Implementation and Complexity Analysis of List Sphere Detector for MIMO-OFDM systems

Markus Myllylä and Markku Juntti
Centre for Wireless Communications
P.O. Box 4500, FIN-90014 University of Oulu, Finland
{markus.myllyla, markku.juntti}@ee.oulu.fi

Joseph R. Cavallaro
Dept. of Electrical & Computer Engineering
Rice University, Houston, TX 77251-1892, USA
cavallar@rice.edu

Abstract—A list sphere detector (LSD) is an enhancement of a sphere detector (SD) that can be used to approximate the soft output maximum *a posteriori* probability (MAP) detector used in the detection of the multiple-input multiple-output (MIMO) signals. The LSD consists of three different parts: the preprocessing unit, the LSD algorithm unit and the log-likelihood ratio (LLR) calculation unit. Architecture design is the key point to enable an efficient implementation of the LSD. In this paper, we design the architecture for the whole detector structure and exploit the parallelism and pipelining possibilities of the presented architecture units. The designed architecture is implemented in a field programmable gate array (FPGA) using Mentor Graphics Catapult C tool. We show that a scalable architecture can be designed for the LSD. The LSD is also shown to be feasible for practical implementation, and the implementation complexity and latency results are presented.

I. INTRODUCTION

The ever increasing data rates in wireless communication systems require the use of the available bandwidth as efficiently as possible to maximize the capacity of the system. The multiple-input multiple-output (MIMO) concept in combination with orthogonal frequency division multiplexing (MIMO-OFDM) has been adapted to multiple wireless telecommunication standards, such as the 3rd generation partnership project (3GPP) long term evolution (LTE) and IEEE 802.16e. The optimal joint detection and decoding of a MIMO signal with forward error coding (FEC) can be approximated with an iterative (turbo type) receiver with separate detector and decoder [1], where the optimal soft output detector is the maximum *a posteriori* probability (MAP) detector. However, the computational complexity of the MAP detector is an exponential function of the number of transmit antennas and modulation levels, and, thus, it is not typically promising in practical implementation. A list sphere detector (LSD) [1] is a variant of the sphere detector (SD) [2], [3] that can be used to approximate MAP detector with much lower computational complexity [1], [4].

The architecture design is a key point in efficient implementation of an algorithm. In this paper, we identify and introduce the key functional units of the LSD, and design a highly parallel and scalable architecture for MIMO-OFDM systems. The possibilities for parallelism and pipelining in the microarchitecture units are introduced and analyzed. The designed architecture is implemented on a Virtex-IV field programmable gate array (FPGA) chip for 4×4 MIMO

system with 16- quadrature amplitude modulation (QAM). The implementation is done using Mentor Graphics' Catapult $\text{\textcircled{R}}$ C Synthesis tool [5] with high-level ANSI C++ language, which is then completely synthesized to produce the resulting RTL. We present the complexity and latency results of the implementation and analyze the major challenges of the implementation.

The paper is organized as follows. The MIMO signal model, the SD principles, and the LSD are presented in Section II. The list sphere detector architecture details are introduced in Section III. The LSD implementation trade-offs and results are presented in Section IV. Conclusions are drawn in Section V.

II. MIMO SIGNAL DETECTION

An OFDM based multiple-antenna system with N_T transmit (TX) antennas and N_R receive (RX) antennas is considered with assumption $N_R \geq N_T$ and QAM constellation. The received signal at baseband can be expressed in terms of symbol interval as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \boldsymbol{\eta}, \quad (1)$$

where the received signal vector $\mathbf{y} \in \mathbb{C}^{N_R \times 1}$, the transmit symbol vector $\mathbf{x} \in \Omega^{N_T} \subset \mathbb{C}^{N_T \times 1}$ and the noise vector $\boldsymbol{\eta} \in \mathbb{C}^{N_R \times 1}$ are defined in the frequency domain. The elements of $\boldsymbol{\eta}$ are independent and complex zero-mean Gaussian with equal power σ^2 for both real and imaginary parts. The channel matrix $\mathbf{H} \in \mathbb{C}^{N_R \times N_T}$ contains complex Gaussian fading coefficients with unit variance. The entries of $\mathbf{x}$ are chosen independently from a complex QAM constellation Ω with Q bits per symbol, i.e., the uncoded transmission rate is $R = N_T Q$ bits per channel use (bpcu). The complex system model in (1) can be reduced into an equivalent real model with new real dimensions $M_T = 2N_T$, $M_R = 2N_R$ [3]. We assume a practical case of system with forward error coding (FEC) and with separate soft-input soft-output (SISO) detector and decoder at the receiver as shown in Figure 1. The turbo principle can be applied in the receiver so that the detector and decoder exchange the information in iterative fashion to approximate the optimal joint detector and decoder [1].

A. Sphere Detection

The sphere detectors (SDs) achieve the hard output maximum likelihood (ML) solution of $\mathbf{x}$ with a reduced number of considered candidate symbol vectors in the search compared to traditional exhaustive search algorithms. Then the sphere

Fig. 1. A coded MIMO-OFDM system model with an iterative receiver structure.

search is done by limiting the search to points that lie inside a M_R -dimensional hyper-sphere $S(\mathbf{y}, \sqrt{C_0})$ centered at $\mathbf{y}$. After QR decomposition (QRD) of the channel matrix $\mathbf{H}$, the condition can be written as [3]

$$\|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{x}\|_2^2 \leq C_0, \quad (2)$$

where C_0 is the squared radius of the sphere, $\mathbf{R} \in \mathbb{R}^{M_R \times M_T}$ is an upper triangular matrix with positive diagonal elements, $\mathbf{Q} \in \mathbb{R}^{M_R \times M_R}$ is an orthogonal matrix, and $\tilde{\mathbf{y}} = \mathbf{Q}^H \mathbf{y}$.

Due to the upper triangular form of $\mathbf{R}$ the values of $\mathbf{x}$ can be solved from (2) level by level using the back-substitution algorithm. Let $\mathbf{x}_i^{M_T} = (x_i, x_{i+1}, \dots, x_{M_T})^T$ denote the last $M_T - i + 1$ components of the vector $\mathbf{x}$. The squared partial ED (PED) of $\mathbf{x}_i^{M_T}$ can be calculated as [4]

$$\begin{aligned} d(\mathbf{x}_i^{M_T}) &= d(\mathbf{x}_{i+1}^{M_T}) + |\tilde{y}_i - \sum_{j=i}^{M_T} R_{i,j} x_j|^2 \\ &= d(\mathbf{x}_{i+1}^{M_T}) + |b_{i+1}(\mathbf{x}_{i+1}^{M_T}) - R_{i,i} x_i|^2, \end{aligned} \quad (3)$$

where $d(\mathbf{x}_{M_T}^{M_T}) = 0$, $b_{i+1}(\mathbf{x}_{i+1}^{M_T}) = \tilde{y}_i - \sum_{j=i+1}^{M_T} R_{i,j} x_j$, $R_{i,j}$ is the (i, j) th term of $\mathbf{R}$ and $i = M_T, \dots, 1$. Depending on the search strategy and the channel realization, the SD searches a variable number of nodes in the tree structure, and aims to find the point $\mathbf{x} = \mathbf{x}_1^{M_T}$, also called a leaf node, for which the ED $d(\mathbf{x}_1^{M_T})$ is minimum.

B. List sphere detector

The performance of a channel coded system may suffer significantly with hard output detector compared to the optimal soft output MAP detector. The list sphere detector (LSD) [1] can be used for obtaining a list of the most probable candidate symbol vectors $\mathcal{L} \in \mathbb{Z}^{N_{\text{cand}} \times N_T}$ as an output, where N_{cand} is the size of the candidate list so that $1 \leq N_{\text{cand}} \leq 2^{Q N_T}$. The list can then be used to approximate the soft output MAP solution with reduced complexity. Depending on the list size N_{cand} , it provides a tradeoff between the performance and the computational complexity. A high level architecture of the list sphere detector, which consists of the preprocessing unit, the LSD algorithm unit and the LLR calculation unit.

The preprocessing unit decomposes the channel matrix $\mathbf{H}$ into upper triangular form as in (2), which enables the symbol-by-symbol tree search. Typically QR decomposition (QRD) is

assumed in literature to perform the channel matrix decomposition into an upper triangular matrix $\mathbf{R}$ and an orthogonal matrix $\mathbf{Q}$, which are given as an input with received signal $\mathbf{y}$ to the LSD algorithm. However, it has been shown that the detection order of the transmitted spatial streams effects to the number of visited nodes, i.e., the algorithm complexity [3], [6]. We assume the use of sorted QRD (SQRD) [7] as preprocessing, where the ordering of the spatial layers is included into modified Gram-Schmidt decomposition process. The SQRD algorithm leads to close to optimal detection order so that the strongest signal is located at the top of the sphere search tree.

The LSD algorithm unit executes the tree search and gives the candidate list $\mathcal{L}$ as an output. In this paper, we consider the increasing radius (IR) - LSD algorithm [8], [9], which is a modification of Dijkstra's algorithm [10] to a LSD algorithm: Dijkstra's algorithm is optimal in the sense of visited number of nodes in the tree structure [10], [9] and the output candidate list $\mathcal{L}$ includes the most probable candidates. The algorithm operates in a sequential fashion, and extends the partial candidate $\mathbf{s} = \mathbf{x}_{i+1}^{M_T}$ and the father candidate $\mathbf{s}_f = \mathbf{x}_{i+2}^{M_T}$ always with the next best admissible nodes x_i and x_{i+1} , if admissible node exists [3]. The algorithm uses two memory sets, where the extended partial of final candidates are stored: the final candidate memory $\mathcal{L}$, which is the size of N_{cand} candidates and the partial candidate memory $\mathcal{S}$, which size is dependent on the executed algorithm iterations. After each iteration, the algorithm continues with the partial candidate with the minimum PED from $\mathcal{S}$ until the PED is larger than the radius C_0 .

The approximation of soft output information $L_D(b_k)$ is calculated in the log-likelihood ratio (LLR) calculation unit using the given candidate list. The *a posteriori* log-likelihood ratio (LLR) can be decomposed by using the Bayes' theorem as [1]

$$\begin{aligned} L_D(b_k) &= \ln \frac{P(b_k = 1)}{P(b_k = 0)} + \ln \frac{p(\mathbf{y}|b_k = 1)}{p(\mathbf{y}|b_k = 0)} \\ &= L_A(b_k) + L_E(b_k|\mathbf{y}), \end{aligned} \quad (4)$$

where $L_A(b_k)$ is the *a priori* information and $L_E(b_k)$ is the extrinsic information of the bits provided by the detector or decoder. The probability $p(\mathbf{y}|b_k)$ can be determined for a system containing Gaussian noise directly from the cost information known about the candidates and then the max-log-MAP approximation can be calculated as [11]

$$L_E(b_k|\mathbf{y}) \approx \max_{\mathbf{x} \in \chi_{k,1}} \left(\frac{-d(\mathbf{x})}{2\sigma^2} \right) - \max_{\mathbf{x} \in \chi_{k,0}} \left(\frac{-d(\mathbf{x})}{2\sigma^2} \right). \quad (5)$$

where $\chi_{k,1} = \{\mathbf{x}|b_k = 1\}$ is the set of bit vectors $\mathbf{x}$ in $\mathcal{L}$ having $b_k = 1$. The performance loss due to max-log-MAP approximation is rather small compared to the more complex log-MAP algorithm.

Fig. 2. A high level architecture of a sorted QR decomposition unit.

III. ARCHITECTURE

A. SQRD

The SQRD algorithm architecture is illustrated in Figure 2. The architecture operates in a sequential fashion, and calculates one row of the $\mathbf{R}$ and one column $\mathbf{q}_i$ of the $\mathbf{Q}$ at a time.

The norm calculation unit calculates the channel matrix column norms, which are used to determine the initial permuting order of the columns. The norm calculation requires a total of M_T^2 multiplication (MUL) operations, and, thus, different levels of parallelism and pipelining can be applied for the microarchitecture of the unit. The control logic unit defines the permutation order of columns at iteration i as $i = 1 \dots M_T$, and controls the calculation units and the memory access. The memory unit is used for storing the $\mathbf{Q}$ and $\mathbf{R}$ matrices during the decomposition. The registers are used to temporarily store the current used rows of $\mathbf{R}$ and columns of $\mathbf{Q}$, and the norm values. The actual calculation of the diagonal element $R_{i,i}$ and the column $\mathbf{q}_i$ is executed in the calculation unit, which requires a square-root, a reciprocal division operation and M_R MULs. Parallelism and pipelining can be applied in the MUL operations. The iterative update unit updates the elements in $R_{i,k}$, the columns $\mathbf{q}_k$, and the norm values $|\mathbf{h}_k|^2$, where $k = 1 \dots M_T$. The update of the variables can be carried out by multiply-and-add (MAC) units, but the number of computations depend on the current iteration i . We designed an efficient time-sharing microarchitecture for the calculations, which enables different levels of parallelism and pipelining, and it is illustrated in Figure 3. The parallel MAC units are time-shared to calculate first the $R_{i,k}$ variable with given k , and then the column $\mathbf{q}_k$ and the norm value $|\mathbf{h}_k|^2$ are updated. The architecture calculates iteratively all k values. As the number of different values assigned for the parameter k varies depending on the decomposition phase, the maximum efficient level of parallelism is to use M_T MAC units.

Fig. 3. The designed microarchitecture for SQRD iterative update unit.

Fig. 4. A parallel and scalable architecture for the IR-LSD algorithm.

B. IR-LSD

The architecture for the soft-output IR-LSD algorithm is designed to include parallel and pipelined operations and it is scalable for different antenna and constellation configurations. The designed architecture, which is designed to have as much parallel processing as possible, operates in sequential fashion, and the main units and the connections between units are illustrated in Figure 4. The SEE and PED units define and extend the selected partial candidate and its father node with the next best admissible nodes and calculates the PEDs of the updated candidates. The partial candidate memory unit is used to store the already extended partial candidates while up to N_{cand} leaf candidates with lowest EDs are stored to the final candidate memory unit. The logic unit defines the candidate(s) to be extended and stored in the next algorithm iteration. This means that the candidates extended in the iteration $D = 1$ are stored to the memory at the same time as the next iteration round candidates are extended. The storing of the partial candidates to the partial memory unit and the storing of final candidate to the final memory unit are then executed in parallel with the SEE and PED units, and, thus, the total latency of one algorithm iteration is then equal to the latency of the highest latency parallel unit plus the latency of the control unit. The total latency of one signal vector detection process, i.e., one algorithm run, is dependent on the required number of algorithm iterations, which is also relative to the number of checked nodes in the search tree. After the algorithm search the output final list $\mathcal{L}$ is given to the LLR calculation unit.

1) *SEE and PED unit*: The two SEE and PED units in the IR-LSD architecture execute the extension of the partial candidate and the father candidate in the algorithm description in parallel. Each SEE and PED unit is divided into two sub-

Fig. 5. The designed microarchitecture for the SEE and PED unit.

units as shown in Figure 5 in detail. The first unit calculates the $b_{i+1}(\mathbf{x}_{i+1}^{M_T})$, which is the part of PED calculation that is independent from the new symbol x_i , as in (3). A total of $M_T - i - 1$ MULs, which can be implemented with different levels of parallelism, are required in the calculation of $b_{i+1}(\mathbf{x}_{i+1}^{M_T})$, where i is the current layer in the search tree and $i_{max} = M_T$. The Schnorr-Euchner enumeration (SEE), which is done the second unit, is designed in a slightly modified fashion from the way presented in (14) in [3]. Instead of calculating the costly and high latency division operation, we calculate the absolute value in (3) with Ω_R different symbols x_i . The calculation can be done with different levels of parallelism, i.e., $1 - |\Omega_R|$ separate parallel MAC units. The desired n th best node is determined by defining first the the node, i.e., symbol, with minimum PED. The information with the sign of the value is used to determine the desired n th best node [3] and the PED is calculated by square operation and the added to the PED of the previous nodes.

2) *Memory units:* The memory units are designed as binary heap [12] data structures, which keep the stored elements in order according to selected definition. The partial candidate memory set $\mathcal{S}$ is implemented as min-heap, where the stored partial candidates are ordered so that the candidate with minimum PED is always sorted to be at the top of the heap. The final memory set $\mathcal{L}$ is implemented as max-heap, where the candidates are sorted according to the maximum PED.

C. LLR calculation

The soft output information $L_D(b_k)$ is calculated with the IR-LSD algorithm output list $\mathcal{L}$ by using the max-log-MAP approximation as in (5). The LLR calculation unit microarchitecture is illustrated in Figure 6. The architecture can be divided into two main parts: the scaling of the ED values and the search of maximum values for each bit. The units can be pipelined to increase the execution speed.

The ED values in the candidate list $\mathcal{L}$ are scaled by multiplying them with the inverse of the noise variance $1/(2\sigma^2)$, i.e., a reciprocal division and a total of N_{cand} MULs are required. Different levels of parallelism and pipelining can be applied for the MUL operations in order to speed up the calculations. The max-log-MAP approximation is calculated for each bit b_k and the calculation requires that all the N_{cand}

Fig. 6. The designed microarchitecture for LLR calculation unit that uses the max-log-MAP approximation.

ED values in the candidate list $\mathcal{L}$ are checked for each QN_T bits in order to determine the maximum values for both bit counterparts. Thus, two sequential logic loops are required in the calculation, what are illustrated with m and k variables in the architecture description. The latency of the loops can be decreased by applying parallel and pipelined logic to check multiple ED values or bits in parallel.

IV. IMPLEMENTATION

The FPGA implementation of the IR-LSD architecture is done for $N_R = N_T = 4$ system with 16-QAM constellation by ANSI C++ language and then synthesized through Mentor Graphics' Catapult C Synthesis tool [5] to produce bit-accurate, parallel hardware. Catapult is used to create complex, high-performance hardware, and allows to quickly experiment with different design specifications for an application specific integrated circuit (ASIC) or an FPGA.

A. Trade-offs and word lengths

The IR-LSD algorithm is a sequential search algorithm and requires a variable number of algorithm iterations to execute the tree search depending on the channel realization. The number of visited nodes can be fixed in the hardware implementation in order to determine the hardware resources and latency of the implementation. An effective and straightforward way to fix the number of iterations is to define a maximum limit for the algorithm iterations D_{max} [13]. We performed Monte Carlo simulations in order to verify the performance of the IR-LSD with the limited search and fixed-point word lengths. A 1/2 rate [13,15] turbo coded MIMO-OFDM system was assumed with $N_T = N_R = 4$, 16-QAM constellation in an uncorrelated typical urban (UNC) 6 tap channel with a velocity of 120 kmph. The receiver includes a IR-LSD with a list size $N_{cand} = 15$, where the absolute values of the soft output LLRs are limited to $|L_D(b_k)| < 8$, and a max-log-MAP turbo decoder with 8 iterations. The performance of the IR-LSD based receiver with different detector configurations is presented in left subplot in Figure 7. The effect of limited IR-LSD search is studied by setting a maximum value D_{max} for the number of executed algorithm iterations. The results show that the IR-LSD works also with limited search and max-log-MAP approximation, and the required maximum and average algorithm iterations D_{max} and D_{avg} with different SNR are shown in right subplot in Figure 7. The determined

Fig. 7. Throughput vs SNR: Performance of the real IR-LSD based receiver in a 4×4 antenna system with 16-QAM constellation (left). The required number of LSD algorithm iterations for target FER (right).

TABLE I
WORD LENGTHS FOR THE IR-LSD IN A 4×4 SYSTEM WITH 16-QAM.

SQRD (W,I)	H (11,3)	R (26,4)	Q (26,3)	norm (27,5)	sqrt() (23,3)	div() (21,3)
LSD alg. (W,I)	$\tilde{y}$ (10,4)	R (9,3)	Ω (8,1)	$b_i(s)$ (12,5)	$d(s)$ (10,5)	
LLR (W,I)	σ (8,0)	LLR (10,5)	$L_D(b_k)$ (6,4)			

word lengths for the IR-LSD are listed in Table I, where the $\mathbf{W}$ and the $\mathbf{I}$ refer to the number of bits used in total and to the integer parts of the representation. It can be seen that the SQRD requires up to 27 bits in internal word lengths to produce accurate enough decomposition of the real 8×8 channel matrix $\mathbf{H}$. We note that the high word lengths could be decreased by introducing internal scaling of the SQRD variables. A maximum of 12 bits is feasible in the IR-LSD internal processing, and the LLR calculation requires 10 bits.

B. Implementation results

The Catapult C[®] Synthesis tool output RTL was synthesized with Mentor Graphics Precision RTL. The design was targeted for Xilinx Virtex-4 chip, and the device utilization of the FPGA chip and the latencies of the main units are shown in Table II. The resource allocations are listed in control logic block (CLB) slices, block random access memories (RAMs), and DSP48 units. The SQRD implementation, which used two parallel MULs in the main calculation units, is able to calculate 110k QRD operations in a second. The maximum throughput of the IR-LSD algorithm implementation, where two and four parallel MULs and MACs were used in the SEE and PED unit, is 13.3Mbps. The throughput of the LLR calculation unit implementation, where two parallel MULs and full parallelism for one bit max calculation and pipelining was used, is 75.5Mbps. It should be noted that parallel units can be used in an OFDM system to obtain higher total throughput.

V. CONCLUSIONS

We designed and introduced a parallel and scalable architecture units for the IR-LSD. It was shown that the main

TABLE II
THE DEVICE UTILIZATION FOR XILINX VIRTEX-IV CHIP AND LATENCIES.

Resource	SQRD	LSD alg.	LLR calc.
Slices	2848	1595	1841
BRAMs	2	6	0
DSP48s	24	6	3
Latency	9.06 μ s	0.133 μ s $\cdot D$	0.212 μ s
Throughput	110k oper./s	13.3Mbps@19dB	75.5Mbps

operations of the algorithm can be run in parallel and are scalable to different configurations with minor changes. The designed architecture was implemented for 4×4 system with 16-QAM on a Virtex-IV FPGA chip. The results show that the LSD is feasible for practical implementation.

ACKNOWLEDGEMENTS

This work was done in MITSE project which was supported by Elektrobit, Nokia, Nokia-Siemens Networks, Texas Instruments and the Finnish Funding Agency for Technology and Innovation, Tekes. The authors thank Mentor Graphics for the possibility to evaluate Catapult C[®] Synthesis tool.

REFERENCES

- [1] B. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, Mar. 2003.
- [2] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Comput.*, vol. 44, no. 5, pp. 463–471, May 1985.
- [3] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.
- [4] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI Implementation of MIMO Detection Using the Sphere Decoding Algorithm," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 7, Jul. 2005.
- [5] M. G. Datasheet, "Catapult Synthesis," Mentor Graphics, Tech. Rep., 2008, http://www.mentor.com/products/esl/high_level_synthesis/catapult_synthesis/index.cfm.
- [6] M. Myllylä, M. Juntti, and J. Cavallaro, "The Effect of Preprocessing to the Complexity of List Sphere Detector Algorithms," in *Proc. Int. Symp. Wireless Pers. Multimedia Commun. (WPMC)*, Saarisekka, Finland, 8–11 September 2008.
- [7] D. Wübben, R. Böhnke, V. Kühn, and K. Kammeyer, "MMSE extension of V-BLAST based on sorted QR decomposition," in *Proc. IEEE Veh. Technol. Conf. (VTC)*, vol. 1, Orlando, Florida, Oct. 6–9 2003, pp. 508–512.
- [8] M. Myllylä, J. Cavallaro, and M. Juntti, "A List Sphere Detector based on Dijkstra's Algorithm for MIMO-OFDM Systems," in *Proc. IEEE Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Athens, Greece, Sep 12–19, 2007.
- [9] W. Xu, Y. Wang, Z. Zhou, and J. Wang, "A Computationally Efficient Exact ML Sphere Decoder," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, vol. 4, Nov. 29–Dec. 3 2004, pp. 2594–2598.
- [10] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Numerische Mathematik*, vol. 1, Mathematisch Centrum, Amsterdam, Netherlands, 1959, pp. 269–271.
- [11] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1009–1013, 1995.
- [12] D. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition*. Addison-Wesley, 1997.
- [13] M. Myllylä, M. Juntti, and J. Cavallaro, "Implementation Aspects of List Sphere Detector Algorithms," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Washington, D.C., USA, Nov 26–30, 2007.