Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 351962, 10 pages
doi:10.1155/2008/351962

Research Article

Enabling VLSI Processing Blocks for MIMO-OFDM Communications

Barbara Cerato,[1] Guido Masera,[2] and Emanuele Viterbo[1]

[1] *Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), Università degli Studi della Calabria, via P. Bucci, 87036 Rende (CS), Italy*
[2] *Dipartimento di Elettronica, Politecnico di Torino, C.so degli Abruzzi 24, 10129 Torino, Italy*

Correspondence should be addressed to Barbara Cerato, bcerato@deis.unical.it

Received 30 April 2007; Revised 3 December 2007; Accepted 17 January 2008

Recommended by Jean-Baptiste Begueret

Multi-input multi-output (MIMO) systems combined with orthogonal frequency-division multiplexing (OFDM) gained a wide popularity in wireless applications due to the potential of providing increased channel capacity and robustness against multipath fading channels. However these advantages come at the cost of a very high processing complexity and the efficient implementation of MIMO-OFDM receivers is today a major research topic. In this paper, efficient architectures are proposed for the hardware implementation of the main building blocks of a MIMO-OFDM receiver. A sphere decoder architecture flexible to different modulation without any loss in BER performance is presented while the proposed matrix factorization implementation allows to achieve the highest throughput specified in the IEEE 802.11n standard. Finally a novel E_8 sphere decoder approach is presented, which allows for the realization of new golden space time trellis coded modulation (GST-TCM) scheme. Implementation cost and offered throughput are provided for the proposed architectures synthesized on a 0.13 μm CMOS standard cell technology or on advanced FPGA devices.

1. INTRODUCTION

MIMO-OFDM (Multi-input multi-output—orthogonal frequency-division multiplexing) is a very promising communication technique that enables to establish very high throughput and reliable wireless links. In order to achieve this goal, space-time (ST) codes are used, since they can conjugate both transmission rate and reliability enhancement of the communication system. ST codes have been considered for some recently proposed standards such as IEEE 802.11n WLAN and 802.16e WMAN.

However, the computational complexity of MIMO-OFDM receivers is much higher than in the single-input single-output (SISO) OFDM approach; as a consequence the potentials offered by MIMO-OFDM are still far from being fully exploited in actual implementations.

Figure 1 depicts the structure of a 2×2 transmit-receive antenna MIMO-OFDM communication scheme. At the receiving side, after the RF/Analog front-end, multiple OFDM demodulation stages, implemented as FFT processors (one per antenna) are allocated, followed by the MIMO signal detector. The adoption of a full-rate and full-diversity ST code demands specific demapping and decoding capabilities, which are covered in Figure 1 by the "ST-code decoder and demapper" block. Finally, a trellis coded modulation (TCM) channel decoder implements forward error correction.

The MIMO channel is modeled by its impulse response between each transmit-receive antenna pair. Assuming h_{ij} represents the time-varying channel fading coefficient between the jth transmit antenna and the ith receive antenna, the MIMO channel with M_t transmit and M_r receive antennas is described through a $M_r \times M_t$ matrix $\mathcal{H}$, where $h_{ij} \sim \mathcal{N}_c(0,1)$. Transmitted space-time codewords $\mathbf{X}$ are $M_t \times L$ matrices, where L is the number of channel uses required by the ST code. Assuming the "block fading" channel model, each transmitted $\mathbf{X}$ will be affected by an independently varying channel matrix $\mathcal{H}$. Then, the $M_r \times L$ received matrix is

$$\mathbf{Y} = \mathcal{H}\mathbf{X} + \mathbf{Z}, \tag{1}$$

where $\mathbf{Z}$ is the additive white Gaussian noise matrix with entries $\sim \mathcal{N}_c(0, N_0)$.

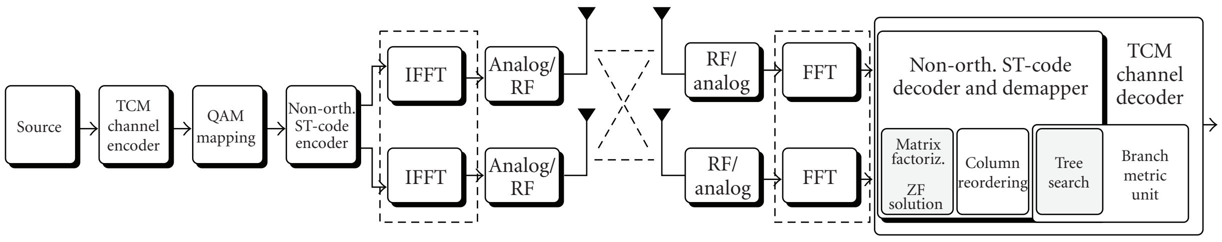


FIGURE 1: ST-Code MIMO System.

When data symbols belong to a Q^2-QAM modulation, it is convenient to represent the codewords $\mathbf{X}$ in vectorized form,where real and imaginary components of the Q^2-QAM are separated in two Q-PAM modulations, resulting in real component codewords $\mathbf{x}$. Consequently, the channel matrix $\mathcal{H}$ is rearranged in a real-valued matrix $\mathbf{H}$ and $\mathbf{Y}$ is replaced with the real-valued $\mathbf{y}$. For a linear ST block code, $\mathbf{x}$ can be obtained as $\mathbf{x} = \mathbf{Bs}$, where $\mathbf{B}$ is the ST-block code generator matrix and $\mathbf{s}$ is the vectorized data vector with entries in Q-PAM. Note that $(Q\text{-PAM})^N$ with $N = 2M_r \times L$ is a hypercubic-shaped constellation carved from a multidimensional integer grid $\mathbb{Z}^N$.

Provided that $\mathbf{H}$ is perfectly known at the receiver, the optimal detector, able to minimize the codeword error rate in a MIMO channel, is the maximum likelihood (ML) detector, which solves the problem

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s}\in(Q\text{-PAM})^N} ||\mathbf{y} - \mathbf{Ms}||^2, \tag{2}$$

where $\mathbf{M} = \mathbf{HB}$ and $N = 2 \times M_r \times L$. The cardinality of the search space, Q^N, depends on the number of receive antennas, the chosen modulation scheme, and the number of channel uses, while the factor 2 comes from the decomposition in real and imaginary components.

Hereinafter, in order to consider a currently practical situation, we will consider a two transmit and two receive antennas system, with a two-channel use ST block code ($M_t = M_r = L = 2$). An example of such a code is the golden code proposed in [1–3] and adopted by the IEEE 802.16e WMAN standard. We then have $N = 8$, $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{s}$ are 8×1 real vectors and $\mathbf{H}$ is a 8×8 real-valued matrix. Thus, when using 16-QAM symbols, the direct computation of (2) results in the evaluation of $4^8 = 65,536$ possible solutions.

Due to the high complexity of the exhaustive search, more efficient methods were proposed. Most of these approaches rely on the rearrangement of (2). In particular, a linear transformation such as QR or Cholesky decomposition allows to rewrite $\mathbf{M}$ as the product of two matrices, one of which is upper triangular [4]. Imposing $\mathbf{M} = \mathbf{QR}$, (2) can be rewritten as

$$\begin{aligned}\arg \min_{\mathbf{s}\in(Q\text{-PAM})^N} ||\mathbf{y} - \mathbf{QRs}||^2 &= \arg \min_{\mathbf{s}\in(Q\text{-PAM})^N} ||\mathbf{Q}^T\mathbf{y} - \mathbf{Rs}||^2 \\ &= \arg \min_{\mathbf{s}\in(Q\text{-PAM})^N} ||\tilde{\mathbf{y}} - \mathbf{Rs}||^2,\end{aligned} \tag{3}$$

where we have exploited the orthogonality of $\mathbf{Q}$ and $\tilde{\mathbf{y}} = \mathbf{Q}^T\mathbf{y}$.

One of the most interesting consequences of this interpretation is that the exploration of the constellation lattice can be thought as a tree traversal. This search tree has N levels and each node in a level has exactly Q sons, representing the points in one dimension of the Q-PAM's. This traversal can be done with polynomial complexity adopting the so-called *sphere decoder* (SD), [5].

Recently proposed concatenated ST coding schemes [6] offer a further reliability enhancement by adopting a combined forward error correction approach based on a high rate bandwidth-efficient trellis coded modulation (TCM) scheme. This Golden ST TCM (GST-TCM) scheme for 2×2 MIMO provides a reasonable ML decoding complexity solution by using Viterbi algorithm and a branch metric computer based on several parallel sphere decoders. A modified sphere decoder is required to support this kind of concatenated scheme, which is an unexplored subject of investigation, from the implementation point of view.

This paper deals with the implementation issues of main processing tasks that enable the development of MIMO receivers. A MIMO detector is organized in two key processing tasks, *matrix factorization* and *sphere decoding* (or *tree traversal*): we then propose efficient architectures for these two key functions. The latter function is the core function of a high performance MIMO detector and its hardware implementation tends to be critical in terms of both throughput and complexity, especially in high data rate systems.

The matrix factorization task operates on the lattice generator matrix $\mathbf{M}$. Since the code generator matrix is constant, the processing must be performed at the channel estimation update frequency, which can change significantly according to the scenario and is generally one or two orders of magnitude lower than the signaling rate. However, in a MIMO-OFDM scheme space, time decoding has to be carried out independently on each subcarrier, determining a dramatic growth of the throughput demand even for matrix factorization.

In Section 2, the sphere decoding algorithm is briefly overviewed, while Section 3 deals with the hardware design of three key building blocks: a sphere decoder, a matrix factorization architecture, and an enhanced sphere decoder for GST-TCM. Finally, Section 4 points out the implementation results achieved for the proposed architectures.

2. THE SPHERE DECODING ALGORITHM

Sphere decoding algorithms are a family of algorithms originally proposed to search the closest point to a given one in a lattice. Their use in wireless communications was suggested for the first time in [5], where the lattice structure of multidimensional constellation is exploited to find the closest point to the received vector.

When solving the minimization problem (2), sphere decoding algorithms achieve a polynomial average complexity by exploring only a subset of the solution space [4].

In particular, a hypersphere is constructed around the received vector $\mathbf{y}$ and only points inside it are taken into account. This constraint can be expressed as

$$||\mathbf{y} - \mathbf{M}\mathbf{s}||^2 \leq C_0, \tag{4}$$

where C_0 is the square radius of the hypersphere [5, 7, 8].

The upper triangular structure of the matrix $\mathbf{R}$ in (3) enables every component to be separately considered for the computation of the distance between the two points. The distance $d^2(\mathbf{s}) = \|\widetilde{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2$ can also be computed recursively as follows. We consider the partial metrics

$$T_l(\mathbf{s}^{(l)}) = \begin{cases} 0 \quad \text{if } l = N+1 \\ T_{l+1}(\mathbf{s}^{(l+1)}) + \left|\widetilde{y}_l - \sum_{j=l}^{n} R_{lj}s_j\right|^2 \\ \quad = T_{l+1}(\mathbf{s}^{(l+1)}) + \left|\widetilde{y}_l - \sum_{j=l+1}^{N} R_{lj}s_j - R_{ll}s_l\right|^2 \\ \quad = T_{l+1}(\mathbf{s}^{(l+1)}) + \left|\psi_{l+1} - R_{ll}s_l\right|^2 \\ \quad \text{if } l = 1,\dots,N, \end{cases} \tag{5}$$

where $\mathbf{s}^{(l)} = [s_l, s_{l+1}, \dots, s_N]$, $\psi_{l+1} = \widetilde{y}_l - \sum_{j=l+1}^{N} R_{lj}s_j$ with $l = 1,\dots,N$. Since the term $\sum_{j=l+1}^{N} R_{lj}s_j = 0$ for $l = N$, then we have $\psi_{N+1} = \widetilde{y}_l$. After N steps, the distance $d^2(\mathbf{s})$ is obtained as $d^2(\mathbf{s}) = T_1(\mathbf{s})$.

As an example, a three-level tree for a 4-PAM modulation is depicted in Figure 2. T_l is the distance metric at level l defined in (5). At every level, the radius constraint (4) must be verified and satisfied, otherwise the branch is pruned. In general, the radius is progressively reduced every time a leaf is reached at a distance that is smaller than current radius.

Several algorithms have been studied in order to make the tree traversal efficient. First algorithm, proposed by Fincke and Pohst in [7], needs to chose explicitly an initial radius.A more efficient solution was proposed by Schnorr and Euchner(SE) [9]. In this case, the initial radius is selected as the distance from the (ZF-DFE) solution and a "depth and best first" traversal of the tree is performed. Originally thought for infinite lattices, the SE algorithm was then adapted to finite lattices [4, 10].

The SE algorithm has intrinsically variable throughput and this makes it not very suitable for hardware implementation. The key to make this algorithm efficient or, at least, with a predictable throughput, is to make an effective pruning. Many theoretical studies in recent literature aim at reaching this goal [11]. A very interesting approach consists in an effective column reordering, which uses heuristic methods to reduce the search complexity with limited performance loss [12]. This technique results in very efficient tree search circuits but additional area is necessary for the preprocessing phase.

On the contrary, the approach proposed in this paper is based on the computational complexity reduction of the tree search algorithm with no column reordering. This solution is suitable for a flexible implementation that can adapt to different modulation sizes.

3. VLSI ARCHITECTURES

Implementation architectures for two key building blocks in MIMO detectors are presented in this section (tree search processing and matrix factorization). An enhanced sphere decoder is then described to be applied in the concatenated GST-TCM scheme.

3.1. Tree search processing block

Given the choice of adopting a fully ML detection algorithm for (2), several implementation options have been proposed in the literature.

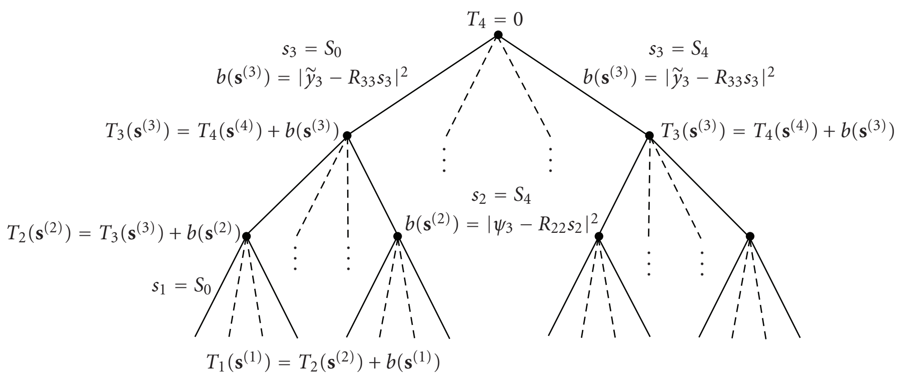
A first classification can be done with respect to the choice of real- or complex-valued tree construction. In the real case, the tree is twice deeper than the complex one. In complex trees, on the contrary, every node has the square of the number of sons with respect to the real tree. As an example, with $M_t = M_r = 4$ and 16-QAM modulation, a complex-valued tree construction would lead to a 4-level tree, where each node has 16 sons, while 8 levels and 4 sons per node appear in the corresponding real-valued tree. Although [13] demonstrates that a complex-valued tree results in a lower number of visited nodes, the construction of a real-valued tree allows for a more flexible solution, adaptable to different modulation schemes.

Another classification criterion is with respect to the implementation parallelism:

(i) parallelism at the level of tree exploration;
(ii) parallelism at the level of the metric computation for all sons of a given node and in the selection of the most probable son.

The first technique can be adopted only with suboptimal algorithms, while the second approach is not feasible with large cardinality QAM modulation schemes, as it implies a large number of concurrent multiplications. Hence, parallelism is not viable for the implementation of flexible architectures. A serial architecture, designed for high throughput, can achieve both flexibility and low area cost.

Detailed descriptions of the proposed architecture can be found in [14, 15]. The proposed architecture adopts a real-valued tree construction and a serial organization. This key advantage offered by this choice is the possibility of a run-time selection of the modulation scheme. The system is furthermore adaptable to different transmitting schemes including the golden code through the use of some



$T_4 = 0$
$s_3 = S_0$
$b(\mathbf{s}^{(3)}) = |\tilde{y}_3 - R_{33}s_3|^2$
$s_3 = S_4$
$b(\mathbf{s}^{(3)}) = |\tilde{y}_3 - R_{33}s_3|^2$
$T_3(\mathbf{s}^{(3)}) = T_4(\mathbf{s}^{(4)}) + b(\mathbf{s}^{(3)})$
$T_3(\mathbf{s}^{(3)}) = T_4(\mathbf{s}^{(4)}) + b(\mathbf{s}^{(3)})$
$s_2 = S_4$
$b(\mathbf{s}^{(2)}) = |\psi_3 - R_{22}s_2|^2$
$T_2(\mathbf{s}^{(2)}) = T_3(\mathbf{s}^{(3)}) + b(\mathbf{s}^{(2)})$
$s_1 = S_0$
$T_1(\mathbf{s}^{(1)}) = T_2(\mathbf{s}^{(2)}) + b(\mathbf{s}^{(1)})$

FIGURE 2: Tree organization for the sphere decoder. S represent the vector of symbol value in 4-PAM, $[-3, -1, 1, 3]$.

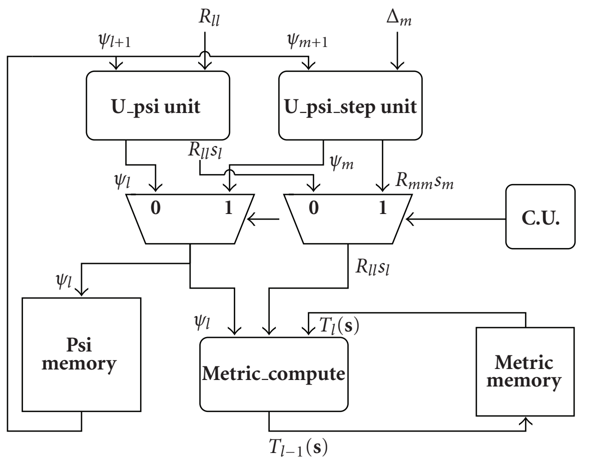


FIGURE 3: *Sphere decoder* block scheme (case of a node expanded in the depth-first mode, with no pruning).

instantiation parameters, which allow to choose the datapath width and the number of levels of the search tree.

The SE algorithm adopts the "depth and best first" traversal of the tree and the minimization of $|\psi_{l+1} - R_{ll}s_l|$ is required according to the problem formulation given in (5). The computation of the $|\psi_{l+1} - R_{ll}s_l|$ values for all possible s_l tends to become infeasible when the order of the modulation increases due to the large number of required operations. Core of the proposed approach is the selection of the s_l that minimizes $|\psi_{l+1} - R_{ll}s_l|$ by means of the division ψ_{l+1}/R_{ll}.

In particular, the iterative evaluation of (5) is rearranged in two steps. At the first step, the value ψ_{l+1} is received as an input from the previous iteration and the desired s_l for the analyzed node is directly obtained through the division ψ_{l+1}/R_{ll}; moreover, the output ψ_l is calculated for the selected s_l as $\psi_l = \tilde{y}_{l-1} - \sum_{j=l}^{N} R_{(l-1)j}s_j$. The second processing step receives ψ_l and T_l, to actually compute T_{l-1}, according to (5). The two operations are performed by units **U_psi_Unit** and **Metric_Compute** in Figure 3, where memories required to store ψ_l amounts and T_l metrics are also shown.

It is worth noting that the result of the division ψ_{l+1}/R_{ll} is rounded to the closest Q-PAM constellation points $\pm1, \pm3, \ldots$. As a consequence, a general purpose hardware divisor is not necessary and the required operation can be executed by means of the first $\log_2 Q$ steps of a successive subtraction divider [16]. This divider has a very simple architecture that employs only shifts and subtractions; although it tends to be very slow for a complete division, this solution can be effectively used when only a few shift and add elementary operations are required.

In a high throughput sphere decoder, a new T_l metric must be evaluated at each clock cycle. In order to achieve this target, the two steps exploit a pipelined architecture. Additionally, an alternative metric must always be ready, also when a pruning of the tree occurs; therefore, in the proposed architecture, two "candidate" nodes are selected in parallel when processing a given father node. The first one is a direct son of the current node, selected by the **U_psi** unit of Figure 3 by descending along the tree. The "alternative" node, selected by **U_psi_Step Unit**, is placed at a higher level in the tree and it is chosen when the branch has to be pruned, that is when the current metric exceeds the best current metric evaluated in the tree traversal. The procedure adopted to select the alternative node is described below.

In the **U_psi Unit**, the evaluation of the direct son of the current node makes use of the division ψ_{l+1}/R_{ll} and the result is approximated either by defect or by excess to the nearest PAM constellation point: the best choice for s_l is given by (see Figure 4)

$$s_{l_{(1)}} = \frac{\psi_{l+1}}{R_{ll}} + \Delta_l, \tag{6}$$

where Δ_l is the correction term. The sign of Δ_l is exploited to select the second (and following) nearest point in the PAM constellation, according to the following rule:

$$s_{l_{(k)}} = s_{l_{(k-1)}} - (-1)^k \operatorname{sign}(\Delta_l)(k-1)A, \tag{7}$$

where A is the distance between two consecutive points.

Thus, **U_psi_Step Unit** simply computes (7) to find the second most probable value of s_l. Figure 4 shows the sequence of alternative nodes selected at a given tree level, after the occurrence of pruning.

Summarily, we have the following.

(i) The division approach achieves low complexity and flexibility in terms of supported modulation schemes.

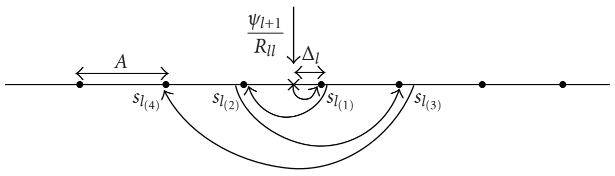


FIGURE 4: Method used to select alternative nodes in U_psi_step unit.

(ii) The concurrent evaluation of two "candidate" nodes provides a significant speed-up to the inherently serial SE sphere decoding algorithm and has a limited impact on complexity.

3.2. *Matrix factorization*

Understanding of throughput requirements is fundamental in the architectural study of this processing block. The IEEE 802.11n WLAN standard, which adopts space-time coding, implies that a new channel estimation is performed whenever a packet arrives; this means that the number of matrix factorizations ranges from a minimum of 64 in a time period of 36 microseconds, to a maximum of 128 in 28 microseconds.

In the design of the matrix factorization block, a first choice between householder transformations and Givens-rotations-based algorithms [17] has to be made. The latter approach results in a sequence of rotation operations that cancel elements under the main diagonal of the matrix. Givens rotations require a larger number of floating-point operations compared to householder transformations; nevertheless they may be implemented using parallel systolic arrays and for this reason they are usually preferred for hardware implementation.

Every single processing element (PE) of the systolic array must perform the angle calculation and the rotation to cancel the matrix elements. Several alternatives exist to accomplish these two tasks, and the most common ones are

(1) computation of *sine* and *cosine* of the angle by means of operations including square roots and divisions;
(2) direct angle calculation and rotation using CORDIC processors [18].

The main advantage of the *sine* and *cosine* approach is that primitives can be optimized resulting in an efficient, although expensive, implementation. The second technique is less expensive, but outputs are generated with longer latencies and data dependency between operations. The very high throughput required by this application can hardly be achieved by iterative CORDIC-based algorithms. Other alternatives have to be explored to reduce the latency of every single processor. Among the square root-free algorithms, the squared Givens rotations (SGR) proposed by Dölher [19] constitute a good compromise between complexity and speed [20, 21].

Let us indicate with $\mathbf{a} = (0, \dots, 0, a_k, \dots, a_n)$ the row of an $n \times n$ matrix, where a 0 must be introduced in the kth position and with $\mathbf{r} = (0, \dots, 0, r_k, \dots, r_n)$ another row having the same number of leading zeros; the standard Givensrotations (StdGR) algorithm employs this set of updating equations to cancel the element a_k:

$$\overline{\mathbf{a}} = q^{-1}(-a_k\mathbf{r} + r_k\mathbf{a}), \qquad \overline{\mathbf{r}} = q^{-1}(r_k\mathbf{r} + a_k\mathbf{a}), \qquad q = \sqrt{r_k^2 + a_k^2}. \tag{8}$$

The SGR algorithm takes advantage of the observation that $\overline{r}_k = q$ introduces the matrix $\mathbf{U} = \text{diag}(\mathbf{R}) \cdot \mathbf{R}$ and exploits the relations $\mathbf{u} = r_k\mathbf{r}$ and $\overline{\mathbf{u}} = \overline{r}_k\overline{\mathbf{r}}$. Then, to simplify the notation, the new vectors $\mathbf{v} = \mathbf{a}/\sqrt{w}$ and $\overline{\mathbf{v}} = \overline{\mathbf{a}}/\sqrt{\overline{w}}$ are introduced for some $w, \overline{w} > 0$. After some algebra, we can express (8) with a new set of updating equations:

$$\overline{\mathbf{u}} = \mathbf{u} + wv_k\mathbf{v}, \qquad \overline{\mathbf{v}} = \mathbf{v} - \frac{v_k}{u_k}\mathbf{u}, \qquad \overline{w} = wu_k/\overline{u}_k. \tag{9}$$

When compared to StdGR, SGR algorithm shows half the number of multiplications and no square-root operation. The updating sequence can be arranged in a systolic array of PEs performing the aforementioned computations.

The PE array can be arranged according to different structures, namely the triangular (TA), square, and linear (LA) shapes: each of them shows a different percentage of PE reuse and a different throughput. Slightly different functions are then associated in the array organization to boundary and internal PEs.

Figure 5 pictures a generic systolic array layout, able to perform QR decomposition of a 4×4 matrix. The identity matrix must enter the systolic array immediately after the matrix to be processed, in order to produce the $\mathbf{Q}$ matrix. During the processing of the input matrix $\mathbf{M}$, the coefficients of $\mathbf{Q}$ are already computed and stored in the internal registers.

Depending on (9), boundary and internal processing elements must behave differently when a diagonal element of the matrix enters a node. In Table 1, the computations performed by the nodes in the different operating modes are listed. In the table, Reg and Reg2 are two registers needed to store the parameters between different steps. The subscript in indicate that a parameter takes origin from the preceding PE in accordance with the connections in Figure 5, while subscript out indicates that a parameter takes origin in the current PE. It must be also noted that the parameter w_{out} is updated only in diagonal mode, while in the other modes it maintains the registered value.

The internal processing element (IPE) appears to be the most computationally intensive block of the entire system. Figure 6 depicts the architecture of the IPEs derived from

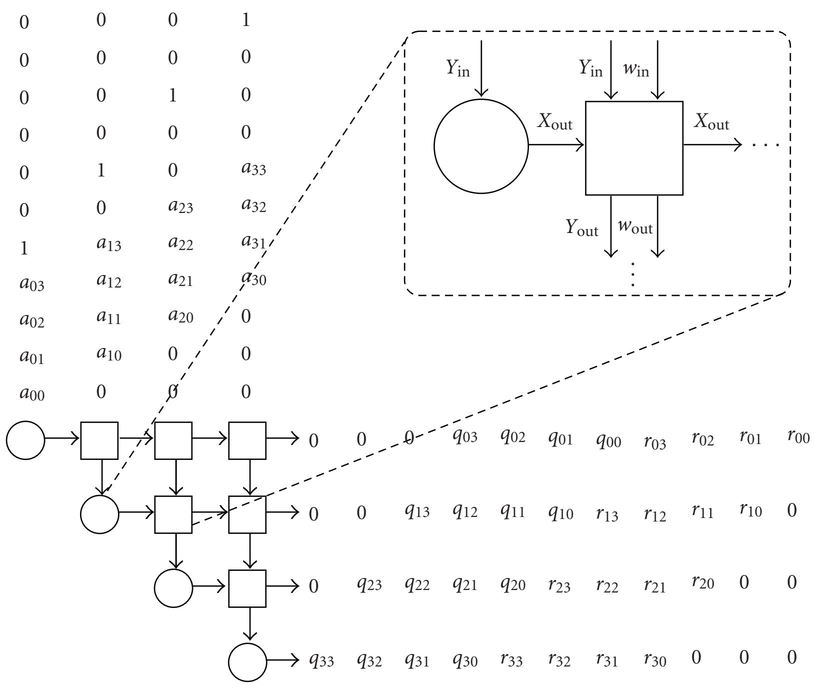


Figure 5: Systolic array for QR decomposition of a 4 × 4 matrix.

Table 1: Operations performed by the PE's.

Mode	Operation
Boundary PE	
Diagonal	Reg $\Leftarrow Y_{\text{in}}$; $X_{\text{out}} \Leftarrow Y_{\text{in}}^2 \cdot w_{\text{in}}$
Nondiagonal	Reg $\Leftarrow$ Reg; $X_{\text{out}} \Leftarrow Y_{\text{in}} \cdot \text{Reg} \cdot w_{\text{in}}$
Internal PE	
Diagonal	Reg $\Leftarrow Y_{\text{in}}$; Reg2 $\Leftarrow \dfrac{Y_{\text{in}}}{X_{\text{in}}}$; $w_{\text{out}} \Leftarrow w_{\text{in}} \cdot \dfrac{X_{\text{in}}}{X_{\text{in}} + Y_{\text{in}}^2}$ $X_{\text{out}} \Leftarrow X_{\text{in}} + Y_{\text{in}}^2 \cdot w_{\text{in}}$ $Y_{\text{out}} \Leftarrow Y_{\text{in}} - \dfrac{Y_{\text{in}}}{X_{\text{in}}} \cdot X_{\text{in}}$
Nondiagonal	Reg $\Leftarrow$ Reg; Reg2 $\Leftarrow$ Reg2, $w_{\text{out}} \Leftarrow w_{\text{out}}$ $X_{\text{out}} \Leftarrow X_{\text{in}} + Y_{\text{in}} \cdot \text{Reg} \cdot w_{\text{in}}$ $Y_{\text{out}} \Leftarrow Y_{\text{in}} - \text{Reg2} \cdot X_{\text{in}}$

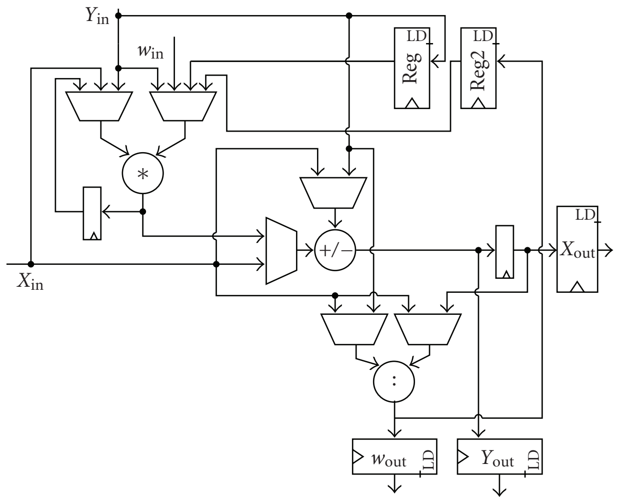


Figure 6: Block diagram of internal PE.

Table 1. Although the divisor has a latency of two clock cycles and two divisions are needed in the diagonal mode, a proper overlapping with the nondiagonal mode guarantees a total latency of three clock cycles.

The method proposed in [22] is adopted to realize the division operation. Using the a Taylor series, the divisor (Y) expressed on $2m$ bits is decomposed into two m-bit groups, higher (Y_H) and lower bits (Y_L). Since $(Y_H)^2 \gg (Y_L)^2$, we can write

$$\frac{X}{Y} = \frac{X}{Y_H + Y_L} = \frac{X(Y_H - Y_L)}{Y_H^2 - Y_L^2} \simeq \frac{X(Y_H - Y_L)}{Y_H^2} \tag{10}$$

with maximum fractional error $< 2^{-2m}$. This divisor takes two clock cycles to complete the division on 16 bit fixed-point data [23]; it requires a multiplier, an adder/subtracter, and a 256 8-bit entries LUT to store the inverse of Y_H^2. The overall complexity of the internal PE is therefore given by two 16 bit multipliers, two adders/subtracters and a LUT.

In this paper, we considered 8 × 8 real matrices as required by the 2 × 2 MIMO system with two channel uses per codeword. With a plain triangular architecture, which allows to obtain the highest throughput, a new matrix can enter the array after 16 steps (8 for computing **R** matrix and 8 for **Q**), that is every 48 clock cycles. In order to factorize 64 matrices in 28 microseconds we need to maintain the clock period shorter than 9 nanoseconds, while a period of 4.5 nanoseconds is required to factorize 128 matrices.

3.3. Enhanced sphere decoder for E_8 lattices

In this section, we address concatenated bandwidth efficient coding schemes for MIMO channels, where a space-time code with nonvanishing determinant is used as inner code and an outer trellis code is concatenated to further increase the reliability of the communication [6].

This TCM exploits the basic idea of partitioning the inner constellation; at each channel use, a signal is selected from one of the partitions. In standard TCM for AWGN channels, the Euclidean distance between points in the same subset is made as large as possible [24]. Full rank ST code design is based on the maximization of the minimum determinant

$$\Delta_{\min} = \min_{\mathbf{X}\neq\hat{\mathbf{X}}} \det\left[(\mathbf{X}-\hat{\mathbf{X}})(\mathbf{X}-\hat{\mathbf{X}})^{\dagger}\right], \tag{11}$$

where $\mathbf{X}$, $\hat{\mathbf{X}}$ are distinct codeword matrices. This pseudodistance replaces the role of the Euclidean distance. In [6] $\Delta_{\min}$ is optimized using set-partitioning that increases the minimum determinant with the partitions. The $\mathbb{Z}^8$ lattice structure of the inner golden code is used, so that sublattices and their cosets are used as partitions. The outer convolutional encoder guarantees that signals are selected properly from different cosets. Among the possible 8-dimensional sublattices considered in GST-TCM, we choose the Gosset lattice E_8 (the densest packing in 8 dimensions [25]).

Any received point has to be decoded to one of the 16 possible cosets of E_8 compounding $\mathbb{Z}^8$. The decoder needs to compute the branch metrics of the inner code to perform Viterbi ML decoding of the concatenated codeword. This is obtained by ML lattice decoding of the received vector in each coset of the E_8 sublattice.

In order to decode the E_8 lattice, we consider that $E_8 \subset \mathbb{Z}^8$ and adapt the classical sphere decoder (as that in [14]) operating on $\mathbb{Z}^8$.

Consequently, this decoding problem can be solved by thinking of E_8 as a punctured $\mathbb{Z}^8$ lattice and setting proper constraints to discriminate the relevant points E_8 within $\mathbb{Z}^8$. This means that at a given tree level, the integer signal vector cannot assume all values; actually it is constrained by the selections that have already been made at upper levels.

These constraints can be derived directly from the construction A of E_8 based on the (8,4,4) extended Hamming code [6]. Let $\mathbf{c} = [c_0, \ldots, c_7]$ denote one of the 16 binary codewords that are used as coset leaders of $2\mathbb{Z}^8$ to obtain E_8.

Taking into account that the tree must be traversed starting from the last dimension, we have

$$\mathbf{c} = \begin{cases} c_7 = \text{free}, \\ c_6 = \text{free}, \\ c_5 = \text{free}, \\ c_4 = c_7 \oplus c_6 \oplus c_5, \\ c_3 = \text{free}, \\ c_2 = (c_4 \oplus c_3)\cdot c_5, \\ c_1 = (c_4 \oplus c_3)\cdot c_6, \\ c_0 = (c_4 \oplus c_3)\cdot c_7. \end{cases} \tag{12}$$

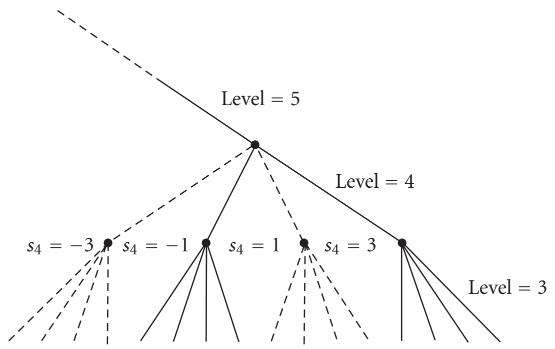


Figure 7: Cross-section at levels 4 and 3, assuming c_7 = "1," c_6 = "0," c_5 = "1" and $\bar{c}_4$ = "1" we obtain $c_4 = c_7 \oplus c_6 \oplus c_5$ = "0" while $c'_4 = \bar{c}_4 \oplus c_4$ = "1" where c is the output of the convolutional encoder and represents a coset leader of $2\mathbb{Z}^8$ in E_8 and $\bar{\mathbf{c}}$ is a coset leader of E_8 in $\mathbb{Z}^8$.

Table 2: Synthesis results at 0.13 μm technology for SD and matrix factorization blocks.

	SD	Matrix factor
Core Area [GE]	61 k	198 k
Max. Clock [MHz]	213	223
Throughput at SNR = 20 dB	148.6 Mbps 16-QAM	4.63 Mmat/s

If, at level i, c_i is free, then the signal can assume any value in the original QAM constellation, otherwise its value is constrained.

In order to perform the ML detection, we have to derive the proper evolution of the received signal among the different sublattices. In particular, we can define $\mathbf{c}$ as the output of the convolutional encoder, which is related to the current state of the encoder, and $\bar{\mathbf{c}}$ as one of the 16 coset leaders of E_8 in $\mathbb{Z}^8$. Combining $\mathbf{c}$ with the coset leader $\bar{\mathbf{c}}$, we obtain a binary vector $\mathbf{c}' = \bar{\mathbf{c}} \oplus \mathbf{c}$ that gives the 256 distinct coset leaders of $2\mathbb{Z}^8$ in $\mathbb{Z}^8$. Thus, all $\mathbf{c}'$ vectors identify the actual allowed points inside $\mathbb{Z}^8$. From the practical point of view, $\bar{\mathbf{c}}$ is fixed for the considered E_8 decoder, while the allowed and interdicted values of the signal s_i depend on the value of $\mathbf{c}'$. If c'_i = "0," then s_i can take the values $[\ldots,-7,-3,1,5,\ldots]$, otherwise it can take the values $[\ldots,-5,-1,3,7,\ldots]$; the bounds of this sets depend on the constellation used for the transmission. It is worth noting that, when c_i is free, c'_i can assume both the values 0 and 1, leading s_i to assume any value in the original PAM constellation.

Figure 7 shows levels 3 and 4 of a tree for the sphere decoding of 4-PAM systems: solid lines are practicable edges, while dashed lines correspond the interdicted ones. For this cross-section, we assume c_7 = "1," c_6 = "0," and c_5 = "1," resulting in c_4 = "0," $\bar{c}_4$ = "1," and c'_4 = "1." Therefore, values $[-1, 3]$ are allowed in this example. At level 3, instead, c_3 is free, and as a consequence c'_3 can assume both values "0" and "1" and the four branches are all admissible.

Table 3: Comparison results for SD building block.

SD	Our		[26]	[27]
Antennas	2 × 2 per two channel uses		4 × 4	4 × 4
Modulation	4,16,64-QAM	16-QAM	16-QAM	16-QAM
Detector	Depth-first sphere		Depth-first sphere	K-best sphere
BER perf.	ML		ML	Quasi-ML
Tech. μm	0.13	0.25	0.25	0.35
Core area GE	61 k +preproc.	56 k +preproc.	117 k +preproc.	91 k +preproc.
Max. clock	213 MHz	109 MHz	51 MHz	100 MHz
Throughput @ SNR = 20 dB	148.6 Mbps 16-QAM	83 Mbps	73 Mbps	52 Mbps

Table 4: FPGA synthesis results for matrix factorization building block.

Tech. μm	xc4vlx200	xc2v1000	xc4vlx200
Handled matrices	8 × 8 Real	4 × 4 Complex	4 × 4 Complex
Array	TA	LA	SE
no. of PEs	32	4	2
f_{clk} MHz	89	101	115
Area	8321(9%) 92 DSP48	1666 Slices (32%) 4 BRAM (10%)	9117 Slices (10%) + 22 DSP48 (23%) + 9 BRAM (3%)
Throughput	1.85 Mmat/s	0.45 Mmat/s	0.15 Mmat/s

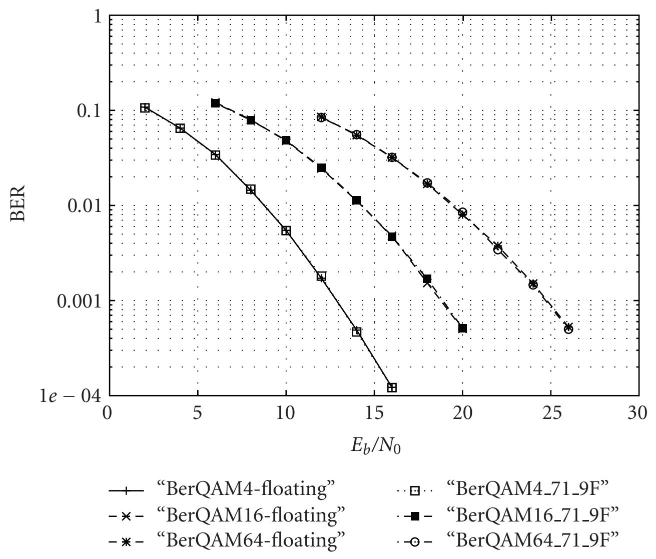


Figure 8: Proposed system performance with different modulations.

The proposed scheme allows to realize with a unique circuit the branch metric computer unit required in the Viterbi algorithm necessary for the decoding of the $\mathbb{Z}^8/E_8$ TCM transmission scheme in [6]. Note that, at each stage of the trellis, 16 different E_8 decoders are required.

The adopted architecture is very similar to the architecture described in Section 3.1 and in [14]. The only difference is the additional functional block, the "constraint maker," able to realize (12).

4. IMPLEMENTATION RESULTS

Sphere decoder performance in the 2 × 2 golden code scenario described in Section 1 is reported in Figure 8 in terms of bit error rate (BER) versus SNR, for 4-, 16-, and 64-QAM modulations. Fixed-point results are also plotted for the case of a 16-bit data representation (7 bits for integer and 9 for fractional parts): in accordance with [23], these results prove that, for this particular application, 16-bit representation is sufficient to achieve the floating-point performances and thus it has been adopted for all the processing blocks here described.

The proposed architectures have been synthesized on a 0.13 μm commercial CMOS standard cell technology with synopsys design compiler. The synthesis results are presented in Table 2: the sphere decoder synthesis results here listed are obtained with a flexible architecture able to decode 4 to 64 QAM modulations, while the matrix factorization block has been realized with a triangular array architecture (Mmat/s indicates millions of matrices processed in a second). It must be noticed that synthesis results differ from those in [14], although referred to the same implementation, due to the use of different synthesis libraries.

For comparison purposes, the tree search block has been also synthesized on 0.25 μm CMOS Standard cell technology (Table 3): we then compare our architecture to the ML implementation described in [26] and the quasi-ML implementation in [27]. It must be noted that two different implementations are presented in [26], one is ML, while the other has close to ML BER performance: as the latter implementation adopts a completely different approach and maps a suboptimal algorithm, only the first implementation figures are included in Table 3 for comparison purposes.

Analyzing data in Table 3, it can be observed that our rearranged approach for the sphere decoder with a single metric computation per cycle allows a significant complexity reduction (approx. 50% for 16 QAM modulation) with respect to parallel structures. At the same time, thanks to the pipelined architecture, we can achieve a remarkable average decoding throughput without any highly specialized structure. Moreover, our flexible decoder is not limited to a single modulation scheme, but it can adapt to different modulations (4-, 16-, and 64-QAM).

Fair comparisons to other implementations cannot be done for the matrix factorization block, as published solutions adopts completely different architectures. For the sake of completeness, we report here two FPGA developments, [21, 28], which implement the SGR algorithm. The main features of these architectures are summarized in Table 4, together with the synthesis results of our solution mapped onto a Xilinx Virtex4 (xc4vlx200) FPGA device. Both [21, 28] carry out the computation of 4×4 complex matrices, while we process 8×8 real-valued ones. This means that, while the single PE complexity will be greater in the complex scenario, the number of PE the data flow pass through is twice and with the basic TA topology, while for a 4×4 matrix there will be 8 PEs, and 32 PEs are required for 8×8 a matrix.

Another difference among these implementations is related to the processing topology; while our solution adopts a TA processing topology with 32 PEs, [28] makes use of a linear array (LA) organization with 4 PEs and two single PEs are used in [21], one for boundary processing and the second one for internal processing.

A further difference with respect to [28] is that in our implementation weight w is updated according to (9) while in [28] it is fixed to a constant value.

In conclusion, the standard cell version fully reaches both the 64 matrices in 36 microseconds and the 128 matrices in 28 microseconds goals and the throughput of the proposed approach compares favourably to that of the other implementations showing high performances at a limited additional cost. On the contrary, the FPGA implementation enables only to reach the 64 matrices in 36 microseconds.

The E_8 decoder, instead, adopting the same architecture as the $\mathbb{Z}_8$ sphere decoder presents a comparable complexity. A little increase in area is due to the addition of the functional block "constraint maker," leading the overall complexity to 62 kGates, and the maximum achievable frequency to 196 MHz.

5. CONCLUSIONS

The hardware implementation of key building blocks in a MIMO-OFDM receiver has been presented. The analysis of the blocks shows their high level of complexity, which justifies the ASIC design approach. The sphere decoder architecture enables to manage different modulations without any loss in BER performance while the proposed matrix factorization algorithm and arrangement allow to achieve the highest throughput specified in the 802.11n standard. Finally, the design of an enhanced sphere decoder, capable of supporting E_8 decoding in a ST-TCM concatenated schemes, has been proposed.

REFERENCES

[1] J.-C. Belfiore, G. Rekaya, and E. Viterbo, "The golden code: a 2×2 full-rate space-time code with nonvanishing determinants," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1432–1436, 2005.

[2] H. Yao and G. Wornell, "Achieving the full MIMO diversity-multiplexing frontier with rotation-based space-time codes," in *Proceedings of the 41st Annual Allerton Conference on Communication, Control, and Computing*, Monticello, Ill, USA, October 2003.

[3] P. Dayal and M. K. Varanasi, "An optimal two transmit antenna space-time code and its stacked extensions," in *Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 987–991, Pacific Grove, Calif, USA, November 2003.

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

[5] E. Viterbo and E. Biglieri, "A universal decoding algorithm for lattice codes," in *Proceedings of the 14th GRETSI Symposium on Signal and Image Processing*, pp. 611–614, Juan-les-Pins, France, September 1993.

[6] Y. Hong, E. Viterbo, and J.-C. Belfiore, "Golden space-time trellis coded modulation," *IEEE Transactions on Information Theory*, vol. 53, no. 5, pp. 1689–1705, 2007.

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

[8] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1639–1642, 1999.

[9] C. P. Schnorr and M. Euchner, "Lattice basis reduction: improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, no. 2, pp. 181–199, 1994.

[10] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Transactions on Information Theory*, vol. 48, no. 8, pp. 2201–2214, 2002.

[11] K.-W. Wong, C.-Y. Tsui, R. S.-K. Cheng, and W.-H. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 3, pp. 273–276, Phoenix, Ariz, USA, May 2002.

[12] C. Hess, M. Wenk, A. Burg, et al., "Reduced-complexity MIMO detector with close-to ML error rate performance," in *Proceedings of the 17th Great Lakes Symposium on VLSI (GLSVLSI '07)*, pp. 200–203, Stresa, Lago Maggiore, Italy, March 2007.

[13] A. Burg, M. Borgmann, C. Simon, M. Wenk, M. Zellweger, and W. Fichtner, "Performance tradeoof in the VLSI implementation of the sphere decoding algorithm," in *Proceedings of the 5th IEE International Conference on 3G Mobile Communication Technologies*, pp. 93–97, London, UK, October 2004.

[14] B. Cerato, G. Masera, and E. Viterbo, "A VLSI decoder for the golden code," in *Proceedings of the 13th IEEE International Conference on Electronics, Circuits and Systems (ICECS '06)*, pp. 549–552, Nice, France, December 2006.

[15] B. Cerato, G. Masera, and E. Viterbo, "Decoding the Golden Code: a VLSI design," http://arxiv.org/abs/0711.2383v1.

[16] B. Parhami, *Computer Arithmetic. Algorithms and Hardware Designs*, Oxford University Press, Oxford, UK, 2000.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, Md, USA, 1996.

[18] B. Haller, J. Götze, and J. R. Cavallaro, "Efficient implementation of rotation operations for high performance QRD-RLS filtering," in *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP '97)*, pp. 162–174, Zurich, Switzerland, July 1997.

[19] R. Dölher, "Squared givens rotation," *IMA Journal of Numerical Analysis*, vol. 11, no. 1, pp. 1–5, 1991.

[20] R. W. G. Lightbody and R. Woods, "Design of parametrizable silicon intellectual property core for qr-based rls filtering," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 4, pp. 659–678, 2003.

[21] M. Karkooti, J. R. Cavallaro, and C. Dick, "FPGA implementation of matrix inversion using QRD-RLS algorithm," in *Proceedings of the 39th Asilomar Conference on Signals, Systems and Computers*, pp. 1625–1629, Pacific Grove, Calif, USA, October-November 2005.

[22] P. Hung, H. Fahmy, O. Mencer, and M. Flynn, "Fast division algorithm with a small lookup table," in *Proceedings of the 33th Asilomar Conference on Signals, Systems, and Computers*, vol. 2, pp. 1465–1468, Pacific Grove, Calif, USA, October 1999.

[23] L. M. Davis, "Scaled and decoupled Cholesky and QR decompositions with application to spherical MIMO detection," in *Proceedings of the IEEE Wireless Communications and Networking (WCNC '03)*, vol. 1, pp. 326–331, New orleans, La, USA, March 2003.

[24] S. Benedetto and E. Biglieri, *Principles of Digital Transmission With Wireless Applications*, Kluwer Academic/Plenum Publishers, New York, NY, USA, 1999.

[25] J. Conway and N. Sloane, *Sphere Packings, Lattices and Groups*, Springer, New York, NY, USA, 1992.

[26] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1576, 2005.

[27] Z. Guo and P. Nilsson, "A VLSI architecture of the Schnorr-Euchner decoder for MIMO systems," in *Proceedings of the 6th IEEE Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, vol. 1, pp. 65–68, Shanghai, China, May-June 2004.

[28] F. Edman, "Digital hardware aspects of multiantenna algorithms," Ph.D. dissertation, Department of Electroscience, Lund University, Lund, Sweden, February 2006.