A 50 MBPS 4x4 MAXIMUM LIKELIHOOD DECODER FOR MULTIPLE-INPUT MULTIPLE-OUTPUT SYSTEMS WITH QPSK MODULATION

A. Burg †‡, N. Felber †, W. Fichtner †

† Integrated Systems Laboratory, ETH-Zurich
‡ Bell-Labs Wireless Research, Lucent Tech.

ABSTRACT

In this article we present an efficient approach for the implementation of optimum maximum likelihood decoding of QPSK modulated multiple-input-multiple-output data streams. The proposed method does not compromise optimality of the detection algorithm. Instead it uses the special properties of QPSK modulation together with algebraic transformations and architectural optimizations to achieve very low hardware complexity and high speed. To our knowledge it is the fastest and most area efficient reported VLSI implementation of a hard decision Maximum Likelihood decoder for QPSK based MIMO systems. It can be applied to a variety of narrow band and wideband systems in many different configurations, including different degrees of spatial multiplexing and receive diversity.

1. INTRODUCTION

In recent years Multiple-Input Multiple-Output (MIMO) systems have attracted considerable attention in the wireless communication community. Multiple antennas at the transmitter and at the receiver allow to directly increase the channel capacity through spatial multiplexing [1]. With this method multiple data streams are transmitted concurrently in the same frequency band from each of the \(N_T\) transmit antennas. A MIMO symbol is thereby described as a vector \(s^{(i)} = [s_0, s_1, \ldots, s_{N_T-1}]^T\) with \(s_0\) chosen from a list of complex symbols according to a modulation scheme. While in general many schemes are possible it is important to note that throughout this paper QPSK modulation is used, defining \(s_0 \in \{1, j, -j, -1\}\), where \(j = \sqrt{-1}\). This is crucial for most of the proposed algorithmic and architectural optimizations, however as MIMO systems are a means to increase throughput without the need for higher order constellations the restriction to QPSK is not a major limitation. The total number of symbols that can be taken by the transmitted symbol vector \(x \in \mathcal{S}^{(i)}\) is given as \(M = 4^{N_T}\). At the receiver the signal is picked up by \(N_R\) receive antennas. As opposed to a Single-Input-Single-Output system a flat channel is now described as an \(N_R \times N_T\) matrix \(H\) and a complex additive noise vector \(n\). With these definitions the received signal vector \(y\) can be defined through a system of equations in matrix notation:

\[
y = Hx + n. \tag{1}
\]

When no noise is present \(x\) can always be obtained from \(y\) provided that \(H\) is not singular. Under the influence of \(n\) the best possible solution for \(x\) has to be found. Thereto, a variety of methods is available which are reviewed and compared in detail in [2]. In the following list the most common approaches are summarized with their particular advantages and disadvantages:

- **Zero-Forcing** is a suboptimal method based on finding the inverse of the channel matrix. Its performance is rather poor due to noise enhancement.

- **MMSE** is also a suboptimal method similar to zero-forcing, however the noise is taken into account. It requires an accurate estimate of the amount of noise present in the system which is hard to obtain in practical systems.

- **V-BLAST** is an iterative application of zero-forcing or MMSE, effectively implementing iterative interference cancellation. It has higher complexity than the original algorithms but can provide significantly better performance.

Maximum-Likelihood (ML) is always the optimum decoding method from a mathematical point of view. A search is performed over all possible symbols and the most likely one is chosen. No noise enhancement takes place and numerical issues are virtually not present, as no matrix inversions or divisions are necessary. The disadvantage, however, is that its complexity grows exponentially with the number of transmit antennas.

In this paper the efficient VLSI implementation of a hard decision maximum-likelihood MIMO decoder is considered. Despite its at first glance very high complexity its excellent performance and stability under all channel conditions make it an appealing choice. The algorithm finds the MIMO symbol \(s^{(i)}\) that minimizes the distance between an ideal (noise free) received candidate signal \(Hs^{(i)}\) and the actual received vector \(y\), maximizing \(P \left( x = s^{(i)} \mid y \right)\):
\[
\arg\min_i \left\{ \left\| (y - Hs^i) \right\|^2 \right\}.
\]

An initial VLSI implementation of such a decoder has recently been described in [3]. The approach taken therein is based on a straightforward realization of Equation 2, taking advantage of the constellation symmetries. The implementation in this paper takes an architecturally different approach and suggests numerous optimizations based on the assumption of QPSK modulation. This results in a highly optimized circuit that achieves full ML performance at a very low complexity.

The rest of the paper is organized as follows: In the next section essential algorithmic transformations are discussed that aim at reducing the hardware complexity and determine the high-level architecture. Subsequently the subblocks are considered on a more detailed level and additional optimizations are suggested for each block together with detailed register transfer level implementations. After a discussion of the implementation results for different design parameters the paper is concluded.

2. HIGH-LEVEL ARCHITECTURE

2.1. Algorithmic Transformations

Rewriting Equation 2 more explicitly yields three mathematically fully equivalent expressions where each immediately suggests a slightly different hardware architecture:

\[
\arg\min_i \left\{ \left\| (y - Hs^i) \right\|^2 \right\}^{H} \quad (3)
\]

\[
\arg\min_i \left\{ \left\| Hs^i \right\|^2 - \Re \left\{ y^H (Hs^i) \right\} \right\}^{H} \quad (4)
\]

\[
\arg\min_i \left\{ \left\| Hs^i \right\|^2 - \Re \left\{ y^H H (s^i) \right\} \right\}^{H} \quad (5)
\]

An intuitive understanding of them can be given as follows: The first equation finds the vectors between the received symbol and all possible candidates and computes their norm. The second expression essentially projects the received symbol vector onto all candidate vectors and subtracts the result from the corresponding bias. The third alternative is very similar to the second, however it first projects \(y\) onto the columns of \(H\) before its projects all possible transmitted symbols onto the result. To compare the efficiency of the three approaches all operations are first divided into channel-rate and symbol-rate computations. This is an important step, as the update rate (related to the coherence time of the channel \(H\)) is in general significantly lower than the symbol-rate (depending on the Doppler speed typically every 0.1 – 10 ms). All partial expressions that include \(H\) and are independent of \(y\) fall into the first category and can therefore be precomputed with significantly relaxed timing constraints. Consequently only the symbol rate operations are considered relevant to the complexity analysis. Four types of resources are considered as criteria: \(adders, multipliers, QPSK-multipliers and memory\) to store precomputed values. A full complex multiplication is assumed to be composed of 5 adders and 3 real-valued multipliers. A \(QPSK\)-multiplication is a simplified complex multiplication which can be built entirely using two \(4 \to 1\) multiplexers and 2 \(NEG\) circuits. Its complexity is comparable to two adders and is therefore very low. Table 1 summarizes the results with the number of transmit and receive antennas as parameters. Note that the numbers represent only the straightforward implementation and do not include any further optimizations that are possible along the line. However they provide a good basis for initial architectural decisions.

<table>
<thead>
<tr>
<th>Symbol rate operations</th>
<th>ADD</th>
<th>MULT</th>
<th>QPSK-MULT</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eq. 3 (M(4N_x - 1))</td>
<td>2MN_x</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Eq. 4 (2MN_x)</td>
<td>2MN_x</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Eq. 5 (N_y(7N_x + M - 2))</td>
<td>3N_yN_x</td>
<td>(MN_y)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Channel rate computations</th>
<th>ADD</th>
<th>MULT</th>
<th>QPSK-MULT</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eq. 3 (2MN_y(N_y - 1))</td>
<td>-</td>
<td>(MN_yN_x)</td>
<td>2MN_y</td>
<td>-</td>
</tr>
<tr>
<td>Eq. 4 (M(2N_x - 1))</td>
<td>2MN_y</td>
<td>(MN_yN_x)</td>
<td>(M(2N_y + 1))</td>
<td>-</td>
</tr>
<tr>
<td>Eq. 5 (M(2N_xN_y + N_z - 1))</td>
<td>2MN_y</td>
<td>(MN_yN_x)</td>
<td>(MN_zN_y)</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1. Complexity comparison of Eq.3-5

It is obvious from Table 1 that Equation 5 uses by far the lowest number of full multiplications. At the same time the number of \(adders\) is only marginally increased, even if the additional \(QPSK\)-multipliers are considered. This is essentially achieved by performing the necessary \(M\) projections with the candidate vectors \(s^i\) which are only QPSK constellations. This approach is therefore taken as a starting point for the hardware architecture and the subsequent optimizations that are proposed in the following.

2.2. Hardware Architecture

The overall architecture of the proposed ML Decoder implementation is depicted in Figure 1. Besides the input buffers (not included in the final implementation results) for the channel coefficients and the soft-symbols it contains a block for the preprocessing of the candidate constellation norms \(Q^{(i)} = \left\| Hs^i \right\|^2\) (Channel-Preprocessor), for the projection of the received vector onto the columns of the channel matrix, yielding \(T = \left\{ y^H H \right\}\) (Symbol-Preprocessor) and for the ML decision (ML-Detector). The interfaces between the blocks are realized using double buffered memories to maximize the throughput. It is noted that the double buffering is not included in the initial estimates that were presented in Table 1 as it is not necessary for the implementation, but chosen for performance reasons and for easier system integration. In the following

\(^1\)Note that the \(\arg\min\) operator allows the exclusion of some terms that are independent of \(i\)

\(^2\)Note that this is reasonable for narrow-band and CDMA wideband systems but does in general not hold for OFDM systems, where a large number of independent channels exist.
the architectural details and algorithmic optimizations of each block are described.

Figure 1. ML Decoder Architecture

3. DETAILED OPTIMIZATION AND RTL-IMPLEMENTATION

3.1. Channel Preprocessor

The channel preprocessor essentially computes

\[ Q^{(i)} = \| H s^{(i)} \|^2 = s^{(i) H} \left( H^H H \right) s^{(i)} \]  

(6)

The computation of the matrix L involves a single complex matrix multiplications that is carried out only once for all candidate symbols together. In addition, note that L is Hermitian (L_{m,n} = L_{n,m}^H) with dimension NT \times N_T. Consequently only \( \frac{1}{2} N_T (N_T + 1) \) elements need to be computed, reducing the complexity to \( \frac{1}{2} N_T N_T (N_T - 1) \) complex multiplications for the off diagonal elements of L and \( 2 N_T N_T \) real multiplications for its diagonal elements. The restriction to QPSK constellations, where all symbols are located on a circle (i.e. are of equal power) allows to further optimize the above expression by realizing that the elements on the diagonal of L contribute equally to all \( Q^{(i)} \). They can therefore be ignored in the minimization process in Equation 5.

Efficiently computing \( Q^{(i)} \) for all possible MIMO constellations \( s^{(i)} \) is achieved using an iterative approach. Its pseudo code description is given in Algorithm 1.

Furthermore taking advantage of the rotational symmetry of the QPSK constellations, \( s \) can be written as \( s = s_0 \left[ 1, s_{1 \leftarrow 0}, \ldots, s_{N_T-1 \leftarrow 0} \right]^T \) and \( \tilde{Q}^{(i)} \) is computed as a function of \( \tilde{s} = \left[ 1, s_{1 \leftarrow 0}, \ldots, s_{N_T-1 \leftarrow 0} \right]^T \) according to the above algorithm by simply limiting the outer for-loop (a) in the first iteration to a single round, considering only \( s_0 \) with \( h_0 = 0 \). The value of \( Q^{(i)} \) is obtained from \( \tilde{Q}^{(i)} \) by setting \( t = \sum_{\nu=0}^{N_T-1} 4 \left( s_0 \right)^\nu \left( s_0 \right)^{-1 - \nu} \). The corresponding register transfer level (RTL) architecture is depicted in Figure 2. A single QPSK multiplier and an adder are sufficient to compute all \( Q^{(i)} \) and therefore also all \( \tilde{Q}^{(i)} \) in only \( \sum_{\nu=0}^{N_T-1} 4 \nu (N_T - k) \) cycles.

3.2. Symbol Preprocessor

The symbol preprocessor computes \( T = y^{H} H \) as a simple matrix times vector multiplication. In general, \( N_T N_T \) multiplications are required. Its implementation is straight forward and does not offer too much room for optimizations except for the usual parallelization or iterative decomposition to trade its delay for silicon area. It is adapted to match the throughput of the subsequent ML-Detector stage.

3.3. ML-Detector

The ML-Detection stage finally computes the complete argument of the min-function in Equation 5 and finds the minimum among all candidate symbols (i.e. the candidate symbol with the highest probability). The remaining operations can be subdivided into the computation of \( T^{(i)} = T s^{(i)} \) and finding the minimum distance among all \( Q^{(i)} = \Re \{ T^{(i)} \} \).

3.3.1. Computing \( T^{(i)} \)

Similar to the computation of \( Q^{(i)} \) it is again possible to take advantage of the symmetry of the QPSK modulation, limiting the set of candidate MIMO constellations to the previously defined subset \( s^{(i)} \). The resulting expression for \( T^{(i)} \) can be evaluated with different degrees of parallelism trading decoding speed (measured solely in number of cycles) for circuit complexity. The fully decomposed solution computes the vector product in a recursive way (sim-

Algorithm 1 Iterative computation of \( Q^{(i)} \)

1. Define the iterative function \( \text{QIteration}(k) \) as follows:

   \[
   \text{QIteration}(k) = \left\{ \begin{array}{ll}
   a: & t = [k \mod N_T - 1] \Rightarrow \text{QIteration}(k+1) = \text{QIteration}(k+1) \\
   b: & t = 0 \Rightarrow \text{QIteration}(k+1) = \text{QIteration}(k+1) \\
   c: & t = N_T - 1 \Rightarrow \text{QIteration}(k+1) = \text{QIteration}(k+1)
   \end{array} \right. 
   \]

2. Initialize \( Q^{(i)}_{\text{min}} = 0 \), \( k = -1 \ldots N_T - 1 \)

3. Define \( s_0 = \exp \left( j \frac{2\pi n}{4} \right) \)

4. Call \( \text{QIteration}(0) \)

5. The index \( i \) for the MIMO symbol \( s = \left[ s_0, s_1, \ldots, s_{N_T-1} \right]^T \) is given by

   \[ i = \sum_{\nu=0}^{N_T-1} 4 \nu (N_T - k) \]
3.3.2. Distance computation and ML decision

The final step is to find the complete set of distances \( d^i_s = \arg \min_j \left( \min_{s_{0,j}} \left\{ \bar{D}^i_j - \Re \left\{ s_{0,j} T^i_j \right\} \right\} \right) \) for all \( i \) from the subsets \( \bar{D}^i_j \) and \( T^i_j \) and to determine the minimum among them. Mathematically this operation is described as:

\[
\arg \min_j \left( \arg \min_{s_{0,j}} \left\{ \bar{D}^i_j - \Re \left\{ s_{0,j} T^i_j \right\} \right\} \right)
\]

As \( s_{0,j} \in \{1, j, -1, -j\} \) the inner part of this expression can be further decomposed into:

\[
\min \left\{ \bar{D}^i_j - \Re \left\{ s_{0,j} T^i_j \right\}, \bar{D}^i_j - \Im \left\{ s_{0,j} T^i_j \right\} \right\}
\]

(7)

Herewith \( s_{0,j} \) is fully described by the outcome of Equation 8 and by the sign of either the real or the imaginary part of the respective \( T^i_j \). The corresponding RTL-implementation is shown in Figure 4.

4. IMPLEMENTATION RESULTS

The proposed architecture was synthesized for a \( 2 \times 2 \) and a \( 4 \times 4 \) antenna configuration with different tree depths in the ML-Detector stage. The respective number of transmit/receive antenna pairs is the maximum that can be supported by the implemented hardware. However, runtime reconfiguration to a lower degree of spatial multiplexing (less transmit antennas) is always possible. Additional antennas at the terminal can still be used to gain diversity, only the number of bits per MIMO symbol is reduced. The implementation results are shown in Table 2 in terms of throughput, silicon area, and gate equivalents\(^3\) (GE) to ease comparison to other technologies. Timing was optimized for 100MHz and was met easily in all cases for the targeted 0.25\(\mu\)m process. All storage elements with the exception of the input buffers were realized as latch-based register files and are included in the gate count. In the \( 4 \times 4 \) implementation they account for 50\% of the total area.

<table>
<thead>
<tr>
<th>( N_T \times N_R )</th>
<th>Depth ( D )</th>
<th>Throughput (^4)</th>
<th>Area</th>
<th>GE</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 2 \times 2 )</td>
<td>0</td>
<td>100 Mbps</td>
<td>0.32mm(^2)</td>
<td>13K</td>
</tr>
<tr>
<td>( 4 \times 4 )</td>
<td>0</td>
<td>12.5 Mbps</td>
<td>0.94mm(^2)</td>
<td>40K</td>
</tr>
<tr>
<td>( 4 \times 4 )</td>
<td>1</td>
<td>50 Mbps</td>
<td>1mm(^2)</td>
<td>42K</td>
</tr>
</tbody>
</table>

Table 2. Implementation, 100MHz clock, 0.25\(\mu\)m process

5. CONCLUSION

In this paper an architecture has been presented for the VLSI implementation of maximum likelihood decoding for multiple-input multiple-output communication systems. Very high throughput is achieved at a comparatively low complexity by exploiting the special properties of the frequently used QPSK modulation. The implemented algorithm achieves optimum maximum likelihood performance. It allows various tradeoffs between throughput and silicon area. In addition to the hardware efficiency the proposed algorithm was also proven to yield a very efficient software implementation on a DSP.

6. REFERENCES


\(^3\)Total gate area divided by the area of a drive-1 NAND gate

Figure 3. Computing \( \bar{T}^i \)

Figure 4. Partial minimization function for \( \bar{T}^i \)

Figure a) shows the tree like structure with depth \( D \) and \( T \). The number of cycles per MIMO symbol decreases to \( 4^{N_T-D-1} \).

Figure b) shows the partial minimization function for \( \bar{T}^i \).

The final result is obtained from taking the minimum across all \( i \) and appropriately combining them with the respective outcomes of \( s_{0,j} \). The degree of parallelism for this process is hereby determined by the tree depth of the \( \bar{T}^i \) computation.