Nonlinear Soft-Output Signal Detector Design and Implementation for MIMO Communication Systems with High Spectral Efficiency

Sizhong Chen, Fei Sun, and Tong Zhang
ECSE Department, Rensselaer Polytechnic Institute, Troy, NY

***Abstract*— VLSI implementations of nonlinear MIMO signal detectors are not trivial, particularly for systems with high spectral efficiency. For example, realization of such a detector for 4×4 MIMO with 64-QAM still remains missing in open literature. To tackle this challenge, we developed a nonlinear soft-output detector design solution, based on which a detector for up to 4×4 MIMO with 64-QAM has been designed using 0.13μm CMOS technology. Above 75 Mbps detection throughput has been verified based on post-layout results.**

I. Introduction

Due to its potential of largely increasing the spectral efficiency, multiple-input multiple-output (MIMO) technology is being considered for a wide use in future wireless communication systems [1]. Nevertheless, the computational complexities of the optimum maximum-likelihood (ML) hard-output and maximum *a posteriori* (MAP) soft-output MIMO detection grow exponentially with spectral efficiency. To largely reduce the computational complexity while maintaining (near-)optimum detection performance, researchers have developed several nonlinear detection schemes that realize hard-/soft-output detection through certain *non-exhaustive tree search*. Although the past several years experienced a significant progress on nonlinear detector VLSI design (e.g., see [2]–[6]), prior work can only support up to 16 quadrature amplitude modulation (QAM) modulation for moderate-size MIMO (such as 4×4). This leaves the design solution that can support 64-QAM for moderate-size MIMO missing in the open literature, whereas the support of 64-QAM is essential for realizing high spectral efficiency in future communication systems.

As an attempt to fill this gap, this paper presents a nonlinear detector design solution that can support 4×4 MIMO with 64-QAM modulation. This is realized by developing two algorithm level techniques, including: (1) We replace the strict sorting in the conventional *K*-best detectors [5], [6] with a *distributed* and *approximate* sorting in order to largely improve the operational throughput and reduce the power consumption while maintaining good detection performance; (2) We modify the PSK enumeration technique proposed in [3], [7] to further reduce the computational complexity for higher order modulation such as 64-QAM. VLSI architectures in support of these algorithm level techniques are further developed, and the resulted detector is referred to as relaxed *K*-best detector. To demonstrate its effectiveness, we designed a soft-output relaxed *K*-best detector with 0.13μm CMOS standard cell and SRAM libraries. Post-layout simulation and analysis results show that this detector can achieve above 75 Mbps for 4×4 MIMO with 64-QAM with the silicon area of about 20mm^2 and power consumption of 847mW.

II. Background

A. System Model

This work considers the MIMO system with spatial multiplexing signaling (i.e., the signals transmitted from individual antennas are independent of each other). Let N_t and N_r represent the number of transmit and receive antennas, respectively. Assume that the transmitted symbol is taken from a W-QAM constellation with $W = 2^q$. At once, the transmitter maps one $qN_t \times 1$ binary vector $\mathbf{x}$ to an $N_t \times 1$ symbol vector $\mathbf{s}$. The transmission of each vector $\mathbf{s}$ over flat-fading MIMO channels can be modeled as $\mathbf{y} = \mathbf{H} \cdot \mathbf{s} + \mathbf{n}$, where $\mathbf{y}$ is an $N_r \times 1$ signal vector received by a MIMO detector, $\mathbf{H}$ is an $N_r \times N_t$ channel matrix, and $\mathbf{n}$ is a noise vector whose entries are independent complex Gaussian random variables with mean zero and variance $N_0/2$.

B. Soft-Output MIMO Signal Detection

The task of a soft-output detector is to compute the log-likelihood ratio (LLR) value of each bit, defined as $L(x_i|\mathbf{y}) = \ln \frac{P(x_i=+1|\mathbf{y})}{P(x_i=-1|\mathbf{y})}$, where x_i denotes the i-th bit of the binary vector $\mathbf{x}$. Through standard simplification [7], [8], $L(x_i|\mathbf{y})$ can be approximated as:

$$\begin{aligned} L(x_i|\mathbf{y}) &\approx \max_{x_i=+1}\{\Lambda(\mathbf{x},\mathbf{y}\} - \max_{x_i=-1}\{\Lambda(\mathbf{x},\mathbf{y})\}, \\ \text{where } \ \Lambda(\mathbf{x},\mathbf{y}) &= -\frac{1}{N_0}\|\mathbf{y} - \mathbf{H}\cdot\mathbf{s}\|^2. \end{aligned} \tag{1}$$

Using standard matrix decompositions such as Cholesky or QR decomposition, we can obtain $\mathbf{H}^*\mathbf{H} = \mathbf{L}^*\mathbf{L}$, where $\mathbf{L} = (l_{i,j})$ is a lower triangular matrix and $(\cdot)^*$ denotes the complex conjugate transpose. Let $\hat{\mathbf{s}} = (\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^*\mathbf{y}$, we have

$$\begin{aligned} \|\mathbf{y} - \mathbf{H}\cdot\mathbf{s}\|^2 &= (\mathbf{s}-\hat{\mathbf{s}})^*\mathbf{L}^*\mathbf{L}(\mathbf{s}-\hat{\mathbf{s}}) \\ &\quad + \mathbf{y}^*(\mathbf{I} - \mathbf{H}(\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^*)\mathbf{y}. \end{aligned} \tag{2}$$

Since the second term in (2) is independent of $\mathbf{s}$ and the matrix $\mathbf{L}$ is lower triangular, we can rewrite $\Lambda(\mathbf{x},\mathbf{y})$ in (1) as

$$\Lambda(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N_t}\Big(-\frac{1}{N_0}\Big|\sum_{j=1}^{i} l_{i,j}(s_j - \hat{s}_j)\Big|^2\Big) = \sum_{i=1}^{N_t}\Lambda_i^s. \tag{3}$$

Hence, we obtain *additive metrics* with the metric increment Λ_i^s that only depends on s_j for $j \le i$. Thus, reduced-complexity nonlinear soft-output MIMO detection can be formulated as an N_t-depth W-ary tree search problem: non-exhaustively search through this tree and find a list of tree leaves, based on which the L-values can be evaluated according to (1). Since the term $-\frac{1}{N_0}$ in (3) can be omitted in the tree search, we define the metric increment Λ_i as $|\sum_{j=1}^{i} l_{i,j}(s_j - \hat{s}_j)|^2$ and the metric of a depth-n path as $\Gamma^{(n)} = \sum_{i=1}^{n}\Lambda_i$. The non-exhaustive tree search can be realized in either depth-first or breath-first manner. This work is interested in nonlinear detector based on breadth-first non-exhaustive tree search.

C. Breadth-First Tree Search K-Best Detector

Broadly speaking, breadth-first tree search extends all the survivor paths at each tree depth at once, purge some paths according to certain criterion, and then continue on to the next tree depth. Based on the principle of *M*-algorithm, a *K*-best MIMO signal detector

[5], [6] performs the following operations at each depth: (i) *Path Extension*: Extend each survivor path from the previous depth with the W modulation points. (ii) *Radius Check*: Delete the extended paths whose metrics are larger than a fixed radius r. (iii) *Path Sort*: Let R denote the number of the remaining extended paths. If $R > K$, then sort these R paths in ascending order based on the path metric and select the first K paths as survivors depth, otherwise all the R extended paths are survivors. After reaching the tree leaves at depth N_t, the detector keeps all the survivors as a list of candidates, based on which the L-values are calculated according to (1).

III. Proposed Relaxed K-Best Detector Design

A. Distributed and Approximate Sorting

For the VLSI implementation of a K-best detector, due to its serial nature and hence large delay, the sorting operation at each depth fails to match the inherent parallelism within the path extension and radius check and becomes the essential detector throughput bottleneck. Furthermore, strict sorting will also incur large silicon overhead and a large amount of data movement, which will directly lead to high power consumption.

To tackle this issue, we propose to modify the K-best detector by replacing the original strict sorting with distributed and approximate sorting to improve the overall operational parallelism: Given a detector parallelism factor β, at the same time β survivor paths can be extended in parallel and all the extended paths from the same survivor path are processed by one individual sorter. All the β sorters perform approximate sorting in parallel and independently from each other. The basic idea of approximate sorting can be described as *sorting with a coarse granularity*: Given the fixed threshold r, we divide the entire range of the path metric (i.e., [0, r]) into a certain number of regions and group the paths whose path metrics fall into the same region. The paths in the same group are not sorted at all. Such approximate sorting only involves the comparison with fixed threshold values, which can be directly implemented in parallel.

As illustrated in Fig. 1, we can use a single-port memory to implement the approximate sorting as follows: We uniformly partition the memory address space into d consecutive segments. Each segment S_i is associated with a configurable threshold range $(u_{i-1}, u_i]$. A path is stored into the memory segment S_i if its metric falls into the range $(u_{i-1}, u_i]$. Each segment has one counter to hold the address of the next available memory location. Initially, we set $u_i = t_i$ for $i = 0, 1, \cdots, d$ where $t_i = i \cdot r/d$. In case that one memory segment, e.g., S_i, becomes full, we hand over the range of segment S_i to the next segment S_{i+1} by setting u_i equal to u_{i-1}, i.e., the range of S_{i+1} is extended from $(t_i, t_{i+1}]$ to $(t_{i-1}, t_{i+1}]$ and S_i is closed to further write since its range becomes $(t_{i-1}, t_{i-1}]$.

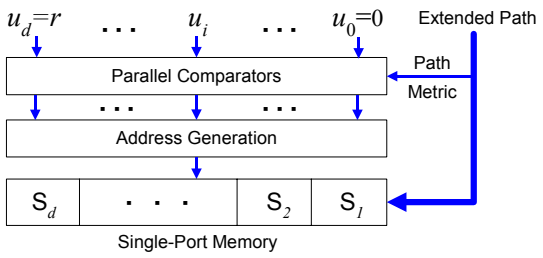

Fig. 1. Realization of the approximate sorting.

From the hardware implementation standpoint, this memory based distributed and approximate sorting has the following main advantages: (i) The distributed structure well matches the parallelism in the path extension and hence helps to realize high throughput; (ii) The computational complexity is much less than the strict sorting, leading to a great potential of higher throughput and significant power savings; (iii) There is no data movement at all, which will help to further reduce the power consumption.

B. Improved PSK enumeration

From Section II, we have that each metric increment Λ_i at depth-i is calculated as $|P_c - P_s|^2$, where $P_c = G_i - \sum_{j=1}^{i-1} l_{i,j} s_j$, $G_i = \sum_{j=1}^{i} l_{i,j} \hat{s}_j$ and $P_s = l_{i,i} s_i$. G_i is common to each received vector and can be pre-computed. P_c is common to all the paths extended from the same survivor, while P_s depends on the modulation point to which the survivor is extended. Due to the complex computation involved in each path extension, *explicitly* examining the extension of each survivor towards all the modulation points will incur a large computational complexity overhead. To tackle this issue, a technique called PSK enumeration [3], [7] has been developed, where the basic idea can be described as follows: For QAM modulation, all the modulation points locate on several circles concentric with the origin, e.g., there are 1, 3, and 9 concentric circles in QPSK, 16-QAM, and 64-QAM, respectively. All the modulation points on the same circle that satisfy the radius check are always adjacent and hence form a single admissible region along that circle. By identifying the boundary of such an admissible region through a zigzag search on each circle, we do not need to explicitly examine the modulation points outside the admissible region, hence the computational complexity will be reduced. However, each concentric circle still has to be explicitly examined for the search of an admissible region.

In this work, we improved the PSK enumeration method so that some concentric circles may be excluded from explicit examination. This will further reduce the computational complexity, particularly for higher order modulation such as 64-QAM that has a relatively large number of concentric circles. Denote the circles that contain at least one modulation point that satisfies the radius check as *valid circles*. From the discussion on PSK enumeration in [7], it can be readily derived that all the valid circles fall into a continuous region. Therefore, by identifying the boundary of such a continuous region, we will not need to explicitly examine the circles outside of this region. Such a boundary can be identified as follows: For a modulation point at depth-i to survive the radius check, it must satisfy $|\frac{P_c}{l_{i,i}} - s_i|^2 < \frac{r - \Gamma^{(i-1)}}{l_{i,i}^2}$. We obtain the boundary of the valid circle region by locating the circle closest to the point $\frac{P_c}{l_{i,i}}$ on each side and extending from these two circles with the distance of $r_e = \frac{r - \Gamma^{(i-1)}}{l_{i,i}^2}$ along both inward and outward directions. Any circle that falls outside does not contain any modulation points that may satisfy the radius check, hence can be simply excluded. Note $1/l_{i,i}$ can be pre-computed during channel estimation and the computation here involves multiplication instead of division.

C. Detector Architecture

Using the above two design techniques, i.e., distributed and approximate sorting and improved PSK enumeration, a relaxed K-best detector can be realized by integrating several identical recursive detector cores that operate in parallel and independently on different received signal vectors, as illustrated in Fig. 2(a). The structure of each detector core is shown in Fig. 2(b), which iterates N_t times to finish the detection of one received signal vector for a MIMO system with N_t transmit antennas. We note that this detector does not contain the pre-computation blocks that perform channel matrix $\mathbf{H}$ decomposition and calculate $\hat{\mathbf{s}} = (\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^*\mathbf{y}$ and $G_i =$

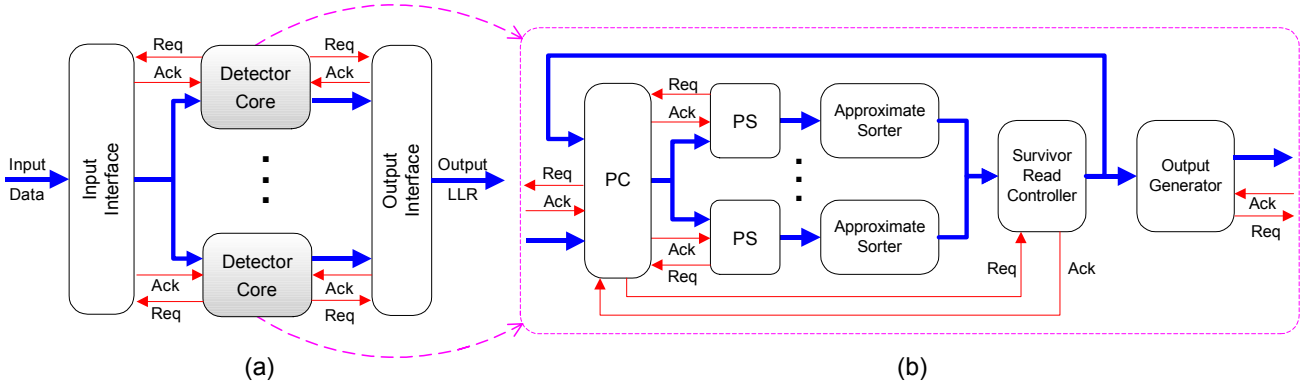

Fig. 2. Structure diagrams of (a) overall detector and (b) each recursive detector core.

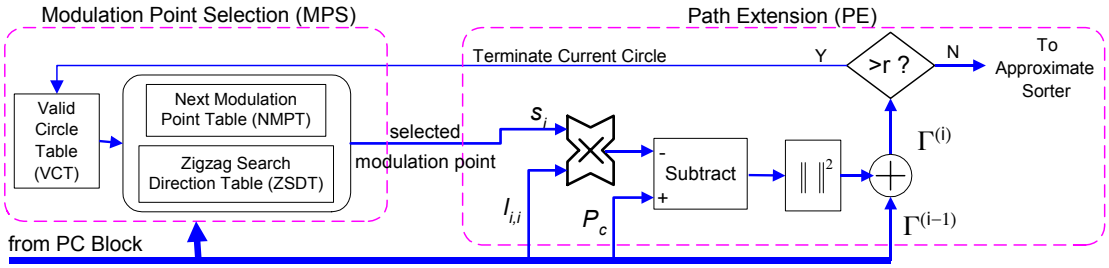

Fig. 3. Structure diagram of a PS block.

$\sum_{j=1}^{i} l_{i,j}\hat{s}_j$. Each recursive detector core contains one PC block that is shared by several PS blocks, where the PC block performs certain pre-calculation for the extension of each survivor path and each PS block carries out the path extension using the improved PSK enumeration method. Each PS block has its own approximate sorter. To accommodate the dynamically varying computational load for PSK enumerations, as illustrated in Fig. 2, the entire detector data flow is driven by handshake through pairs of *Req* and *Ack* signals. The recursive detector core design is further described as follows.

Upon receiving one survivor path, each PC block performs the following operations: (a) calculates $P_c = \sum_{j=1}^{i-1} l_{i,j}(\hat{s}_j - s_j) + l_{i,i}\hat{s}_i$, as mentioned earlier, which is shared among the succeeding path extensions, (b) identifies the boundary of valid circle region, and (c) determines the starting point and initial direction for the zigzag search on each valid circle. Although the involved computation tends to be very complex, the PC block can be deeply pipelined to support a high data processing throughput.

Extending one survivor at a time, each PS block non-exhaustively examines the modulation points for path extension using the improved PSK enumeration method and sends the extended paths to its own approximate sorter. Fig. 3 shows the structure diagram of one PS block, which contains two main sub-blocks including modulation point selection (MPS) and path extension (PE). Each clock cycle, MPS selects and feeds one modulation point to PE for path extension. When PE detects a modulation point out of admissible region (i.e., the extended path metric $\Gamma^{(i)}$ is larger than the radius r), it will send a termination request to MPS so that MPS will not feed any other modulation points on the same circle to PE. Nevertheless, since PE should be deeply pipelined in order to achieve high throughput, if MPS keeps feeding the modulation points on the same circle to PE, the PE pipeline will be filled with modulation points out of admissible region when MPS receives the termination request. This may largely degrade the PE pipeline utilization efficiency and reduce the computation saving of PSK enumeration. To solve this problem, we make MPS feed the modulation points to PE alternatively among all the valid circles: As shown in Fig. 3, MPS maintains three tables, including valid circle table (VCT) that stores the indices of valid circles, next modulation point table (NMPT) that stores the next modulation point to be extended on each valid circle, and zigzag search direction table (ZSDT) that stores the present zigzag search direction on each valid circle. All these tables are initialized by the data sent from the PC block. Each clock cycle, MPS feeds PE with one modulation point on the valid circle pointed by a valid circle pointer, updates the NMPT and ZSDT accordingly, and then make the valid circle pointer point to the next valid circle in VCT. If MPS receives a circle termination request from PE, it simple removes the corresponding circle index from VCT. This alternative modulation point fetching approach can largely improve the PE pipeline utilization efficiency for high order modulations.

Each approximate sorter contains two single-port memory blocks that receive the data from the current depth and provide the data to the next depth, alternatively. Controlled by the survivor read controller as shown in Fig. 2(b), approximate sorters send the survivors back to PC block as follows: We start with fetching one path at a time from segment S_1 in each single-port memory as survivor, alternatively among all the single-port memories, until we have fetched K paths or all the paths stored in the S_1 segments have been fetched. If latter happens, we move on to the segment S_2, and so on. When the last depth N_t is reached, an output generator is invoked to generate LLR according to (1). In case that all the last-depth survivors agree on one bit position, i.e., the corresponding $\Gamma_{i,+1}$ or $\Gamma_{i,-1}$ remains as undefined, we use $|\Gamma_{wst} - \Gamma_{bst}|$ as the magnitude of the L-value for those bits, where Γ_{bst} and Γ_{wst} represent the best and worst metrics of the last-depth survivors, respectively. When the output

generator is calculating LLR, the detector core requests the next received signal vector and those associated pre-computed data from the input interface.

IV. Detector ASIC Design

We designed a soft-output relaxed *K*-best detector for 4×4 MIMO with 64-QAM modulation. The design parameters are outlined as follows: It contains 9 detector cores, each one has 1 PC block and 8 PS blocks. Each single-port memory in the approximate sorters can store 128 path data (including the path symbols and path metric) and is partitioned into 16 segments. The path metric is represented by 8 bits, and each entry in the matrix $\mathbf{L}$ and vector $\hat{\mathbf{s}}$ is represented by 15 and 16 bits, respectively. Our computer simulations suggest that such finite wordlength configuration incurs negligible performance degradation from using floating-point precision.

The detection performance is evaluated based on fixed-point computer simulations with the following configurations: We consider LDPC-coded MIMO-OFDM system with 64-point FFT. Each subcarrier MIMO channel remains constant during transmission of a complete frame and is flat fading, i.e., all the entries in the MIMO channel matrix are independent random Gaussian variables. The LDPC code has a code rate of 1/2 and code length of 2304, and the LDPC code decoder performs up to 20 decoding iterations. There is no iteration between detector and decoder. For the definition of the MIMO channel SNR, we follow the one proposed in [7]. For the purpose of comparison, we also performed the simulation using soft-output exhaustive search sphere detectors that *exhaustively* examines all the paths that satisfy the sphere radius check. Both detectors use the same radius r calculated as $2\alpha N_r\sigma^2$ [7], where α is empirically set as 6 and σ is the noise standard deviation. Fig. 4 shows the simulated frame error rate (FER).

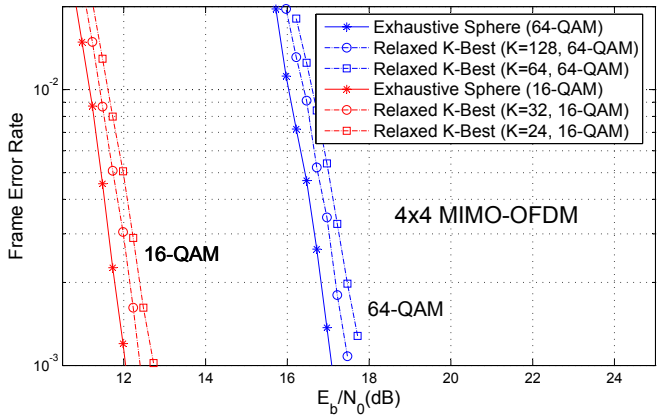

Fig. 4. Simulated FER vs. SNR.

Using Chartered 0.13μm CMOS standard cell and SRAM libraries with 8 metal layers, we designed one recursive detector core. Synopsys tools were used throughout the design hierarchy down to place and route. The post-layout simulation results have been verified against fixed-point C testbench. Fig. 5 shows the detector core layout, where the dark area in the layout is occupied by SRAM.

By integrating 9 identical recursive detector cores, this relaxed-*K* detector has the implementation metrics summarized in Table I based on the post-layout power estimation and static timing analysis. Due to the use of PSK enumeration method, the computational load and instantaneous throughput of the detector dynamically vary and depend on run-time channel conditions. Therefore, we carried out extensive post-layout simulations to estimate the average detection throughput under different SNRs. Table II shows the estimated average detection throughput at four different SNRs for 4×4 MIMO 64-QAM when $K = 64$.

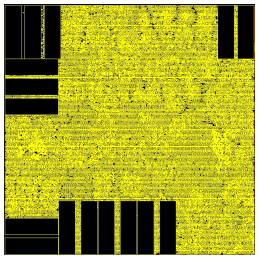

Fig. 5. Layout of one detector core.

TABLE I
IMPLEMENTATION METRICS OF THE DETECTOR.

# of cores	V_{DD}	f_{clk}	Area	Gate Count	Power
9	1.2 V	270 MHz	21.4 mm^2	1.79 M	847 mW

TABLE II
AVERAGE THROUGHPUT OF THE DETECTOR.

E_b/N_0 (dB)	17.7	17.2	16.7	16.2
Throughput (Mbps)	77.1	74.0	71.4	69.1

V. Conclusions

This paper presents a near-optimum nonlinear breadth-first signal detector design solution for MIMO systems with high spectral efficiency. Algorithm level techniques and corresponding VLSI architectures are developed to ensure good hardware implementation efficiency while maintaining near-optimum detection performance. With 0.13μm CMOS standard cell and SRAM libraries, we designed a soft-output detector that, for the first time ever reported in the open literature, can support 4×4 MIMO with 64-QAM at reasonable silicon area and power consumption cost.

References

[1] N. Jayant (ed.), "Special Issue on Gigabit Wireless," *Proceedings of the IEEE*, vol. 92, Feb. 2004.

[2] D. Garrett et al., "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 1544–1552, Sept. 2004.

[3] A. Burg et al., "VLSI implementation of MIMO detection using the sphere decoding algorithm," *Journal of Solid-State Circuits*, vol. 40, pp. 1566 – 1577, July 2005.

[4] D. Garrett et al., "A 28.8 Mb/s 4×4 MIMO 3G CDMA receiver for frequency selective channels," *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 320–330, Jan. 2005.

[5] K.-W. Wong, C.-Y. Tsui, R. S. Cheng, and W.-H. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *IEEE International Symposium on Circuits and Systems*, May 2002, pp. III–273–III–276.

[6] Z. Guo and P. Nilsson, "Algorithm and implementation of the k-best sphere decoding for mimo detection," *IEEE Journal on Selected Areas in Communication*, vol. 24, pp. 491–503, March 2006.

[7] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Transactions on Communications*, vol. 51, pp. 389–399, March 2003.

[8] S. Baro, J. Hagenauer, and M. Witzke, "Iterative detection of MIMO transmission using a list-sequential (LISS) detector," in *Proc. of IEEE International Conference on Communications*, May 2003, pp. 2653–2657.