

Vectorization of the Sphere Detection Algorithm

Björn Mennenga and Emil Matus and Gerhard Fettweis

Vodafone Chair Mobile Communication Systems

Technische Universität Dresden

Dresden, Germany

Email: mennenga,matus,fettweis@ifn.et.tu-dresden.de

Abstract—In this paper we present concepts for vectorization of sphere detection algorithms based on regularization of depth first tree search algorithms. Due to data dependant control flow, these tree search algorithms exhibit a highly irregular structure not allowing an efficient collaborative detection of multiple received symbols in parallel. In order to enable parallel symbol processing, a transformation of the irregular tree search algorithm is proposed resulting in a novel regular algorithm structure. Based on this, a concept for a vectorized List Sphere Detector is introduced, employing a SIMD computational model. In addition to this, limiting effects of vector processing are studied, leading to concepts which ease these effects and enable the utilization of vectorization’s benefits.

I. INTRODUCTION

Near Maximum-Likelihood (ML) detection is essential for good system performance in MIMO communications. However, its algorithmic complexity is a major problem [1], [2]. Many recent research try to solve this problem by introducing complexity reduced tree search algorithms, with promising results e.g. [1]–[5]. Though, algorithmic complexity is still limiting the the achievable throughput demanding an adaptation of complexity, performance or the transmission system to the channel characteristics. However, flexible implementations are still too inefficient to be applied [6]. Consequently, often multiple parallel detectors are needed to meet the demands of latest mobile communication systems.

As possible solution to utilize flexible high performance detectors, we propose a vectorization [7] of sphere detectors, employing the SIMD paradigm and processing independent received symbols in parallel. In addition to this, we introduce programmable application-specific processor architectures as enabling technology for the design of flexible receivers adaptable to heterogeneous MIMO transmission scenarios and channel conditions. In case of most tree search algorithms like metric-first or depth-first search, exemplarily described in Sec. II, this vectorization is frustrated by their inherent irregular structure. We solve this problem by the regularization of these algorithms, Sec. III-A. This enables the desired vectorization, a novel parallelization for these algorithms, described for the List Sphere Detector in Sec. III-B. Based on this, the effects of the vectorization, especially on the cycle count, are carried out in Sec. III-C. To ease an increasing average complexity, caused by the degree of parallelization, different solutions are finally proposed and evaluated in Sec. IV.

II. SPHERE DETECTION

A. Basics of the tree based MIMO Detection

Considering a MIMO system model equal to the one described in [3] with N_T transmit antennas and N_R receive antennas, a symbol detections can be summarized as follows:

The received symbol vector is given by $\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}$, for a channel matrix \mathbf{H} , a send vector \mathbf{x} and a noise vector \mathbf{n} of variance N_0 . The detection of the transmitted bits $c_{m,l}$, $m = 1..N_T$, $l = 1..L$, where L donates the number of bits per transmit symbol, is achieved by a

Soft-In Soft-out detector via calculation of the L-values

$$L(c_{m,l}|\mathbf{y}) = \ln \left(\frac{P(c_{m,l} = +1|\mathbf{y})}{P(c_{m,l} = -1|\mathbf{y})} \right) \approx -\frac{1}{N_0} \min_{c|c_{m,l}=+1} \{\lambda_0\} + \frac{1}{N_0} \min_{c|c_{m,l}=-1} \{\lambda_0\}, \quad (1)$$

over the metrics λ_0 of a hypothesis and a counter-hypothesis for each bit. Via QR-decomposition (QRD) of a known channel matrix $\mathbf{H} = \mathbf{Q}\mathbf{R}$, the detection can be interpreted as a tree search with the layered distance metrics

$$\lambda_i = \lambda_{i+1} + \left| y'_i - \sum_{j=i+1}^{N_T-1} r_{ij}\hat{x}_j - r_{ii}\hat{x}_i \right|^2 - \frac{N_0}{2} \sum_{j=1}^L c_{i,j} L_a(c_{i,j})$$

[1], [2], with a modified receive symbol $\mathbf{y}' = \mathbf{Q}^H \mathbf{y}$, an estimated sent symbol \hat{x}_i , a-priori knowledge $L_a(c_{i,j})$ and $i = (N_T - 1)..0$, whereas the root metric is defined to zero: $\lambda_{N_T} = 0$. The aim of the tree search is to solve (1) by analyzing only favorable nodes with their λ_i in the tree.

B. List Sphere Detection

As underlying detector a List Sphere Detector (LSD), a depth-first detector, is selected [2] and extended by complexity reduction methods like sorted QRD [4], to estimate most reliable signals first, analysis of most likely nodes first (known as Schnorr-Euchner enumeration) [8], MMSE detection with Bias reduction [5], and search radius adaptation to the envisages detection quality. For simulation and implementation the system model of [3] is assumed: 4×4 MIMO, flat fading channel, information block size of 9216 bits, gray mapping, Rate 1/2 PCCC, decoder using a BCJR with 8 internal iterations, ... Further we assume a non iterative detection, a 64 QAM transmission and an execution corresponding to the “one-node-per-cycle” principle [9]. Counter-hypotheses for the L-value calculation (1) are generated of leaf nodes examined during the tree search. Hence, possibly not all needed counter-hypotheses were found and the L-values have to be clipped. The definition of a proper clipping level is crucial for good performance [10]. The clipping level is chosen such that the average mutual information at the detector output is maximized.

The resulting detector achieves a BER-performance close to optimal¹ with only about 0.5dB loss (Fig. 1(a)). The optimal search maybe accomplished via bit-specific radiuses as described in [6]. As measurement of the detection complexity the cycle count e is chosen, since the cycle count respectively the number of extended nodes correspond to the amount of operations performed during the detection. The average detection complexity of the chosen detector is suitable small below 45 cycles, see Fig. 1(b).

¹Optimal in this sense means: A search finds the ML hypothesis and the ML counter-hypotheses for each bit.

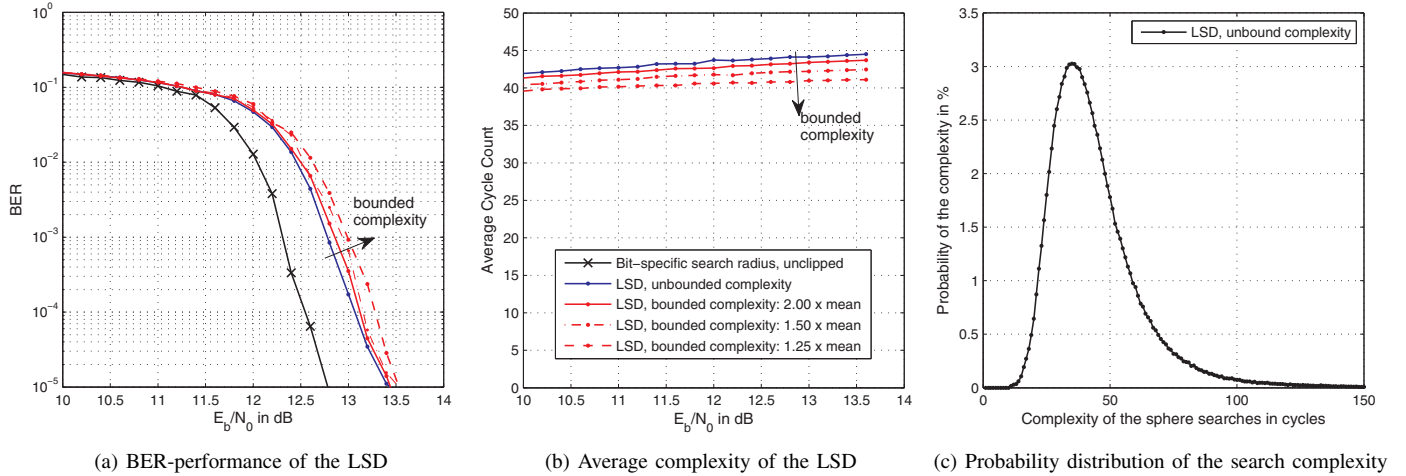


Fig. 1. Properties of the List Sphere Detector for the system model of [3] with a 64 QAM, 4×4 MIMO and without detection-decoding iterations

C. Sphere Detection Properties

The sphere search is started from the highest level of the tree $i = N_T$, the root of the tree, with an unlimited sphere. At each level i the algorithm analyzes the children nodes with their λ_i and selects one of the nodes within the search sphere, which wasn't extended so far. The selected node is extended by analyzing its children nodes in the layer $i - 1$. Whenever a leaf node is reached ($i = 0$), the radius is readjusted and the analyzed leaves are stored for the calculation of the L-values. Whenever all children nodes from a parent node within the sphere were extended, the search level is increased and the search continued. As soon the root is reached again, the search is completed. The operations of a sphere detection and the relevant notations are drafted in Fig. 2 for a BPSK and $N_T = 4$ transmit antennas. In this example the radius is adapted after finding the first leaf in ④ and finally the ML symbol is found in operation ⑨. Paths excluded from the search are illustrated with thin branches, paths examined with thick ones.

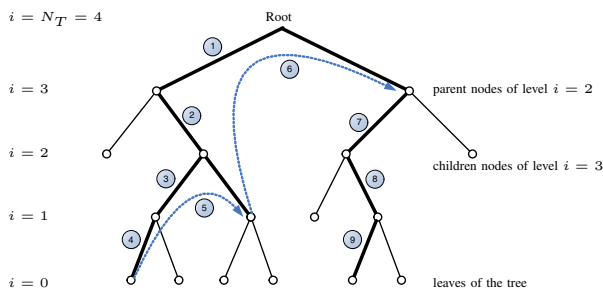


Fig. 2. Example sphere based tree search

The resulting flowchart of a LSD algorithm is shown in Fig. 3. Obvious is the problematic inherent irregular data & control flow structure of the LSD. Depending on status information (e.g. λ_i, i) different processes have to be performed. Besides this, the complexity the different processes is highly divergent reaching from negligible small (e.g. selection of a node) to numerous operations needed for the calculations and sortations. As a result, the complexity of a single loop in Fig. 3 (equivalent to one cycle in an one-node-per-cycle architecture) and the number of loops to perform for the tree search is heavily dependent on the actual data & control flow. It is

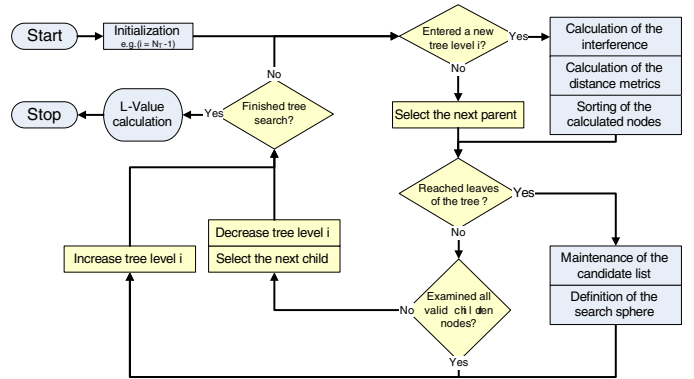


Fig. 3. Flowchart of a List Sphere Detector

impossible to predict the data & control flow and hence the operations to be performed. Figure 1(c) shows the probability density function of the detection complexity e for the described LSD, illustrating this variation in e .

III. VECTORIZATION OF THE SPHERE DETECTION

A. Regularization

Crucial for the envisaged vectorization of the sphere detection is the regularization of the flowchart in Fig. 3. In order to enable SIMD-like parallel detection of multiple symbols, all parallel detectors have to perform the same processes at once. Therefrom, the processes to be performed within a loop have to be independent of the status information and the number of loops has to be equal for all parallel data paths. To resolve the problematic irregular structure, we propose a regularization of the LSD's flowchart. As indicated in Fig. 3 by pale-yellow elements, various control operations route the processing flow within a sphere search. Separation of these processes in conjunction with combining and merging of the operations in separated modules enables an integrated selection of nodes to be analyzed in the next loop. In addition to this, we integrate all parallel operations in processing units, were a separation isn't possible. This permits the regularization of the LSD dataflow as illustrated in Fig. 4. By the prerequisite, in every analyzed tree level with $level > i$ the next parent node, if existing, is already determined, this enables for the described

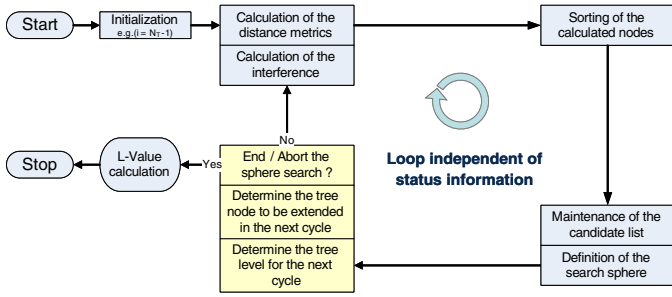


Fig. 4. Regularized data flow of a List Sphere Detector

LSD the determination of the next layer and the related node within one processing module each. This permits to extend one node in each loop. Consequently all processes needed for the node extension have to be performed in every loop. This leads to the elimination of the parallel data path and hence to the best possible utilization of the calculation and sorting entities. The stop criterium is extended to support a synchronization across the vector elements (stop finished tree searches until all searches of the vector are finished) and an early termination to ease unfavorable rising average complexity caused by the vectorization. Finally, all processing modules used for the leaf processing are executed independently of the actual level. By realizing a proper implementation of these entities only maintain one sequential comparison in the critical path, the resulting complexity enhancement is negligible small compared to the complexity of the other processing modules in the critical path.

B. Vectorized LSD (VLSD)

Assuming a p -parallel SIMD processor architecture for the proposed vectorization, each of the p vector-slices executes a separate tree search as drafted in Fig. 5. The processing modules are pipelined within a slice and all p detection slices are executed in parallel, such that each pipeline stage is processing a separate tree search. Due to the proposed regularization, a collective control of all slices in a SIMD fashion is feasible by a single VLIW instruction, enabling the aimed one-node-per-cycle architecture.

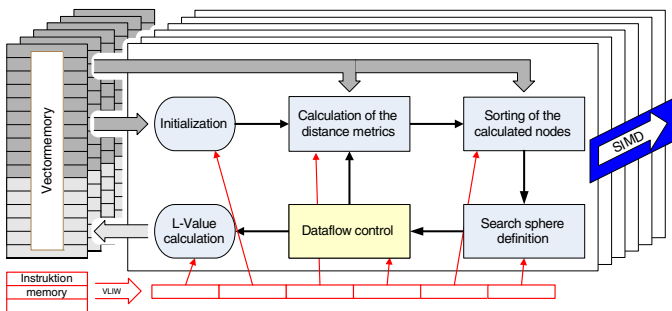


Fig. 5. Simplified structure of a SIMD List Sphere Detector

C. VLSD Properties

Besides the positive effects of the vectorization, like the reduction of control overhead, vectorized memory accesses, increased computational performance, the enabling of flexible implementations with reduced power consumption [7], the vectorization of the LSD leads to an increasing average complexity per vector element. Originating from the fluctuating number of loops needed for a single tree search, as depicted in Fig. 1(c), and the fact that a vector is completely

processed as soon as all parallel slices are finished, see section III-B, the slowest path is decisive for the amount of loops needed to process an input vector. The complexity of one vector e_{vec} is hence given by: $e_{vec} = \max(e_i), i = 1..p$. Resulting from this the average complexity \bar{e}_{vec} increases with the degree of vectorization as shown in Fig. 6(a) and dramatically affects the achievable speedup as depicted in Fig. 6(b).

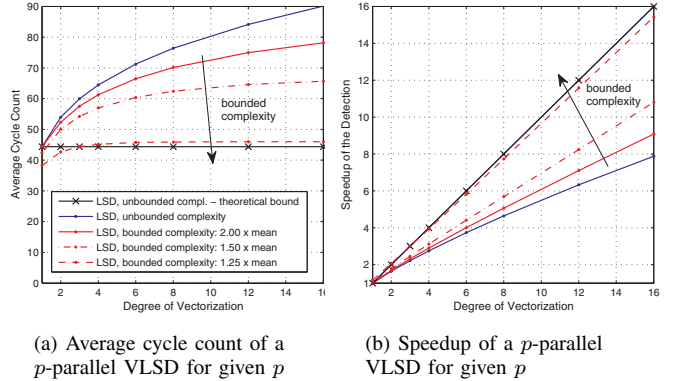


Fig. 6. Influence of the vectorization and the bounded complexity on the cycle count and the speedup of a List Sphere Detector

IV. VLSD IMPLEMENTATION ISSUES

In order to gain full advantages of the vectorization we propose two approaches to ease the rising average complexity: Bounding the maximum loop count and load balancing techniques.

A. Bounded Complexity

One possible approach to ease the increasing average cycle count is to bound the maximum number of cycles available for a single tree search: $e_i \leq e_{max}$. Limiting the complexity involves an early termination of tree searches with $e_i > e_{max}$ and causes a loss in the BER performance. Since the complexity distribution, Fig. 1(c), is quite a slim function and also aborted searched still generate valuable output due to the clipped L-value calculation, even a tight bound leads to a minor loss in the BER performance, as shown in Fig 1(a). Although, as a result the average cycle count of a single search is only reduced negligibly, the bounded complexity has a major effect on the average complexity of the vectorized sphere search \bar{e}_{vec} , see Fig. 6(a), and hence on the feasible speedup illustrated in Fig. 6(b). As consequence, by bounding the maximum cycle count almost the full speedup p is feasible, involving only little performance loss and nearly no additional hardware complexity.

B. Load Balancing

As described in section III-C, the average complexity and hence the speedup is affected by the slices becoming idle after finishing their tree searches. As second possibility to ease this effect and increase processor utilization we propose to start selectively new tree searches immediately after the previous one is finished. Therefore, we propose the inclusion of n fifo buffer elements within the vector slices for the in- and out-put data, as depicted in Fig. 7. Albeit this is easy to realize, the effects of the buffers are limited due to the probability of $\sum_{t=0}^{T_0} e_{i,t}^2$ differs from at least one $\sum_{t=0}^{T_0} e_{j \neq i,t}^3$ more than the corresponding cycle count of the last $n, 0 \leq n \leq T_0$, searches

²Sum complexity of the tree searches in slice i at time T_0 .

³Sum complexity of the tree searches in slices $j \neq i$ at time T_0 .

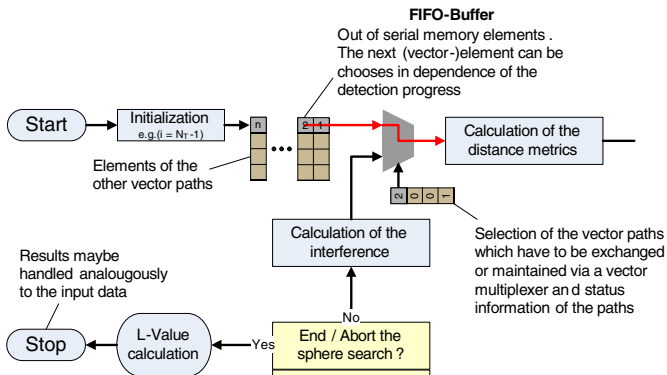
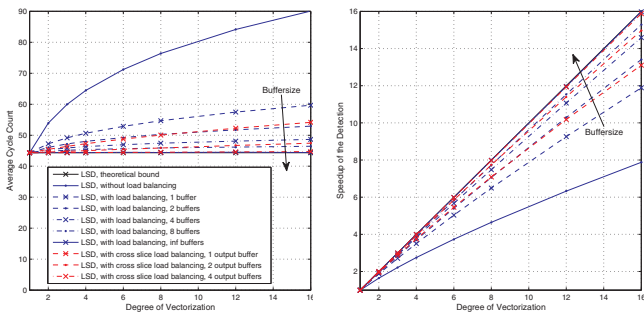


Fig. 7. Concept of a SIMD LSD with load balancing via input buffers

$\sum_{t=(T_0-n+1)}^{T_0} e_{j \neq i, t}^4$, whereas t is the time index, T_0 is the actual time and i, j are the slice numbers $1 \leq i \leq p, 1 \leq j \leq p$. The probability of an idle mode of slice i caused by slice j is given by:

$$P_{idle}(i, j) = P\left(\sum_{t=0}^{T_0} e_{i,t} < \sum_{t=0}^{T_0-n} e_{j,t}\right), \forall j = 1..p, j \neq i.$$

With an unlimited T_0 and a distribution of e given in Fig. 1(c), the probability of an idle mode is minimized for small p or big n . As result, increasing buffer sizes lead to rapidly decreasing cycle count, Fig. 8(a), but a speedup close to optimum is only feasible for very large buffers, see Fig. 8(b), leading to high hardware complexity.



(a) Average cycle count of a p -parallel VLSD for given p

(b) Speedup of a p -parallel VLSD for given p

Fig. 8. Influence of the proposed load balancing on the Vectorized List Sphere Detector (VLSD)

To improve the speedup achievable per vector slice, it is possible to distribute the incoming data across the vector slices as drafted in Fig 9(a) for incoming vector data and Fig 9(b) for an input data stream. Due to this distribution, only a minimum of buffers at the input are needed and the detector can be ideally supported with input data. Resulting from the deviations of the cycle count, a more complex structure at the output is needed. Each of the vector slices requires a full interconnect to the available buffer elements to minimize delays. The number of buffers is hence decisive for the achievable speedup. The resulting cycle counts and corresponding speedups are shown in Fig. 8(a) and Fig. 8(b) for an input stream with $n \times p$ buffer elements at the output, $n = 1, 2, 4$. The performance of the corresponding detector with vector input is similar. While the speedup is improved per available buffer element compared to

⁴Sum complexity of the last n tree searches in slices $j \neq i$.

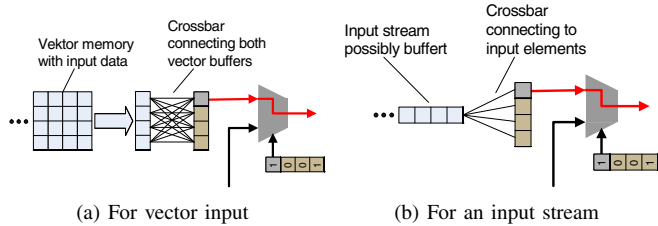


Fig. 9. Example concepts for cross slice load balancing

the load balancing of Fig. 7 and hence less buffers are needed, the complexity for the interconnection is clearly increased. The choice of a proper structure is hence application specific.

V. CONCLUSION

In this contribution we showed the proposed vectorization to be a suitable parallelization for sphere detection. Based on the recommended regularization of the sphere detection algorithm, a novel degree of parallelization - the vectorization - is feasible for depth-first tree searches. This enables efficient algorithm implementations based on the SIMD paradigm and hence more efficient and flexible high performance MIMO detectors. The keys for a powerful realization of these concepts are the proposed techniques to ease the increasing average complexity: limiting the cycle count of a tree search and load balancing. Although both approaches incur negative effects on the implementation, a combination of both enables a speedup close to the theoretical bound, at almost no performance loss and increasing hardware complexity. Since the concepts are transferable to other tree search types like breath-first or metric-govern search, this enables the usage of the benefits of the vectorization, independently of the degree of parallelization and the underlying tree search.

REFERENCES

- [1] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Transactions on Information Theory*, vol. 45, pp. 1639–1642, Jul. 1999.
- [2] B. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Transactions on Communications*, vol. 51, pp. 389–399, Mar. 2003.
- [3] E. Zimmermann and G. Fettweis, "Generalized Smart Candidate Adding for Tree Search Based MIMO Detection," in *Proceedings of the ITG/IEEE Workshop on Smart Antennas (WSA'07)*, Vienna, Austria, 2007.
- [4] D. Wubben, R. Bohnke, V. Kuhn, and K.-D. Kammeyer, "MMSE extension of V-BLAST based on sorted QR decomposition," in *Proceedings of the IEEE 58th Vehicular Technology Conference, 2003 (VTC'03-Fall)*, vol. 1, pp. 6–9, Oct. 2003, pp. 508–512.
- [5] E. Zimmermann and G. Fettweis, "Unbiased MMSE Tree Search Detection for Multiple Antenna Systems," in *Proceedings of the International Symposium on Wireless Personal Multimedia Communications (WPMC'06)*, San Diego, USA, 2006.
- [6] J. Jalden and B. Ottersten, "Parallel Implementation of a Soft Output Sphere Decoder," in *Asilomar Conference on Signals, Systems, and Computers*, 2005.
- [7] G. Fettweis, "Embedded SIMD Signal Processor Design," in *International Workshop on Systems, Architectures, Modeling, and Simulation (SAMOS'03)*, Samos, Greece, 21–23. Jul. 2003, pp. 71–76.
- [8] C. Schnorr and M. Euchner, "Lattice basis reduction: Improving practical lattice basis reduction and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–199, Aug. 1997.
- [9] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.
- [10] Y. de Jong and T. Willink, "Iterative tree search detection for MIMO wireless systems," *Communications, IEEE Transactions on*, vol. 53, no. 6, pp. 930–935, 2005.