# JOINT LIST POLAR DECODER WITH SUCCESSIVE CANCELLATION AND SPHERE DECODING

*Xiao Liang*[1,2,3]*, Huayi Zhou*[1,2,3]*, Zaichen Zhang*[2,3]*, Xiaohu You*[2]*, and Chuan Zhang*[1,2,3,*]

[1]Lab of Efficient Architectures for Digital-communication and Signal-processing (LEADS)
[2]National Mobile Communications Research Laboratory, Southeast University
[3]Quantum Information Center of Southeast University, Nanjing, China
Email: {xiao_liang, hyzhou, zczhang, xhyu, chzhang}@seu.edu.cn

## ABSTRACT

For polar codes, both successive cancellation list (SCL) decoding and list sphere decoding (LSD) aim to balance performance and complexity. The same list structure but different decoding schedules of SCL and LSD can lead to a combination of both schemes. In this paper, an efficient joint list decoder with SCL and LSD (JLSCD) is proposed to reduce time complexity. We apply SCL and LSD schemes simultaneously but independently, then merge them at the middle point of the decoding. Numerical results have demonstrated JLSCD scheme's advantage in complexity. FPGA implementation of JLSCD decoder is also given in this paper.

***Index Terms***— Polar codes, joint decoding, successive cancellation, sphere decoding, VLSI implementation.

## 1. INTRODUCTION

Polar codes, proposed by Arıkan [1, 2], can provably achieve the symmetric capacity of binary-input discrete memory-less channels (B-DMCs). Recently, polar codes have been selected for control channel of eMBB by 3GPP.

Though successive cancellation (SC) has been proposed for polar decoding, its performance is unsatisfactory compared with maximum likelihood (ML) decoder. For better performance, successive cancellation list (SCL) decoder, which always keeps $L_C$ best candidate paths, was proposed [3, 4]. ML decoder implemented with sphere decoding (SD) [5, 6] was studied in [7–9]. However, the computational complexity of SD decoders is still high. List SD (LSD) proposed in [10] applies breadth-first-search (BFS) to keep $L_D$ candidate paths with the minimum distances. It has a lower complexity than SD decoder but with worse performance.

Since both SCL and LSD decoder try to balance performance and complexity with BFS scheme over list structure, it is expected to combine them for better balance. Since their decoding orders are inverse, we apply them simultaneously but merge them at the middle point of decoding. Because the SCL and LSD schemes can decode independently, the time complexity will reduce.

The remainder of the paper is organized as follows. Section 2 reviews the preliminaries. Section 3 presents the proposed the decoding scheme. Section 4 gives the performance and complexity analysis of different decoding schemes. FPGA implementation is shown in Section 5. Finally, Section 6 concludes the entire paper.

## 2. PRELIMINARIES

### 2.1. Polar Codes

Denote a polar code by $(N, K, \mathcal{A})$ [1], where $N$, $K$ and $\mathcal{A}$ are the block length of a polar code, the information length in one block and the set of information bits, respectively. Let $\mathcal{A}^C$ denote the complement of $\mathcal{A}$ on $\{1, 2, ..., N\}$. The input alphabet, output alphabet, and transition probabilities of B-DMCs can be defined as $\mathcal{X}$, $\mathcal{Y}$, and $W(y|x)$, respectively, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. The information vector, encoded vector, and received vector are denoted by $u_1^N = (u_1, u_2, ..., u_N)$, $x_1^N = (x_1, x_2, ..., x_N)$, and $y_1^N = (y_1, y_2, ..., y_N)$, respectively. The encoded vector can be generated as $x_1^N = u_1^N \mathbf{G_N} = \mathbf{F}^{\otimes n} u_1^N$, where $n = \log_2 N$ and $\mathbf{F} = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$.

### 2.2. SC and SCL Polar Decoders

SC decoding calculates $u_1^N$ successively. If $u_i$ is a frozen bit, then $\hat{u}_i = 0$. Otherwise, the SC polar decoder usually computes the log-likelihood ratio (LLR) of each bit channel:

$$L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) = \log \frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1} \mid u_i = 0)}{W_n^{(i)}(y_1^N, \hat{u}_1^{i-1} \mid u_i = 1)}. \quad (1)$$

and decides as $\hat{u}_i = \begin{cases} 0, & \text{if } L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) \geq 0; \\ 1, & \text{otherwise.} \end{cases}$

The SC decoding algorithm updated the LLRs by applying the two equations (Type A and Type B) listed in (2) recursively, where $\max^*$ denotes the Jacobi logarithm:
$\max^*(x_1, x_2) \triangleq \ln(e^{x_1} + e^{x_2})$.

Instead of keeping one path with the most likely bit estimation in each step in SC decoding, the SCL decoding expands and selects $L_C$ bast survival paths on the full binary-tree. With list size $L_C$, SCL decoding improves the performance than SC decoding [3] and [11].

### 2.3. SD and List SD Polar Decoding

Let $\mathcal{U}$ denote the set of all information sequences $u_1^N$. Let $g_{j,i}$ denote the $(i, j)$-th entry of $\mathbf{G_N}$. The aim of SD decoding is to solve the following minimization problem $\hat{u}_1^N = \arg\min_{u_1^N \in \mathcal{U}} \left\| \hat{y}_1^N - u_1^N \mathbf{G_N} \right\|^2$ [9]. Let $D(y_1^N)$ denote the Euclidean distance between $y_1^N$ and the codeword. SD algorithm enumerates all points $u_1^N \in \mathcal{U}$ that satisfy

$$
\begin{aligned}
Type\ A: L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}|u_{2i-1}) = max^*(L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}|u_{2i-1}) + L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}|0), \\
L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}|\bar{u}_{2i-1}) + L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}|1)), \\
Type\ B: L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}|u_{2i}) = L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}|u_{2i-1} \oplus u_{2i}) + L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}|u_{2i}).
\end{aligned} \tag{2}
$$

the constraint

$$
\begin{aligned}
&D(y_1^N) = \left\| y_1^N - u_1^N \mathbf{G_N} \right\|^2 \\
&= \sum_{i=1}^{N} (y_i - \overset{N}{\underset{j=i}{\oplus}} (g_{j,i} u_j))^2 \leq {r_0}^2
\end{aligned} \tag{3}
$$

where $r_0$ is the initial radius for the search and $\overset{b}{\underset{j=a}{\oplus}}(\cdot)$ is the summation over GF(2). Since $\mathbf{G_N}$ is the a lower triangular matrix, SD algorithm processes in a back manner from $i = N$ to $i = 1$ to find the optimal $u_1^N$ which has the minimum Euclidean distance to the received codeword. The average complexity of SD algorithms is $\mathcal{O}(N^3)$ in many scenarios [12].

Since the complexity of SD is high, LSD algorithm is proposed [10]. LSD algorithm abandons the radius and applies BFS to keep a list of $L_D$ candidate paths with the minimum distances. Compared with SD algorithm, LSD has a performance loss but successfully reduces the complexity.

## 3. JOINT LIST DECODING WITH SC AND SD

SCL decoding and LSD algorithms have their own tradeoff between performance and complexity. The similarity between them is that they are both BFS process. The differences between them are metric measurement and decoding order. SCL decoding calculates LLR as the standard metric while LSD algorithm compares the Euclidean distance among the candidate paths. Coincidentally, the decoding order of SCL decoding is from $u_1$ to $u_N$ while LSD method decodes from $u_N$ to $u_1$. This fact leads to an efficient decoding method: we apply SCL decoding and LSD algorithm simultaneously and merge them together at the middle point. We name this decoding algorithm as joint list decoding with SC and SD (JLSCD). Since the SCL and LSD can decode independently at the same time, the time complexity reduced by half. In this section, we first propose JLSCD algorithm and then carry its performance and complexity analysis.

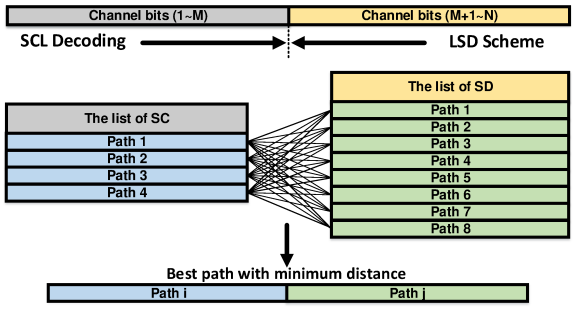


**Fig. 1**. JLSCD scheme ($L_C = 4$, $L_D = 8$).

### 3.1. An Efficient Joint Decoding Scheme

Assume the middle point of the channel is $M$, which indicates that SCL decoding is applied for the first bit to the $M$-th bit and LSD algorithm is applied for the last bit to the $(M+1)$-th bit. Define SCL($y_1^N$, $M$, $L_C$) as the SCL decoding procedure with received vector $y_1^N$, middle point $M$ and list size $L_C$. Define $\mathcal{U}_C$ as the the output paths set of SCL($\cdot$). Define LSD($y_1^N$, $M$, $L_D$) as the LSD algorithm procedure. Define $\mathcal{U}_D$ as the the output paths set of LSD($\cdot$). Define distance($p_i, p_j$) as the Euclidean distance calculation procedure between the joint path $p_i, p_j$ and the codeword. Define $d$ as the current minimum distance parameter. The JLSCD algorithm can be listed as follows:

**Algorithm 1** JLSCD polar decoding

**Require:** $y_1^N$, $M$, $L_C$, $L_D$
1: $d = +\infty$;
2: $\mathcal{U}_C$ = SCL($y_1^N$, $M$, $L_C$);
3: $\mathcal{U}_D$ = LSD($y_1^N$, $M$, $L_D$);
4: **forall** $p_i \in \mathcal{U}_C$
5: **forall** $p_j \in \mathcal{U}_D$
6: $d$ = min($d$, distance($p_i, p_j$));
7: refresh $\hat{u}_1^n$;
8: **end for**
9: **end for**
**Ensure:** $\hat{u}_1^n$

The input parameters are $y_1^N$, $M$, $L_C$ and $L_D$. The algorithm sets $d = +\infty$ as the initialization, then perform SCL decoding and LSD algorithm independently and simultaneously. At the middle point, dual loops are applied to find the best joint path with the minimum distance through matching. Fig. 1 illustrates the JLSCD scheme with $L_C = 4$, $L_D = 8$.

## 4. PERFORMANCE AND COMPLEXITY ANALYSIS

### 4.1. Performance Analysis

As SD algorithm enumerates all possible points, it has the best performance. SC decoding performs worst for it is a greedy algorithm. Since SCL decoding and LSD algorithm are dynamic schemes with the list size, their performances are between SD and SC decoding. Proposed JLSCD decoding consists of SCL decoding and LSD algorithm, and the matching procedure picks up the best joint path. Therefore, its performance is between SCL decoding and LSD algorithm.

Fig. 2 indicates FER performance comparison different schemes via simulation over binary-input additive white Gaussian noise channel. For all the five different schemes, simulations are carried out for the $(N = 32, K = 26)$ polar code in [2]. The information set $A$ is selected using the method in [13] for polar codes. The proposed JLSCD scheme (blue star line) performs between SCL decoding (black left triangle line) and LSD algorithm (green square line).

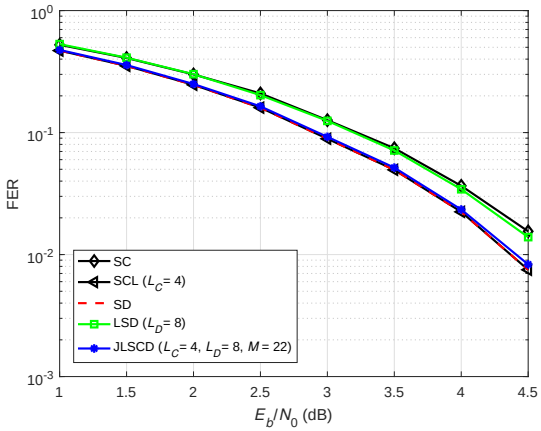


**Fig. 2**. FER performance comparison.$(N = 32, K = 26)$.

Compared with the SD scheme (red dash line), the JLSCD scheme has a less than $0.1$ dB performance loss at the FER of $0.01$.

### 4.2. Complexity Analysis

We calculate addition times, multiplication times and comparison times as the complexity analysis of SC, SCL, SD, LSD and JLSCD schemes.

#### *4.2.1. Complexity of SC Decoding*

In Eq. (2), there are $i$ additions and 1 multiplication in Type A. In Type B, there are $(i-1)$ additions, 2 multiplications and 1 comparison. The information set determines the number of visited Type A and Type B nodes. Here 'visited node' means the node we need to calculate its LLR, and 'non-visited node' means the node we don't need to calculate its LLR because none of the information nodes needs its value.

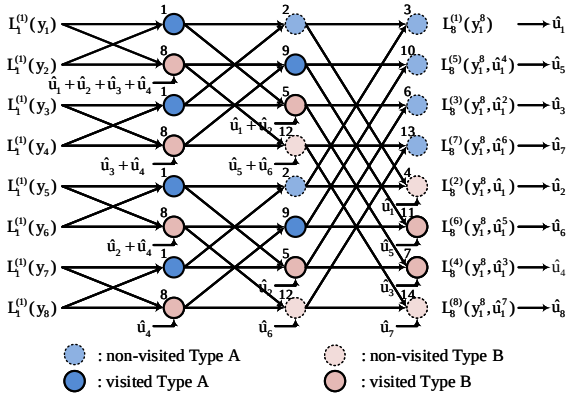


**Fig. 3**. SC decoding process $(N = 8, \mathcal{A} = \{4, 6\})$.

Fig. 3 indicates SC decoding process of given information set $\mathcal{A} = \{4, 6\}$. To determine $\hat{u}_4, \hat{u}_6$, all the visited Type A nodes (solid blue circle) and the visited Type B nodes (solid red circle) are required to calculate the LLRs. Let $Sa_c, Sm_c, Sc_c$ denote the addition times, multiplication times and comparison times in SC decoding. In general, assume there are $T_1$ Type A nodes and $T_2$ Type B nodes in SC decoding process. We have

$$\begin{cases} Sa_c = \sum_{j=1}^{T_1} ia_j + \sum_{j=1}^{T_2} (ib_j - 1), \\ Sm_c = T_1 + 2T_2, \\ Sc_c = T_2, \end{cases} \tag{4}$$

where $ia_j, ib_j$ denotes the corresponding '$i$' for $j$-th Type A and Type B node.

#### *4.2.2. Complexity of SCL Decoding*

Let $Sa_{cl}, Sm_{cl}, Sc_{cl}$ denote the addition times, multiplication times and comparison times in SCL decoding. For SCL decoding keeps $L_C$ candidate paths, the operation numbers should times $L_C$. Assume we apply bubble sorting for the list, which adds $K \times L_C(L_C - 1)/2$ comparisons. We have

$$\begin{cases} Sa_{cl} = L_C \times (\sum_{j=1}^{T_1} ia_j + \sum_{j=1}^{T_2} (ib_j - 1)), \\ Sm_{cl} = L_C \times (T_1 + 2T_2), \\ Sc_{cl} = L_C \times T_2 + K \times L_C(L_C - 1)/2. \end{cases} \tag{5}$$

#### *4.2.3. Complexity of SD Scheme*

Let $Sa_d, Sm_d$ denote the addition times and multiplication times in SD algorithm. As shown in Eq. (3), each $i$ produces $(N-i)$ addition times and $(N - i + 1)$ multiplication times. Assume SD algorithm visits totally $D$ times on channel bits, and the bit-position of $i$-th visit is $k_i$. We have

$$\begin{cases} Sa_d = \sum_{i=1}^{D} (N - k_i), \\ Sm_d = \sum_{i=1}^{D} (N - k_i + 1). \end{cases} \tag{6}$$

#### *4.2.4. Complexity of LSD Scheme*

Let $Sa_{dl}, Sm_{dl}, Sc_{dl}$ denote the addition times, multiplication times and comparison times in LSD algorithm. Since LSD algorithm is a BFS process, it keeps a list and decodes every information bit on the choices '0' or '1', and set '0's for frozen bits. We have

$$\begin{cases} Sa_{dl} = 2 \times \sum_{i \in \mathcal{A}} (N - k_i) + \sum_{i \in \mathcal{A}^C} (N - k_i), \\ Sc_{dl} = K \times L_D(L_D - 1)/2, \\ Sm_{dl} = 2 \times \sum_{i \in \mathcal{A}} (N - k_i + 1) + \sum_{i \in \mathcal{A}^C} (N - k_i + 1). \end{cases} \tag{7}$$

#### *4.2.5. Complexity of JLSCD Scheme*

Let $Sa_{Jc}, Sm_{Jc}, Sc_{Jc}$ denote the addition times, multiplication times and comparison times in partial SCL part of JLSCD decoding. Let $Sa_{Jd}, Sm_{Jd}, Sc_{Jd}$ denote the addition times, multiplication times and comparison times in partial LSD part of JLSCD decoding.

We need to obtain the partial information bits in partial SCL decoding, then determine the number of visited Type A and Type B nodes. Assume there are $T_1'$ Type A nodes and $T_2'$ Type B nodes in JLSCD decoding. Let $ia_j', ib_j'$ denotes the corresponding '$i$' for j-th Type A and Type B node. Assume $K'$ is the number of information bits from the first bit to the $M$-th bit. Let $\mathcal{B}$ denote the partial information set from $M+1$-th bit to $N$-th bit. Let $\mathcal{B}^C$ denote the complement of $\mathcal{B}$ on $\{M+1, M+2, ..., N\}$. Let $k_i'$ denote the bit-position for $i \in \mathcal{B}$. We have

$$\begin{cases} Sa_{Jc} = L_C \times (\sum_{j=1}^{T_1'} ia_j' + \sum_{j=1}^{T_2'} (ib_j' - 1)), \\ Sa_{Jd} = 2 \times \sum_{i \in \mathcal{B}} (N - k_i') + \sum_{i \in \mathcal{B}^C} (N - k_i'), \\ Sm_{Jc} = L_C \times (T_1' + 2T_2'), \\ Sm_{Jd} = 2 \times \sum_{i \in \mathcal{B}} (N - k_i' + 1) + \sum_{i \in \mathcal{B}^C} (N - k_i' + 1), \\ Sc_{Jc} = L_C \times T_2' + \frac{1}{2}(K' \times L_C(L_C - 1)), \\ Sc_{Jd} = \frac{1}{2}(K - K') \times L_D(L_D - 1)). \end{cases} \tag{8}$$

**Table 1**. Operation number of different schemes ($N = 32$, $K = 26$, $L_C = 4$, $L_D = 8$, $M = 22$).

| Schemes | Additions | Multiplications | Comparisons |
|---|---|---|---|
| SC | 966 | 450 | 146 |
| SD | 10,853 | 11,532 | 0 |
| SCL | 3,864 | 1,800 | 740 |
| LSD | 6,312 | 6,728 | 728 |
| JLSCD SCL part | 1,848 | 1,408 | 576 |
| JLSCD LSD part | 1,728 | 1,840 | 280 |

Table. 1 shows the number of three operations in different schemes. Both SCL part and LSD part of JLSCD scheme have complexity reduction compared with other schemes. We can choose a proper $M$ to merge them together at the same running time in real applications.

## 5. VLSI IMPLEMENTATION RESULTS

SC as well as SCL hardware designs in [14–17] have $\log_2 N$ stages. Since LSD is an encoding process, it also has $\log_2 N$ stages. JLSCD scheme includes both SCL part and LSD part, the corresponding architecture needs both SCL and LSD hardware modules. Because there are usually more information bits in $\{N/2+1..N\}$ than that in $\{1..N/2\}$, the value of $M$ is usually larger than $N/2$ to merge them together at the same time. Therefore, the SCL part needs complete $(\log_2 N)$-stage SCL hardware design [14], and LSD part needs only $(\log_2 N - 1)$-stage LSD design. Our proposed JLSCD architecture applies distributed sorting algorithm [18, 19] for list sorting both for SCL and LSD. The decoding latency is based on the larger latency between SCL and LSD. The merging process would not influence the final latency, since the merging process of current block and decoding process of next block implement simultaneously.

The hardware platform is set up on Altera Stratix V FPGA. Set the quantization length of LLR as 1 sign bit, 6 integer bits and 1 decimal bit for SCL part. Tab. 2 shows SCL decoder ($L_C = 4$), LSD decoder ($L_D = 8$) and JLSCD decoder ($L_C = 4$, $L_D = 8$, $M = 22$) are all based on polar codes with ($N = 32, K = 26$). The latency of our proposed JLSCD decoder costs only $54.6\%$ compared with SCL decoder with a $10.8\%$ adaptive logical modules (ALMs) increase.

**Table 2**. Implementation results comparison with ($N = 32, K = 26, M = 22, L_C = 4, L_D = 8$).

| Decoding Module | ALMs | Register | Latency |
|---|---|---|---|
| SCL decoder | 8,784 | 2,744 | 174 |
| LSD decoder | 1,228 | 640 | 268 |
| JLSCD decoder | 9,729 | 3,176 | 95 |

## 6. CONCLUSIONS

JLSCD scheme is proposed to reduce the time complexity. Based on the similarities and differences of SCL and LSD schemes, the JLSCD scheme applies these two schemes independently and simultaneously, then merges them together. Numerical results have shown the complexity reduction of proposed JLSCD scheme. The VLSI implementation results show the significant latency reduction of proposed JLSCD decoder compared with SCL decoder with an acceptable ALMs increase.

# Acknowledgement

This work is supported in part by NSFC under grant 61501116, Jiangsu Provincial NSF under grant BK20140636, Huawei HIRP Flagship under grant YB201504, the Fundamental Research Funds for the Central Universities, the SRTP of Southeast University, State Key Laboratory of ASIC & System under grant 2016KF007, ICRI for MNC, and the Project Sponsored by the SRF for the Returned Overseas Chinese Scholars of MoE.

# References

[1] Erdal Arıkan and Emre Telatar, "On the rate of channel polarization," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2009, pp. 1493–1495.

[2] Erdal Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, July. 2009.

[3] I. Tal and A. Vardy, "List decoding of polar codes," in *Proc. IEEE International Symposium on Information Theory Proceedings (ISIT)*, 2011, pp. 1–5.

[4] K. Chen, K. Niu, and J.R. Lin, "List successive cancellation decoding of polar codes," *Electronics Letters*, vol. 48, no. 9, pp. 500–501, April 2012.

[5] Michael Pohst, *On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications*, ACM, 1981.

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

[7] Sinan Kahraman and M. Erturul Çelebi, "Code based efficient maximum-likelihood decoding of short polar codes," in *Proc. IEEE International Symposium on Information Theory Proceedings (ISIT)*, 2012, pp. 1967–1971.

[8] Kai Niu, Kai Chen, and Jiaru Lin, "Low-complexity sphere decoding of polar codes based on optimum path metric," *IEEE Commun. Lett.*, vol. 18, no. 2, pp. 332–335, 2014.

[9] Jing Guo and Albert Guillén I Fàbregas, "Efficient sphere decoding of polar codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2015, pp. 236–240.

[10] S. A. Hashemi, C. Condo, and W. J. Gross, "List sphere decoding of polar codes," in *Asilomar Conference on Signals, Systems and Computers*, 2015, pp. 1346–1350.

[11] Kai Niu and Kai Chen, "CRC-aided decoding of polar codes," *IEEE Commun. Lett.*, vol. 16, no. 10, pp. 1668–1671, 2012.

[12] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm i. expected complexity," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2806–2818, 2005.

[13] Irina Tal and Alexander Vardy, "How to construct polar codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, 2013.

[14] Chuan Zhang, Bo Yuan, and Keshab K. Parhi, "Reduced-Latency SC polar decoder architectures," in *Proc. IEEE International Conference on Communications (ICC)*, 2011, pp. 3471–3475.

[15] Chuan Zhang and Keshab Parhi, "Low-latency sequential and overlapped architectures for successive cancellation polar decoder," *IEEE Trans. Signal Process.*, vol. 61, no. 10, pp. 2429–2441, 2013.

[16] Chuan Zhang, Xiaohu You, and Jin Sha, "Hardware architecture for list successive cancellation polar decoder," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 209–212.

[17] Menghui Xu, Xiao Liang, Chuan Zhang, Zhizhen Wu, and Xiaohu You, "Stochastic BP polar decoding and architecture with efficient re-randomization and directive register," in *Proc. IEEE International Workshop on Signal Processing Systems (SiPS)*, 2016, pp. 315–320.

[18] X. Liang, J. Yang, C. Zhang, W. Song, and X. You, "Hardware efficient and low-latency CA-SCL decoder based on distributed sorting," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[19] Yifei Shen, Chuan Zhang, Junmei Yang, Shunqing Zhang, and Xiaohu You, "Low-latency software successive cancellation list polar decoder using stage-located copy," in *Proc. IEEE International Conference on Digital Signal Processing (DSP)*, 2016, pp. 84–88.