System Architecture and Implementation of MIMO Sphere Decoders on FPGA

Xinming Huang, *Member, IEEE*, Cao Liang, *Student Member, IEEE*, and Jing Ma, *Member, IEEE*

***Abstract*—Multiple-input–multiple-output (MIMO) systems use multiple antennas in both transmitter and receiver ends for higher spectrum efficiency. The hardware implementation of MIMO detection becomes a challenging task as the computational complexity increases. This paper presents the architectures and implementations of two typical sphere decoding algorithms, including the Viterbo–Boutros (VB) algorithm and the Schnorr–Euchner (SE) algorithm. Hardware/software codesign technique is applied to partition the decoding algorithm on a single field-programmable gate array (FPGA) device. Three levels of parallelism are explored to improve the decoding rate: the concurrent execution of the channel matrix preprocessing on an embedded processor and the decoding functions on customized hardware modules, the parallel decoding of real/imaginary parts for complex constellation, and the concurrent execution of multiple steps during the closest lattice point search. The decoders for a 4×4 MIMO system with 16-QAM modulation are prototyped on a Xilinx XC2VP30 FPGA device with a MicroBlaze soft core processor. The hardware prototypes of the SE and VB algorithms show that they support up to 81.5 and 36.1 Mb/s data rates at 20 dB signal-to-noise ratio, which are about 22 and 97 times faster than their respective implementations in a digital signal processor.**

***Index Terms*—Field-programmable gate array (FPGA), lattice point search, multiple-input–multiple-output (MIMO) detection, parallel structure, sphere decoding, system-on-chip (SoC).**

I. Introduction

WIRELESS communication systems are dense compositions of signal processing and VLSI technologies. With the ever increasing demand of higher data rate and better quality of service, VLSI design and implementation method for wireless communications becomes more challenging, which urges researchers to provide new architectures and efficient implementations to meet high performance requirements. In recent years, the interests in multiple-input–multiple-output (MIMO) systems have exploded. It is well known that MIMO systems are able to increase system capacity and improve communication reliability [1]–[3]. The applications of MIMO technology have emerged at the forefront of the developing standards for next-generation mobile communications and wireless networks. Combined with the orthogonal frequency-division multiplexing (OFDM) technique, MIMO is proposed to be incorporated into the fourth generation (4G) mobile communications system architectures to enhance voice and data transmissions. Recent initiatives for standardization of future MIMO communication systems including UMTS (3GPP Release 7) [4], IEEE 802.11n wireless LAN [5], and IEEE 802.16e WiMax [6] reflect the importance of MIMO techniques.

The information theory for MIMO systems has been well studied on performance parameters such as data rate and bit error rate (BER) [7]. The layered space-time receiver structures and coding schemes have allowed the MIMO systems to approach the theoretical capacities on a multiantenna channel [8]. On the receiver end, one of the key functions is to perform channel decoding to recover the original data stream corresponding to each of the transmitted antennas from the receiving signal vector and estimated channel information. Both lattice theory and coding theory are applied in the design of MIMO detection algorithms. In a multiple antenna channel environment, each of the transmitted signal vectors is aligned on the modulated constellation points. Therefore, a multilayered lattice is formed with a set of finite points and the MIMO detection is essentially an algorithm to search for the closest lattice point to the received vector. There are two typical classes of comprehensive search algorithms for a lattice without an exploitable structure. One is the Pohst strategy that examines lattice points lying inside a hypersphere [9], [10]. The lattice decoding algorithm developed by Viterbo and Boutros is based on the Pohst strategy [11]. Another class of lattice search strategy is suggested by Schnorr and Euchner [12], based on examining the points inside the aforementioned hypersphere in zig zag order of lattice layers with nondecreasing distance from the received signal vector. A representative lattice decoding algorithm based on Schnorr–Euchner (SE) strategy is applied by Agrell *et al.* [13]. Both lattice search algorithms solve the maximum-likelihood (ML) detection problem. Both algorithms are considered the most promising approaches for MIMO detection, and are also commonly referred as sphere decoders since the algorithms search for the closest lattice point within a hypersphere.

Due to the complexity of the lattice decoding algorithms and the high data dependency among the decoding procedures, the MIMO decoders are generally implemented on digital signal processors (DSPs), such as the Bell Labs layered space-time (BLAST) system [14], [15]. Because it does not support parallel computation, the speed of the DSP implementation is often limited, especially as the number of antennas increases. The VLSI architectures of MIMO systems have been investigated recently. It is a challenging task to reduce the complexity of the

Manuscript received May 2, 2006; revised June 13, 2007. This work was supported in part by the National Science Foundation under Grant EPS-0346411.

X. Huang and C. Liang are with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA 01609 USA (e-mail: xhuang@ece.wpi.edu; cliang@ece.wpi.edu).

J. Ma was with the Department of Electrical Engineering, University of New Orleans, New Orleans, LA 70148 USA. She is now with The MathWorks Inc., Natick, MA 01760 USA (e-mail: jing.ma@mathworks.com).

Digital Object Identifier 10.1109/TVLSI.2007.912042

VLSI implementation in order to achieve maximal performance in real-time. Several hardware implementations have been reported by prototyping the VB algorithm, the SE algorithm, or their modified versions [16]–[18]. Most recently, the application-specific integrated circuit (ASIC) implementation by Berg *et al.* has demonstrated the decoding rate up to 73 Mb/s at 20-dB signal-to-noise ratio (SNR) [19]. It is a modified searching algorithm based on the Pohst strategy and incorporates the depth-first tree traversal with progressive radius reduction. However, an ASIC implementation is generally refined for a fixed number of antennas and a certain signal constellation, and is optimized for low power high frequency circuit design. The limitation of an ASIC implementation is lack of flexibility when the number of antennas or the signal constellation changes.

Field-programmable gate-array (FPGA) devices are widely used in signal processing, communications, and network applications because of their reconfigurability and support of parallelism. FPGA has at least three advantages over a DSP processor: the inherit parallelism of an FPGA is equipped for vector processing; it has reduced instruction overhead; the processing capacity is scalable if the FPGA resource is available. The disadvantage is that the development cycle of the FPGA design is usually longer than the DSP implementation. But once an efficient architecture is developed and the parallel implementation is explored, FPGA is able to significantly improve the processing speed because of its intrinsic density advantage [20]. Furthermore, FPGA also has several advantages over an ASIC implementation: an FPGA device is reconfigurable to accommodate system configuration changes even in run-time; it has significantly reduced prototyping latency comparing to ASIC; it is a cost-effective solution to meet the low volume short cycle product requirement [21].

In addition, the system-on-chip (SoC) concept has been adopted to FPGA lately by introducing one or more embedded processors into the FPGA design [22], e.g., the PowerPC hard processor cores and the MicroBlaze soft processors on Xilinx FPGAs as well as the Nios soft processors on Altera FPGA devices.[1,2] The SoC architecture significantly improves the interporability and reduces the design complexity of many complex computational algorithms. Consequently, the hardware/software codesign technique can be applied to partition the computational algorithm into customized hardware and embedded software. For instance, one or more embedded processors can be instantiated in an FPGA to execute processing tasks that are less time critical but highly sequential or considerably complicated for direct circuit implementation. In this paper, we introduce the FPGA-based SoC architectures for two typical sphere decoding algorithms.

The main contributions of this paper are summarized as follows.

1) We present the FPGA-based system architectures and implementations for MIMO sphere decoding. To the authors' knowledge, the real-time performance results of the system prototypes are among the fastest MIMO decoders reported thus far. The FPGA implementations provide the flexibility for varying the number of antennas and signal constellations.
2) The SoC approach is introduced to simplify the design of the sphere decoder and to improve the efficiency. The hardware/software codesign techniques partition the complicated preprocessing tasks such as matrix factorizations and inversions to an embedded processor and the real-time decoding functions to customized hardware modules.
3) Both the VB and the SE decoding algorithms are implemented on an FPGA platform and are evaluated for decoding rates, BER performance, power consumptions, and area utilizations. The implementations on a DSP are also presented for comparison.
4) Three levels of parallelism are explored to accelerate the processing: the concurrent execution of the preprocessing on an embedded processor and the decoding on hardware cores; the parallel decoding of real/imaginary parts if complex constellation applies; and the concurrent execution of multiple steps during the closest lattice point search.

This paper is organized as follows. Section II reviews the principles of the VB and SE decoding algorithms. The hardware/software codesign architecture is described in Section III. Three levels of parallelism are explored in Section IV. The data dependency among the iterative lattice search procedures is also analyzed in this section. A comparison of the experimental results between these two algorithms among the FPGA and DSP implementations is presented in Section V. The conclusion is given in Section VI.

II. Sphere Decoding Algorithm

Considering an MIMO system with M transmit and N receive antennas, the received signal $\mathbf{y}$ is given by

$$\mathbf{y} = \mathbf{uH} + \mathbf{n} \tag{1}$$

where $\mathbf{u}$ is a transmitted signal vector and $\mathbf{n}$ is an additive white Gaussian noise vector. $\mathbf{H}$ is the $M \times N$ channel matrix that can be assumed as known from perfect channel estimation and synchronization. For selected modulation scheme, each element of the transmit vector $\mathbf{u}$ is a constellation point and the channel matrix $\mathbf{H}$ generates a lattice. The ML decoding algorithm is to find the minimal distance between the received point and the examining lattice point that

$$\hat{\mathbf{u}} = \arg \min_{\mathbf{u} \in \Omega^{M_T}} \|\mathbf{y} - \mathbf{uH}\|^2 \tag{2}$$

where $\hat{\mathbf{u}}$ is the decoded vector. The entries of $\mathbf{u}$ are chosen from a complex constellation Ω. The set of all possible transmitted vector symbols is denoted by Ω^M. Thus, the ML-based sphere decoding system can be summarized as follows.

Input: The channel lattice generation matrix $\mathbf{H}$ and the received signal vector $\mathbf{y}$.

Output: A $1 \times M$ vector $\hat{\mathbf{u}}$ such that $\hat{\mathbf{u}}\mathbf{H}$ is a lattice point that is the closest to $\mathbf{y}$.

In this section, we summarize the two typical sphere coding algorithms for MIMO detection, namely the VB algorithm and the SE algorithm. Both algorithms are ML lattice decoding algorithms, which try to enumerate the lattice points inside a sphere,

[1][Online]. Available: http://www.altera.com/technology/embedded/emb-index.html

[2][Online]. Available: http://www.xilinx.com/ise/embedded_design_prod/index.htm

and find the best lattice point with minimum distance to the received point. The main difference between these two algorithms is the investigating order inside the lattice structure. The VB algorithm searches from the lower bound to the upper bound in each search layer and examines all possible lattice points falling into a certain sphere in the lattice structure with an initial radius $\sqrt{C}$. Meanwhile, the SE algorithm spreads out from a nearby lattice point of the received signal and terminates once the total distance is greater than the best distance and the search procedure reaches the bottom layer. No initial radius is needed in the SE algorithm, and it is not required to upgrade the upper and lower bounds of each layer using the time consuming square root functions as needed in the VB algorithm. The detail procedures for each of the sphere decoding algorithms are presented as follows.

A. VB Decoding Algorithm

The direct approach to find the closest lattice point is to enumerate all lattice points falling inside a sphere centered at the received point so as to identify the closest lattice point in the Euclidean metric. The VB decoding algorithm is developed based on the aforementioned principle. Basis reduction can be performed on the lattice generation matrix $\mathbf{H}$ to reduce the complexity of the decoding procedure. In this case, Cholesky factorization is applied to the Gram matrix $\mathbf{G} = \mathbf{H}\mathbf{H}^T$ and it yields $\mathbf{G} = \mathbf{W}^T\mathbf{W}$, where $\mathbf{W}$ is an upper triangular matrix. The closest lattice point search problem is formulated as

$$\hat{\mathbf{u}} = \arg \min_{\mathbf{u}\in\Omega^{M_T}} \|(\rho - \mathbf{u})\mathbf{W}^2\|^2 \tag{3}$$

where vector $\rho = \mathbf{y}\mathbf{H}^{-1}$ is the least-mean-square (LMS) solution of (1). The predefined parameter C denotes the squared radius of an M-dimensional sphere centered at the receiving vector. So the closest lattice point u must satisfy

$$d(\rho, \mathbf{u}) = \sum_{j=1}^{M} \left(\sum_{l=j}^{M} \mathbf{W}_{j,l}(\rho_j - u_j) \right)^2 \leq C. \tag{4}$$

Starting from the bottom row of matrix $\mathbf{W}$ and working backwards, the upper and lower bound of the examining lattice point can be determined from (4). We use u_k to represent the index of the examined layer and L_k to denote the upper bound of u_k. The closest lattice point search starts from the bottom layer to the top layer and scans each lattice index from the lower bound to the upper bound. When the algorithm reaches the top layer without violating the bound constraint, a valid lattice point is found. Then the new distance d_{new} between the valid lattice point and the received vector is calculated and compared with the current best distance d_{best}. A closer lattice point to the received point is found if d_{new} is less than d_{best}. This lattice point is saved and the search radius is upgraded as d_{new}. The process iterates until all of the lattice points within the sphere are examined. More details about the theory of the decoding algorithm can be founded in [11].

The step-by-step procedures of the *VB algorithm* are listed in the following.

1) *Preprocessing*: Transform $\mathbf{H}$ into an upper triangular matrix $\mathbf{W}$ by Cholesky decomposition algorithm [10]. Initialize the sphere radius by an adaptive method [11], set dimension index $k = N$ and $d_{\text{best}} = C$, and find the upper bound L_k and index u_k.
2) *Finite-State Machine (FSM)*: Upgrade $u_k = u_k + 1$. If ($u_k < L_k$ and $k > 1$), then go to State A; If ($u_k < L_k$ and $k = 1$), then go to State B; If $(u_k \geq L_k)$, then go to State C.
3) *State A*: Expand the search into $(k-1)$ sublayer, find the parameters used to upgrade u_k and L_k, and go to State D.
4) *State B*: Compute d_{new}. If $d_{\text{new}} < d_{\text{best}}$, record the currently best distance and the best lattice point, set $k = N$, and go to State D. If $d_{\text{new}} \geq d_{\text{best}}$, then go to FSM.
5) *State C*: If $k = N$, stop the algorithm. Otherwise, move the search one layer up $k = k + 1$ and go to FSM.
6) *State D*: Upgrade u_k and L_k that involve square root computations, and then go to FSM.

B. SE Decoding Algorithm

Instead of examining the lattice points within a sphere, the SE algorithm searches the closest lattice point in the nearest hyperplane. So no initial radius is required in the SE algorithm. The search of the closest lattice point starts from the Babai point [23], and spreads out within the distance between the Babai point and the received vector $\mathbf{y}$. The lattice generation matrix $\mathbf{H}$ is factorized into a lower triangular matrix $\mathbf{R}$ and an orthonormal matrix $\mathbf{Q}$ using KZ reduction or LLL reduction [24], where $\mathbf{H} = \mathbf{R}^{-1}\mathbf{Q}$. The closest lattice point problem is formulated as

$$\hat{\mathbf{u}} = \arg \min_{\mathbf{u}\in\Omega^{M_T}} \|\mathbf{y}\mathbf{Q}^{\mathbf{T}} - \mathbf{u}\mathbf{R}^{-\mathbf{1}}\|^2. \tag{5}$$

The index u_k is calculated and examined in two steps as in (6) and (7)

$$\mathbf{e_k} = \mathbf{y}\mathbf{Q}^{\mathbf{T}}\mathbf{R} \tag{6}$$

$$u_k = \{[e_{kk}], [e_{kk}] \pm 1, [e_{kk}] \pm 2, \ldots\} \tag{7}$$

where the round function $[z]$ finds the closest integer to $z \in \Re$. The orthogonal distance d_k to the current k-dimensional layer can be found as

$$d_k = (e_{kk} - u_k)/r_{kk} \tag{8}$$

where r_{kk} is the diagonal element of $\mathbf{R}$. Thus, the partial Euclidean distance (PED) up to the k-dimensional layer is calculated as

$$\text{PED}_k = \sum_{i=k}^{M} d_i^2. \tag{9}$$

If the PED_k is less than the current best distance d_{best}, it is stored and the search procedure expands to $(k-1)$ dimensional sublayer with an update

$$e_{k-1,i} = e_{k,i} - d_k r_{k,i}, \qquad \text{where } i = 1, 2, \ldots, k-1. \tag{10}$$

Conversely, if the PED_k is greater than d_{best}, the searching procedure steps back by one dimensional layer, followed by an update of the examining index in zig-zag order as in (7), which leads to a nondecreasing distance from the examining index to the current layer. The procedure exits when the search moves down to the bottom layer without finding a shorter distance. Therefore, the best lattice point found so far becomes the output.

The step-by-step procedures of the *SE algorithm* are illustrated in the following.

1) *Preprocessing:* Perform $\mathbf{QR}$ factorization on channel matrix $\mathbf{H}$ and inversion of the triangular matrix $\mathbf{R}$. Initialize dimensional index $k = M$, find the bounded index u_k and the distance d_k from $\mathbf{y}$ to the current sublayer M, and set $d_{\mathrm{best}} = \infty$.
2) *FSM*: Updates PED_kusing d_k; let $d_{\mathrm{new}} = \mathrm{PED}_k$:
   a) if $d_{\mathrm{new}} < d_{\mathrm{best}}$ and $k > 1$, go to State A;
   b) if $d_{\mathrm{new}} < d_{\mathrm{best}}$ and $k = 1$, go to State B;
   c) if $d_{\mathrm{new}} \geq d_{\mathrm{best}}$, go to State C.
3) *State A:* Expand the search into $k - 1$ sublayer, find bounded u_k and distance d_k, and go to FSM.
4) *State B:* Record current best distance $d_{\mathrm{best}} = d_{\mathrm{new}}$ and the lattice point $\hat{\mathbf{u}}$; Set $k = 2$, find bounded u_k and distance d_k, and go to FSM
5) *State C:* Exit if $k = M_T$. Otherwise, move the search into $k-1$ sublayer, find bounded u_k and distance d_k, and go to FSM.

Because there is no bound constraint in the SE algorithm, it does not need to calculate and to update the bound in each layer, thus the time-consuming square root operations are avoided. Second, the chance of early identifying the correct layer is maximized using the nondecreasing order of investigation. Furthermore, the lattice point search starts from the Babai point, so no initial radius is required. Based on numerical results, it is claimed that the SE algorithm is in average about 2 to 8 times faster [13] than the VB algorithm.

III. SoC Architecture on FPGA

This section presents a general system-on-chip architecture for sphere decoding algorithms. In order to improve the efficiency of the closest point lattice search, basis reduction is generally performed to transform the lattice generation matrix into an upper or lower triangular matrix, as shown in both VB and SE algorithms. This process is called preprocessing, which involves matrix factorizations such as Cholesky or QR decomposition and matrix inversions. These complex matrix manipulations are difficult to be mapped directly into digital circuits. Fortunately, the preprocessing stage of the sphere decoding algorithms does not need to be executed frequently. Generally, we can assume that the channel matrix $\mathbf{H}$ is static during the transmission of one frame length of data. Therefore, the preprocessing of the lattice generation matrix only needs to be executed once for a period of time (typically 10 ms). A standard embedded processor on the latest FPGA running about 100 MHz is able to complete the preprocessing task gracefully within such a period of time. In our FPGA-based SoC architecture shown in Fig. 1, we dedicate a soft core embedded processor for the preprocessing task.

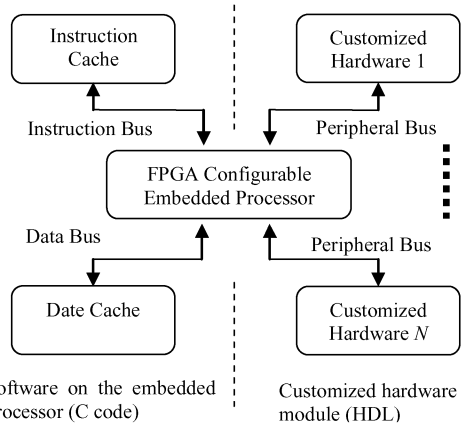

Fig. 1. FPGA-based SoC architecture for sphere decoder implementations.

Although the preprocessing only performs once per frame length of data, the actual closest lattice point search procedure needs to be executed for every received signal vector. The decoding rate of the sphere decoder is determined by the average completion time of the search procedure. As shown in Fig. 1, the SoC architecture includes an embedded processor, customized hardware module(s), and a shared peripheral bus between them for data communication. Subsequently, hardware/software codesign technique is applied to partition the decoding algorithm into each of the processing units. A straightforward coarse-grain partition yields the implementation of preprocessing on an embedded processor and the search procedures on customized hardware units. The preprocessing results (the upper or low triangular matrix) are transferred from the processor to the customized hardware units periodically. In practice, the SoC architecture can achieve considerably faster processing speed than a general DSP because of the VLSI implementation of the lattice search procedures. Furthermore, the FPGA-based SoC architecture provides more flexibility and programmability than an ASIC implementation. The number of customized hardware units can be changed and each of them can be reconfigured at run-time. The software implementation of preprocessing greatly reduces the design complexity and silicon area and the preprocessing algorithm can be conveniently updated if necessary.

Fig. 1 shows the SoC architecture based on the Xilinx Virtex-II Pro FPGA with an embedded MicroBlaze (MB) soft core processor, which is a 32-bit RISC processor implemented using general logic primitives.[2] The on-chip peripheral bus (OPB) is chosen as the interface between the MicroBlaze and customized hardware cores. The OPB is synchronous to the MicroBlaze processor and can achieve a high data transfer rate. The maximum number of the customized modules that can be attached to the soft processor is determined by the size of the hardware cores and the available resources on an FPGA.

IV. Exploration of Parallel Structures

This section presents the parallel structures of the sphere decoding algorithms. Three levels of parallelism are explored to accelerate the decoding rate of the hardware implementations.

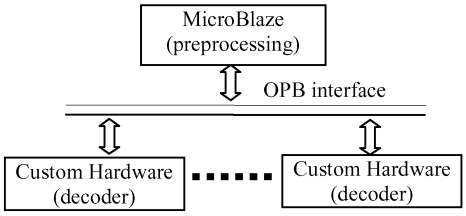

Fig. 2. Diagram of the processor-level parallelism for sphere decoder.

A. Processor Level Parallelism

The hardware/software codesign architecture for both VB and SE algorithms are identical and illustrated in Fig. 2. The preprocessing part, including matrix factorization and inversion, is programmed on the MicroBlaze soft processor for both algorithms. The lattice search procedures are implemented in customized hardware modules to speed up the decoding throughput. The OPB interface is used to transfer data between them. The processor level parallelism refers to the concurrent execution of the preprocessing and decoding blocks.

Furthermore, multiple decoding blocks can be mapped into the FPGA if the hardware resources are available. This is feasible because all signals received within the same frame length are decoded using the same triangular matrix supplied by the same preprocessing unit. Thus multiple received signals can be decoded simultaneously. Given t_p as the execution time of preprocessing and t_d as the time to decode a single received vector, the system architecture with a single embedded processor and N customized hardware units can achieve a speed up of

$$S_k = \frac{(t_p + t_d L)}{\max(t_p, t_d L/N)} \tag{11}$$

where L is the number of received vector per frame length. Based on the FPGA resource usage shown in Section V, the XC2VP30 FPGA can accommodate three hardware decoders for the VB algorithm and four decoders for the SE algorithm in parallel to accelerate the decoding speed.

B. Complex Constellation Parallelism

If the transmitted signal is modulated using complex quadrature amplitude modulation (QAM), the channel matrix $\mathbf{H}$ becomes an $M \times N$ complex matrix, and the received signal $\mathbf{y}$ and the transmitted signal $\mathbf{u}$ are $1 \times N$ and $1 \times M$ complex vectors respectively. In this section, we only consider the case for the same number of transmit and receive antennas ($M = N$).

Generally, the M-dimensional complex problem can be solved via decomposing the complex matrix into an equivalent $2M$-dimensional real matrix as

$$\begin{bmatrix} \Re\{\mathbf{y}\} \\ \Im\{\mathbf{y}\} \end{bmatrix}^T = \begin{bmatrix} \Re\{\mathbf{u}\} \\ \Im\{\mathbf{u}\} \end{bmatrix}^T \begin{bmatrix} \Re\{\mathbf{H}\} & -\Im\{\mathbf{H}\} \\ \Im\{\mathbf{H}\} & \Re\{\mathbf{H}\} \end{bmatrix} + \begin{bmatrix} \Re\{\mathbf{n}\} \\ \Im\{\mathbf{n}\} \end{bmatrix}^T. \tag{12}$$

The inputs of the sphere decoder become a $2M$-dimensional channel matrix and a $1 \times 2M$ received signal vector. Doubling the dimension of the system enlarges the search space, thus significantly increasing the number of iterations during the closest lattice point search procedure. Fig. 3 shows the number of states visited as the dimension of the channel matrix grows for the SE algorithm for E_b/N_0 of 10 and 20 dB, respectively. Number of states visited is an important factor of decoding rate. The decoding rate will be considerably degraded if the dimension of the search space is doubled. In addition, decomposing into a $2M$-dimensional real lattice ignores the orthogonal feature of real and imaginary parts of a complex system and the symmetry of the QAM constellation, thus reducing the possibility of parallel implementation on FPGA. Consequently, the conventional $2M$-dimensional real lattice method is not suitable for hardware implementation.

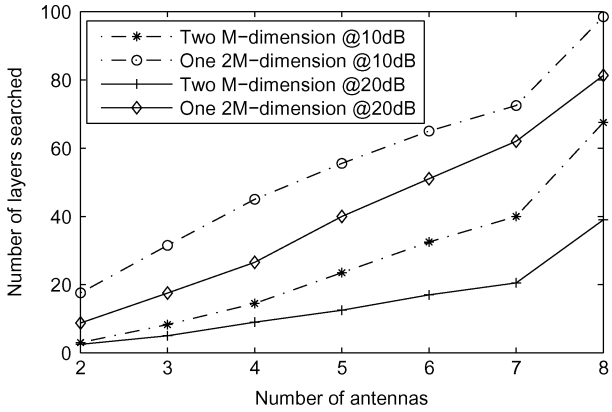

Fig. 3. Number of states visited versus number of antennas.

An efficient complex channel matrix transformation is developed to explore the orthogonal feature of the real and imaginary parts of a complex problem. (12) can be rewritten in (13) and (14) using linear transformation as

$$\mathbf{y}_1 = \Re\{\mathbf{u}\}\mathbf{S} + \mathbf{n}_1 \tag{13}$$

$$\mathbf{y}_2 = \Im\{\mathbf{u}\}\mathbf{S} + \mathbf{n}_2 \tag{14}$$

where $\mathbf{S}$ is given by $\Re\{\mathbf{H}\} + \Im\{\mathbf{H}\}\Re(\mathbf{H})^{-1}\Im(\mathbf{H})$, and $\mathbf{y}_1$ and $\mathbf{y}_2$ can be computed similarly. All lattice points remain the same after the linear transformation. It is worth mentioning that the linear transformation above creates a new lattice generation matrix $\mathbf{S}$ and new received signal vectors $\mathbf{y}_1$ and $\mathbf{y}_2$. It also implies that the property of the new noise vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ may be different from those of the original noise vectors $\Re(\mathbf{n})$ and $\Im(\mathbf{n})$. Compared to the simulation results of the direct search in (12), separate decoding of real and imaginary parts may result small differences in BER performance at high SNR due to the effect of the noise property changes. This will be further explained in the experimental study in Section V.

It is clearly indicated in (13) and (14) that the $2M$-dimensional complex problem has been decomposed into two independent M-dimensional real problems. This transformation is well suited for FPGA-based implementation because the real and the imaginary parts of the received signal can be decoded in parallel. Since the number of search iterations for each part remains the same as an M-dimensional real system, the decoding rate can be improved for an MIMO system with complex quadrature amplitude modulation (QAM) constellation using the linear transformation method.

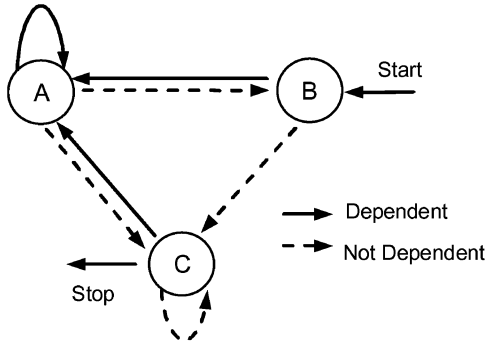

Fig. 4. Dependency graph of the SE algorithm.

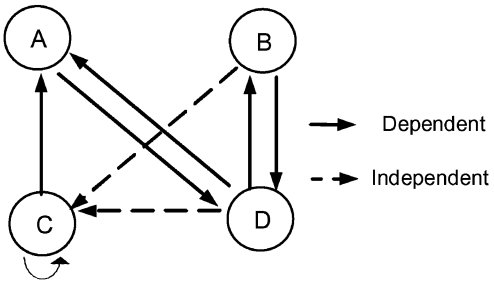

Fig. 5. Dependency graph of the VB algorithm.

C. State Level Parallelism

To further improve the decoding speed, we explore the intrinsic parallel structure for the sphere decoding algorithm. Due to the irregular structure of the lattice, the lattice search procedures are complicated and not explicit for mapping into concurrent execution threads. To facilitate the parallel implementation, each sphere decoding algorithm is partitioned into a number of states as listed in Section II, with an FSM controlling the transition among them. Subsequently, the data dependency analysis can be performed on the state transition diagram.

1) Data Dependency Analysis: There are three states involved in the decoding procedure as described in the step-by-step procedures for the SE algorithm. Fig. 4 shows the data dependency among all possible state transitions. State A is dependent on both states B and C if the search procedure switches to A from B or C, because the orthogonal distance calculated in B or C is used to upgrade the layer index u_k in State A. Similarly, State A is self dependent. State B has no data dependency on A because these two states always work on different search dimensions if B follows A during the search. For the same reason, State C is not dependent on any other state. Based on the data dependency analysis, the possibility of the parallelism among these three states is found as: $\mathrm{A}||\mathrm{B}$, $\mathrm{A}||\mathrm{C}$, $\mathrm{B}||\mathrm{C}$, and $\mathrm{C}||\mathrm{C}$. These conditions are used to implement the state level parallelism.

Similarly to the SE algorithm, Fig. 5 gives all possible state transitions and the data dependency of the VB algorithm. It shows that only state C can be executed simultaneously with states B–D. Therefore, the parallel states are $\mathrm{B}||\mathrm{C}$, $\mathrm{C}||\mathrm{C}$, and $\mathrm{D}||\mathrm{C}$.

2) Implementation of Parallel Structure: Based on the data dependency analysis, parallel structures are developed to further accelerate the closest lattice point search procedure. As an example, Fig. 6(a) shows a typical search process in the SE decoding algorithm, where the state transition is executed sequentially. In the state-level parallel structure, four hardware modules are created, with one for each state and a duplicated module for state C because two continuous C states can be executed concurrently. These four modules are executed simultaneously to speed up the search procedure as shown in Fig. 6(b). All possible states after the next state are enabled if they are allowed to execute in parallel. For example, if state A is the next state, then modules A, B, and C are all enabled because the search procedure could jump to either B or C after state A, making A execute in parallel with both B and C. At the end of the processing in state A, the results from either B or C will be accepted based on the state transition condition. The next group of the selected state is then enabled and the search procedure continues. For a typical search process shown in Fig. 6, seven cycles are required to complete the computation if executed sequentially as demonstrated in Fig. 6(a), but the same process only takes four cycles in the parallel structure as shown in Fig. 6(b).

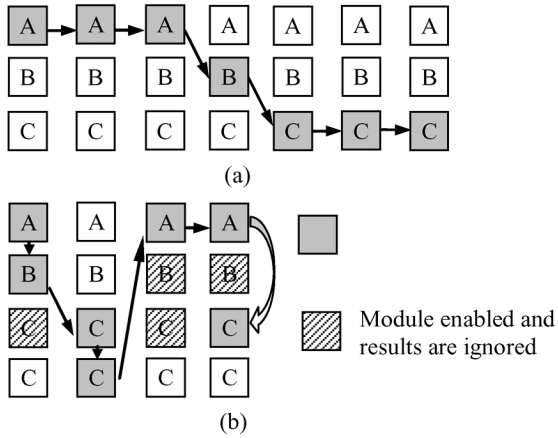

Fig. 6. Example of the closest point search process: (a) sequential implementation and (b) parallel implementation.

The state parallel structure for the VB algorithm is similar to that of the SE algorithm. There are five state modules (A, B, C, C, and D) with double instantiations of state C. Concurrent execution becomes possible providing that data dependency does not exist among the selected states. A moderate performance gain can be achieved based on the state level parallel implementations.

V. Experimental Results

Both SE and VB algorithms are developed in EDK[3] and prototyped on Xilinx Virtex-II Pro developing board with an XC2VP30 FPGA.[3] The resource usage and the maximum frequency of the decoder are compared between these two algorithms. The BER performance is obtained by the simulation of fixed-point HDL models. Test vectors are used to verify the design implementation on the FPGA-based hardware prototype. For performance comparisons, both algorithms are also implemented on a DSP processor.

[3][Online]. Available: http://www.xilinx.com/univ/xupv2p.html

TABLE I
EMBEDDDED PROCESSOR TIMING PERFORMANCE OF PREPROCESSING FOR A 4×4 MIMO SYSTEM WITH 16-QAM MODULATION

	SE Algorithm	VB Algorithm
Device	MicroBlaze soft processor	
Frequency	100 MHz	
Decomposition	36,277 cycles (QR)	4,621 cycles (Cholesky)
Inversion	12,287 cycles	
Transpose	1,129 cycles	
Total Time	53,422 cycle (0.534 ms)	32,548 cycle (0.325 ms)

TABLE II
FPGA RESOURCE USAGE OF THE LATTICE DECODING MODULES FOR A 4×4 MIMO SYSTEM WITH 16-QAM MODULATION

	SE Algorithm	VB Algorithm
Target FPGA	XC2VP30	
No. of Slices	3880 out of 13696	5614 out of 13696
No. of bonded IOBs	205 out of 556	227 out ot 556
No. of 18×18 Multipliers	36 out of 136	58 out of 136
Max Frequency	251 MHz	257 MHz

A. Soft Core Processor Performance

As described in Section III, the preprocessing part of the sphere decoding algorithms is partitioned to the MicroBlaze soft core processor on FPGA. The software code for the preprocessing includes QR or Cholesky factorization, matrix inversion and transpose, and other operations. From the synthesis results, the MicroBlaze processor can operate at 100 MHz when prototyped on the FPGA. The execution time of the preprocessing task for the 4×4 MIMO system is shown in Table I.

The results demonstrate that the preprocessing part of the sphere decoders can be completed within 0.534 ms, which is much less than the typical transmission of a frame length of data (about 10 ms) in a MIMO system. A system with eight transmit and eight receive antennas has also been examined. The results show that the preprocessing task can be completed within 2.7 ms for both SE and VB algorithms. These results prove the feasibility of the processor-level parallel architecture with an embedded processor.

B. Hardware Resource Utilization

Based on the SoC architecture described in Section III, the lattice point search procedures are mapped as customized hardware units on the FPGA to speedup the decoding rate. For the 4×4 MIMO system, the FPGA resource usages and the post place-and-route (PAR) frequencies are compared between the SE and VB algorithms as shown in Table II. The results show that the VB-based decoder uses more embedded multipliers and more FPGA slices than the SE decoder because the VB algorithm involves more complicated computations.

C. Decoding Rate Performance

The data rate for the MIMO sphere decoder with M transmit and M receive antennas is determined by

$$R = \frac{M b_{\dim} f}{c_{\text{avg}} n_{\text{state}}} \tag{15}$$

TABLE III
DATA RATES FOR THE FPGA-BASED SPHERE DECODING ALGORITHM IMPLEMENTATIONS

	SE Algorithm	VB Algorithm
Platform	XC2VP30 FPGA	
Max Frequency	251 MHz	257 MHz
Cycles/State	8.5 cycles	7.7 cycles
No. of States	5.8	14.5
Bits/Dimension	4	
Dimension	4	
Decoding Rate	81.5 Mbps	36.8 Mbps

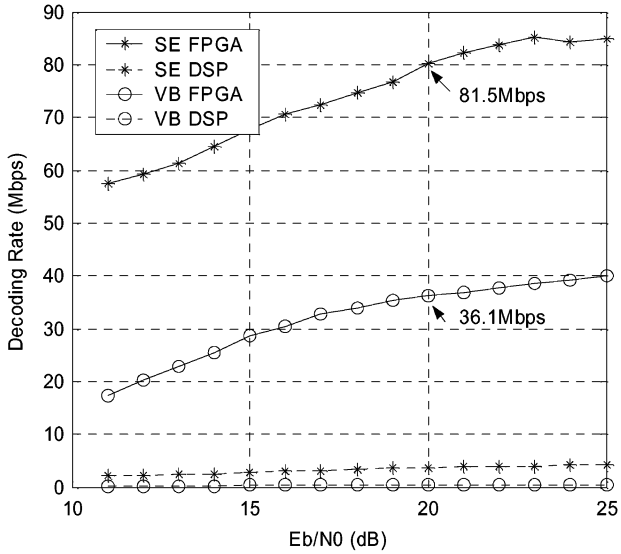

Fig. 7. Decoding rate comparison for VB and SE sphere decoders on FPGA and DSP implementations for a 4×4 MIMO system with 16-QAM modulation.

where f is the system frequency of the circuits in megahertz, bits per dimension $b_{\dim}$ is determined by the modulation scheme, and n_{state} is the number of state visits required to decode a receive vector at certain SNR. In our design with state level parallelism, two or more states executed at the same time are counted as one state visit. c_{avg} is the average number of clock cycles per state visit that is calculated as

$$c_{\text{avg}} = \sum (P_{i,\text{state}} c_{i,\text{state}}) \tag{16}$$

where P_i, state denotes the statistic percentage for visiting state i, which can be obtained from high-level simulation. c_i, state is the number of clock cycles used for state i captured from the FPGA circuit simulation. The simulation results show that the average processing time per state visit is 8.5 cycles for the SE algorithm and 7.7 cycles for the VB algorithm. Based on these parameters, the decoding rate of the sphere decoder for the 4×4 system with 16-QAM modulation at 20-dB SNR is given in Table III. The results show that the SE based decoder is more than 2.21 times faster than the VB based decoder. These results match the theoretical analysis in [13].

These two algorithms are also implemented on DSP in comparison with the FPGA-based decoders. A fixed-point DSP processor TMS320C6201 is chosen as the benchmark platform, which operates at 200 MHz [25]. Fig. 7 gives a data rate comparison between the FPGA and DSP implementations. The results

TABLE IV
DATA RATES FOR THE DSP-BASED SPHERE DECODING ALGORITHM IMPLEMENTATIONS

	SE Algorithm	VB Algorithm
Platform	TMS320C6201	
Frequency	200 MHz	
Cycles/State	94 cycles	474 cycles
No. of States	9.3	18
Bits/Dimension	4	
Dimensions	4	
Decoding Rate	3.66 Mbps	0.38 Mbps

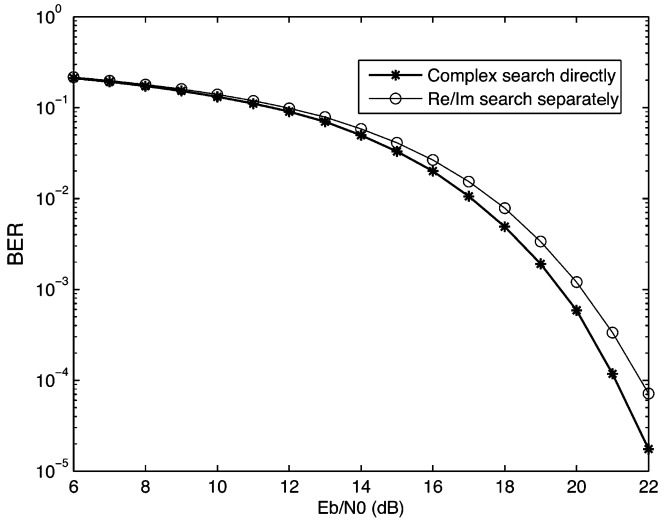

Fig. 8. BER performance comparison of complex search and Re/Im separation for a 4×4 system with 16-QAM modulation using SE algorithm. Separate decoding of real and imaginary parts provides parallel execution by results in a performance degradation of 0.8 dB at high SNR.

show that the DSP-based SE and VB decoders can only reach about 3.66 and 0.38 Mb/s, respectively for a 4×4 system with 16-QAM modulation at 20-dB SNR. Compared to the FPGA design, the key reason for the low decoding rate in DSP is the lack of parallelism support and high instruction overhead. As shown in Table IV, the DSP-based VB decoder takes 474 clock cycles in average for a state visit, which is only 7.7 clock cycles in the corresponding FPGA implementation. The number of state visits of the DSP implementation is significantly larger than that of the FPGA implementation because DSP does not support the state level parallelism. Similarly, the DSP-based decoders can not implement the current execution of preprocessing and decoding. It does not support the parallel processing of real/image parts of decoding algorithms either. As shown in Fig. 7, the data rates of the FPGA-based sphere decoders are about 22 times and 97 times faster than the DSP implementations of SE and VB algorithms, respectively, at the same system and channel settings.

D. BER Performance

The BER performance is evaluated using different lattice generation matrices with Gaussian distribution and a large number of source data at different SNRs with additive white Gaussian noise (AWGN). Fig. 8 shows the BER performance for the hardware implementation of the SE sphere decoder. The BER performance of the VB algorithm is almost identical to the SE algorithm. The FPGA implementations apply the proposed complex transformation method and decode the real and imaginary parts seperately. The software simulations directly conduct the lattice search using complex numbers. The BER performance of the FPGA implementations is then compared to the software simulation. The experimental results in Fig. 8 show that the FPGA-based sphere decoders match the BER performance of the software simulations at low SNR. It is also shown that a small degradation up to 0.8 dB at high SNR for the transformed parallel implementations on the FPGA. As explained in Section IV-B, the small difference of BER performance at high SNR is due to the changes of noise property through the linear transformation.

TABLE V
POWER AND AREA COMPARISON OF THE FPGA AND DSP IMPLEMENTATIONS

	FPGA-SE	FPGA-VB	DSP
Technology	0.13 μm		0.18 μm
(I/O)/Core Voltage	2.5V/1.5V		3.3V/1.8V
Power	1950 mW	1520 mW	1940 mW
Area	25.98 mm^2	37.59 mm^2	95.64 mm^2

E. Power and Area Estimation

The power consumption of the design mapped on FPGA can be estimated using the XPower tool.[4] The estimated power is calculated by summing up the consumed power of all used elements in the design using the device specific capacitance model and the circuit switching activity taken from the VCD files. Since the die size of the XC2VP30 FPGA and silicon area of each slice is not available in the public domain, the area estimation is based on the data provided in [26] for a similar FPGA device XC2V1000, which uses 0.15 μm process. In particular, the area of each tile, which includes 128 slices, 1 embedded multiplier, and 1 RAM block, is 1.141 mm^2. Since the prototyped XC2VP30 device uses a 0.13-μm process, the tile area is scaled by multiplying a factor of $(.13/.15)^2$ to yield an estimated size of 0.857 mm^2. The total area utilization on the FPGA is then computed by the number of slices in use as in Table V.

The power consumption of the TMS320C6201 DSP is based on the typical values provided in [27] for high activity operations (75% high and 25% low). The die size of the DSP device is provided in [28]. The area of the DSP is larger than the FPGA design. One obvious reason is the difference in process technology. The C6201 DSPs use a 0.18-μm process and XC2VP30 FPGAs use a 0.13-μm process. The most important reason is that the area of FPGA is calculated based on the number of slices mapped for a particular algorithm. But the die size of the entire DSP is counted as the silicon area, which is application-independent. In particular, some elements on the DSP device may not actively used for the MIMO decoding algorithm, such as phase-locked loop (PLL) and direct memory access (DMA) controller.

VI. CONCLUSION

The system architectures and implementations of two typical sphere decoding algorithms for MIMO detection are pre-

[4][Online]. Available: http://www.xilinx.com/prs_rls/0153xpowercplds.htm

sented in this paper. Hardware/software codesign technique is applied when partitioning the decoding algorithm on a single FPGA device, with channel matrix preprocessing on an embedded processor and the decoding functions on customized hardware modules. Multiple levels of parallelism are explored in the system design in order to achieve faster throughput. The decoders for a 4×4 MIMO system with 16-QAM modulation are prototyped on a Xilinx XC2VP30 device with a MicroBlaze soft core processor. Compared to the VB algorithm, the SE algorithm has lower circuit complexity, smaller silicon area, and fewer lattice search iterations. The results show that the SE based decoder is more than 2.21 times faster than the VB based decoder and uses 30% fewer FPGA slices. The same algorithms are also implemented on a TMS320C6201 DSP processor to compare the performance. The FPGA prototypes of the SE and VB algorithms show that they support up to 81.5 and 36.1-Mb/s data rates at 20-dB SNR, which are about 22 and 97 times faster than their respective implementations in a DSP. Power consumptions of the FPGA and DSP implementations are comparable, but the areas of the FPGA designs are smaller. The throughput advantage of the FPGA is due to its rich computational resources and the parallel processing using the customized circuits. Future research will further optimize the VLSI implementations to reduce the power and area utilization and to achieve faster system throughput on a single chip.

ACKNOWLEDGMENT

The authors would like to thank Y. Liu (University of Colorado, Boulder) for the generous help on the decoding algorithms from the information theory prospective. They would also like to thank the anonymous reviewers for their careful review and precious comments.

REFERENCES

[1] G. J. Foschini and M. Gans, "On the limits of wireless communication in a fading environment when using multiple antennas," *Wireless Personal Commun.*, vol. 6, pp. 311–335, 1998.

[2] B. M. Hochwald and S. Ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[3] D. Gesbert, M. Shafi, S. Da-shan, P. J. Smith, and A. Naguib, "From theory to practice: An overview of mimo space-time coded wireless systems," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 3, pp. 281–302, Mar. 2003.

[4] "Multiple-input multiple-output in UTRA (Release 7)," 3GPP TR25.876 (V7.0.0), 2006.

[5] *Draft Amendment to Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications: Enhancements for Higher Throughput*, IEEE P802.11n/D1.0, 2006.

[6] *Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems*, IEEE802.16e standard for local and metropolitan area, 2006.

[7] A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications*. New York: Cambridge Univ. Press, 2003.

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

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

[10] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *SIGSAM Bull.*, vol. 15, no. 1, pp. 37–44, 1981.

[11] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1639–1642, May 1999.

[12] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Program.*, vol. 66, no. 2, pp. 181–191, 1994.

[13] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[14] A. Adjoudani, E. C. Beck, A. P. Burg, G. M. Djuknic, T. G. Gvoth, D. Haessig, S. Manji, M. A. Milbrodt, M. Rupp, D. Samardzija, A. B. Siegel, I. Sizer, T. , C. Tran, S. Walker, S. A. Wilkus, and P. W. Wolniansky, "Prototype experience for MIMO blast over third-generation wireless system," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 3, pp. 440–451, Mar. 2003.

[15] M. Guillaud, A. Burg, M. Rupp, E. Beck, and S. Das, "Rapid prototyping design of a 4×4 blast-over-umts system," in *Proc. 35th Asilomar Conf. Signals, Syst. Comput.*, 2001, pp. 1256–1260, vol. 2.

[16] Z. Guo and P. Nilsson, "A VLSI architecture of the soft-output sphere decoder for MIMO systems," in *Proc. IEEE Int. Midwest Symp. Circuits Syst.*, 2005, pp. 1195–1198.

[17] O. Damen, A. Chkeif, and J. C. Belfiore, "Lattice code decoder for space-time codes," *IEEE Commun. Lett.*, vol. 4, no. 5, pp. 161–163, May 2000.

[18] D. Garrett, L. Davis, S. ten Brink, B. Hochwald, and G. Knagge, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE J. Solid-State Circuits*, vol. 39, no. 9, pp. 1544–1552, Sep. 2004.

[19] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "Vlsi implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.

[20] A. DeHon, "The density advantage of configurable computing," *IEEE Comput.*, vol. 33, no. 4, pp. 41–49, Apr. 2000.

[21] R. Hartenstein, "A decade of reconfigurable computing: A visionary retrospective," in *Proc. Int. Conf. Des., Autom. Test Eur.*, 2001, pp. 642–649.

[22] M. Vuletid, L. Pozzi, and P. Ienne, "Seamless hardware-software integration in reconfigurable computing systems," *IEEE Des. Test Comput.*, vol. 22, no. 2, pp. 102–113, Feb. 2005.

[23] W. W. Hager, *Applied Numerical Linear Algebra*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[24] H. Cohen, *A Course in Computational Algebraic Number Theory*. Berlin, Germany: Springer-Verlag, 1993.

[25] Texas Instruments Inc., Dallas, TX, "TMS320C6201 fixed-point digital signal processor," 2004.

[26] K. Compton and S. Hauck, "Automatic design of area-efficient configurable asic cores," *IEEE Trans. Comput.*, vol. 56, no. 5, pp. 662–672, May 2007.

[27] Texas Instruments Inc., Dallas, TX, "TMS320C62X/C67X power consumption summary (SPRA486C)," 2002.

[28] J. Neel, S. Srikanteswara, J. H. Reed, and P. M. Athanas, "A comparative study of the suitability of a custom computing machine and a VLIW DSP for use in 3G applications," in *Proc. IEEE Workshop Signal Process. Syst.*, 2004, pp. 188–193.

Xinming Huang (M'01) received the Ph.D. degree in electrical engineering from Virginia Polytechnic Institute and State University, Blacksburg, VA in 2001.

He is an Assistant Professor with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute (WPI), Worcester, MA. He was a member of Technical Staff with the Wireless Advanced Technology Laboratory, Bell Labs of Lucent Technologies, Whippany, NY, from 2001 to 2003. He was also an Assistant Professor with the Department of Electrical Engineering, University of New Orleans, New Orleans, LA, from 2003 to 2006.

Dr. Huang was a recipient of a DARPA/MTO Young Faculty Award in 2007, the IBM Faculty Fellowship Award in 2004, and the Central Bell Labs Annual Excellence and Teamwork Award in 2002. His research interests include the areas of circuit design and system architecture, with an emphasis on reconfigurable computing, wireless communications, and networked embedded systems.

Cao Liang (S'06) received the B.S. degree in communication and information engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2001, and the M.S. degree in electrical engineering from the University of New Orleans, New Orleans, LA, in 2006. He is currently pursuing the Ph.D. degree in electrical and computer engineering from Worcester Polytechnic Institute, Worcester, MA.

His research interests include high performance computing architecture, VLSI design, and routing in collaborative computing networks.

Jing Ma (M'02) received the B.S. degree from Northwestern Polytechnic University, Xi'an, China, in 1993, the M.S. degree from the National University of Singapore, Singapore, in 1998, and the Ph.D. degree from the Virginia Polytechnic Institute and State University, Blacksburg, in 2002, all in electrical engineering.

She is currently with The MathWords Inc., Natick, MA. She was an Assistant Professor with the Department of Electrical Engineering, University of New Orleans, New Orleans, LA, from 2003 to 2006. Her research interests include reconfigurable computing, wireless sensor networks, and computer-aided design.