An Optimised Parallel Tree Search for Multiuser Detection with VLSI Implementation Strategy

Geoff Knagge[1]
School of Electrical Engineering
and Computer Science
University of Newcastle
Callaghan, NSW 2308, Australia
gknagge@ee.newcastle.edu.au

Graeme Woodward[2]
Bell Labs Australia
68 Waterloo Road
North Ryde, NSW 2113, Australia
graemew@agere.com

Steven Weller, Brett Ninness
School of Electrical Engineering
and Computer Science
University of Newcastle
Callaghan, NSW 2308, Australia
{steve, brett}@ee.newcastle.edu.au

***Abstract*— Multiuser detection (MUD) strategies have the potential to significantly increase the capacity of wireless communications systems, but for these to be useful they must also be practical for implementation in very large scale integrated (VLSI) circuits. In particular, while the maximum-likelihood (ML) solution is optimal in bit error rate, it cannot be directly implemented due to its exponential computational complexity.**

Lattice decoders, such as the sphere search, exhibit near-optimal ML performance with reduced complexity, but their application is still limited by computational requirements. Here, a number of optimisations are presented, designed to reduce the computational cost of the sphere search, in the context of VLSI implementation. We also propose parallel implementation strategies for such a detector, suitable for implementation in VLSI. This is then combined with a single-pass tree search approach that can be designed to not significantly impair error-rate performance. While the design is targeted towards a MUD application, the concepts may also be applied to a multiple-input multiple-output (MIMO) system, or similar applications.

I. Problem Background

The problem of optimal multiuser detection (MUD) for Code Division Multiple Access (CDMA) systems is important in terms of significantly increasing error-rate performance and capacity. In a direct sequence CDMA (DS-CDMA) system, the kth user transmits a data bit s_k of value ± 1, which is spread by a signature sequence, $\mathbf{h}_k$ of length p. In addition, as the signal is transmitted across an air interface, it is also subject to additive white Gaussian noise (AWGN), $\mathbf{n}$. If each time interval is defined to be equal to one chip, a discrete-time model of the received signal is then represented by

$$\mathbf{y}_k = \mathbf{h}_k s_k + \mathbf{n}, \tag{1}$$

where $\mathbf{y}_k$, $\mathbf{h}_k$ and $\mathbf{n}$ are $p \times 1$ vectors.

Using this technique, K users may share a communications channel by using different signature sequences. If all users are controlled so that they all transmit synchronous symbols, then the discrete time model becomes

$$\mathbf{y} = \sum_k \mathbf{h}_k \mathbf{s}_k + \mathbf{n} = \mathbf{H}\mathbf{s} + \mathbf{n}, \tag{2}$$

[1]This work was partially supported by the Australian Research Council Linkage Project Grant LP0211210. [2]Now with Agere Systems, 11-17 Khartoum Rd, North Ryde, NSW 2113, Australia.

where $\mathbf{H} = [\mathbf{h}_1 \mathbf{h}_2 \cdots \mathbf{h}_K]$ is a $p \times K$ matrix of channel estimates, $\mathbf{s} = [s_1 s_2 \cdots s_K]$ is the unknown length K column vector, and both $\mathbf{y}$ and $\mathbf{n}$ are $p \times 1$ vectors.

The multiuser detection problem is to solve (2) for $\mathbf{s}$, with knowledge of the channel $\mathbf{H}$ and the received signal $\mathbf{y}$.

The bit error rate optimality of the maximum-likelihood (ML) multiuser detector was proven by Verdú [1], and involves finding

$$\tilde{\mathbf{s}} = \arg\min_{\mathbf{s}\in\Lambda} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2, \tag{3}$$

where Λ is the set of possible decisions over all users.

This is a combinatorial optimisation problem that is NP-hard, and hence impractical for all but the smallest of problems. In particular, with K users, each transmitting from a constellation of size 2^q, the complexity of the problem becomes $O\left(2^{qK}\right)$.

Lattice decoding, especially the sphere search variant, is regarded as a very promising candidate for practical, high performance, near-optimal ML detection algorithms. It has recently received wide attention for its potential application to space-time decoding, and MUD for multi-carrier CDMA (MC-CDMA) [2] and DS-CDMA [3] systems. An equivalent algorithm, the closest point search, is well described by [4], and the applications to Multiple Input-Multiple Output (MIMO) channels have been studied in [5].

The sphere search can be described as a variation of a tree search, making use of simplifications to greatly reduce the search space to only a few percent of the points considered in the full ML problem. However, the algorithm and the associated preprocessing is still computationally intensive, and optimisations need to be found to further reduce its complexity.

To address this problem, this paper proposes a variation in the sphere search technique, and investigates strategies that may be used to assist in the VLSI implementation of such a detector. Section II introduces the sphere search algorithm and its computational requirements. In Section III, the proposed parallel search strategy and single-pass simplification is described, and Section IV describes how this allows for a simplified architecture. Section V presents the relative complexity comparisons of this technique against standard algorithms.

Finally, the paper is concluded with a summary and evaluation of these proposals.

II. Standard Sphere Search Algorithm

The optimal solution, $\tilde{\mathbf{s}}$, in (3) may be expressed as:

$$\tilde{\mathbf{s}} \quad = \quad \arg\min_{\mathbf{s}\in\Lambda} (\mathbf{s}-\hat{\mathbf{s}})^H \mathbf{H}^H \mathbf{H} (\mathbf{s}-\hat{\mathbf{s}}). \qquad (4)$$

Here, Λ is the lattice of possible decisions over all users, $\mathbf{H}^H$ denotes the conjugate transpose of $\mathbf{H}$, and $\hat{\mathbf{s}}$ is the unconstrained ML estimate of $\mathbf{s}$, given by

$$\hat{\mathbf{s}} \quad = \quad (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H\mathbf{y}. \qquad (5)$$

The sphere search has the additional concept of a radius, r, which is a threshold that defines the maximum distance around the search centre, $\hat{\mathbf{s}}$, that will be searched. The problem becomes one of solving (4) for cases $\mathbf{s} \in \Lambda$ that satisfy

$$(\mathbf{s}-\hat{\mathbf{s}})^H \mathbf{H}^H \mathbf{H} (\mathbf{s}-\hat{\mathbf{s}}) \quad \leq \quad r^2. \qquad (6)$$

By utilising either a Cholesky or QR decomposition, an upper triangular matrix $\mathbf{U}$ can be obtained such that $\mathbf{U}^H\mathbf{U} = \mathbf{H}^H\mathbf{H}$, with the added constraint that the diagonals of $\mathbf{U}$ are non-negative, so that (6) may be rewritten as

$$\sum_{i=1}^{K} \left| u_{ii}(s_i - \hat{s}_i) + \sum_{j=i+1}^{K} u_{ij}(s_j - \hat{s}_j) \right|^2 \quad \leq \quad r^2, \qquad (7)$$

where u_{ij} is the (i,j)-th element of $\mathbf{U}$.

The upper triangular nature of $\mathbf{U}$ allows the optimisation problem to be structured as a tree search, with each user representing one level of the tree, and the branches representing a choice of one of the constellation points available for each user. Associated with each branch is a cost contribution, represented by one term of the outer summation in (7). Each leaf then represents the entire collection of decisions, and has a cost that is the sum of the cost contributions associated with each of the branches taken to reach that leaf from the root of the tree.

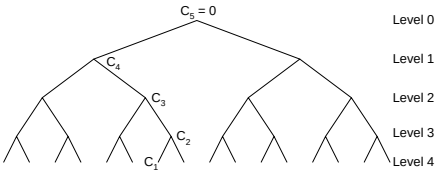

Fig. 1. Example of cost allocations for tree search decisions. Note that "Level 0" does not actually exist in the search, since the first decision is represented by "Level 1".

Fig. 1 presents an example for a $K = 4$ user problem, with binary decisions for each user. The first decision is represented by level 1, which corresponds to the last row of $\mathbf{U}$. Defining cost C_{K+1} as 0, the cost for the node at level $K + 1 - n$ of the tree is

$$C_n = C_{n+1} + \left| u_{nn}(s_n - \hat{s}_n) + \sum_{j=n+1}^{K} u_{nj}(s_n - \hat{s}_n) \right|^2. \qquad (8)$$

It is apparent from (8) that all descendants of a given node will have a cost that is not smaller than the cost of that parent node. Therefore, if a given node has a cost greater than the current radius, all of its descendants will also have a greater cost, and so the tree may be pruned at that point. It is via this pruning that the sphere search significantly reduces the search space and therefore the complexity of the detection problem.

When a leaf node is evaluated, it may be added to a "leading candidates list" of an arbitrary fixed length n, which contains the best n leaves found to date. This list is used, after the search is complete, to generate soft information about the decision for each transmitter, as described in Section IV-C. Once the leading candidates list is filled, the radius becomes equal to the cost of the highest cost leaf in that list. Further additions to the list will result in that highest cost leaf being discarded, and the radius being adjusted to match the new highest cost leaf within the list.

III. Proposed Parallel Single-Pass Search

To make the processing of larger problems more practical, a distributed processing approach may be used to consider multiple paths of the tree at one time in an attempt to generate the leading candidates list in a shorter time. A key requirement of this approach is that all parallel searchers act on the same level of the tree at any one time, providing a simple solution to memory contention issues. This means that all searchers will require the same row of the decomposed matrix, and will also be utilising the same cells of that matrix at any given time.

In the parallel scheme considered in Fig. 2, each of the two parallel searchers evaluates the cost of its two children ([A, B] and [C, D] respectively) on each step. The two most promising of these children are then chosen and assigned to the distributed entities on the next step of the search, while the others are stored for later consideration.

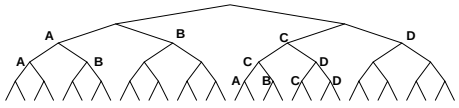

Fig. 2. On each level, the children of the current nodes (labelled A-D) are evaluated by the two parallel searchers. The best two are then selected as the current nodes for the next iteration.

Once the leaves of the tree are reached, a traditional sphere search back-tracks up the tree to evaluate the validity of the other unexplored paths. The "single-pass" approach proposed in this paper simply terminates the search as soon as the first set of leaf nodes is reached, thus entirely eliminating the need for back-tracking. While the architecture in [6] uses a similar underlying idea, the approach, implementation strategies, and analysis presented here are significantly different.

Using the parallel search strategy described, the first set of candidates produced by the searchers are very likely to be "good", even though they may not be optimal. Since the only purpose of these candidates is to produce soft information, as described in section IV-C, it is sufficient to do this as long as there are enough candidates generated. A larger number of parallel searchers will provide better soft information, but will incur an increase in complexity. However, as described in section IV, an increase in the number of searchers does not necessarily involve significant duplication of hardware.

A. Performance

To demonstrate that the single-pass approach does not significantly impact error-rate performance, a simplified model of a DS-CDMA transmission system has been used. Each user transmits one bit per symbol with a common spreading code, and each symbol is then encoded with a scrambling sequence that is pseudo-random and unique to each user.

All experiments described here are for a fully synchronous system. This is a simplification that reduces the complexity of the detection problem but, due to the properties of the scrambling sequences, was not observed to distinctly affect the bit error rate of the detectors in our simulations. For the purpose of demonstrating these concepts, a flat AWGN channel was assumed, but the technique is generally applicable to other channels.

For a system of 20 users spread by 32 chips, Fig. 3 shows that as few as 4 parallel searchers are required to obtain a result near to the optimal unconstrained full search. However, for a fully loaded system, Fig. 4 shows that many more candidates are needed in order to achieve such a result.

To confirm the trend, other experiments were conducted, including a system spread by 64 chips. In Fig. 5, when this system is half loaded comparatively few searchers are required to obtain a near optimal result, while the fully loaded case in Fig. 6 requires significantly more searchers.

A near optimal result can always be obtained, however it involves a trade-off against how much parallelism is feasible for a specific application. This will depend on the acceptable BER performance, the processing time allowed, and the limits on the complexity of the VLSI circuit.

IV. Architectural Strategies

The implementation benefits of the single-pass approach are very significant in comparison to the architecture described in [7], which implements a standard sphere search. Since the search is terminated as soon as the first leaf nodes are reached, no back-tracking is required and so no stack structure is necessary to remember branches that were not followed. In addition, there no longer needs to be any concept of a radius, nor any complete sorting of the leading candidates list. Thus, by removing the need for back-tracking, the associated overhead costs that previously limited the feasibility of implementation have been eliminated. In addition, our search strategy allows other optimisations to achieve a feasible multiuser detector.

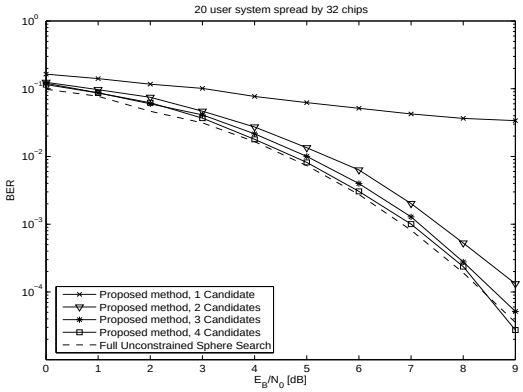

Fig. 3. Performance of single-pass approach on a 20 user system.

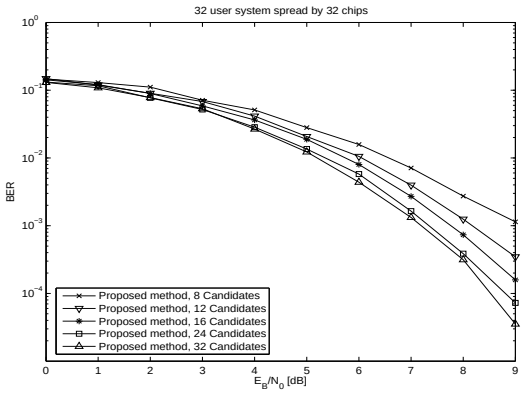

Fig. 4. Performance of single-pass approach on a 32 user system.

While a detailed architecture is beyond the scope of this paper, Fig. 7, shows that implementation simply consists of a number of parallel search engines. The content of these engines is illustrated in Fig. 8. Each engine implements a series of simple node cost calculations, which are optimised as follows.

A. Calculation of Node Costs

From (8), the cost of any node consists of the cost of its parent, plus the result of a multiplication between one row of the decomposed matrix, **U**, and the vector of the difference between the search centre and the current estimate. The number of multiplications required may seem large, but can be greatly reduced by careful examination of the algorithm.

The key features that allow a simplified implementation strategy are the binary tree characteristic of the multi-user detection problem, and by requiring that all searchers process the same level of the tree at any given time.

Since all searchers are processing nodes on the same level of the tree, then from the expansion $u_{nj}\left(s_n - \hat{s}_n\right) = u_{nj}s_n - u_{nj}\hat{s}_n$ it can be observed that only the first term is unique to each searcher. Furthermore, since the search tree is binary, the multiplication becomes a trivial matter of calculating $\pm u_{nj}$.

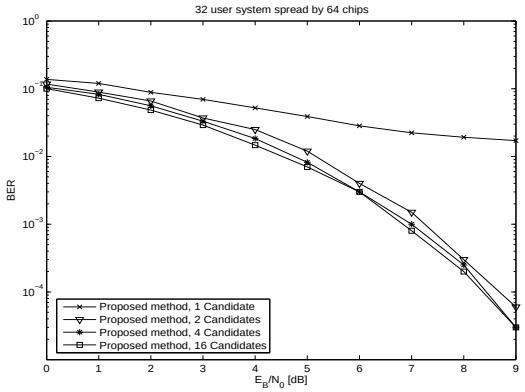

Fig. 5. Performance of single-pass approach on a 32 user system.

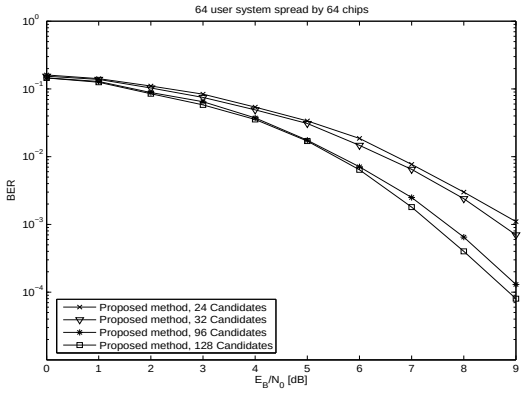

Fig. 6. Performance of single-pass approach on a 64 user system.

The latter term is common to all searchers, and so only needs to be calculated once.

By examination of (8), we can also see that the entire inner summation is common to the children of a given node. The computational cost then consists of

- A group of multiplications and additions common to all searchers;
- A group of additions for the first child of each searcher;
- A single addition for subsequent children of each searcher;
- Two squaring operations to find $|x|^2$ for each child.

Similar optimisations are also possible for a tree with quaternary decisions, where the decisions are of the form $\pm 1 \pm j$ with a precomputed scaling of $\sqrt{2}$.

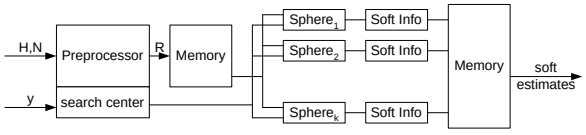

Fig. 7. A possible one-pass architecture, containing a number of parallel search engines to obtain required throughput.

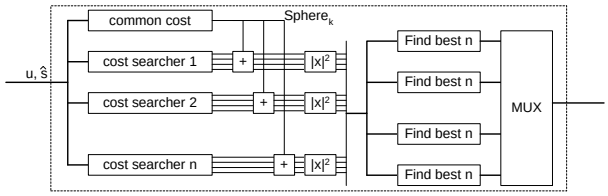

Fig. 8. Architecture of a single search engine.

B. Searcher Implementation

The single-pass approach allows for a very simple architecture of searcher hardware, allowing a large number of searchers to be easily instantiated. The additions required for node cost calculations can be performed in parallel as previously described, and it is not necessary to maintain a stack of unexplored options. The multiplications required to find the final cost of each node may either be performed in parallel, or time multiplexed, depending on the number of multipliers that may be feasibly implemented.

The only point where the number of searchers introduces possible complexity concerns is in the sorting of evaluated children on each level of the tree. With m parallel searchers, only the m best children are required. While standard full-sorting methods such as the bubble sort may be used, it is also acceptable to find the m best children in any order.

C. Generation of Soft Information

The required output of the sphere search is a soft decision for each user's symbol, with the sign of the value representing the decision and the magnitude representing the reliability. Generally, a likelihood ratio of probabilities is required:

$$\mathrm{LR}(\mathbf{y}) = \frac{P(s_k = -1|\mathbf{y})}{P(s_k = +1|\mathbf{y})}. \tag{9}$$

In a sphere list search, these probabilities can be determined directly from the leading candidates list. We note that the difference between the squared Euclidean distance in (3), and the leaf cost minimised in (4) is a constant, Δ. By defining the cost of a given leaf as $d_{\mathbf{s}}$ and applying Bayes' rule, we obtain

$$p(\mathbf{y}|\mathbf{s}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\Delta}{2\sigma^2}} e^{-\frac{d_{\mathbf{s}}^2}{2\sigma^2}}. \tag{10}$$

The probability of a "1" being transmitted by a particular user is equal to the sum of the probabilities of all of the combinations containing a "1" for that given user, and similarly for a "-1". It is not necessary to calculate the constant term $\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\Delta}{2\sigma^2}}$, since it cancels when the likelihood ratio is computed.

If the costs of the best n solutions are known, then the others may be estimated by observing that their cost is at least as high as that of the worst known point. This value can then be substituted in place of the unknown costs. Alternatively, these unknown results may be ignored completely, since their contribution is likely to be relatively small. The variance σ^2

may be also approximated without significantly affecting the performance, and in our experiments was approximated as $\frac{1}{2}$.

The soft output log-likelihood ratio associated with the kth user can then be determined by

$$\begin{aligned} \text{LLR}_k &= \ln\frac{P(s_k=1|y)}{P(s_k=-1|y)} && (11)\\ &= \ln P(s_k=1|y) - \ln P(s_k=-1|y) && (12)\end{aligned}$$

A hard decision can be determined from the soft outputs by simply recording the sign of the output, with the magnitude representing the relative confidence of the decision.

V. Complexity

When considered for VLSI implementation, detection algorithms are generally compared in terms of the number of multiplications required per detection operation. In the case of the sphere search, this involves the calculation of (8) as the search progresses down the tree.

The sum of $u_{ij}\hat{s}_j$ is constant regardless of which branches are taken, and the calculation of $u_{ij}s_j$ is trivial in the case where s_j is taken from a binary or quaternary constellation. Therefore, the number of multiplications in the searching operation consists entirely of :

- p calculations of $|x|^2$ on each level, where p is the number of parallel evaluations at each level of the tree.
- $\frac{K(K+1)}{2}$ multiplications to evaluate $u_{ij}\hat{s}_j$. Since most of these are full complex multiplications, they are considered to be twice as complex as the $|x|^2$ calculations.

If the single pass approach is used, then the total amount of multiplications is at most $Kz + K(K+1)$, where z is the product of the size of the constellation and the number of parallel searchers. This analysis ignores the preprocessing requirements, such as performing the Cholesky decomposition, which are not changed by the proposed method and are approximately $O(K^3)$ in complexity.

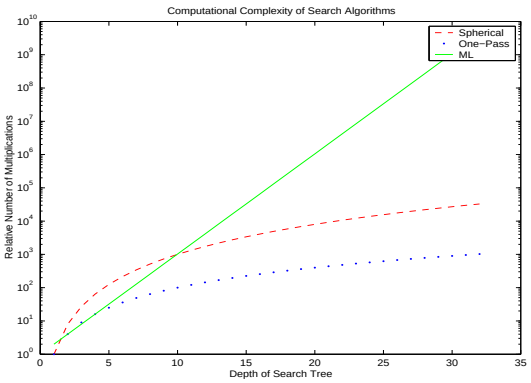

Fig. 9. Comparison of relative complexity of the various alternative detectors.

Fig. 9 shows the general trend of the computational complexity of each type of tree search technique. Due to its exhaustive search, the standard ML detector can only be used on very small problems, with K users requiring 2^K cost calculations. The sphere search algorithm can be configured to have an approximate complexity of $O(K^3)$ [5]. Hence, it can handle significantly more users, particularly if parallelism is exploited in a hardware implementation.

The single-pass approach offers not only computational savings, but the complexity of the hardware required is also significantly reduced. In particular, by removing the need for back-tracking, there is no need for stacks of unexplored options and associated control mechanisms.

VI. Application to MIMO

While presented here as a multiuser detector, this algorithm is equally suited to MIMO applications with small constellations, since the discrete time model in (2) is the same. In the MIMO case, $\mathbf{H}$ is a matrix of channel coefficients between transmit and receive antennae.

Furthermore, this algorithm may also be of use in any similar application that uses a similar discrete model, and involves binary or quaternary decisions.

VII. Conclusion

The results of our single-pass experiments have demonstrated that the number of searchers needed to obtain a good result is primarily dependent on the loading of the system, rather than the specific number of users present. The calculation sharing strategies proposed in this paper will significantly reduce the complexity of the implementation of the parallel searchers, allowing more to be implemented. The single-pass approach also removes the need to generate and store information that would otherwise be required for back-tracking in the search tree. For an implementation to remain practical, the amount of parallel searchers needs to be small enough to build in hardware, but large enough to provide near-optimal results.

Our design is specifically aimed towards the multi-user detection problem, however the techniques could be applied to larger dimensioned MIMO detectors and other similar lattice searching problems.

References

[1] S. Verdú, "Minimum probability of error for asynchronous Gaussian multiple-access channels," vol. 32, pp. 85–96, Jan. 1986.

[2] L. Brunel, "Multiuser detection techniques using maximum likelihood sphere decoding in multicarrier CDMA systems," *IEEE Trans. Wireless Commun.*, vol. 3, pp. 949–957, May 2004.

[3] L. Brunel and J. J. Boutros, "Lattice decoding for joint detection in direct-sequence CDMA systems," *IEEE Trans. Inform. Theory*, vol. 49, pp. 1030–1037, September 2003.

[4] E. Agrell, T. Eriksson, A. Vardy, and K. Zager, "Closest point search in lattices," *IEEE Transactions on Information Theory*, vol. 48, pp. 2201–2213, Aug 2002.

[5] B. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Information Theory*, vol. 51, pp. 389–399, Mar 2003.

[6] K. Wong, C. Tsui, R. Cheng, and W. Mow, "A VLSI architecture of a k-best lattice decoding algorithm for MIMO channels," in *IEEE International Symposium on Circuits and Systems*, pp. III–273 – III–276, May 2002.

[7] B. Widdup, G. Woodward, and G. Knagge, "A highly-parallel VLSI architecture for a list sphere detector," in *International Conference on Communications (ICC 2004)*, vol. 5, pp. 2720–2725, Jun 2004.