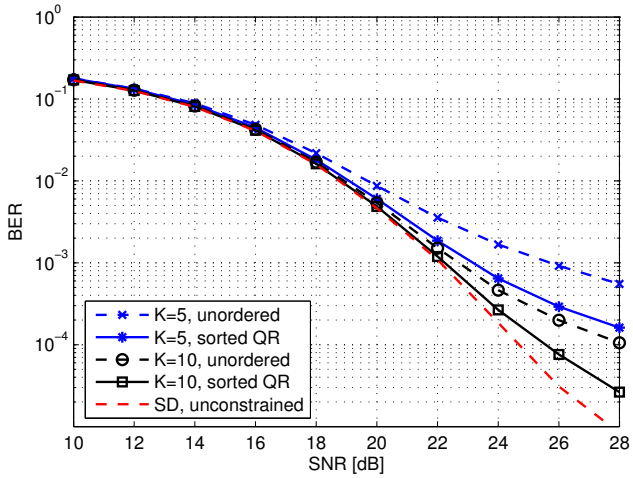

Figure 2. BER performance of K-best detection with and without ordering for $K = 5$ and $K = 10$.

sures that the detector visits the children of a node in ascending order of their PEDs.

SD with Early Termination. A VLSI implementation of SD based on depth-first tree traversal and SE enumeration can achieve ML performance with high average throughput at reasonable silicon area [6].

A significant disadvantage of SD is that the effort required to find the ML solution depends on the realization of the channel and the noise and sometimes even corresponds to an exhaustive search. In practice, the maximum detection effort must be limited by a constraint on the algorithm run time, which ultimately prevents the detector from achieving ML performance. We call such an approach *constrained SD*. A straightforward way to impose a run-time constraint is to terminate the detection process for each received vector after a maximum number of visited nodes $D_{\max}$; with the architecture described in Section 3.2, the number of visited nodes is a direct measure for the run time of the detector. The detector then returns the best solution found so far. Unfortunately, this simplistic approach results in frequent over- or under-allocation of computing resources, since the variance of the run time required to find the ML solution is large. Because early termination is likely to prevent the decoder from finding the ML solution, the bit error rate (BER) performance is severely impaired (compare the top curve in Fig. 1 against the bottom one, which corresponds to ML detection).

Block Processing. A better solution to the variable run time problem is to impose an aggregate run-time constraint of ND_{avg} visited nodes for an entire block of N vector symbols. While the average run time remains the same as for SD with early termination and $D_{\max} = D_{\text{avg}}$, block processing leads to a more efficient allocation of computing resources.

The main difficulty with the application of a per-block run-time constraint lies in the fact that the detection effort for the individual vector symbols is not known a priori. Hence, a scheduling algorithm must be used to distribute the available run time over the vector symbols in the block. The recursive *maximum-first* strategy [9] allocates a maximum run-time equivalent of $D_{\max}(n)$ to the nth vector symbol in a block according to

$$D_{\max}(n) = ND_{\text{avg}} - \sum_{i=1}^{n-1} D(i) - (N-n)M_T \qquad (6)$$

where $D(i)$ denotes the actual number of visited nodes for the ith vector symbol. The concept behind (6) is that a vector symbol is allowed to use up all of the remaining run time within the block up to a safety margin of $(N-n)M_T$ visited nodes, which allows to find at least the zero-forcing decision-feedback solution [10] for the remaining vector symbols.

The improvement in BER performance due to block processing depends on the block length N and on the channel variation within the block. The third curve from the top in Fig. 1 shows the BER performance for $N = 64$ and independent channel realizations for each vector symbol, corresponding to a fast-fading scenario. Note the significant improvement over per–vector symbol early termination.

2.3. K-Best Detection

The K-best algorithm traverses the tree breadth-first, i.e., the detector visits all siblings of a node before it proceeds to the next level. The tree pruning strategy is designed such that the detection effort is constant for each vector symbol: the detector visits only K nodes on each level of the tree and computes the PEDs of all their children. Among these children, it then selects the K ones with the smallest PEDs as the parent nodes to be visited on the next level.

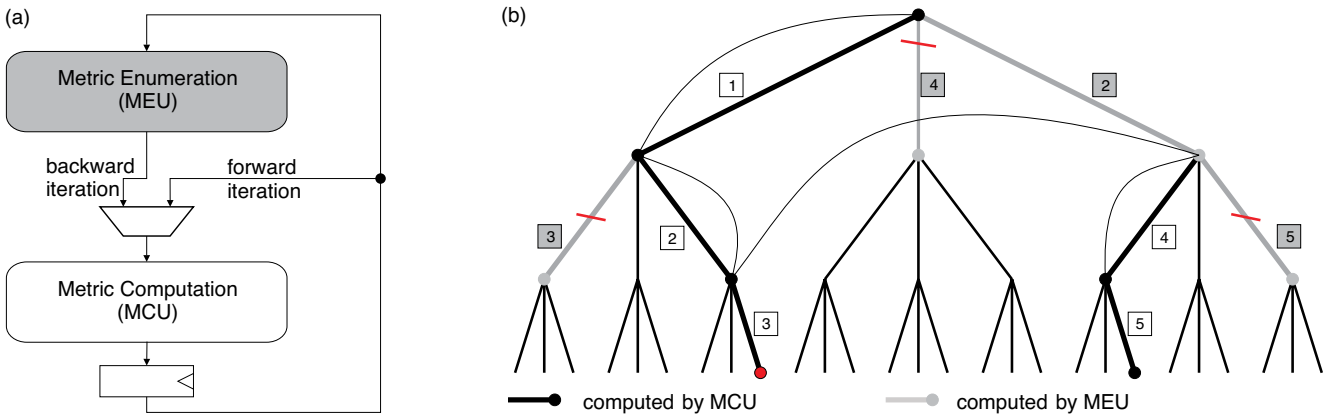

Figure 3. (a) One-node-per-cycle VLSI architecture for SD. (b) Tree-pruning example. The numbers on the branches indicate the cycles in which the corresponding branches are selected by MCU and MEU.

Clearly, the choice of the design parameter K determines the trade-off between silicon area, throughput, and BER performance. Computational complexity and memory requirements grow with increasing K, but the BER performance also improves, as illustrated in Fig. 2. The allocation of computing resources and hence the run time per vector symbol is constant, determined by the design parameter K. An important drawback of K-best detection against SD is that a flexible allocation of resources across multiple vector symbols, which results in improved performance as demonstrated above, is not immediately possible.

2.4. Ordering of Spatial Streams

Adapting the detection order of the individual spatial streams to the geometry of the matrix channel is a well-known means to improve the BER performance of SIC algorithms. For K-best detection, ordering has the same beneficial impact on BER. The effect of ordering on unconstrained SD (i.e., with ML performance) is a reduction of the average number of visited nodes. For constrained SD, ordering translates into an improved BER performance because, compared to unordered SD, chances are higher that the detector finds the ML solution within the run-time limit.

Ordering strategies for tree-search algorithms include column-norm ordering [3], the application of the V-BLAST scheme [11], and the sorted QR decomposition [12]. The latter algorithm is of particular interest for practical implementations because of the negligible additional silicon complexity and the associated significant improvement in BER performance. The corresponding gains are quantified in Figs. 1 and 2 for SD and K-best detection, respectively.

3. Implementation of Tree-Search Algorithms

In this section, we briefly describe suitable VLSI architectures for SD and K-best detection.

3.1. Modified Norm Algorithm

We shall first discuss the *modified norm algorithm* [6], which is a method to reduce the considerable complexity associated with the VLSI implementation of the arithmetic operations in (4) and (5). The square root of the PED (4), denoted by $x_i = \sqrt{d_i}$, can be interpreted as an ℓ^2-norm of a vector with elements x_{i+1} and e_i; the same interpretation holds for the computation of $|e_i|$ from $\Re\{e_i\}$ and $\Im\{e_i\}$. If we approximate the ℓ^2-norm by the ℓ^1-norm or by the ℓ^∞-norm,[1] the silicon area and the length of the critical path can be significantly reduced. A detailed discussion of the corresponding implications for BER performance can be found in [6].

3.2. VLSI Architectures for Sphere Decoding

The SD algorithm can be implemented efficiently using a one-node-per-cycle VLSI architecture [6], for which the high-level block diagram is shown in Fig. 3(a). The design is composed of two main entities, which operate in parallel to ensure that a new node of the tree is visited in each cycle. The metric computation unit (MCU) handles the forward iterations by identifying the preferred child of the current node. The metric enumeration unit (MEU) follows the MCU on its path through the tree with one cycle delay, to prepare for the moment when the forward iteration stalls because a leaf is reached or the SC is violated by all children of a node. In this case, the MEU immediately supplies a new node (together with its PED), from which the MCU can continue in the next cycle. An example of this traversal procedure is given in Fig. 3(b).

Unfortunately, the recursive nature of the SD algorithm precludes the use of pipelining to speed up the processing of a single vector symbol. However, the introduction of pipeline

1 The ℓ^1-norm of a vector with real-valued components a and b is given by $|a| + |b|$, the corresponding ℓ^∞-norm by $\max(|a|, |b|)$.

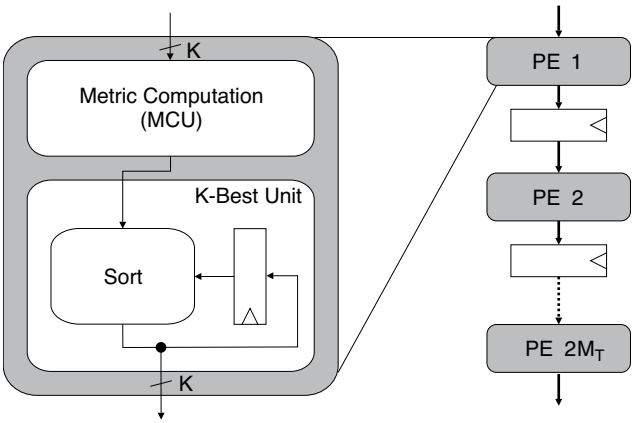

Figure 4. Parallel architecture for K-best detection.

stages allows to process subsequent vector symbols concurrently in an interleaved fashion, resulting in an overall higher throughput.

When the modified norm algorithm is used, the ℓ^∞-norm results in a better BER performance than the ℓ^1-norm [6].

3.3. VLSI Architectures for K-Best Detection

For the implementation of the K-best algorithm, in contrast to SD, the M_T-dimensional MIMO detection problem with complex-valued QAM constellations should generally be decomposed into a $2M_T$-dimensional real-valued problem [13].

A K-best architecture consisting of a pipelined linear array of processing elements (PE) has originally been proposed in [5], which was later adopted for the implementation of different variations of the same algorithm, among others in [14,13]. The basic concept behind the architecture, shown in Fig. 4, is that each of $2M_T$ PEs (i.e., each pipeline stage) considers one level of the tree. To this end, a PE comprises two main entities: The K-best MCU receives (from the preceding PE) the PEDs of K admissible nodes and computes the PEDs of all associated children. The *K-best unit* (KBU) finds the K children with the smallest PEDs from the output of the MCU and forwards this list to the next pipeline stage.

In [5, 14], a fully serial architecture is used, in which the PED computation takes one cycle per child node. The recent design in [13] takes a more parallel approach and computes the PEDs of all children of a given parent node in one cycle. Hence, the delay of each pipeline stage reduces to only K cycles, which allows for a much higher throughput compared to previous implementations.

In contrast to SD, the modified norm algorithm yields better error-rate performance if the ℓ^1-norm is employed instead of the ℓ^∞-norm [13].

4. Implementation Results and Comparison

The area-throughput trade-off of our VLSI implementations for SD (with three pipeline stages) and for K-best detection, both in a 0.25 μm technology, is depicted in Fig. 5. One gate equivalent (GE) corresponds to the area of a two-input drive-one NAND gate.

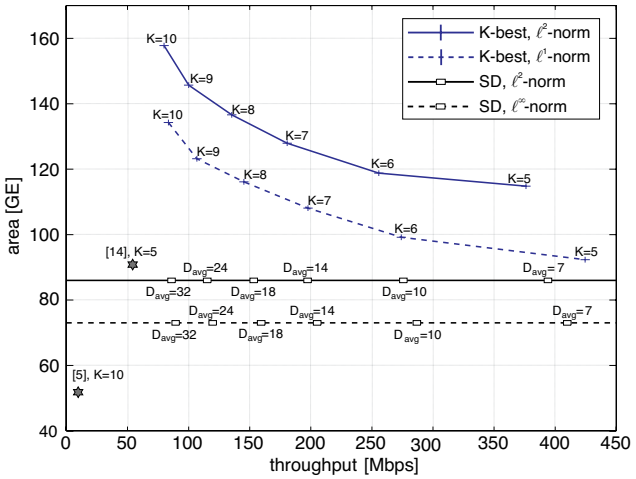

Figure 5. Area-throughput trade-off for SD and K-best detection.

4.1. Sphere Decoder Implementation

For SD, the constraint on the maximum number of visited nodes has no implications for the hardware implementation. Hence, the parameter D_{avg} can be adjusted during operation to provide the lowest possible error rate for a given throughput requirement.

Fig. 5 shows that the use of the ℓ^∞-norm instead of the ℓ^2-norm for the PED computation leads to a 15% reduction in silicon area; a reduction of the length of the critical path additionally yields a 4% higher throughput for the same choice of D_{avg}. However, the modified norm algorithm incurs an SNR penalty of about 0.15 dB ($D_{\text{avg}} = 7$) and 1.4 dB ($D_{\text{avg}} = 18$) for SNR below 24 dB. Interestingly, at higher SNR, the ℓ^∞-norm SD performs better than the ℓ^2-norm SD, because the ℓ^∞-norm results in more efficient tree pruning [6], which helps to mitigate the negative impact of the run-time constraint.

4.2. K-Best Detection Implementation

For K-best detection, silicon area as well as throughput are significantly influenced by the design parameter K, as shown in Fig. 5. The impact on area is mainly due to a larger K requiring more storage registers between the PEs of the pipelined architecture. The reasons for the considerable throughput reduction with increasing K are twofold: First, the number of parent nodes to be visited by the MCUs, and hence the number of cycles required by each pipeline stage, is directly proportional to K. Second, the length of the critical path in the KBU also increases almost linearly with K [13]. The combination of these two effects causes

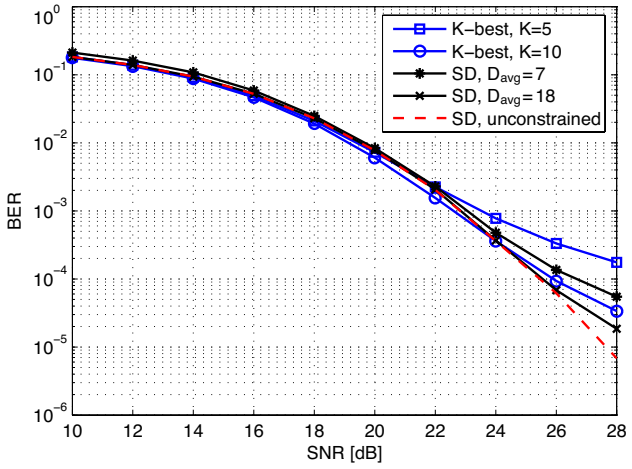

Figure 6. BER performance in a fast-fading scenario of SD with per-block ($N = 64$) run-time constraint and K-best detection.

the throughput of the parallel K-best architecture to decay as $1/K^2$. The throughput of fully serial architectures [5, 14] decays according to $1/K$, because the length of the critical path of the sorting units is almost independent of K. However, implementation results indicate that the parallel approach is still preferable for small K.

The use of the ℓ^1-norm instead of the ℓ^2-norm causes an SNR penalty of 0.4 dB for K-best detection, but reduces silicon area by approximately 16% and allows for an up to 13% higher throughput.

4.3. Comparison of SD and K-Best Detection

To compare SD and K-best detection, consider the area-throughput trade-off in Fig. 5 together with the BER performance depicted in Fig. 6, where both implementations use the modified norm algorithm. The implementations with $K = 5$ and $D_{\text{avg}} = 7$, respectively, both yield a high throughput in excess of 400 Mbps, but SD provides better BER performance at a 20% lower silicon area. If lower BER is required, a K-best detector with $K = 10$ can slightly outperform the SD with $D_{\text{avg}} = 7$, but only at a 5-fold throughput penalty. On the other hand, when the run-time limit is increased to $D_{\text{avg}} = 18$, SD performs again better than the K-best detector with $K = 10$, providing almost twice the throughput at half the silicon area of the K-best detector.

5. Conclusion

K-best detection as well as SD with early termination and block processing allow for the efficient implementation of MIMO detection with close-to-ML error-rate performance. In a fast-fading MIMO system, SD appears to be superior to K-best detection, since for a given throughput requirement the algorithm yields a lower error rate at lower silicon area. Moreover, SD allows to adjust the run time during operation to provide the best error-rate performance for a given throughput requirement, which is a particularly desirable feature in systems supporting variable bandwidths such as IEEE 802.11n or IEEE 802.16.

References

[1] G. Foschini and M. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Personal Commun.*, vol. 6, no. 3, pp. 311–335, 1998.

[2] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *SIGSAM Bull.*, vol. 15, no. 1, pp. 37–44, Feb. 1981.

[3] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Comp.*, vol. 44, no. 170, pp. 463–471, Apr. 1985.

[4] E. Viterbo and E. Biglieri, "A universal lattice decoder," in *Proc. 14th GRETSI Symp. Signal and Image Process.*, Sep. 1993, pp. 611–614.

[5] K. Wong, C. Tsui, R. S. K. Cheng, and W. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, vol. 3, 2002, pp. 273–276.

[6] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoder algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.

[7] Q. Liu and L. Yang, "A novel method for initial radius selection of sphere decoding," in *Proc. IEEE Veh. Technol. Conf. (VTC)-Fall*, vol. 2, Sep. 2004, pp. 1280–1283.

[8] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improving practical lattice basis reduction and solving subset sum problems," *Math. Program.*, vol. 66, pp. 181–191, 1994.

[9] C. Studer, "Sphere decoding with resource constraints," Master's thesis, ETH Zurich, Zurich, Switzerland, Aug. 2005.

[10] M. O. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[11] L. Beygi, A. R. Ghaderipoor, and K. Dolatyar, "A new lattice decoding for space time block codes with low complexity," in *Proc. IEEE Int. Symp. Personal Indoor Mobile Radio Commun. (PIMRC)*, vol. 1, Sep. 2002, pp. 428–430.

[12] A. Wiesel, X. Mestre, A. Page, and J. Fonollosa, "Efficient implementation of sphere demodulation," in *Proc. IEEE Workshop Signal Process. Advances Wireless Commun. (SPAWC)*, Jun. 2003, pp. 36–40.

[13] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner, "K-best MIMO detection VLSI architectures achieving up to 424 Mbps throughput," submitted to IEEE Int. Symp. Circuits Syst. (ISCAS), 2006.

[14] Z. Guo and P. Nilsson, "A 53.3 Mb/s 4×4 16-QAM MIMO decoder in 0.35μm CMOS," in *Proc. IEEE Int. Symp. Circuits and Syst. (ISCAS)*, vol. 5, May 2005, pp. 4947–4950.