Algorithm and Implementation of the K-Best Sphere Decoding for MIMO Detection

Zhan Guo and Peter Nilsson, *Member, IEEE*

***Abstract*—K-best Schnorr–Euchner (KSE) decoding algorithm is proposed in this paper to approach near-maximum-likelihood (ML) performance for multiple-input–multiple-output (MIMO) detection. As a low complexity MIMO decoding algorithm, the KSE is shown to be suitable for very large scale integration (VLSI) implementations and be capable of supporting soft outputs. Modified KSE (MKSE) decoding algorithm is further proposed to improve the performance of the soft-output KSE with minor modifications. Moreover, a VLSI architecture is proposed for both algorithms. There are several low complexity and low-power features incorporated in the proposed algorithms and the VLSI architecture. The proposed hard-output KSE decoder and the soft-output MKSE decoder is implemented for 4×4 16-quadrature amplitude modulation (QAM) MIMO detection in a 0.35-μm and a 0.13-μm CMOS technology, respectively. The implemented hard-output KSE chip core is 5.76 mm^2 with 91 K gates. The KSE decoding throughput is up to 53.3 Mb/s with a core power consumption of 626 mW at 100 MHz clock frequency and 2.8 V supply. The implemented soft-output MKSE chip can achieve a decoding throughput of more than 100 Mb/s with a 0.56 mm^2 core area and 97 K gates. The implementation results show that it is feasible to achieve near-ML performance and high detection throughput for a 4×4 16-QAM MIMO system using the proposed algorithms and the VLSI architecture with reasonable complexity.**

***Index Terms*—Multiple-input–multiple-output (MIMO), Schnorr–Euchner algorithm, sphere decoder, very large scale integration (VLSI).**

I. Introduction

IT IS KNOWN that an extraordinary spectral efficiency near Shannon bound is able to be achieved in multiple-input–multiple-output (MIMO) systems [1]. MIMO is one of the hot technologies for fourth-generation (4G) because it can increase the capacity (coverage or link quality in other sense) at no cost in frequency spectrum. MIMO is becoming a key part in almost every new wireless standard, such as HSDPA, 802.11n, 802.16e and 802.20. To exploit the potentials of MIMO, one of the challenges is the very high computing power that is required at the receiver end. This exceeds the capabilities of the typical chips that are currently being employed in the wireless communication community.

The optimal maximum-likelihood (ML) decoders, using exhaustive search, have been shown to be feasible for 4×4 MIMO systems with quadrature phase-shift keying (QPSK) modulation [2], [3]. To simplify the exponentially complex search problem in ML decoders for MIMO systems with higher modulation constellations, lattice (sphere) decoders are shown in [4] and [5] to be capable of achieving near-ML performance with reasonable complexity. Moreover, the lattice decoders can be extended to support soft decision outputs and, hence, be used in an iterative MIMO receiver [6]. The lattice decoding algorithms have two kinds of implementation strategies, i.e., Fincke–Pohst strategy [4], [7], [8] and Schnorr–Euchner strategy [9]–[11]. To avoid confusion in this paper, the lattice decoder using the Fincke–Pohst strategy is called SD (sphere decoder), and the lattice decoder using the Schnorr–Euchner strategy is called SE.

In this paper, we mainly focus on very large scale integration (VLSI) implementation aspects of lattice decoding algorithms for MIMO detection. An early VLSI implementation of the SD algorithm is presented in [12] for a 4×4 16-quadrature amplitude modulation (QAM) MIMO system, in which a breadth-first search method is employed and coined as K-best SD algorithm. The drawback with the VLSI architecture of [12] is that the decoding throughput is limited to 10 Mb/s at a 100 MHz clock frequency. Inspired by [12], the K-best SE (KSE) algorithm, proposed in this paper, is shown to be more suitable for VLSI implementations. Furthermore, a modified KSE (MKSE) algorithm is proposed to support soft outputs with only minor modifications to the KSE.

Sections II and III of this paper briefly describe the lattice model and the lattice decoding algorithms for MIMO systems. Section IV proposes the KSE algorithm supporting hard outputs. Section V further extends the KSE algorithm to support soft outputs and proposes the MKSE algorithm to improve its performance. Section VI provides simulation results of both algorithms. Section VII proposes a VLSI architecture for both algorithms. Section VIII presents VLSI implementation results of both algorithms. The conclusion is given in Section IX.

II. Lattice Model of MIMO Systems

Consider a symbol synchronized MIMO system with M transmit antennas and N receive antennas. The baseband equivalent model for the MIMO channel is

$$\tilde{\mathbf{x}} = \tilde{\mathbf{H}}\tilde{\mathbf{s}} + \tilde{\mathbf{v}} \tag{1}$$

where $\tilde{\mathbf{s}} = [\tilde{s}_1, \tilde{s}_2, \ldots \tilde{s}_M]^T$ is the transmitted symbol vector, in which each component is independently drawn from a complex constellation such as QAM, $\tilde{\mathbf{x}} = [\tilde{x}_1, \tilde{x}_2, \ldots \tilde{x}_N]^T$ is the received symbol vector, and $\tilde{\mathbf{v}} = [\tilde{v}_1, \tilde{v}_2, \ldots \tilde{v}_N]^T$ is an independent identically distributed (i.i.d.) complex zero-mean Gaussian

Manuscript received January 11, 2005; revised April 14, 2005. This work was supported by the INTELECT Program within the Foundation of Strategy Research, Sweden (SSF), in part by the Competence Center for Circuit Design (CCCD), Department of Electroscience, Lund University, and in part by the EU Sixth Framework IST-MAGNET Program.

The authors are with the Department of Electroscience, Lund University, SE-221 00 Lund, Sweden (e-mail: zhan.guo@es.lth.se; peter.nilsson@ es.lth.se).

Digital Object Identifier 10.1109/JSAC.2005.862402

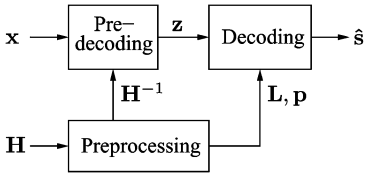


Fig. 1. Block diagram of a typical lattice decoder for MIMO detection.

noise vector with variance σ^2 per dimension. Moreover, $\tilde{\mathbf{H}}$ denotes the $N \times M$ channel matrix, whose elements $\tilde{h}_{ij}$ represent the complex transfer functions from the jth transmit antenna to the ith receive antenna, and are all i.i.d. complex zero-mean Gaussian with variance 0.5 per dimension. The channel matrix is assumed to be perfectly known to the receiver, and $M = N$ is assumed in the sequel.

The complex matrix (1) can be transformed to its real matrix representation $\mathbf{x} = \mathbf{H}\mathbf{s} + \mathbf{v}$, i.e.,

$$\begin{bmatrix} \Re(\tilde{\mathbf{x}}) \\ \Im(\tilde{\mathbf{x}}) \end{bmatrix} = \begin{bmatrix} \Re(\tilde{\mathbf{H}}) & -\Im(\tilde{\mathbf{H}}) \\ \Im(\tilde{\mathbf{H}}) & \Re(\tilde{\mathbf{H}}) \end{bmatrix} \begin{bmatrix} \Re(\tilde{\mathbf{s}}) \\ \Im(\tilde{\mathbf{s}}) \end{bmatrix} + \begin{bmatrix} \Re(\tilde{\mathbf{v}}) \\ \Im(\tilde{\mathbf{v}}) \end{bmatrix} \tag{2}$$

where $\Re(\cdot)$ and $\Im(\cdot)$ denote the real and imaginary part of $(\cdot)$, respectively. The elements of $\tilde{\mathbf{H}}$ are assumed to be i.i.d. complex Gaussian. The set $\{\mathbf{H}\mathbf{s}\}$ can be considered as the lattice $\Lambda(\mathbf{H})$ generated by $\mathbf{H}$ [10]. The columns of $\mathbf{H}$ are called basis vectors for $\Lambda(\mathbf{H})$, while the transmitted vector $\mathbf{s}$ acts as the coordinates of a lattice point. If the received vector $\mathbf{x}$ is considered as a perturbed lattice point due to the Gaussian noise $\mathbf{v}$, the objective of the MIMO detection is to find its closest lattice point $\hat{\mathbf{s}}$ for a given lattice $\Lambda(\mathbf{H})$, i.e.,

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s} \in \Omega^{2M}} \|\mathbf{x} - \mathbf{H}\mathbf{s}\|^2 \tag{3}$$

where Ω is the set of real entries in the constellation, e.g., $\Omega = \{-3, -1, 1, 3\}$ in the case of 16-QAM. Exploiting the lattice properties of the MIMO system model, the optimal ML decoder for MIMO systems can be simplified to a lower complexity lattice decoder with near-ML performance.

III. Lattice Decoders for Mimo Detection

A typical lattice decoder for MIMO detection consists of a preprocessing unit, a predecoding unit and a decoding unit, as shown in Fig. 1. The preprocessing unit takes the estimated channel matrix $\mathbf{H}$, and generates its inverse $\mathbf{H}^{-1}$, a triangular matrix $\mathbf{L}$, and a correspondingly optimal ordering $\mathbf{p}$ if needed. The task of the predecoding unit is simply to generate a zero-forcing (ZF) point $\mathbf{z} = (\mathbf{H}^{-1}\mathbf{x})^T$ as an initial estimate for the decoding unit. The computational complexity of the predecoding unit is omitted in the following complexity analysis for all the lattice decoders. The differences among various lattice decoders for MIMO detection depend largely on the design of the decoding unit.

In a lattice decoder, an $n = 2M$-dimensional lattice is decomposed into n sublattices. Let k be the dimension of the sublattice that is currently being investigated, and y the orthogonal distance between two points in the adjacent sublattices. The objective of the decoder is to search for the lowest possible squared distance $bestdist$ between $(k = n)$-dimensional and $(k = 1)$-dimensional sublattice [10].

In theory, the BER performance of the SE and the SD algorithms should be the same for MIMO detection, since the difference between the SE and the SD lies in the searching order among the sublattices [10]. According to the searching direction instead, the lattice decoders can be divided into two types, the depth-first type with variable throughput and the breadth-first type with fixed throughput.

A. Depth-First Algorithms

The depth-first algorithm searches for the $bestdist$ in both forward and backward directions among the sublattices. The currently lowest distance $newdist$ is first searched in the forward direction of $k = n, n-1, \ldots, 1$, compared with an initially lowest distance criterion $bestdist$ of infinity (or sphere radius C). Each time the $(k = 1)$-dimensional sublattice is reached, the $bestdist$ is replaced with the $newdist$. Then, the algorithm moves backward in the direction of $k = 1, 2, \ldots, n$. As soon as the $newdist$ is smaller than the current criterion of $bestdist$, the algorithm moves forward to $k = 1$ again. From a sequential decoding point of view [13], the depth-first lattice decoding algorithm actually uses depth-first and metric-first mixed searching scheme to find the closest lattice point to the received symbol. Consequently, this type of lattice decoding algorithms for MIMO detection can also be called sequential algorithms, e.g., sequential SD [4], [6], [14] and sequential SE [11], [15].

B. Breadth-First Algorithms

Instead of the metric-first and depth-first mixed searching scheme, the breadth-first searching scheme can also be employed for MIMO detection. The breadth-first algorithm searches for the $bestdist$ in the forward direction only, but the best K candidate $newdist$ are kept at each level of the sublattice. Hence, the breadth-first algorithms can result in a constant decoding throughput. A strict breadth-first algorithm should keep K as large as possible without compromising on the optimality, compared with the exhaustive-search ML algorithm. However, limiting K can reduce the complexity of the breadth-first algorithm [12], [16]–[18], that is called K-best algorithm in this paper. The bit-error rate (BER) performance of the K-best algorithm is expected to be close to that of the ML algorithm if K is sufficiently large, as in the well-known M-algorithm for sequential decoding [13].

The principle of the K-best type of algorithm is outlined as below.

- Step 1) At the root sublattice, initialize one path with metric zero.
- Step 2) Extend each survivor path, retained from the previous sublattice, to M_c contender paths, and update the accumulated metric for each path.
- Step 3) Sort the contender paths according to their accumulated metrics, and select the K-best paths.
- Step 4) Update the path history for each retained path, and discard the other paths.
- Step 5) If the iteration arrives at the end sublattice, stop the algorithm. Otherwise, go to Step 2).

The best path at the last iteration is, thus, the hard decision output of the decoder. The advantage of the K-best algorithm over the sequential algorithm is its fixed throughput, since it is easily implemented in a parallel and a pipelined fashion.

IV. Proposed K-Best SE Algorithm

The proposed K-best SE algorithm is a modification to the K-best SD using the Schnorr–Euchner searching strategy, which is formulated as the following. The matrix $\mathbf{L}$ with positive diagonal elements is the inverse and transpose of the matrix $\mathbf{R}$, i.e., $\mathbf{L} = \mathbf{R}^{-T}$, where $\mathbf{R}$ is the upper triangular matrix in the QR-decomposition of the channel matrix $\mathbf{H} = \mathbf{QR}$. The other notations used below are in conformity with those in [10].

1) Input $\mathbf{z}$, $\mathbf{L}$, $\mathbf{p}$, K, and C, and initialize

$$\begin{aligned} \mathbf{e}_1 &= \mathbf{z} \\ bestdist &= 0 \\ k &= n. \end{aligned}$$

2) For $i = 1, 2, \ldots, length(bestdist)$, where the function $length(\cdot)$ returns the number of elements in $(\cdot)$, calculate

$$u_{t,k} = \omega \quad \forall\, \omega \in \Omega \tag{4}$$

$$y_t = \frac{(e_{i,k} - u_{t,k})}{l_{k,k}} \tag{5}$$

$$newdist_t = bestdist_i + y_t^2. \tag{6}$$

3) Let $T = length(bestdist) \cdot length(\Omega)$, sort $newdist_t$ $(t = 1, 2, \ldots, T)$ in ascending order, and choose the best (smallest) $\min(T, K)$ candidate paths with $newdist_t < C$. Discard the other paths. Adjust $\mathbf{U}$ and $\mathbf{y}$ accordingly, and replace $bestdist$ with $newdist$.
4) For $i = 1, 2, \ldots, length(bestdist)$, calculate

$$e_{i,j} = e_{i,j} - y_i \cdot l_{k,j} \quad j = 1, 2, \ldots, k-1. \tag{7}$$

5) If $k \neq 1$, then go to step 2) with $k = k-1$, else return the first row of $\mathbf{U}$ as $\hat{\mathbf{s}}$ which has the smallest $bestdist$ and is to be sorted by $\mathbf{p}$ if needed.

Moreover, it is straightforward to show $(1/(l_{k,k})) = r_{k,k}$, where $r_{k,k}$ is the kth diagonal element of the matrix $\mathbf{R}$. The division in (5) can, thus, be replaced by a multiplication, which is more simple and more numerically stable than the division in hardware implementations. Furthermore, note that the second item $u_{t,k}/l_{k,k}$ on the right-hand side of (5) is irrelevant to the i loop, which is instead needed for the first item $e_{i,k}/l_{k,k}$. Consequently, (4) and (5) can be replaced by (8), in which $\tilde{\mathbf{u}}_k = \Omega \cdot r_{k,k}$

$$y_t = e_{i,k} \cdot r_{k,k} - \tilde{u} = \tilde{y} - \tilde{u} \quad \forall\, \tilde{u} \in \tilde{\mathbf{u}}_k. \tag{8}$$

The advantage of using (8) is that $\tilde{\mathbf{u}}_k$ can be calculated in the preprocessing unit, which results in lower complexity in the decoding unit. The additional overhead to the preprocessing unit is trivial due to the symmetry of Ω for the modulation schemes considered. This property is not available in the K-best SD algorithm. The total complexity of the MIMO detector using the KSE is, thus, lower than that using the K-best SD.

The complexity of the KSE algorithm depends on the number of candidates K and the chosen sphere radius C. A smaller K represents lower complexity in the K-best algorithms. It is straightforward to show that a preprocessing scheme, taking into account both postdetection signal-to-noise ratio (SNR) and channel noise level [19], can reduce the complexity of the decoding unit significantly. On the other hand, the C in KSE is similar to the threshold value in the T-algorithm for sequential decoding [20]. When the value of C is sufficiently large, e.g., 2^{10}, the algorithm achieves its maximal complexity which is a constant provided K and n. When C is smaller, however, the complexity is reduced with the degradation in performance due to the lost lattice points outside the radius. A possible choice of the radius is $C = \gamma n \sigma^2$, where $\gamma \geq 1$ is chosen to guarantee that the true lattice point can be captured [6]. In our experimental evaluations, the coefficient γ is empirically set to 5 when $\mathrm{SNR} < 25$ dB, and 10 when $\mathrm{SNR} \geq 25$ dB.

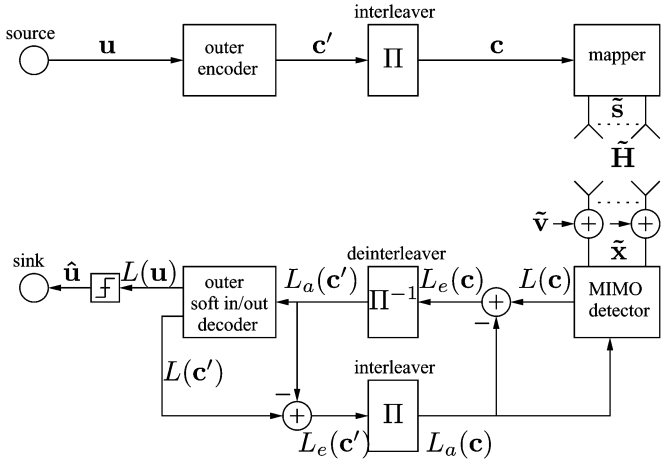


Fig. 2. MIMO transmission and iterative receiver model.

V. Soft-Output Extension of K-Best SE Algorithm

As for the soft-output MIMO detection, Fig. 2 shows a standard flowchart of an iterative MIMO receiver [6]. The information bits $\mathbf{u}$ is encoded and interleaved to become the coded bits $\mathbf{c}$, which is the input to the constellation mapper. The soft-output MIMO detector takes channel observations $\mathbf{x}$ as well as *a priori* information $L_a(\mathbf{c})$ on the inner coded bits, and calculates extrinsic information $L_e(\mathbf{c})$ for each of the coded bits per symbol vector. Then, $L_e(\mathbf{c})$ is deinterleaved to become the *a priori* input $L_a(\mathbf{c}')$ to the outer soft-input/soft-output decoder, which calculates extrinsic information $L_e(\mathbf{c}')$ on the outer coded bits. Then, $L_e(\mathbf{c}')$ is reinterleaved and fed back as *a priori* information $L_a(\mathbf{c})$ to the inner detector, thus completing an iteration. After some iterations, the outer decoder makes decisions $\hat{\mathbf{u}}$ about the information bits by *a posteriori* information $L(\mathbf{u})$.

A. APP Detection of MIMO Signals

In the iterative MIMO receiver, the MIMO detector needs to generate *a posteriori* probability (APP) about the inner coded bits $\mathbf{c}$. The APP is usually expressed as a log-likelihood ratio value (L-value). The L-value of the bit c_k, $(k = 1, 2, \ldots, M * M_c)$, where M_c represents the number of bits per constellation symbol, conditioned on the received symbol vector $\mathbf{x}$, is defined as

$$L(c_k|\mathbf{x}) = \ln \frac{P[c_k = +1|\mathbf{x}]}{P[c_k = -1|\mathbf{x}]}. \tag{9}$$

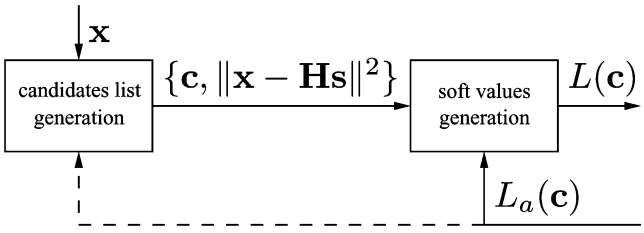


Fig. 3. Block diagram of the soft-output MIMO detector.

Equation (9) can further be simplified [6], [14]

$$L(c_k|\mathbf{x}) = \max_{\mathbf{c}\in\mathbb{C}_{k,+1}}\{\Lambda(\mathbf{c},\mathbf{x},\mathbf{L}_a(\mathbf{c}))\} - \max_{\mathbf{c}\in\mathbb{C}_{k,-1}}\{\Lambda(\mathbf{c},\mathbf{x},\mathbf{L}_a(\mathbf{c}))\} \quad (10)$$

where $\mathbb{C}_{k,\pm1} = \{\mathbf{c}|c_k = \pm 1\}$ and

$$\Lambda(\mathbf{c},\mathbf{x},\mathbf{L}_a(\mathbf{c})) = -\frac{1}{2\sigma^2}||\mathbf{x}-\mathbf{eHs}||^2 + \frac{1}{2}\mathbf{c}\cdot\mathbf{L}_a^T(\mathbf{c}). \quad (11)$$

The first item of (11) can be calculated by a lattice decoder [6]. Furthermore, to avoid the overwhelming complexity of (10) in searching the set $\mathbb{C}_{k,\pm1}$ exhaustively, [6] proposed to search a subset $\mathbb{L}$ of $\mathbb{C}_{k,\pm1}$

$$L(c_k|\mathbf{x}) = \max_{\mathbf{c}\in\mathbb{L}}\{\Lambda(\mathbf{c},\mathbf{x},\mathbf{L}_a(\mathbf{c}))\} - \max_{\mathbf{c}\in\mathbb{L}}\{\Lambda(\mathbf{c},\mathbf{x},\mathbf{L}_a(\mathbf{c}))\}. \quad (12)$$

The candidate list $\{\mathbf{c}, ||\mathbf{x}-\mathbf{Hs}||^2\}$ can, thus, be calculated only once by the lattice decoder. The *a priori* list $\mathbf{L}_a(\mathbf{c})$ and the APP value $L(c_k|\mathbf{x})$ must be updated every iteration. Specifically, the MIMO detector in Fig. 2 can be replaced with a candidates list generation block and a soft values generation block, as shown in Fig. 3. The mostly well-known list type of MIMO decoding algorithms are list sphere decoding (LSD) algorithm [6] and list sequential sphere decoding (LISS) algorithm [14].

B. K-Best SE Decoder With Soft Outputs

As implied in Fig. 3, a detector with the capability of generating a candidates list is simply a soft MIMO detector. Hence, the proposed KSE supports soft outputs, as well as hard outputs. In the KSE algorithm, the K best paths retained at the last iteration is the candidates list, in which the best path is the hard output. Similar to the M-algorithm, the candidates list obtained in the KSE does not necessarily include the closest (ML) point from the received point. Consequently, the expected performance of the KSE is lower than that of the optimal LSD and LISS.

However, the KSE has advantages in hardware implementations compared with the LSD and the LISS. The KSE is a single-direction searching algorithm, i.e., the iteration proceeds in the forward direction only. Consequently, the KSE is easily implemented in a parallel fashion and achieves a fixed throughput. Both the LSD and the LISS are two-direction searching algorithms, i.e., the iteration proceeds forward and backward. Their detection throughput is variable, since it depends on the maximally possible searching time. The variable detection throughput would demand I/O buffers, which may be an extra overhead in a practical system.

Moreover, due to the single-direction property, the KSE is easily implemented in a pipelined hardware architecture [21]. Each pipeline stage corresponds to one iteration in the KSE algorithm. The high detection throughput is, thus, possible in the KSE decoder. In contrast, both the LSD and the LISS have to maintain a long list or stack for all the iterations, which could limit the detection throughput due to the nonpipelined fashion.

In the iterative MIMO system, the KSE detector can be used at every iteration to suppress the number of survivor paths, when (11) is used as the metric. Though the algorithm performance could be improved by this scheme [16], [17], the computational complexity is also significantly increased compared with the case in which the detector is used only at the first iteration. The objective of this paper is to investigate a low complexity implementation of the soft-output MIMO detector. The discussion on the soft-output KSE is, thus, limited to the usage at the first iteration, where only $||\mathbf{x}-\mathbf{Hs}||^2$ is taken into account in the metric calculation.

Similar to other list type of MIMO decoding algorithms, the number of survivor paths K is a key to the KSE algorithm. A larger K would increase the candidates list size and, thus, improve the performance. The penalty is that the complexity is also increased, since a large number of paths has to be extended, sorted and retained. Another disadvantage of larger K is about the detection throughput, since the iteration period of the KSE is proportional to K according to the implementation results of [21].

In summary, the soft-output KSE with a smaller K has lower complexity and higher detection throughput, but has lower performance compared with that with a larger K. A modification to the soft-output KSE is, thus, proposed to improve its performance without increasing K.

C. Modified K-Best SE Decoder With Soft Outputs

Inspired by [14], the MKSE tries to use the information contained in the discarded paths that are not extended to the end sublattice. In other words, The MKSE generates the soft outputs by using the discarded paths, as well as the K survivor paths during some iterations of the algorithm. The MKSE modifies step 3) of the KSE as below.

- Let $T = length(bestdist) \cdot length(\Omega)$, sort $newdist_t$ $(t = 1,2,\ldots,T)$ in ascending order, and choose the best (smallest) $\min(T,K)$ candidate $newdist_t < C$. From the kth iteration, move the discarded $T - \min(T,K)$ paths to the candidates list.…

The modification to the KSE is done with a minor increase in complexity. Only some paths moving should be taken into account in hardware implementations. Since the candidates list is required in all the soft-output MIMO detectors to generate the soft values, it should not be counted in the hardware overhead due to the modification. Consequently, the VLSI architecture proposed in [21] for the KSE can also be applied to the MKSE with minor modifications. The MKSE would have the same detection throughput as the KSE with the same K, since both retain the equal number of survivor paths until the end sublattice. The idea of the MKSE algorithm was partly originated from our previous KSE implementation [21]. Hence, it could be helpful

to understand the MKSE algorithm with its VLSI architecture considered together (cf. Fig. 12).

Note that the paths retained from the kth iteration could be more reliable, since they are more close to the end sublattice. For a 4×4 16-QAM MIMO system, the MKSE ($K = 5$, $k = 4$) retains $20 + 4 \times (20 - 5) = 80$ paths as the candidates list, while the original KSE ($K = 5$) retains at most $KM_c = 20$ fully extended paths. The soft outputs by the MKSE would be more reliable than those by the KSE, since the MKSE has more candidate paths than the KSE with the same K. The performance of the MKSE is, thus, expected to be better than that of the KSE with the same K.

On the other hand, most of paths retained in the MKSE are not fully extended to the end sublattice. These paths have to be virtually augmented to full length based on the assumptions about the remaining undetected symbols, then they can contribute to the soft value generation. Several possibilities are proposed in [14] as assumptions about the undetected symbols. The ZF estimation, also called the Babai point [10], is the most simple method to implement. The ZF estimation can be obtained simply by rounding the received point to the nearest lattice point, which adds little overhead to the hardware implementation.

It should be noted that the path augmenting by ZF estimation is easier to implement in the MKSE than in the LISS. The LISS has to maintain a large stack for the candidate paths. To enable the path augmenting, the LISS has to know whether each path achieves the full length or at which iteration it stops. This may add an extra overhead to the hardware implementation. However, the MKSE inherently knows where each path arrives due to its single-direction property as in the KSE. Consequently, the path augmenting is not an overhead in the MKSE compared with the LISS.

VI. Simulation Results

A 4×4 16-QAM MIMO system is considered in our simulations. Based on BER and numerical complexity in real-value multiplication, the simulation results of the MIMO decoding algorithms mentioned above are presented in this section. To ease the comparison, a division or a square-rooting is considered to be as complex as a multiplication, while a squaring is considered to be half as complex as a multiplication [22].

In our simulations, the SNR per transmitted information bit is defined as

$$\left(\frac{E_b}{N_0}\right)_{dB} = \left(\frac{ME_s}{N_0}\right)_{dB} + 10\log_{10}\frac{1}{R_c M_c} \qquad (13)$$

where R_c is the code rate, and $R_c = 1$ is assumed when the uncoded MIMO system is considered. The average symbol energy $E_s = 2(q-1)/3$ for the q-QAM ($q = 4, 16, \ldots$) modulation when $E_b = 1$.

In the case of hard-output detection, 100 k independent channel realizations (packets) of 40 uncoded 16-QAM symbols are transmitted with ten symbols from each antenna. In the case of soft-output detection, to enable comparisons with the results from [6], the frame length is chosen to be 9216 information bits. To get an insight into the average behavior of an iterative MIMO receiver, all simulations for the soft-output detection are performed until at least 10 frame errors are incurred or at most 200 frames are transmitted. Four receiver iterations are performed for each frame. As discussed in [6], an ergodic channel model is used in the sense that the statistical nature of $\mathbf{H}$ is observed as the channel is used.

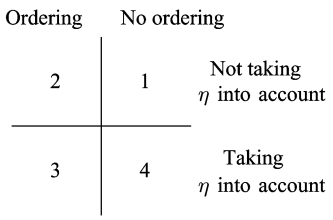


Fig. 4. Four preprocessing modes, where η represents the channel noise level.

A. Effects of Preprocessing

In the lattice detector, as shown in Fig. 1, the complexity of the predecoding unit is fixed to be n^2 regardless of the algorithm used in the decoding unit. The complexity of the predecoding unit is, thus, ignored in the sequel. The effects of the preprocessing unit on the decoding unit using the KSE algorithm are analyzed in this section.

To reduce the computational complexity of the decoding unit, $\mathbf{H}$ is commonly preprocessed in various practical MIMO detectors [23], [24]. The preprocessing can be partitioned into four modes, as shown in Fig. 4, according to the ordering by the postdetection SNR and the consideration of the channel noise level $\eta = \sqrt{\sigma^2/E_s}$. No operation on $\mathbf{H}$ is done in $mode\ 1$. The $mode\ 2$ only takes account of the ordering by the postdetection SNR [25], which represents the effects of interferences from all other received signals. The $mode\ 4$ does no ordering and is essentially a solution based on the minimum mean-square error (MMSE) criterion, in which η is included. The $mode\ 3$ takes account of the postdetection SNR, as well as the MMSE criterion, that is simply the method used in the square root algorithm [19].

For a given BER performance, the value of K correlates strongly with the decoding complexity of the KSE, that can be determined by simple trial and error. Based on the simulations, the minimum value of K is determined to be 16, 14, 5, and 12 for the KSE with $mode$ 1, 2, 3, and 4 preprocessing, respectively. The BER performance of the KSE is shown in Fig. 5(a). Clearly, the BER performances of the KSE are almost the same for all the preprocessing modes at low and middle SNR, with only minor differences at very high SNR.

The decoding complexity of the KSE with various preprocessing modes is shown in Fig. 5(b) as the function of SNR. It is clear from Fig. 5(b) that the KSE with $mode\ 3$ preprocessing has always the lowest decoding complexity, since the value of K of $mode\ 3$ is much smaller than those of the other modes. At high SNR, however, it is interesting to observe that the complexity differences among the four preprocessing modes are not as significant as that the values of K indicate. The reason is that the decoding complexity becomes more dependent on the radius C regardless of K, since the number of survived paths tends to be less than the chosen K at high SNR. Unfortunately, this property cannot be exploited in hardware implementations, although the complexity of $mode$ 1, 2, and 4 is lower than that

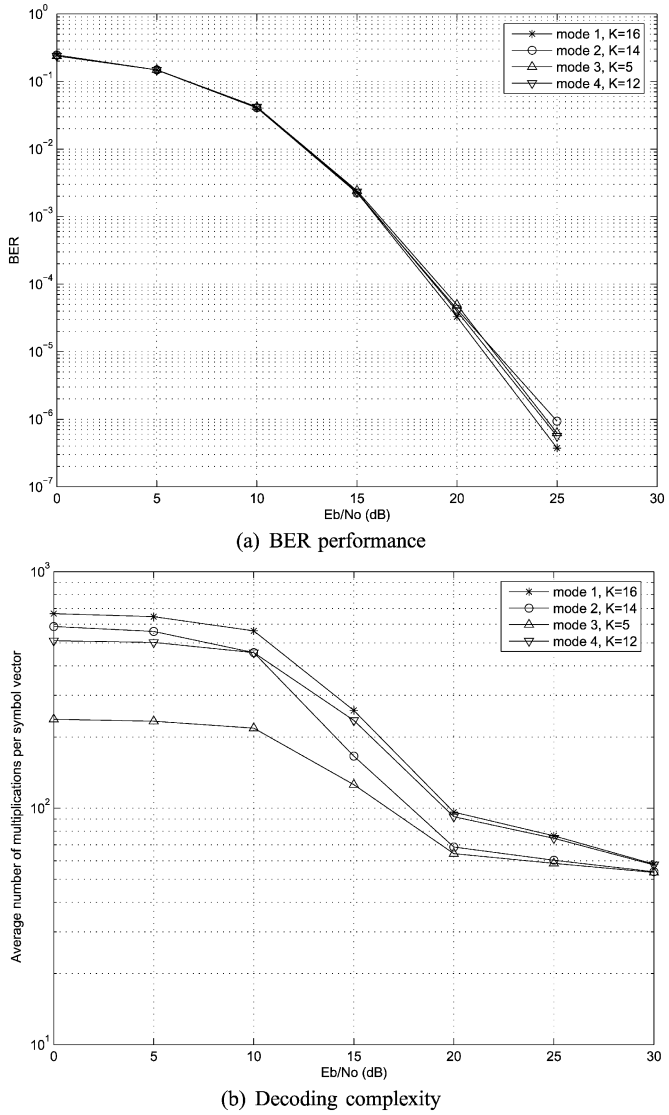


Fig. 5. Effects of preprocessing on KSE, 4×4 16-QAM. The radius C is set as in Section IV.

of $mode\ 3$. One reason is that the memory use and the frequency of memory accesses have been omitted in the complexity analysis. Since the KSE is a sorting-based algorithm, the value of K also correlates strongly with the used memory size. The other reason is that the range of medium SNR is more interesting in practical implementations, especially in the case of soft-output detection as shown later. Consequently, the KSE with $mode\ 3$ preprocessing, due to the much smaller value of K, would be more promising in hardware implementations than those with the other preprocessing modes. The $mode\ 3$ preprocessing is assumed for all the MIMO decoding algorithms in the sequel.

Furthermore, as shown in Fig. 1, the preprocessing unit is operated on packet level, while the decoding unit is operated on symbol level. A common case in current wireless communications is that the packet rate is much less than the symbol rate. This motivates us to reduce the complexity of the decoding unit by moving as many calculations as possible to the preprocessing unit. The derivation of (8) is such an example. The complexity of preprocessing is only critical in wideband MIMO-OFDM systems, where it needs to be performed on a tone-by-tone basis. A partial solution to this problem has recently been presented in [26].

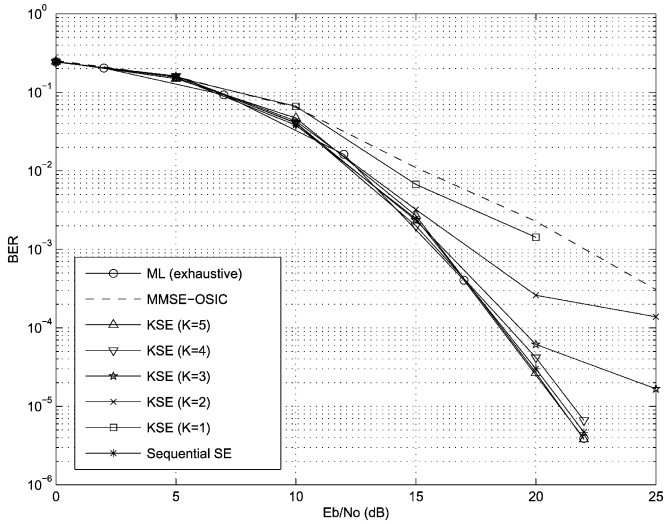


Fig. 6. BER performance comparison of exhaustive-search ML, KSE with various K, sequential SE and MMSE-OSIC, 4×4 16-QAM. The radius C is set to infinity. The sequential SE is adopted from [15] without using Fano-like metric bias and early termination. The MMSE-OSIC (square-root algorithm) is adopted from [19].

B. Hard-Output Case

The K-best algorithm does not necessarily catch the ML point in theory. However, the BER performance of the K-best algorithm can approach to that of the sequential algorithm, if the value of K is chosen suitably. Fig. 6 shows comparative simulation results of KSE with various K. The performances of exhaustive-search ML, sequential SE and MMSE-ordered successive interference cancellation (OSIC) are also shown in Fig. 6 as references. Clearly, both the KSE $(K = 5)$ and the adopted sequential SE can achieve the same performance as the exhaustive-search ML within the investigated range of SNR. As expected, the performance of KSE deteriorates as the value of K decreases. The KSE $(K = 1)$ actually performs like the MMSE-OSIC, since the $mode\ 3$ preprocessing is assumed for the KSE in our simulations.

Fig. 7 shows floating-point complexity comparison of KSE, K-best SD, and sequential SE. Due to the adoption of (8), both maximum and average complexity of the KSE are lower than that of the K-best SD by $2MK(length(\Omega) - 1)$ multiplications per symbol vector.

Fig. 7 also shows the floating-point complexity of the sequential SE adopted from [15]. Clearly, the average complexity of the sequential SE is much lower than that of the KSE $(K = 5)$, especially within the range of low and medium SNR. Moreover, it is observed that the minimum complexity of the sequential SE corresponds to about 50 multiplications per symbol vector. Exploiting the observations above, the sequential SE decoders can achieve a much higher peak decoding throughput compared with the KSE decoders, as proved in [27].

On the other hand, it is clear from Fig. 7 that the maximum complexity of the sequential SE is higher than that of the KSE regardless of the range of SNR. Fig. 7 only shows a snapshot

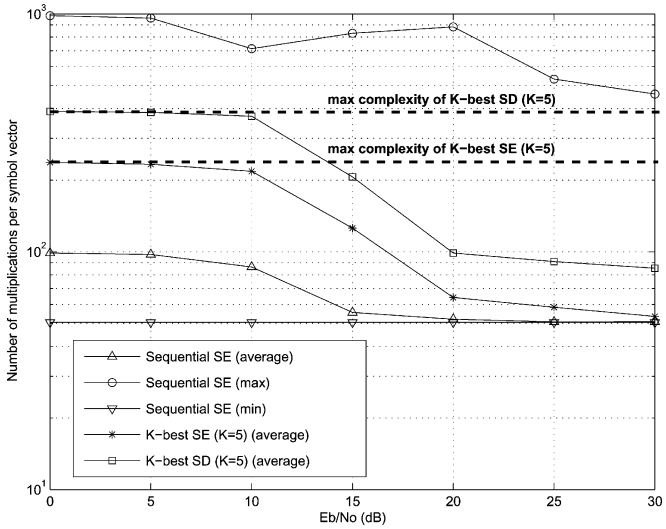


Fig. 7. Floating-point complexity comparison of KSE, K-best SD, and sequential SE, 4×4 16-QAM. The maximum and average complexity of the K-best algorithms are obtained when the radius C is set to infinity and as in Section IV, respectively. The sequential SE is adopted from [15] without using Fano-like metric bias and early termination.

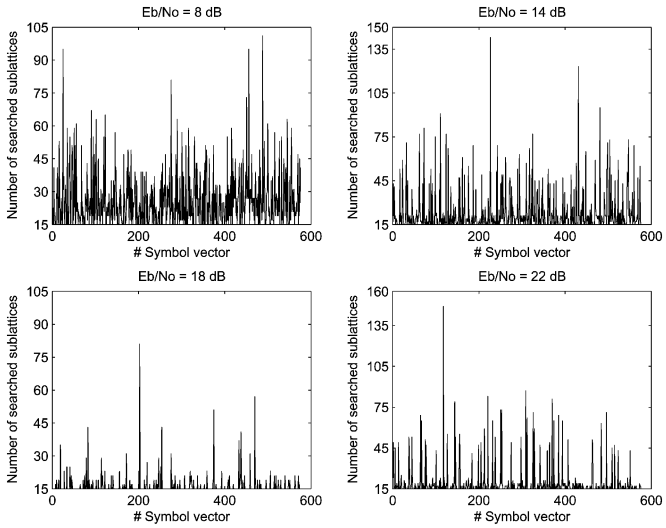


Fig. 8. Snapshots of number of searched sublattices in sequential SE, 4×4 16-QAM. The sequential SE is adopted from [15] without using Fano-like metric bias and early termination.

plot for the maximum complexity of the sequential SE, since it is variable with different simulation runs. Fig. 8 shows snapshots of number of searched sublattices in the sequential SE. Clearly, the maximum number of searched sublattices in the sequential SE is irrelevant to SNR. In summary, the sequential SE can achieve a high peak decoding throughput, which corresponds to searching 15 real-value sublattices (cf. Fig. 8). However, the decoding throughput of the sequential SE is variable due to the variable number of searched sublattices. To alleviate the possible loss in both the instantaneous throughput and BER performance, the sequential SE decoders have to use I/O buffers. A systematic solution is still an open problem.

As shown in Fig. 7, the maximum complexity of the KSE is fixed. Consequently, the I/O buffers can be avoided in the

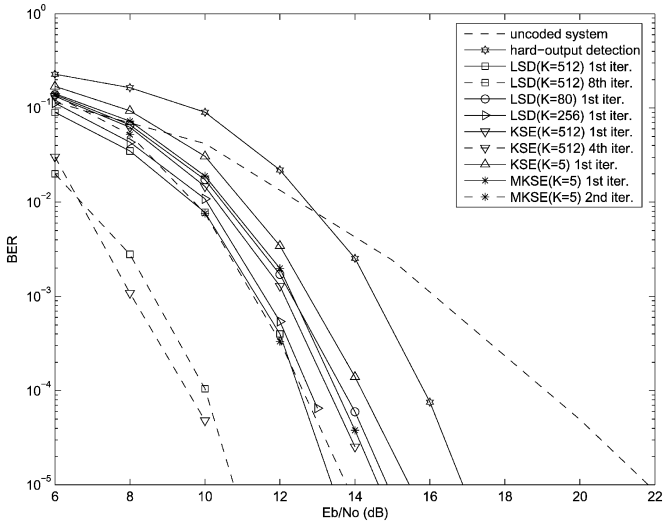


Fig. 9. Comparative simulation results, outer rate 1/2 convolutional code with memory 2, 4×4 16-QAM. The channel capacity is at 3.7 dB.

KSE decoder, when the KSE decoder is designed based on the maximum complexity corresponding to K. At the same time, the KSE decoder can exploit the sphere radius C to reduce operations in the implemented circuits, as implied with the average complexity in Fig. 7. This operation reduction would be beneficial to low-power design without affecting the decoding throughput, as shown later in Section VII-B. How to exploit C to tradeoff BER performance and computational complexity is to be left to a MIMO system designer [cf. Figs. 5(a)–7].

C. Soft-Output Case

Fig. 9 shows comparative simulation results with an outer $R_c = 1/2$ convolutional code with memory 2. It is clear that the coded MIMO system outperforms the uncoded system by 5 dB at $\mathrm{BER} = 10^{-5}$ even with a very simple convolutional code and a hard-output MIMO detection. As expected, the KSE is shown to be a soft-output MIMO detector. At the first iteration, the LSD with a list of 512 candidates [6] outperforms the KSE $(K = 512)$ by about 1 dB. However, the KSE $(K = 512)$ only uses four iterations to outperform the LSD using eight iterations. Even the KSE $(K = 5)$ shows a little capability in the soft values generation, taking into account that it has only $K = 5$ paths as the candidates list.

The parameters of the MKSE simulated are $K = 5$ and $k = 4$ and, hence, the MKSE has 80 candidate paths for the soft values generation. It is clear from Fig. 9 that the performance of the MKSE is close to that of the KSE $(K = 512)$ at the first iteration, and outperforms the KSE $(K = 5)$ by about 1 dB at $\mathrm{BER} = 10^{-5}$. This shows that the performance of the KSE with larger K at the first iteration can be achieved by the MKSE with much smaller K. Both the decoding throughput and the implementation complexity can, thus, be improved significantly in the MKSE due to smaller K. Furthermore, note that the MKSE has the same performance as the LSD with a list of 80 candidates at the first iteration. This shows that the path augmenting used in the MKSE performs well, although the complexity of the MKSE is much lower than that of the LSD with the same size of candidate paths, as shown in Fig. 10.

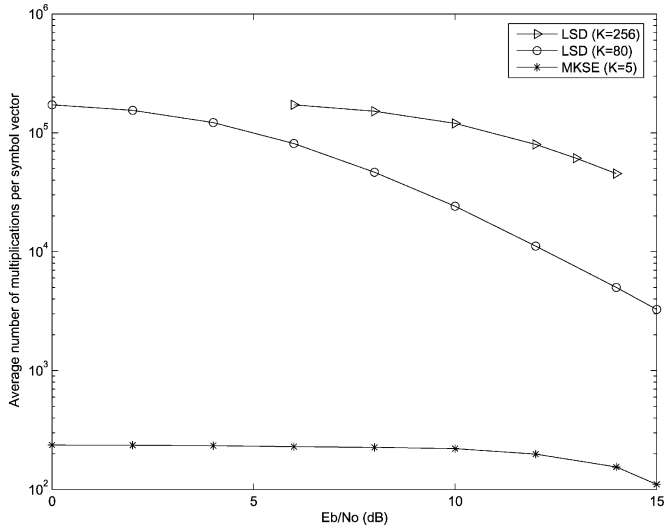


Fig. 10. Complexity comparison between MKSE and LSD, 4×4 16-QAM.

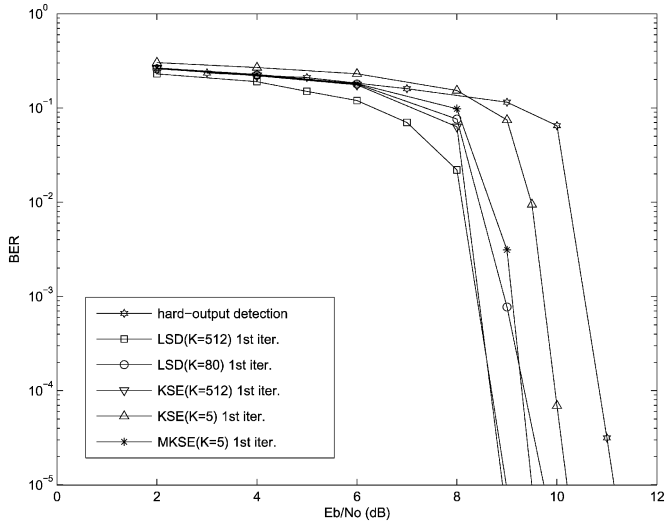


Fig. 11. Comparative simulation results, outer rate 1/2 turbo code with memory 2, 4×4 16-QAM. The channel capacity is at 3.7 dB.

On the other hand, no performance improvement is observed for the MKSE after the second iteration, neither for the KSE $(K = 5)$. The MKSE can gain about 0.7 dB with the second iteration compared with the first iteration. Therefore, the proposed MKSE is suitable for the coded MIMO system without iterative detection and decoding (i.e., no loop between the MIMO detector and the outer decoder), which may be a practical case in MIMO applications with medium to high date rates.

Fig. 11 shows comparative simulation results with an outer $R_c = 1/2$ turbo code with memory 2. The MKSE $(K = 5)$ outperforms the KSE $(K = 5)$ and the hard-output detection by about 0.7 and 1.7 dB, respectively, but has about 0.5 dB loss compared with the LSD at the first iteration. On the other hand, it is clear from Fig. 11 that the performance of the MKSE $(K = 5)$ is close to that of the LSD $(K = 80)$. Therefore, a similar conclusion to the case of convolutional code can be obtained in the case of turbo code. As a low complexity soft-output MIMO detector, the proposed MKSE is also suitable for the turbo coded MIMO receiver without iterations.

VII. Proposed VLSI Architecture

The proposed KSE and MKSE algorithms share the same VLSI architecture, except that the MKSE needs an extra soft-output module. Fig. 12 shows the overall architecture of the decoder for a 4×4 16-QAM MIMO system. The architecture supports both hard outputs and soft outputs. When the soft-output module is not included, the architecture is simply a hard-output KSE decoder. Otherwise, the architecture becomes a soft-output MKSE decoder. In this section, the hard-output KSE decoder is described first, then the soft-output MKSE decoder is analyzed as an extension of the hard-output KSE decoder.

As shown in Fig. 12, the architecture consists of eight pipeline stages. Each stage has a processing element (PE), which implements the operations corresponding to step 2)–step 4) of the algorithm. Stage 1 to stage 8 corresponds to the eighth to the first level of computation in the algorithm. The buffers U, D, Y, and E between adjacent PEs are needed to store the variables $\mathbf{u}_k$, $\mathbf{bestdist}_k$, $\tilde{\mathbf{y}}_k$ and $\mathbf{e}_k$ in the algorithm, respectively. The preprocessed channel information $\mathbf{L}$ and $\mathrm{diag}(\mathbf{R})$ are combined into a single data stream, which is the input to the buffer G.

A. PEs

According to the (6)–(8), the original dataflow graph (DFG) for the kth $(k = 8, 7, \ldots, 1)$ level computation of K-best SE can be derived, as in Fig. 13(a), which mainly consists of two multipliers, one squarer and one sorting unit. An alternative is to time-multiplex the multiplier M_1 and the squarer, as shown in Fig. 13(b). The DFG of Fig. 13(b) can be further simplified as a DFG consisted of only one multiplier, one adder, and one sorting unit, as done in [12]. This is the DFG with the smallest area, but it may result in an implementation with the less efficient throughput due to excessive time-multiplexing.

In contrast, from the second stage in our implementation, the multiplier M_1 is time-multiplexed with the multiplier M_2 at the preceding stage, as shown in Fig. 13(c). This new time-multiplexing scheme alone can improve the decoding throughput by about three times compared with [12], even if the impact of the sorting unit is not taken into account. The penalty is an extra squarer. It should be noted that the hardware complexity of the squarer is only about half of the multiplier [22]. Furthermore, the squarer and the following adder can be integrated into a single unit using carry-save adder (CSA) technique, and the resulting circuit is smaller and faster than a separate squarer and a separate adder.

The eighth PE is also implemented as in Fig. 13(c), but the multiplier M_2 and the following adder are not needed according to the algorithm. The DFG of the first PE is shown in Fig. 13(d). The structure of the first PE is similar to Fig. 13(b), since there is no preceding PE that can share its multiplier with the first PE. On the other hand, the DFG of the first PE should be similar to Fig. 13(c), since the first PE has to share its multiplier M_2 with the second PE.

It is shown in [12] and [20] that the cost of sorting is linear with the value of K, when the bubble sorting algorithm is employed. In the proposed architecture, the data are input to the sorting unit in series. Furthermore, the length of sorting sequence is $length(\Omega) \cdot K = 20$ in our implementation, that is

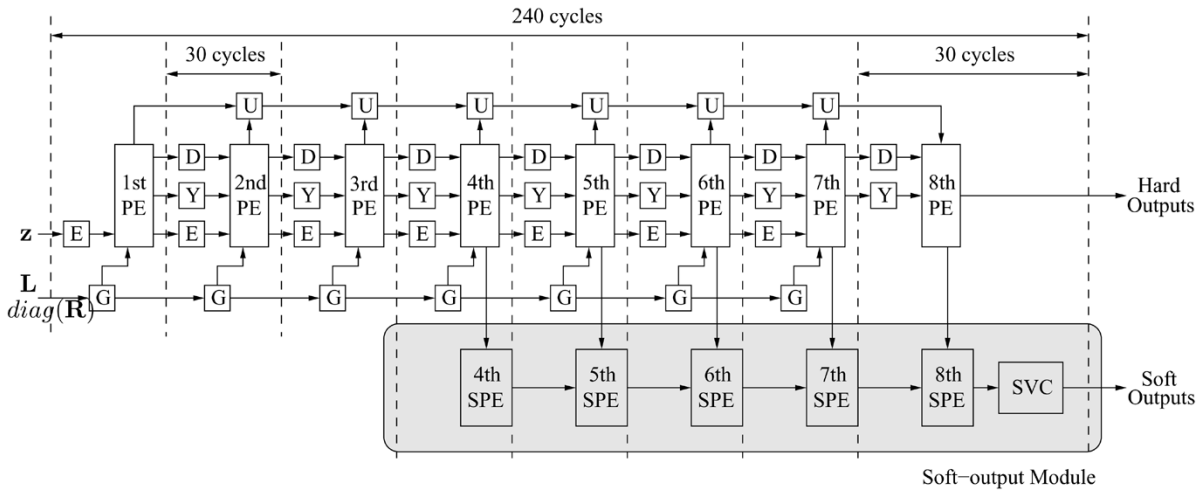


Fig. 12. VLSI architecture of the KSE/MKSE decoder for 4×4 16-QAM MIMO system. The buffers U, D, Y, and E correspond to $\mathbf{u}_k$, $\mathbf{bestdist}_k$, $\tilde{\mathbf{y}}_k$, and $\mathbf{e}_k$ in the algorithm, respectively.

only half of that in [12]. Therefore, the bubble sorting algorithm is the most suitable choice for our sorting unit compared with the other sorting algorithms mentioned in [20].

B. Buffers

Since a large number of small buffers with various sizes exist in the proposed VLSI architecture, all the buffers are implemented in register banks instead of RAMs. The reason is the large number of RAMs could result in the overhead in place and route and the inefficiency in silicon area and power consumption. In this paper, the size of a buffer is denoted as $m \times n$, where m and n represents the number of banks and the number of pipelined registers in each bank, respectively. The default value of n is 1 if ignored. The sizes of all the buffers are shown in Table I.

The first buffer G with size 1×9 stores the value $l_{8,j}$ $(j = 1, \ldots, 7)$, $r_{8,8}$ and $r_{7,7}$, in which $r_{7,7}$ is required by the multiplier M_2 at the first stage to calculate $\tilde{\mathbf{y}}_7$ for the second stage. The input symbol $\mathbf{z}$ is buffered using double register banks to maximize the decoding throughput. The size of the first buffer E corresponds to two continuous symbol vectors. In our implementation, the first element of $\mathbf{z}$ is processed in the first PE directly. The size of the first buffer E is, thus, 2×7. The number of register banks required in the buffers D, Y, and E is four at the second stage, since there are only four candidates available from the preceding stage according to the algorithm. It is straightforward to derive the sizes of the other buffers, taking $K = 5$ into account.

The register banks between the adjacent PEs are not always fully occupied by the transferring data, since the sphere radius C of the algorithm makes it possible that the number of candidates is less than K at some stages. Each of the register banks is designed to be activated only when it is required by the candidates generated at the preceding stage. The power consumption on buffers can, thus, be reduced.

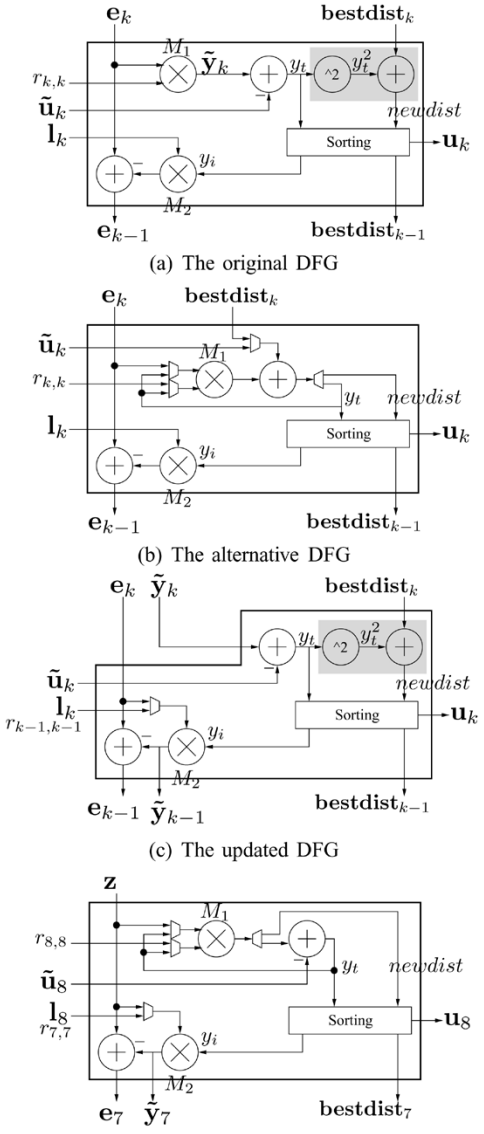
C. Timing Schedule

As far as timing schedule is concerned, it should be noted that all the data are calculated and transferred serially in the proposed architecture. To maximize the decoding throughput, the clock cycles covered by each stage should be minimized. The sorting unit is the bottleneck in the timing schedule, since it covers 20 clock cycles. The other arithmetic units need 5 clock cycles, as shown in Fig. 13(a), assuming that a multiplier or squarer has the same speed as an adder. Therefore, the calculation period of $20 + 5 = 25$ clock cycles is needed for each stage. Moreover, time-multiplexing is extensively used in the proposed architecture, as shown in Fig. 13(c) and (d), which needs 5 more clock cycles in the final implementation. In total, each stage needs 30 clock cycles, in which 25 clock cycles are involved in the calculations, and the remained 5 clock cycles are used for time-multiplexing. The latency for the architecture to process a symbol vector is, thus, $30 * 8 = 240$ clock cycles, as shown in Fig. 12.

The multiplier M_2 at the second stage performs the calculation on $e_{i,j}$ $(k = 7)$ according to (7). On the other hand, it has to calculate $\tilde{\mathbf{y}}_k$ $(k = 6)$ for the third stage due to time-multiplexing. Therefore, the multiplier M_2 at the second stage could be covered by $K*(7-1)+K = 35$ clock cycles for processing a received symbol vector. As analyzed above, however, the calculation period is limited to 25 clock cycles for each stage. The solution is to borrow 10 clock cycles from the sixth stage, since the multiplier M_2 at the sixth stage is covered by 10 clock cycles only. In other words, the multiplier M_2 at the sixth stage is time-multiplexed with the second stage for 10 clock cycles. The same problem would happen to the multiplier M_2 at the third stage, which could be covered by 30 clock cycles. In the final implementation, the multiplier M_2 at the sixth stage is further time-multiplexed with the third stage for 5 clock cycles.

D. Hard-Output KSE Versus K-Best SD

Table II shows a rough comparison on the number of resources between the proposed hard-output KSE $(K = 5)$ and the K-best SD $(K = 10)$ of [12], assuming that the same datapath wordlength of 16 bits is employed. Table II also shows the derived number of resources of K-best SD $(K = 5)$ assuming that the same VLSI architecture as [12] is employed. It is clear from Table II that the proposed architecture has less buffers and comparators than [12], with the penalty of seven squarer/adders and seven adders.



(a) The original DFG

(b) The alternative DFG

(c) The updated DFG

(d) The DFG of the 1st PE

Fig. 13. Illustration of dataflow graph (DFG). The shaded area represents that the squarer and the following adder can be merged into a single unit using CSA technique. (b) Derived by merging M_1 and squarer in (a). (c) Derived by merging M_1 and M_2 in (a), and used for the second PE and the nth ($n = 3 \ldots 8$) PE.

TABLE I
SIZES OF ALL THE BUFFERS IMPLEMENTED IN REGISTER BANKS

Buffer	Stage							
	1	2	3	4	5	6	7	8
G	1×9	1×7	1×6	1×5	1×4	1×3	1×2	
U		5×2	5×3	5×4	5×5	5×6	5×7	
D		4	5	5	5	5	5	5
Y		4	5	5	5	5	5	5
E	2×7	4×7	5×6	5×4	5×3	5×2	5×1	

Moreover, as described above, the latency of the proposed architecture is 240 clock cycles. In contrast, the period of each stage and the total latency of [12] is 160 and 1280 clock cycles, respectively. This shows that the proposed architecture outperforms over [12] in both decoding throughput and decoding latency when the same clock period is employed. The throughput gain of the KSE results from both the decreased K and the improved time-multiplexing DFG scheme (cf. Fig. 13).

TABLE II
COMPARISONS OF THE NUMBER OF RESOURCES BETWEEN THE HARD-OUTPUT KSE AND K-BEST SD, 4 × 4 16-QAM

	KSE	K-best SD [12]	
	K=5	K=5	K=10
Preprocessing assumed	mode 3	mode 3	mode 1
# Multiplier	8	8	8
# Squarer/adder	7	-	-
# Adder	15	8	8
# Comparator	39	40	80
# Buffer (Total bits)	34 (4054)	56 (4300)	56 (8600)
Decoding period (Cycles)	30	80	160
Decoding latency (Cycles)	240	640	1280

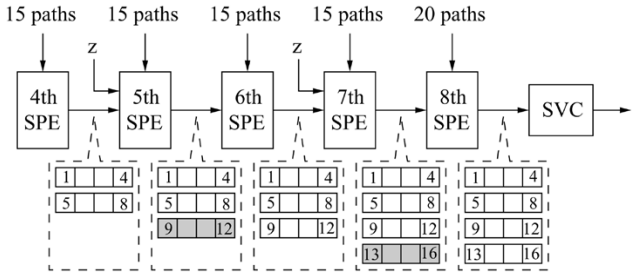


Fig. 14. Block diagram of the soft-output module. The numbered square represents bit metric. The shadowed square represents bit metric estimated by ZF path augmenting.

E. Soft-Output Extension

As shown in Fig. 12, the VLSI architecture proposed above for the hard-output KSE algorithm is easily extended to support the soft-output MKSE algorithm, with only a soft-output module appended. The function of the soft-output module is to exploit the discarded paths from the hard-output module. According to the simulation results for the 4 × 4 16-QAM MIMO system, the MKSE should retain the discarded paths from the fourth stage to the last stage.

Since the MKSE ($K = 5$) is only suitable for the iterative MIMO receiver with no iterations, the soft-output module is designed to calculate the soft outputs of the MIMO detector directly. Specifically, the soft-output module calculates the soft values first based on the discarded paths retained from the fourth stage. The final soft outputs are calculated just after the $KM_c = 20$ paths are obtained at the last stage. In this way, the transferred data in the soft-output module are the soft values (path metrics) instead of the retained paths, and the MKSE does not need to keep up to 80 paths until the last stage. Consequently, the soft values generation unit in Fig. 3 is avoided in the MKSE, and the implementation complexity of the soft-output module is reduced sufficiently. Moreover, the decoding period and latency of the hard-output module are not affected by the soft-output module. Hence, the MKSE has the same decoding throughput as the KSE.

Fig. 14 shows the block diagram of the soft-output module. It consists of five soft-value processing elements (SPE) and a soft-value calculation (SVC) unit. The SPE units accept the discarded paths from the fourth PE to the eighth PE, and calculate

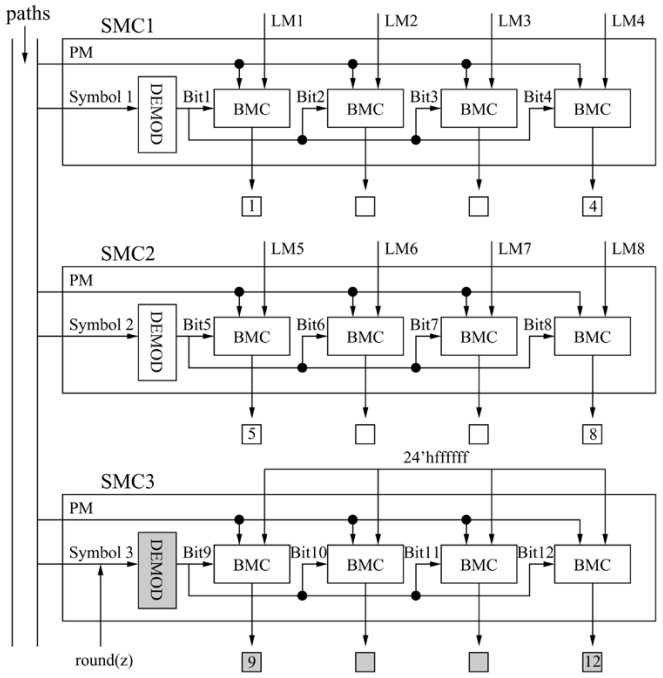


Fig. 15. Block diagram of the fifth SPE unit. The shadowed DEMOD unit accepts estimated symbol, and the shadowed square represents bit metric estimated by ZF path augmenting.

the corresponding bit metrics. Specifically, the fourth SPE accepts 15 discarded paths in series from the fourth PE, and calculates bit metrics for bit 1–bit 8. Then, the fifth SPE accepts the bit metrics from the fourth SPE, as well as 15 paths from the fifth PE, updates the bit metrics for bit 1–bit 8, and calculates bit metrics for bit 9–bit 12. The bit metric wordlength is determined to be 24 bits according to simulations. The higher 12 bits represents the bit metric $BM^{(1)}$ when $bit = 1$, while the lower 12 bits represents the bit metric $BM^{(0)}$ when $bit = 0$. All the 16 bit metrics corresponding to a symbol vector are obtained after the calculation in the eighth SPE. The SVC unit simply subtracts $BM^{(0)}$ from $BM^{(1)}$ for each bit metric, and output the final soft values. The SVC unit uses two subtractors in parallel to calculate the soft outputs. Hence, the calculation period of the eighth SPE combined with the SVC unit is within 30 clock cycles, as shown in Fig. 12.

Note that the bit metrics for bit 9–bit 12 calculated in the fifth SPE are estimation values based on the ZF path augmenting, since the symbol corresponding to bit 9–bit 12 is not detected until the sixth stage. The ZF path augmenting is also used in the seventh SPE, since the symbol corresponding to bit 13–bit 16 is not detected until the eighth stage. Therefore, the fifth and seventh SPE need to estimate the corresponding symbol by rounding the extra $\mathbf{z}$ input.

Fig. 15 shows the structure of the fifth SPE. It consists of three symbol metric calculation (SMC) units. Similarly, the fourth SPE consists of two SMC units, the seventh SPE consists of four SMC units and so on. Each SMC unit consists of four bit metric calculation (BMC) units and one DEMOD unit. The DEMOD unit, implemented in a 4-to-4 combinational decoder, demodulates the detected symbol corresponding to four bits. Each output bit of the DEMOD unit is an input to one of four

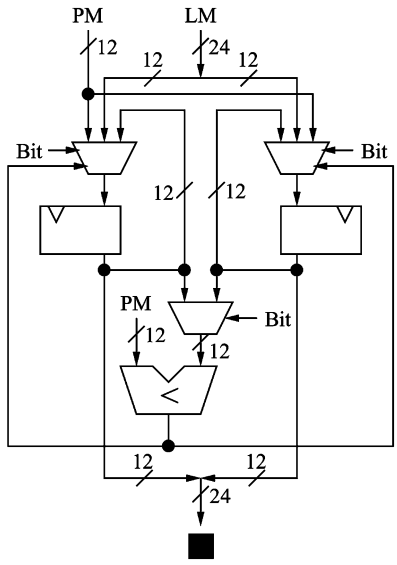


Fig. 16. BMC unit structure.

TABLE III
BUFFER WORDLENGTH COMPARISON BETWEEN HARD-OUTPUT KSE AND SOFT-OUTPUT MKSE, 4×4 16-QAM

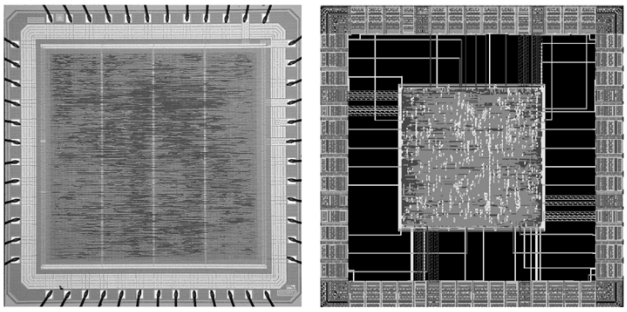
	Wordlength (bits)						Total bits
	G	U	D	Y	E	BM	
Hard-Output KSE	16	2	20	16	16	-	4054
Soft-Output MKSE	10	2	12	10	10	24	4134

BMC units in the SMC unit. The BMC unit accepts the path metrics (PM) in series and the loaded metrics (LM) from the previous stage of SPE, and calculates the corresponding bit metrics. For the unit SMC3 in Fig. 15, the DEMOD unit accepts the symbol estimated by rounding the $\mathbf{z}$ input. Moreover, The LM inputs of SMC3 are fixed to the maximum value corresponding to the chosen wordlength, since the metrics for bit 9–bit 12 are not available from the fourth SPE.

Fig. 16 shows the structure of the BMC unit, which mainly consists of a comparator, two registers and three MUXs. It loads the LM input at the first clock cycles, then accepts the path metrics and performs the comparison in series. After all the path metrics are compared, the bit metrics $BM^{(0)}$ and $BM^{(1)}$ are held at the outputs of the two registers.

F. Soft-Output MKSE Versus Hard-Output KSE

Compared with the hard-output KSE, the additional resources to the soft-output MKSE are 16 4-to-4 combinational decoders, 64 comparators, 64 BM registers, and 2 subtractors used in the SVC unit. However, wordlength requirements are relaxed in the MKSE compared with the hard-output KSE. The MKSE tries to find a range of better points and calculate the soft values, while the hard-output KSE tries to search for the best point corresponding to the hard output. What the MKSE focuses is on the diversity of the searched points, not on the absolute accuracy. That is also why a soft-output MIMO detector always prefers finding as many points as possible. Our fixed-point simulation results prove the statement above. Table III shows the buffer wordlength comparison between the implemented hard-output



(a) Die photo of the hard-output KSE decoder, 0.35-μm CMOS. (b) Layout of the soft-output MKSE decoder, 0.13-μm CMOS.

Fig. 17. Implemented MIMO decoders, 4×4 16-QAM.

TABLE IV
MEASUREMENT RESULTS OF THE HARD-OUTPUT KSE, 4×4 16-QAM, 0.35-μm PROCESS

Clock (MHz)	Throughput (Mb/s)	Latency (μs)	Power (mW @2.8 V)
10	5.3	24	50
20	10.6	12	133
40	21.2	6	272
50	26.7	4.8	335
100	53.3	2.4	626

KSE and soft-output MKSE. Clearly, the wordlength required in the MKSE is much less than that in the KSE. The increase of the total buffer bits in the MKSE is mainly due to the additional BM registers. As proved later in Section VIII, the core equivalent gates number of the MKSE is only 6.5% higher than that of the hard-output KSE. This penalty is worth since the MKSE supports soft outputs.

VIII. IMPLEMENTATION RESULTS

The proposed VLSI architectures are modeled in Verilog HDL, synthesized using Synopsys Design Compiler, and routed using Cadence SoC Encounter/Silicon Ensemble. The RTL and gate level netlists are all verified against the same test vectors generated from the MATLAB fixed-point model. The postlayout timing is verified using Synopsys PrimeTime with net and cell delays back annotated in SDF format [28].

A. Hard-Output KSE Decoder

The hard-output KSE decoder is fabricated in a 0.35-μm CMOS technology. Fig. 17(a) shows the die photo of the chip. The chip core area is 2.4×2.4 mm^2 with 91 K gates. For the 4×4 16-QAM MIMO system, each symbol vector contains 16 bits. The decoding throughput that the chip can support is, thus, equal to $(16/30)f_c$, where f_c denotes the clock frequency. The decoding latency of the chip is equal to $30 * 8/f_c$. The nominal supply is 3.3 V, but the chip was functional at 2.8 V with a 100 MHz clock at room temperature. Table IV shows the decoding throughput, latency, and the measured core power consumption with various clock frequencies.

B. Soft-Output MKSE Decoder

The soft-output MKSE decoder is to be sent for fabrication in a 0.13-μm 6-ML CMOS technology. Fig. 17(b) shows the layout of the chip. The chip core area is 0.75×0.75 mm^2 with 97 K gates. The postlayout timing simulation shows that the chip can be operated at a maximal clock frequency of 200 MHz. The maximal decoding throughput of the MKSE is, thus, expected to be more than 100 Mb/s, and the corresponding decoding latency is 1.2 μs. The implemented MKSE chip can make its soft-output module work in sleeping mode in order to reduce its power consumption. This flexibility is useful when the hard-output MIMO decoding could meet the system requirements.

TABLE V
SILICON IMPLEMENTATION COMPARISON, ALL THE HARD-OUTPUT DECODERS CAN ACHIEVE ML PERFORMANCE WITHIN THE INVESTIGATED RANGE OF SNR, 4×4 16-QAM

	Hard-Output			Soft-Output
Algorithm Preprocessing assumed	Sequential SE [27] mode 1	K-best SD [12] mode 1	KSE mode 3	MKSE mode 3
Process (μm)	0.25	0.35	0.35	0.13
Equivalent gates number (K)	117	109	91	97
Max. clock frequency (MHz)	51	100	100	200
Decoding throughput (Mb/s)	variable	10	53.3	106.6
Decoding latency (μs)	variable	12.8	2.4	1.2

TABLE VI
SILICON COMPLEXITY COMPARISON OF SOFT-OUTPUT DECODING, 4×4 16-QAM, ESTIMATED WITH 0.18-μm AND 122.88 MHz CLOCK FREQUENCY

	LSD (K=256) [29]	MKSE (K=5)
Area (mm^2)	8.38	1.07
Decoding throughput (Mb/s)	38.4	65.5

C. Comparison to Other MIMO Decoders

Table V shows the silicon implementation comparison between the published hard-output decoders and our work. The equivalent gates number is defined as the total core area divided by the area of a drive-1 NAND gate. All the hard-output decoders shown in the table can achieve ML performance within the, respectively, investigated range of SNR. The number of resources of [12], including RAMs, is translated to equivalent gates number. Both KSE and MKSE outperform the K-best SD in complexity and decoding throughput. The decoding throughput of [27] is variable due to employing the sequential SE type of algorithm. [27] can achieve a much higher peak throughput than the KSE or MKSE, but suffers from the instantaneous throughput loss under worse channel conditions.

For soft-output 4×4 16-QAM MIMO decoding, there has been only one silicon complexity estimation published [29]. It uses the LSD $(K = 256)$ algorithm and its complexity is estimated in a 0.18-μm CMOS process. To enable a comparison with [29], we scaled the MKSE decoder from 0.13-μm to 0.18-μm [28]. Table VI shows a rather rough comparison in silicon complexity between [29] and the MKSE. The comparison results show that the MKSE $(K = 5)$ can achieve higher decoding throughput with lower complexity compared with the LSD $(K = 256)$. It should be noted that the BER performance of the LSD $(K = 256)$ is better than that of the MKSE $(K = 5)$ even at the first iteration, as shown in Fig. 9.

IX. CONCLUSION

In this paper, the KSE algorithm is proposed to approach near-ML performance for MIMO detection, which is shown to

be capable of supporting soft outputs. The MKSE algorithm is further proposed to improve the performance of the soft-output KSE with minor modifications. The simulation results show that the MKSE can approach the performance of the LSD proposed in [6] with lower complexity. Moreover, a VLSI architecture is proposed for both algorithms. There are several low complexity and low-power features incorporated in the proposed algorithms and the VLSI architecture. The implementation results show that it is feasible to achieve near-ML performance and high detection throughput for 4×4 16-QAM MIMO detection using the proposed algorithms and the VLSI architecture with reasonable complexity.

ACKNOWLEDGMENT

The authors would like to thank J. Liu (IMEC, Belgium) for the generous help on the soft-output decoding simulations. They would also like to thank A. Burg (ETH, Zurich) for the helpful discussions on the depth-first and breadth-first algorithms. They would also like to thank the anonymous reviewers for their careful review and precious comments.

REFERENCES

[1] G. J. Foschini, "Layered space-time architecture for wireless communication in fading environments when using multiple antennas," *Bell Labs. Tech. J.*, vol. 2, pp. 41–59, 1996.

[2] D. Garrett, L. Davis, S. ten Brink, and B. Hochwald, "APP processing for high performance MIMO systems," in *Proc. IEEE Custom Integrated Circuits Conf.*, San Jose, CA, Sep. 2003, pp. 271–274.

[3] A. Burg, N. Felber, and W. Fichtner, "A 50 mbps 4×4 maximum likelihood decoder for multiple-input multiple-output systems with QPSK modulation," in *Proc. 10th IEEE Int. Conf. Electron., Circuits, Syst. (ICECS)*, Dec. 2003, pp. 322–335.

[4] M. 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.

[5] B. Hassibi and H. Vikalo, "On the expected complexity of sphere decoding," in *Proc. Asilomar Conf. Signals, Syst., Comput,*, 2001, pp. 1051–1055.

[6] 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.

[7] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Comput.*, vol. 44, pp. 463–471, 1985.

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

[9] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Programming*, vol. 66, pp. 181–191, 1994.

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

[11] G. Rekaya and J.-C. Belfiore, "Complexity of ML lattice decoders for the decoding of linear full rate space-time codes," IEEE *Trans. Wireless Commun.*, to be published.

[12] 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 *Proc. IEEE Int. Symp. Circuits Syst.*, May 2002, pp. III-273–III-276.

[13] J. B. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Commun.*, vol. COM-32, pp. 169–176, Feb. 1984.

[14] S. Bäro, J. Hagenauer, and M. Witzke, "Iterative detection of MIMO transmission using a list-sequential (liss) detector," in *Proc. IEEE Int. Conf. Commun.*, 2003, pp. 2653–2657.

[15] Z. Guo and P. Nilsson, "Reduced complexity Schnorr-Euchner decoding algorithms for MIMO systems," *IEEE Commun. Lett.*, vol. 8, no. 5, pp. 286–288, May 2004.

[16] Y. L. de Jong and T. J. Willink, "Iterative tree search detection for MIMO wireless systems," in *Proc. IEEE 56th Veh. Technol. Conf.*, Sep. 2002, pp. 1041–1045.

[17] D. L. Ruyet, T. Bertozzi, and B. Özbek, "Breadth first algorithms for APP detectors over MIMO channels," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2004, pp. 926–930.

[18] Z. Guo and P. Nilsson, "VLSI implementation issues of lattice decoders for MIMO systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, Vancouver, BC, Canada, May 2004, pp. IV-477–IV-480.

[19] B. Hassibi. An efficient square-root algorithm for BLAST. [Online]. Available: http://mars.bell-labs.com/

[20] P. A. Bengough and S. J. Simmons, "Sorting-based VLSI architecuture for the M-algorithm and T-algorithm Trellis decoders," *IEEE Trans. Commun.*, vol. 43, pp. 514–522, 1995.

[21] Z. Guo and P. Nilsson, "A VLSI architecture of the Schnorr-Euchner decoder for MIMO systems," in *Proc. IEEE 6th CAS Symp. Emerging Technol.: Frontiers of Mobile and Wireless Communication*, Shanghai, China, Jun. 2004, pp. 65–68.

[22] B. Parhami, *Computer Arithmetic, Algorithms and Hardware Designs*. Oxford, U.K.: Oxford Univ. Press, 2000.

[23] A. Wiesel, X. Mestre, A. Pages, and J. R. Fonollosa, "Efficient implementation on sphere demodulation," in *Proc. IEEE Workshop on Signal Processing Advances in Wireless Commun.*, Rome, Italy, Jun. 2003, pp. 36–40.

[24] M. O. Damen, H. E. Gamal, and N. C. Beaulieu, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2372–2388, Oct. 2003.

[25] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. ISSSE*, Pisa, Italy, Sep. 1998, pp. 295–300.

[26] M. Borgmann and H. Bölcskei, "Linear interpolation based matrix inversion for MIMO-OFDM receivers," in *Proc. 38th IEEE Asilomar Conf. Signals, Syst. Comput.*, Nov. 2004, pp. 1941–1947.

[27] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, to be published.

[28] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits a Design Perspective*, 2nd ed., 2003.

[29] 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.

Zhan Guo received the B.Sc. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 1996, the M.Sc. degree in microelectronics from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree in electroscience from Lund University, Lund, Sweden, in 2005.

From 1999 to 2000, he was with the Department of ASIC, Shanghai Institute, Huawei Technologies Company, Ltd., China. His main research interests include VLSI design for wireless communications, low-power circuits design, and system-on-chip design.

Peter Nilsson (S'90–M'96) received the M.S. degree in electrical engineering, the Licentiate of Engineering degree in 1992, the Ph.D. degree in engineering in 1996, and the Docent degree in 2003, from Lund Institute of Technology, Lund University, Lund, Sweden.

After receiving the Ph.D. degree, he was an Assistant Professor in the Department of Applied Electronics (now Department of Electroscience), Lund University. In November 1997, he became an Associate Professor at the same department. He is also the Program Manager for Socware Research and Education, a national program for research and Master's education on system-on-chip. His main interest is in the field of implementation of digital circuits.

Dr. Nilsson is also an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I. He is a member of the VLSI Systems and Applications Technical Committee in the IEEE Circuits and Systems Society.