VLSI implementation of the Sphere Decoding Algorithm

Integrated Systems Laboratory, ETH-Zurich
apburg@iis.ee.ethz.ch, mawenk,martinze@ee.ethz.ch,
mwegmuel,felber,fw@iis.ee.ethz.ch

Abstract

Maximum likelihood detection is an essential part of high-performance multiple-input-multiple-output (MIMO) communication systems. While it is attractive due to its superior performance (in terms of BER) its complexity using a straight forward exhaustive search grows exponentially with the number of antennas and the order of the modulation scheme. Sphere decoding is a promising method to reduce the average decoding complexity significantly without compromising performance. This paper discusses the VLSI implementation of the sphere decoder and presents the first implementation of the algorithm that does not compromise BER performance.

1. Introduction

Multiple-input-multiple-output (MIMO) communication systems and spatial multiplexing have recently drawn significant attention as a means to achieve tremendous gains in system capacity and link reliability. In essence a MIMO system takes \( M \) parallel data streams, modulates each of them using a set of complex constellations \( \Lambda \) and arranges them in a vector \( \hat{s} \) whose components are then transmitted in parallel through \( M \) antennas. In the channel these streams interfere with each other and their superposition is distorted by an additive white Gaussian noise component \( n \). The result \( y \) is then picked up at the \( N \) receive antennas:\n
\[
y = H\hat{s} + n
\]  

At the receiver the best possible estimate \( s \) of the transmitted vector \( \hat{s} \) is to be found in order to minimize the error probability. This is achieved using the corresponding maximum likelihood (ML) criterion:

\[
\arg\min_{s} \{ ||y - Hs|| \}
\]  

As the entries of the transmitted vector \( s \) have each been chosen independently from a finite alphabet, Eq. 2 can be solved by an exhaustive search. However, with \( \Lambda \) denoting the cardinality of \( \Lambda (\Lambda = |\Lambda|) \) the complexity grows exponentially with the number of antennas and the order of the modulation scheme. Nevertheless, for numbers of antennas \(< 5\) with low-order modulation schemes such as BPSK and QPSK, exhaustive search methods have been shown to be feasible and efficient [1]. However, for more independent streams or higher-order modulation schemes other methods need to be considered.

The Sphere decoding algorithm is an efficient way to solve Eq. 2 without compromising the optimality of an exhaustive search. In its original form the algorithm was already introduced in 1981 by Finke and Pohst [2]. As opposed to an exhaustive search its average complexity order has been claimed to be only polynomial.

In this paper a new architecture for the efficient VLSI implementation of the sphere decoding algorithm for communication systems (i.e. using complex constellations) is presented and an actual implementation of a \( 4 \times 4 \) decoder for 16-QAM modulation is described. To the best of our knowledge only one ASIC implementation of the sphere algorithm has been reported so far [3]. However, this circuit compromises optimality\(^2\) and the herein presented architecture achieves a significantly higher throughput. The rest of the paper is organized as follows: In the next section the basic algorithm is briefly described together with what is currently considered the most efficient implementation strategy. The third section introduces a new approach and the corresponding architecture which is significantly more suitable for VLSI implementation. In the fourth section optimizations are discussed that can reduce the circuit complexity. The fifth section summarizes some details of the implemented circuit and gives throughput results before the paper is concluded.

2. Algorithm

The Sphere decoding algorithm starts from Eq. 2 and rewrites it as

\[
\min_{s \in \Lambda} (s - \hat{s})^T U^H U (s - \hat{s}) < r^2
\]

whereby \( U \) is an upper triangular matrix such that \( U^H U = H^H H \) and \( \hat{s} = (H^H H)^{-1} H^H y \) is the zero-forcing solution (or unconstraint ML estimate) of Eq. 1. Now, as opposed to

\(^1\) It is noted that with an appropriate choice of \( H \) and \( s \) this equation does not only apply to MIMO systems. Instead it is equally sufficient to describe many other effects such as multipath interference in frequency selective channels and interference from other users in a multiple access scenario.

\(^2\) It can not guarantee to find the ML solution.
to a full exhaustive search, only the points that lie inside a hyper-sphere with radius $r$ are considered. Due to the triangular shape of U Eq. 3 can be written in an iterative, monotonically increasing form, starting from \( i = M \):

\[
T_M = |u_{MM} (s_M - \hat{s}_M)|^2 < r^2
\]

\[
T_i = T_{i+1} + \left| u_i (s_i - \hat{s}_i) + \sum_{j=i+1}^{M} u_{ij} (s_j - \hat{s}_j) \right|^2 < r^2 \quad (4)
\]

The decoding process can now be interpreted as descending down in a tree in which each node has $A$ branches. Each of them is characterized by its partial Euclidean distance (PED) from the received point as given by Eq. 4. It only depends on the path from the root to the current node and on the symbol $s_i \in \Lambda$ which is associated with the current branch. If a PED exceeds the sphere constraint ($T_i \geq r^2$) the entire branch and all its descendents can be cut off and do not need to be considered further. This is the main reason for the complexity reduction in the sphere decoder as opposed to an exhaustive search. However, it is now obvious that the throughput is no longer constant and will depend critically on the choice of $r$.

**Proposed implementation strategies:** Recently, a number of optimizations have been proposed for the original algorithm. However, all of them consider an implementation on a general purpose processor and aim at reducing the overall number of operations, irrespective of their relative complexity and without considering the possibility of achieving high throughput through heavily parallel processing in VLSI. In summary most proposed approaches follow a scheme similar to the one outlined below:

1. Set an initial radius $r_0$ and start from the root of the tree $i = M$

2. Find a valid path to the bottom of the tree (that has not been visited) according to some ordering scheme. On the way down, eliminate as many branches as possible. If a dead-end is reached (no branch meets the sphere constraint) go one level up and try another path (back to 2). The decoder terminates when no other path down can be found.

3. When a leaf is reached, set a new radius as $r^2 = T_i$ (as proposed in [4]) and remember the constellations that lead to the leaf (the taken path down). Then go one level up and back to 2; $i = i + 1$.

The efficiency of the algorithm depends now critically on determining at each level of the tree if at least one valid path exists and which (if any) shall be used next to descend down the tree (step 2). For integer constellations, two approaches have been proposed in the literature: the Finke-Pohst and the Schnorr-Euchner enumeration. In [5] they are compared and the latter has been shown to be the most efficient. Both analytically find an initial point that fulfills the sphere constraint (SC) when they visit a node for the first time. The remaining points are examined in natural or in a zig-zag order respectively.

**Complex constellations:** In the case of complex constellations (as in communication systems) the above algorithm needs to be slightly modified. For the case of QAM constellations this is simply achieved by transforming the complex lattice into a real lattice (based on PAM constellations), with twice the number of dimensions, using the following decomposition for Eq. 1:

\[
\begin{bmatrix}
R \{ y \} \\
I \{ y \}
\end{bmatrix} =
\begin{bmatrix}
R \{ H \} & -3 \{ H \} \\
R \{ \bar{s} \} & 3 \{ s \}
\end{bmatrix}
\begin{bmatrix}
R \{ n \} \\
3 \{ n \}
\end{bmatrix} \quad (5)
\]

However, doubling the dimension also doubles the levels in the tree, thereby increasing the decoding complexity significantly. In addition to that this technique can only be directly applied to QAM constellations. A method to directly decode certain complex constellations has been suggested in [6].

3. **VLSI Architecture**

As opposed to the conventional approach, a parallel implementation strategy is suggested here. While it would require more operations on a standard DSP it is much more suitable for a VLSI implementation. It completely avoids the numerically critical operation of analytically finding an initial branch for the enumeration procedure. Instead, it examines all branches that originate from the current node in parallel and finds the smallest one. This allows to operate directly on arbitrary complex constellations without increasing the dimension of the problem (i.e. the depth of the tree). The architecture is designed to visit one node in each clock cycle. Its main advantage in terms of throughput results from the fact that no node is ever visited twice.

The initial hardware architecture is shown in Fig. 1. The Sphere-ALU concurrently computes the PEDs of all paths that originate from the current node (Eq. 4). The Down-Path unit then finds the smallest among them, and if it meets the SC and the leaf of the tree is not yet reached, it sets it as the path to be followed down in the next cycle. The remaining results, which are not needed immediately, are stored in a register file and the already taken branches are marked as invalid.

If a leaf or a dead-end is reached the decoder has to search the nodes of the current path, trying to find another path that meets the SC. Thereto, it only needs to consider those branches that have not been taken previously and have the smallest PED of the remaining branches at the respective nodes. These however can already be extracted on the way down by the Next-Path unit. They are stored in a separate buffer with only $M - 1$ entries (one for each node on the current path). From this, the next node to be visited can instantly be chosen, giving preference to the deepest node in the tree (i.e. jumping up as few levels as possible as suggested in [5]). However, an important difference to the proposed algorithm is that in this implementation no extra time (cycle) is needed to climb up the tree as no node is visited twice explicitly. Instead, the decoder
can jump immediately to a new node and can continue there.

The drawback of the above implementation is the fact that the critical path (that dictates the overall timing) is given by the Sphere-ALU and the Down-Path unit, which then again determines the next input of the Sphere-ALU. This is highlighted by the dotted red line in Fig. 1. Due to the recursion, pipelining is not possible without modifications to the algorithm. A possible solution is to blindly use any of the possible branches to descend. However, this has been shown to reduce the throughput of the decoder significantly. Instead, the precomputed zero-forcing solution \( \hat{s}_i \) at the corresponding level is used to decide which path the decoder should follow first. Since this value is available immediately, the decision is removed from the critical path. As in the original architecture the decoder still uses the smallest PEDs criterion to decide how to proceed when a dead-end or the bottom of the tree is reached. However at a leaf, an additional cycle is required, as at this point it is desirable to pick the minimum among all candidates instead of the ZF solution to inhibit later revisits to this node. In Fig. 1 block A) is replaced by block B), which simply slices \( \hat{s}_i \) to the nearest constellation point to determine the next path.

4. Sphere ALU

The Sphere-ALU occupies the largest part of the design and determines its critical path. It basically computes Eq. 4 for all possible symbols in the modulation scheme \( (s_i \in \Lambda) \). As these symbols are often convenient constants, multiplication with them can be implemented efficiently with few shift/add operations. Hereon, a much more efficient implementation of Eq. 4 can be obtained by separating all terms that are independent of \( s_i \) from those that need to be computed for all its possible values:

\[
T_i = T_{i+1} + \|u_i s_i - u_0 \hat{s}_i + \sum_{j=i+1}^{M} u_j (s_j - \hat{s}_j) \|^2
\]

(7)

While the first term needs to be computed only once, the complexity of the other terms depends on the modulation scheme. With 16-QAM \( s_i \) is chosen from the set

\[
s_i = \begin{cases} 
1+j, & -1+j, & -1-j, & 1-j \\
1+3j, & -1+3j, & -1-3j, & 1-3j \\
3+3j, & -3+3j, & -3-3j, & 3-3j
\end{cases} = \Lambda
\]

(8)

The symmetries in \( \Lambda \) lead to further simplifications by realizing that \( \|u_j\|^2 (\hat{s}_j s_i) \) can only have three different values, while \( 2\Re \{ (\hat{a} u_j) s_i \} \) has only four different solutions. These can then be combined to obtain the 16 desired PEDs. The resulting block diagram is shown in Fig. 2.

**Square root sphere algorithm** Another approach to increase the efficiency of the Sphere-ALU is to reconsider Eq. 4. By taking its square root a new SC is formulated:

\[
X_i = \frac{X_{i+1}}{\sqrt{\|m^2 + n^2\|^2 + \left(\max(|m|, |n|) - \bar{m} \min(|m|, |n|)\right)^2}} < r
\]

(9)

While in this form the complexity is significantly higher, it allows for the introduction of simple approximations:

\[
\sqrt{m^2 + n^2} \approx \max\left(\max(|m|, |n|) - \frac{3}{8} \left|\max(|m|, |n|)\right|, \frac{5}{8} \max(|m|, |n|)\right)
\]

(10)

Eq. 11 has the highest complexity but also the smallest error, while Eq. 12 leads to a significant error, but its complexity is very low. In a first step, the outer square root can be expressed as \( |X_i| \approx f_k(|X_{i+1}|, |R_i|) \), where \( f_k() \) denotes any of the approximations in Eq. 9-11. In order to obtain \( |R_i| \) from the real and imaginary parts of \( R_i \) another (inner) square root operation is needed. This can again be computed using the proposed substitutions: \( |R_i| \approx f_k(|\Re\{R_i|, |\Im\{R_i|\}) \). The block diagram of the modified ALU is shown in Fig. 3.
To evaluate the influence of the approximation error on the overall BER performance and the throughput (in terms of number of cycles), simulations were carried out. Table 1 summarizes some of the results that use Eq. 11 for the outer square root.

Table 1. Comparison of some sphere approximations

<table>
<thead>
<tr>
<th>Inner Approx.</th>
<th>Original</th>
<th>Eq. 9</th>
<th>Eq. 10</th>
<th>Eq. 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>1.2 mm²</td>
<td>1.6 mm²</td>
<td>1.4 mm²</td>
<td>1.1 mm²</td>
</tr>
<tr>
<td>Critical Path</td>
<td>14.0 ns</td>
<td>13 ns</td>
<td>12.6 ns</td>
<td>11.0 ns</td>
</tr>
<tr>
<td>Cycles @ 20dB SNR</td>
<td>11.2</td>
<td>19.0</td>
<td>19.0</td>
<td>19.7</td>
</tr>
<tr>
<td>BER</td>
<td>0.0050</td>
<td>0.0057</td>
<td>0.0056</td>
<td>0.0061</td>
</tr>
</tbody>
</table>

It can be seen that the square root sphere only slightly increases the BER. More importantly, using the simplest approximation for the outer square root leads to a significant increase in the number of cycles, while it only slightly reduces the critical path. This problem disappears, when Eq. 9 or 10 are used for the outer approximation. However, the area will be larger than the original sphere algorithm with only a small improvement of the critical path and some loss in BER.

5. Implementation Results

The sphere algorithm has been implemented and sent for fabrication for a 4 × 4 system with 16-QAM modulation using a standard 0.25μm CMOS process. It integrates the original sphere criterion from Eq. 4, as it requires a smaller area, guarantees optimum BER performance under all circumstances, and achieves the overall higher throughput. Its average performance is shown as a function of the SNR in Fig. 4 based on the maximum achievable clock frequency of 57MHz². At 20dB the algorithm requires on average 11 cycles for each symbol, which results in a net data rate of 80Mbps. The core occupies an area of 3.56 mm².

Figure 3. Proposed square root sphere ALU for a 16-QAM modulation

Figure 4. Average throughput (M = N = 4, 16-QAM)

6. Conclusion

In this paper the hardware implementation of the sphere decoding algorithm has been discussed. First a novel decoding strategy has been introduced which is particularly suitable for VLSI implementations. It guarantees very high throughput and deals efficiently with arbitrary complex constellations. Subsequently a slightly modified strategy was proposed to reduce the critical path of the circuit. To achieve an efficient implementation some critical RTL optimizations were suggested and the square root sphere algorithm was introduced. Finally an actual VLSI implementation for a 4 × 4 MIMO system using 16-QAM modulation was presented. The decoder achieves an average throughput of 80Mbps at 20dB SNR. It is to the best of our knowledge the fastest reported implementation of the sphere decoding algorithm and the highest throughput MIMO detector.