View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by E-space: Manchester Metropolitan University's Research Repository

# A Method to Enhance the Performance of Successive Cancellation Decoding in Polar Codes

Ammar Hadi*, Emad Alsusa* and Khaled M. Rabie‡
*School of Electrical & Electronic Engineering, The University of Manchester
‡School of Engineering, Manchester Metropolitan University
Email: {ammar.hadi, e.alsusa}@manchester.ac.uk; k.rabie@mmu.ac.uk

***Abstract*—Polar codes are regarded as a major breakthrough in modern channel coding since they are capacity-achieving using simple successive cancellation (SC) decoding. However, this is only possible with significantly large code lengths which may not be applicable for many systems. In this paper, we focus on short length polar codes and present a method which can enhance the performance of the successive cancellation decoder. For the purpose of analysis, we discuss the SC code tree and show how the proposed method can improve the performance by increasing the computational nodes in the code tree. The results quantify the achieved performance improvement over the conventional SC decoder.**

***Index Terms*—Polar code, channel coding, code tree, successive cancellation, channel polarization.**

## I. Introduction

Polar codes were invented by Arikan [1] who proved in his seminal work that this family of codes can achieve the symmetric capacities of the binary discrete memory-less channels (B-DMCs). Polar codes utilize the channel polarization phenomena for the construction. In channel polarization, identical independent copies of the B-DMC bit channels are combined and split into two groups in which the first has pure noisy bits and the second contains the noiseless bits. The heuristic method of polar codes construction was applied on binary erasure channels (BEC). Further studies have develop other construction methods such as [2] where the authors tried to manipulate the construction by giving two approximations of the quantization, and [3] where the authors presented a new method for the construction that uses convolution and has linear complexity. Recently, a practical construction method was presented for polar codes under additive white Gaussian noise (AWGN) channels [4].

Polar codes were initially introduced with the successive cancellation (SC) decoder which has relatively low complexity. Further enhancements to the SC performance have been carried out by the successive cancellation list and successive cancellation stack [5], [6]. Moreover, the belief propagation (BP) decoder was applied for decoding polar codes; however, this decoder entails more complexity and memory due to its recursive nature [7]. Analysis of polar codes in wireless communication systems was reported in [8] where the authors suggested that polar codes might be a good candidate for future wireless communication technologies due to its simple implementation in image and speech transmission. Further studies considered polar codes in MIMO techniques and wireless fading [9], [10]. The code tree of polar decoding was discussed in [11] where the authors provided the number of nodes and edges required in the decoding procedures.

The aim of this paper is to propose a method to enhance the performance of the SC decoder. The proposed method which is referred to in this paper as one-step decision delay (ODSS), is based on adding extra computations nodes to the SC code tree and apply a decision in the code tree. We present a comparison between the conventional SC decoder and the proposed method in terms of complexity and latency. The simulation results for OSDD method reveal a reasonable enhancement to the decoder performance.

The remainder of this paper is organized as follows. In Section II, the preliminaries of the system model are described. In Section III, the decoding code tree is demonstrated and the proposed OSDD decoder is presented in Section IV. Simulation results are presented and discussed in Section V. Finally, conclusions are drawn in Section VI.

## II. Preliminaries

### A. Polar Code Construction

The construction of polar codes exploits the channel polarization in which a number of identical copies of a channel $W$ are combined to one channel $W_N : X_N \rightarrow Y_N$ where $X_N = (x_0, x_1, ..., x_{N-1})$ and $Y_N = (y_0, y_1, ..., y_{N-1})$ are the corresponding inputs and outputs vectors, respectively. The transition probability for the yielded channel can be obtained by $W_N(Y_N|X_N) = \prod_{i=1}^{N} W(y_i|x_i)$. Then, the polarization phenomena suggests that this comprehensive channel can be split back into a set of $N$ channels. For the sake of polarization, the algorithm selects the noiseless channels group for sending the information set $A$ while the noisy channels are fixed to be a frozen set $A^c$ whose values are known to both the sender and the receiver. Hence, the decoding algorithm of polar codes can make a swift decision when $\hat{u} \in A^c$. On the other hand, the decoder makes a decision based on the results of its computation functions when $\hat{u} \in A^c$. Thus, the frozen set does not carry any information but at the same time they are useful for the decoding algorithm. Let polar code $(N, K)$, the codeword vector $X$ can be generated by the polar encoder as

$$X = UG_N \tag{1}$$

where $G_N$ denotes the generator matrix and the $U$ vector includes the entirely information and frozen bits.

### B. The SC Decoding

For a given number of bits $N$, the SC algorithm consists of a number of recursive functions to compute the likelihood ratios. Thus, the algorithm is divided into a number of stages denoted by $s \in (0, 1, .., n)$ in which each stage has a number of groups indexed by $g \in (0, 1, .., 2^s - 1)$ and for each group, there is a number of functions $f \in (0, 1, .., 2^{n-s} - 1)$. Therefore, a single likelihood ratio ($LR$) in the SC algorithm can be denoted by $LR(s, g, f)$. The SC algorithm starts by finding the $LR$s after the channel by

$$LR(0, 0, i) = \frac{w(y_i|x_i = 0)}{w(y_i|x_i = 1)} \tag{2}$$

After the first stage calculations, the decoder calculates the functions inside each single set by (3) and (4). The previous decision $P_d$ for the last stage can be given by

$$P_d(n, g, 0) = \hat{u}_{g-1} \tag{5}$$

On the other hand, the algorithm updates $P_d$ in the stages as

$$P_d(s - 1, g/2, 2f) = P_d(s, g - 1, f) \oplus P_d(s, g, f) \tag{6}$$

$$P_d(s - 1, g/2, 2f + 1) = P_d(s, g, f) \tag{7}$$

The final decision is taken by the algorithm in the last stage where each set has a single computational function which includes $LR(n, i, 0)$; therefore, the general formula for the SC decision can be given as

$$\hat{u}_i = \begin{cases} 0 & i \in U_F \\ 0 & i \in U_I, LR(n, i, 0) \geq 1 \\ 1 & i \in U_I, LR(n, i, 0) < 1 \end{cases} \tag{8}$$

where $0 \leq i \leq N - 1$. For better clarity, we present in Fig. 1 the SC algorithm for eight bits. It is worth mentioning that the complexity of the SC decoder is given by $O(N \log_2 N)$, which is equivalent to the required processing elements to accomplish the SC decoding to one codeword whereas the number of the time elements to the SC decoder can be given as $2N - 1$ [12].

## III. Decoding Code Tree

### A. The Polar Code Tree

One of the most complex problems in any channel code is the design of the decoder. In general, The decoding of polar codes could be defined as the algorithm of a valid path finder inside the code tree which consists of nodes set and edges set. In the trivial case where there are no frozen bits, the number of nodes is given by [11]

$$| V | = 2^{N+1} - 1 \tag{9}$$

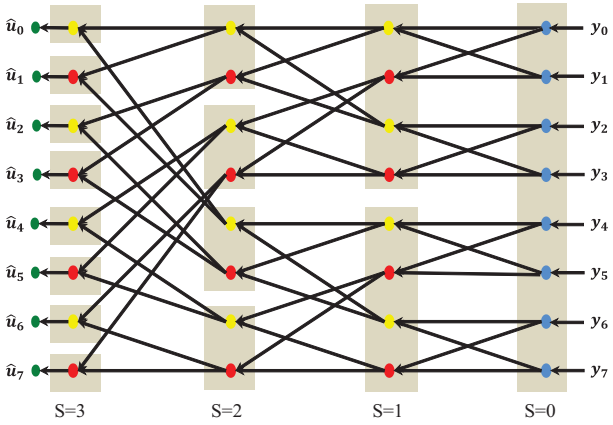



Figure 1. The SC algorithm when $N = 8$.

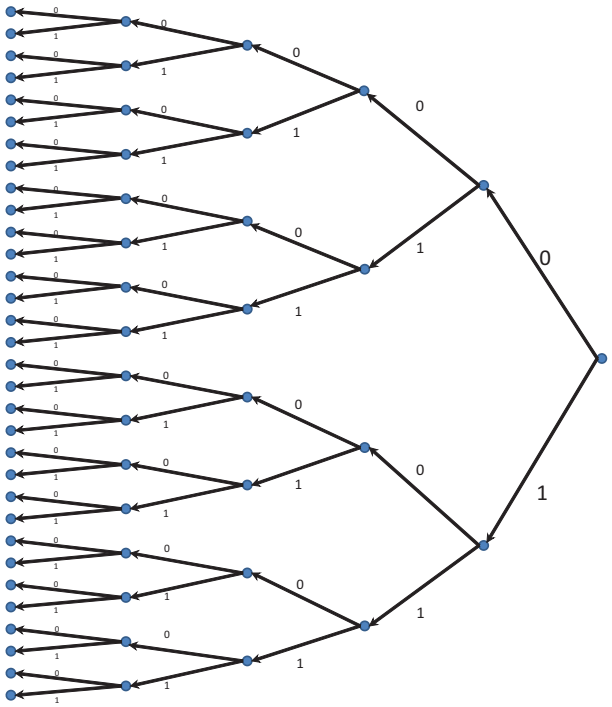


Figure 2. The polar code tree when $N = 5$ and $K = 5$.

Fig. 2 displays the decoder where $N = K = 5$. It can be noticed that each node is divided into two binary options. In total, there are 63 nodes in the code tree in this particular example.

The depth of any specific node ($d$ ) in the code tree could be defined as the length of the path from this node to the root node, i.e. the first node in the tree. It is known in a normal polar code that there is a specific number of nodes behaves as frozen bits, hence, each frozen bit is expressed by only one node in the code tree. As mentioned above, the total number of nodes without any frozen bits could be obtained from (9). Now, suppose a single information bit with depth $d_1$ is fixed as a frozen bit in the the code tree, this bit would decrease the number of nodes after its position in the code tree by half.

$$LR(s,g,f) = \frac{L(s-1,g/2,2f)L(s-1,g/2,2f+1)+1}{L(s-1,g/2,2f)+L(s-1,g/2,2f+1)} \tag{3}$$

$g$ is even

$$LR(s,g,f) = [L(s-1,g/2,2f+1)]^{1-2P_d(s,g-1,r)}\, L(s-1,g/2,2f) \tag{4}$$

$g$ is odd

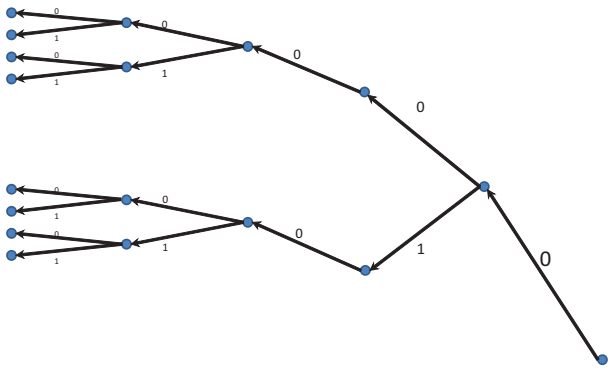


Figure 3. The polar code tree when $N = 5, K = 4, d_1 = 1$, and $d_2 = 3$.

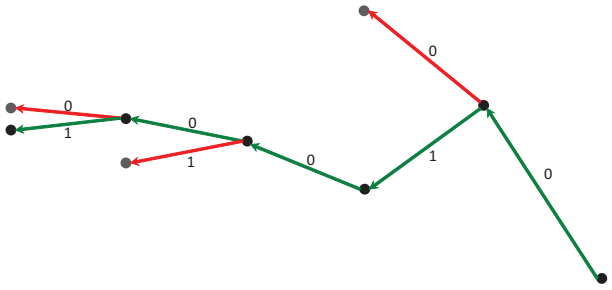


Figure 4. The SC code tree when $N = 5$ and $K = 3$.

In contrast, the number of nodes before this bit will not be affected by this change. It should be pointed out that the nodes before the frozen bit indicates to all nodes between the frozen bit and the root of the code tree. Hence, the single bit reduces the number of nodes by $(-2^N + 2^{d_1-1})$. In case that a second bit changes its status to frozen, the number of the remaining nodes decreases by half, consequently the second frozen bit causes a reduction by $(-2^{N-1} + 2^{d_2-2})$. Now, the general formula for computing the total number of nodes in the tree of polar codes can be given by

$$| V | = 2^{N+1} - 1 + \sum_{i \in A^C} 2^{d_i - i} - 2^{N-i+1} \tag{10}$$

where $d_i < d_{i+1}$.

For example, Fig. 3 illustrates the code tree for $N = 5$, $K = 3$, $d_1 = 1$ , and $d_2 = 3$. It can be noticed that the number of nodes in the polar code tree depends on the number of frozen bits and their depths which are selected according to the polarization phenomena. The full maximum likelihood (ML) decoder is supposed to compute all the nodes in the code tree to make its decisions and choose the most proper path. In general, the ML decoder can be applied only for short codes; otherwise, this decoder is impractical since it needs a huge number of computation nodes as we will show later.

### *B. The SC Tree*

Since the decoder calculates only the $LR$s of the information bits, it needs two nodes for each information bit and one node for a frozen bit. In contrast to the ML decoder, the SC algorithm finds only one valid path inside the code tree; therefore, the number of the nodes visited by the SC algorithm could be defined as

$$| V_{SC} | = 2K + | A^c | + 1 \tag{11}$$

where $V_{SC}$ is the set of nodes in the SC code tree.

Fig. 4 depicts the SC algorithm inside the code tree for a polar code $(5, 3)$ and the example presents the decoding of the codeword $(01001)$. The green color is used for the most successful path while the red one is used for the non-valid paths according to the decision function (8).

## IV. One-Step Decision Delay Method

It has been observed from the previous section that the potential problem of the SC decoder is that if any bit is decoded not correctly, there is no chance for the decoder to correct this bit again. In addition, the wrong bit may affect the following bits estimation and yields more errors. Therefore, the OSDD method tries to improve the performance of the SC by applying one additional calculation to each bit. This could be done by making a delay on the decision and instead of the decision, the OSDD algorithm assumes two options of the binary bit 0 and 1 as follows

$$LR_0(n,i,0) = \frac{w(y_i|u_i=0)}{w(y_i|u_i=1)} | \hat{u}_{i-1} = 0 \tag{12}$$

$$LR_1(n,i,0) = \frac{w(y_i|u_i=0)}{w(y_i|u_i=1)} | \hat{u}_{i-1} = 1 \tag{13}$$

Then, the decoder compares these two $LR$s in the following level and makes the final decision as

$$\hat{u}_{i-1} = \begin{cases} 0 & i-1 \in U_F \\ D_{i-1} & i-1 \in U_I \end{cases} \tag{14}$$

where the final decision $D_{i-1}$ is given by

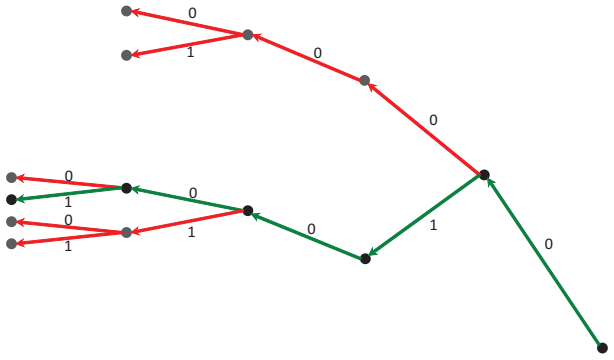


Figure 5. The OSDD code tree $N = 5$ and $K = 3$.

$$D_{i-1} = \begin{cases} 0 & L_o(n,i,0) \geq L_1(n,i,0) \\ 1 & L_o(n,i,0) < L_1(n,i,0) \end{cases} \quad (15)$$

It is clearly seen that the new decision is based on two $LR$s instead of one as in the conventional SC decoder. It is worthwhile mentioning that the OSDD algorithm applies on all bits except the last bit $\hat{u}_{N-1}$ because there is no following bit; therefore, the last bit can be obtained by applying the normal SC decision given in (8).

### A. The OSDD Code Tree

In the code tree of OSDD, the frozen bits are divided into two sets. The first set $A_B^c$ includes all the frozen bits whose positions are before the first information bit while the frozen bits in set $A_A^c$ occur after the first information bit. Hence, the number of nodes inside the OSDD code tree can be calculated by

$$\mid V_{OSDD} \mid = 4K + \mid A_B^c \mid + 2 \mid A_A^c \mid - 1 \quad (16)$$

where $V_{OSDD}$ is the set of nodes in the OSDD code tree.

Fig. 5 shows the code tree for the OSDD algorithm when the codeword $(01001)$ is decoded by OSDD method. It can be noticed that the first frozen bit $\hat{u}_0$ happens before the first information bit, thus it needs only one node while the second frozen bit $\hat{u}_2$ needs two nodes since it occurs after the first information bit.

For the purpose of comparison, Table I lists the number of nodes in the code tree for different codewords of polar codes. It can be seen that there is a significant difference between the ML decoder and the two others as the number of nodes to ML grows rapidly with increasing of the codewords which makes it unpractical. Although, the trees of the OSDD method has more nodes compared to the SC decoder, the increasing number of nodes is not significant.

### B. Complexity and Latency

In this section, we investigate the latency and complexity of the OSDD algorithm. In addition, we give a comparison to the conventional SC decoder. We have found that the complexity for the OSDD algorithm can be given by $2N \log_2 N - (N-1)$.

Table I
THE NUMBER OF NODES IN THE CODE TREES OF ML, SC, AND OSDD DECODERS.

| Codeword $N$ | Code rate | Number of nodes $\mid V \mid$ | | |
|---|---|---|---|---|
| | | ML | SC | OSDD |
| 8 | 0.25 | 16 | 11 | 18 |
| | 0.5 | 36 | 13 | 22 |
| | 0.75 | 136 | 15 | 28 |
| 16 | 0.25 | 56 | 21 | 34 |
| | 0.5 | 576 | 25 | 46 |
| | 0.75 | 8230 | 29 | 56 |
| 32 | 0.25 | 672 | 41 | 74 |
| | 0.5 | 140636 | 49 | 94 |
| | 0.75 | $33.7 \times 10^6$ | 57 | 110 |
| 64 | 0.25 | 151552 | 81 | 146 |
| | 0.5 | $8.63 \times 10^9$ | 97 | 186 |
| | 0.75 | $5.62 \times 10^{14}$ | 113 | 222 |
| 128 | 0.25 | $9.69 \times 10^9$ | 161 | 306 |
| | 0.5 | $3.7 \times 10^{19}$ | 193 | 378 |
| | 0.75 | $1.58 \times 10^{29}$ | 225 | 446 |
| 256 | 0.25 | $4.11 \times 10^{19}$ | 321 | 626 |
| | 0.5 | $6.83 \times 10^{38}$ | 385 | 762 |
| | 0.75 | $1.25 \times 10^{58}$ | 449 | 894 |
| 512 | 0.25 | $9.69 \times 10^{38}$ | 641 | 1250 |
| | 0.5 | $2.32 \times 10^{77}$ | 769 | 1530 |
| | 0.75 | $7.88 \times 10^{115}$ | 897 | 1790 |
| 1024 | 0.25 | $2.32 \times 10^{77}$ | 1281 | 2530 |
| | 0.5 | $2.69 \times 10^{154}$ | 1537 | 3066 |
| | 0.75 | $3.1 \times 10^{231}$ | 1793 | 3578 |

Table II
COMPLEXITY COMPARISON BETWEEN SC AND OSDD METHODS WHEN $N = 8$.

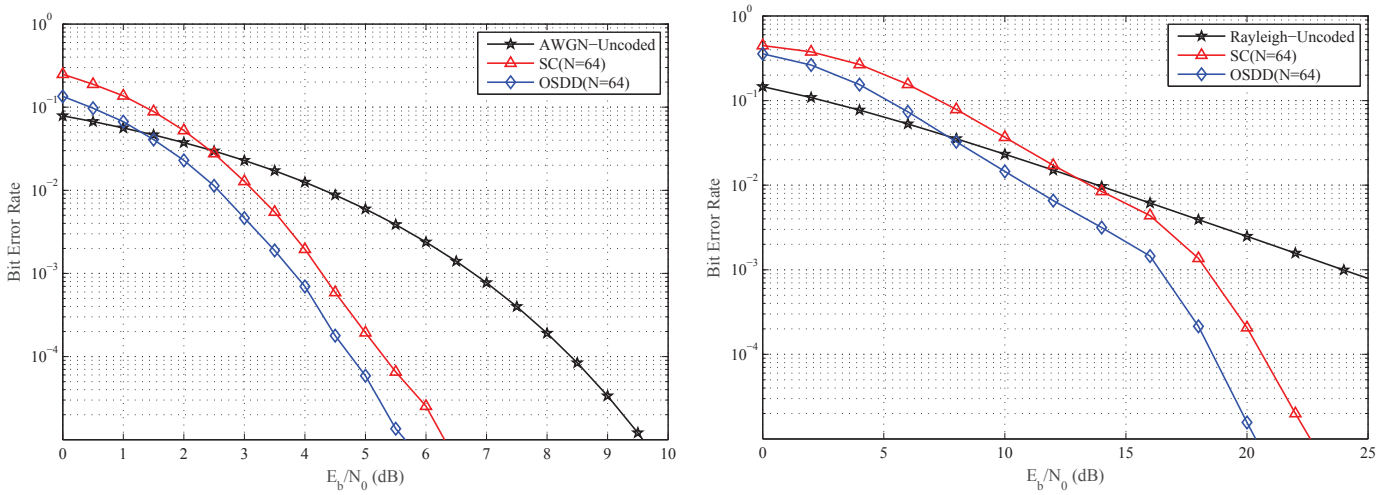
| Bit | Processing elements | |
|---|---|---|
| | SC | OSDD |
| $\hat{u}_0$ | 7 | 8 |
| $\hat{u}_1$ | 1 | 6 |
| $\hat{u}_2$ | 3 | 2 |
| $\hat{u}_3$ | 1 | 14 |
| $\hat{u}_4$ | 7 | 2 |
| $\hat{u}_5$ | 1 | 6 |
| $\hat{u}_6$ | 3 | 2 |
| $\hat{u}_7$ | 1 | 1 |
| Total | 24<br>$N \log_2 N$ | 41<br>$2N \log_2 N - (N-1)$ |

Table II shows a comparison between the proposed method with the SC decoder when $N = 8$. It is apparent from this table that there is no significant increasing in complexity.

The latency of the OSDD method doesn't change since the number of time elements required for the OSDD decoder is the same as the time elements are needed by the SC decoder as illustrated in Table III where the latency for both methods can be given by $2N - 1$.

## V. SIMULATION RESULTS

To illustrate the performance enhancement with the OSDD method, we present in Fig. 6a a comparison between the proposed method with the conventional SC decoder in AWGN channel. The results show the bit error rate (BER) versus $E_b/N_0$ where $E_b$ is the energy per information bit and $N_0$ is the spectral density of the noise considering binary phase shift keying modulation. In addition, the BER performance




(a) AWGN channel.

(b) Rayleigh channel.

Figure 6. BER performance of OSDD method, conventional polar code, and uncoded channel versus $E_b/N_0$ when $N = 64$ and code rate$= 0.5$: (a) AWGN, (b) Rayleigh channel.

Table III
LATENCY COMPARISON BETWEEN SC AND OSDD METHOD WHEN $N = 8$.

| Bit | Time elements | |
|---|---|---|
| | SC | OSDD |
| $\hat{u}_0$ | 3 | 3 |
| $\hat{u}_1$ | 1 | 1 |
| $\hat{u}_2$ | 2 | 2 |
| $\hat{u}_3$ | 1 | 1 |
| $\hat{u}_4$ | 3 | 3 |
| $\hat{u}_5$ | 1 | 1 |
| $\hat{u}_6$ | 2 | 2 |
| $\hat{u}_7$ | 1 | 1 |
| Total | 15<br>$2N-1$ | 15<br>$2N-1$ |

of the OSDD method is illustrated in Fig. 6b with Rayleigh fading channel is adopted. It is clearly seen that the OSDD method can enhance the behavior of the SC decoding in the both channels.

## VI. CONCLUSION

The performance of the SC decoder can be enhanced by applying the OSDD method. The results demonstrated that, in AWGN channels, the $Eb/No$ improvement is close to 0.75 dB, while in fading channel the improvement is more than 2 dB. The analysis shows that the proposed method has the same latency as that of the SC decoder; however, it entails a slight complexity increase.

## REFERENCES


[1] E. Arikan, "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, Jul. 2009.

[2] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, Oct. 2013.

[3] R. Mori and T. Tanaka, "Performance and construction of polar codes on symmetric binary-input memoryless channels," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, Jun. 2009, pp. 1496–1500.

[4] H. Li and J. Yuan, "A practical construction method for polar codes in AWGN channels," in *Proc. IEEE TENCON Spring Conference*, Apr. 2013, pp. 223–226.

[5] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.

[6] K. Niu and K. Chen, "Stack decoding of polar codes," *Electronics Letters*, vol. 48, no. 12, pp. 695–697, Jun. 2012.

[7] E. Arikan, "A performance comparison of polar codes and reed-muller codes," *IEEE Commun. Lett.*, vol. 12, no. 6, pp. 447–449, Jun. 2008.

[8] P. Shi, W. Tang, S. Zhao, and B. Wang, "Performance of polar codes on wireless communication channels," in *Proc. IEEE Communication Technology (ICCT)*, Nov. 2012, pp. 1134–1138.

[9] X. Wang, Z. Zhang, and L. Zhang, "On the polar codes for MIMO," in *Proc. International Conference on Wireless Communications Signal Processing (WCSP)*, Oct. 2013, pp. 1–5.

[10] M. Islam and R. Liu, "Polar coding for fading channel," in *Proc. International Conference on Information Science and Technology (ICIST)*, Mar. 2013, pp. 1096–1098.

[11] K. Chen, K. Niu, and J. Lin, "Improved successive cancellation decoding of polar codes," *IEEE Trans. Commun.*, vol. 61, no. 8, pp. 3100–3107, Aug. 2013.

[12] C. Leroux, A. Raymond, G. Sarkis, and W. Gross, "A semi-parallel successive-cancellation decoder for polar codes," *IEEE Trans. Signal Process*, vol. 61, no. 2, pp. 289–299, Jan. 2013.