EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

An optimized encoding algorithm for systematic polar codes

Xiumin Wang[1], Zhihong Zhang[1], Jun Li[2*], Yu Wang[3], Haiyan Cao[4], Zhengquan Li[5] and Liang Shan[1]

Abstract

Many different encoding algorithms for systematic polar codes (SPC) have been introduced since SPC was proposed in 2011. However, the number of the computing units of exclusive OR (XOR) has not been optimized yet. According to an iterative property of the generator matrix and particular lower triangular structure of the matrix, we propose an optimized encoding algorithm (OEA) of SPC that can reduce the number of XOR computing units compared with existing non-recursive algorithms. We also prove that this property of the generator matrix could extend to different code lengths and rates of the polar codes. Through the matrix segmentation and transformation, we obtain a submatrix with all zero elements to save computation resources. The proportion of zero elements in the matrix can reach up to 58.5% from the OEA for SPC when the code length and code rate are 2048 and 0.5, respectively. Furthermore, the proposed OEA is beneficial to hardware implementation compared with the existing recursive algorithms in which signals are transmitted bidirectionally.

Keywords: Encoding algorithm, Low complexity, Computing units, Systematic polar codes, Generator matrix transformation

1 Introduction

Polar codes proposed by Arikan [1] can theoretically reach the Shannon limit. It has been widely given attention in the communication field because of its low complexity and good decoding performance. As early as 2008, Arikan proposed the concept of channel polarization and carried out a rigorous mathematical proof [2]. For the selection of information bits of polar codes, there are some patterns, such as Bhattacharyya parameters [3] and density evolution [4]. The hardware implementations of the traditional non-systematic polar code (NSPC) encoding were introduced [5–7]. However, the complexity and computing resources were significantly consumed when fully utilizing the generator matrix of the polar codes to design the hardware structure. Hoyoung et al. [8] proposed partially parallel encoder structure that can reduce latency. However, the resource consumption increased due to the length of the polar codes. Compared to NSPC, the SPC proposed by Arikan [9] outperformed them in the bit error rate. Vangala et al. [10] proposed three encoding algorithms and used the recursion method to find a suitable equilibrium between a memory cell and a computing unit. Chen et al. [11] improved the encoding algorithms designed by Vangala et al. [10] to simplify the storage patterns for SPC. In [12], Sarkis et al. introduced SPC and developed a hardware structure of the decoder. The improved hardware structure increased the throughput of the polar decoder. When the hardware was implemented [13], the input and output ports were exchanged to obtain the SPC encoder based on the NSPC encoder. However, the latency and resource consumption were twice than that of the hardware implementation with NSPC. Wang et al. found a method that utilized the property of parity check matrix to reduce the computational units [14–16]. This approach can be utilized for both SPC and NSPC to reduce computational units. By studying the encoding flow graph of SPC, we found out that some of computational units can be omitted. This omission is determined by the encoding structure and the exclusive Or (XOR) operations.

In this paper, we define some variables which are determined by code length, code rate, and the selection for information bits. The two *lemmas* regarding the transformed generator matrix have been proved. After applying the property of the transformed generator

* Correspondence: 07a0303105@cjlu.edu.cn
[2]Binjiang College Nanjing University of Information Science and Technology, Wuxi 214105, China
Full list of author information is available at the end of the article

matrix, the proposed algorithm can reduce the number of XOR computing units according to the iterative property of the generator matrix. The iteration property can make the submatrix with a particular lower triangular structure. This submatrix is a part of the original generator matrix and can have all elements with zero value. Besides the zero submatrices obtained from the iteration process, there are still some other elements with zero value in the original matrix. These extra zero elements can also be omitted when computing XOR logical operations.

The rest sections of this paper are organized as follows. The comparisons between SPC and NSPC are introduced in Section 2. The existing encoding algorithms of SPC are reviewed in Section 3. Our simplified process to use XOR operations is also discussed in this section. The distribution rule of information bits, definition of variables, and the lemma proof are shown in Section 4. The case study of our optimized encoding algorithm (OEA) matrix transformation is presented in Section 5. Results and analysis are discussed in Section 6. Conclusions are found in Section 7.

1.1 Methods/Experimental

The research content of this paper is mainly theoretical derivation and analysis, and specific experimental verification will be carried out in future research.

2 Systemic and non-systemic polar codes

Based on the channel polarization theory, the length of polar codes is $N = 2^n$, $n \geq 1$, where n is a positive integer. Let A represent the set of the indices of the information bits. The code rate is $R = K/N$, where K represents the length of information bits and is the number of the elements in a set of A. Information bits selection is determined by Bhattacharyya parameters [17]. Let Ac represent the complement of a set of A. The index set $Ac = \{0,1,\ldots,N-1\} - A$ is for the frozen bits, and the length of Ac is $N - K$. $u = (u_0, u_1, \cdots, u_{N-1})$ represents the message vector, where u_i denotes an arbitrary element of the vector of u. $x = (x_0, x_1, \cdots, x_{N-1})$ represents a codeword vector, where x_i indicates a random component in the vector of x. The generator matrix G_N is defined as

$$G_N = F^{\otimes n} \tag{1}$$

where $\otimes$ denotes Kronecker power operation, $n = \log_2(N)$, and F denotes two-dimensional matrix $F = [1, 0; 1, 1]$. Applying the property of Kronecker product, we can construct a generator matrix as

$$G_N = \begin{bmatrix} F^{\otimes n-1} & 0 \\ F^{\otimes n-1} & F^{\otimes n-1} \end{bmatrix} = \begin{bmatrix} G_{N/2} & 0 \\ G_{N/2} & G_{N/2} \end{bmatrix} \tag{2}$$

The codeword vector of x for NSPC can be represented by the encoding Eq. (1)

$$x = uG_N = u_A G_A + u_{Ac} G_{Ac} \tag{3}$$

where GA is a submatrix of GN, and it is constructed by the rows of indices in A. GAc is a submatrix of GN and is constructed by the rows of indices in Ac. $uA = (ui\text{: } i \in A)$, uA u and $uAc = (uj\text{: } j \in Ac)$, uAc u, and $uAc = u - uA$. The symbol of $\oplus$ represents a mod-2 addition or a logical XOR operation in the binary domain.

Arikan [9] first proposed the mathematical formula shown in Eqs. (4) and (5) for SPC:

$$x_A = u_A G_{AA} + u_{Ac} G_{AcA} \tag{4}$$

$$x_{Ac} = u_A G_{AAc} + u_{Ac} G_{AcAc} \tag{5}$$

where G_{AA} denotes the submatrix of G_N consisting of the array of elements $(G_{i,j})$ with $i \in A$, $j \in A$, and $G_{AA} = (G_{i,j}\text{: } i \in A,\ j \in A)$. Similarly for the other submatrices of $G_{AcA} = (G_{i,j}\text{: } i \in Ac,\ j \in A)$, $G_{AAc} = (G_{i,j}\text{: } i \in A,\ j \in Ac)$, and $G_{AcAc} = (G_{i,j}\text{: } i \in Ac,\ j \in Ac)$. There is the same denotation for $x_A = (x_i\text{: } i \in A)$ and $x_{Ac} = (x_j\text{: } j \in Ac)$. If the matrix G_{AA} is invertible and the inputs to SPC encoder are u_{Ac} and x_A, then the output x_{Ac} from the SPC encoder is

$$x_{Ac} = (x_A - u_{Ac} G_{AcA}) G_{AA}^{-1} G_{AAc} + u_{Ac} G_{AcAc} \tag{6}$$

Consider that the decoding results are not affected by the value of frozen bits [1], we can simplify the encoding procedure by setting zero values to all frozen bits, namely $u_{Ac} = (u_i = 0\text{: } i \in Ac)$. Then, x_{Ac} in Eq. (6) can be simplified to

$$x_{Ac} = x_A G_{AA}^{-1} G_{AAc} \tag{7}$$

where ${G_{AA}}^{-1}$ is a lower triangular matrix with ones on the diagonal and the submatrix of G_{AAc} also includes a lower triangular matrix. Hence, the matrix product of ${G_{AA}}^{-1} G_{AAc}$ has the same structure as G_{AAc}. It has a submatrix including a lower triangular matrix. We will use the property of ${G_{AA}}^{-1} = G_{AA}$ in the binary domain [9] to prove the Lemma 1 in Section 4 and the matrix transformation in Section 5.

3 Proposed OEA for SPC

The development of the proposed OEA was motivated by [9, 11]. The computational complexity can be decreased by reducing the number of logical XOR computing units. The following example illustrates the procedure of omitting XOR computing units. After logical XOR operations, the output results are the same either from the approach to apply a generator matrix or the method to apply an encoding diagram scheme.

Figure 1 shows the SPC encoding diagram with $N = 8$ and $A = \{1,3,5,6,7\}$. In Fig. 1, the encoding direction is from bottom to top. The encoding process for information bits starts from right to left. Then, the encoding process for frozen bits is from left to right. The gray circles on the rightmost represent the information bits, and the black circles on the leftmost represent the frozen bits. The black arrow on the right represents the computing direction of the information bits. The black arrow on the left represents the computing direction of the frozen bits. Since the value of the frozen bits does not affect the encoding result, we set them to zero. The final outputs of the encoding are the value of the rightmost black circles.

For a SPC encoder in Fig. 1, we can reduce the XOR operation to obtain the output of x_0, x_2, and x_4. From Fig. 1, we can obtain

$$x_0 = u_0 \oplus u_4 \oplus u_2 \oplus u_6 \oplus u_1 \oplus u_5 \oplus u_3 \oplus u_7 \tag{8}$$

Consider that u_0, u_2, and u_4 are frozen bits which are set zero. Then, Eq. (8) can be rewritten as

$$x_0 = u_6 \oplus u_1 \oplus u_5 \oplus u_3 \oplus u_7 \tag{9}$$

From Fig. 1, we also have the following relations

$$\begin{cases} u_7 = x_7, \\ u_6 = x_6 \oplus x_7, \\ u_5 = x_5 \oplus x_7, \\ u_3 = x_3 \oplus x_7, \\ u_1 = x_1 \oplus x_3 \oplus x_5 \oplus x_7 \end{cases} \tag{10}$$

Substitute Eq. (10) into Eq. (9), x_0 becomes

$$x_0 = (x_6 \oplus x_7) \oplus (x_1 \oplus x_3 \oplus x_5 \oplus x_7) \oplus (x_5 \oplus x_7) \oplus (x_3 \oplus x_7) \oplus x_7 \tag{11}$$

After applying logical XOR operations, the output results become zero if the input values are the same. Thus, Eq. (11) can be rewritten as

$$x_0 = x_6 \oplus x_1 \oplus x_7 \tag{12}$$

Similarly, x_2 and x_4 can be derived

$$x_2 = u_2 \oplus u_6 \oplus u_3 \oplus u_7 = u_6 \oplus u_3 \oplus u_7 \tag{13}$$
$$= (x_6 \oplus x_7) \oplus (x_3 \oplus x_7) \oplus x_7 = x_6 \oplus x_3 \oplus x_7$$

$$x_4 = u_4 \oplus u_6 \oplus u_5 \oplus u_7 = u_6 \oplus u_5 \oplus u_7 \tag{14}$$
$$= (x_6 \oplus x_7) \oplus (x_5 \oplus x_7) \oplus x_7 = x_6 \oplus x_5 \oplus x_7$$

By combining Eqs. (12), (13), and (14), we simplify the encoding diagram illustrated in Fig. 1 to the diagram in Fig. 2.

Compared with the computation complexity of original algorithm [11], the proposed OEA reduces the computational units without increasing the memory bits. Figures 1 and 2 show the difference of the number of computing units. There are only four XOR computing units to be reduced in Fig. 1. However, 12 XOR operations are omitted in Fig. 2. u_1, u_3, u_5, u_6, and u_7 are intermediate variables that can be ignored. The gray nodes on the rightmost represent the information bits, and the black nodes on the leftmost represent the frozen bits. The right black nodes are unknown. The outputs of the SPC encoder are x_0, x_2, and x_4.

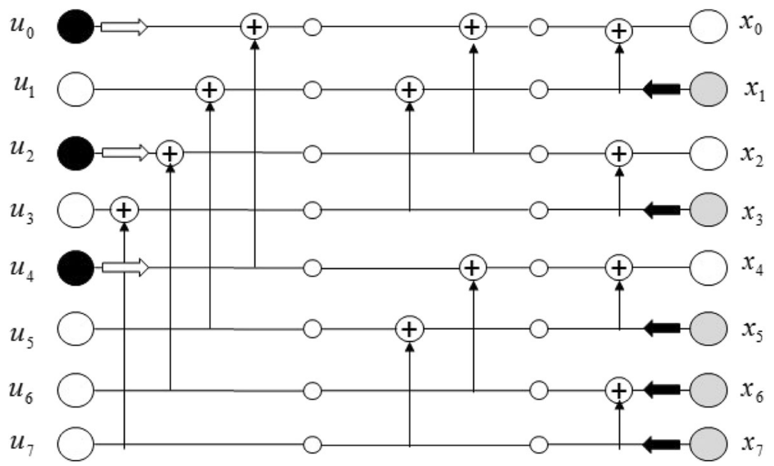

Fig. 1 SPC encoding diagram with $N = 8$ and $A = \{1, 3, 5, 6, 7\}$: The encoding direction is from bottom to top. The encoding process for information bits starts from right to left. Then, the encoding process for frozen bits is from left to right. The gray circles on the rightmost represent the information bits, and the black circles on the leftmost represent the frozen bits. The black arrow on the right represents the computing direction of the information bits. The black arrow on the left represents the computing direction of the frozen bits. The final outputs of the encoding are the value of the rightmost black circles

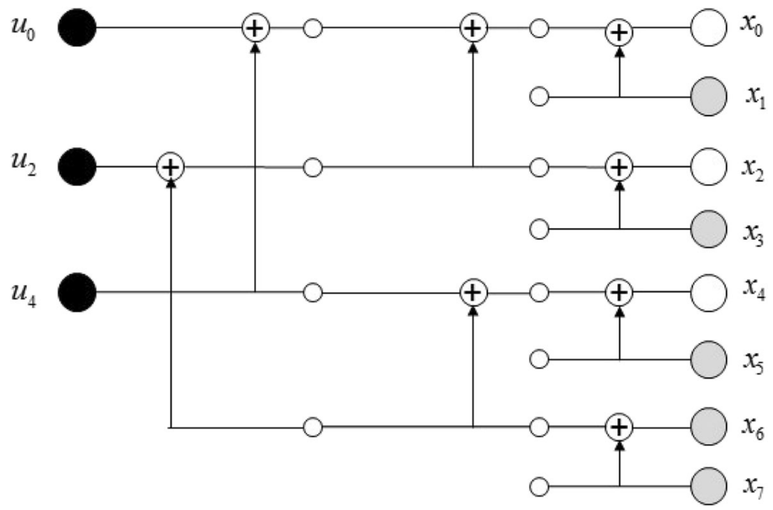

Fig. 2 Our SPC encoding optimization algorithm diagram: The gray nodes on the rightmost represent the information bits, and the black nodes on the leftmost represent the frozen bits. The right black nodes are unknown and the outputs of the SPC encoder are to obtain x_0, x_2, and x_4

The outputs of the SPC encoder also can be generated from a two-dimensional generator matrix. For example, we construct a generator matrix with the polar code length $N = 8$

$$G_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{15}$$

The codeword vector x can be obtained from the matrix product of u (Eq. (3)) and above G_8 (Eq. (15)). For example, we can have the element x_0 (Eq. (16)) by multiplying u (Eq. (3)) with G_8 (Eq. (15))

$$x_0 = u_0 \oplus u_1 \oplus u_2 \oplus u_3 \oplus u_4 \oplus u_5 \oplus u_6 \oplus u_7 \tag{16}$$

Comparing x_0 in Eq. (16) to x_0 in Eq. (8), we found that the encoding result is the same. One is from an SPC encoding diagram (Eq. (8)), and another one is from a generator matrix approach (Eq. (16)). The proposed OEA utilizes the characteristics of the generator matrix and the matrix transformation. For a general case with the code length N, Section 4 shows that the outputs from a generator matrix approach are the same as the outputs from a diagram approach. For example, when $N = 8$ in Eq. (7), $A = \{1,3,5,6,7\}$, and $G_{AA}{}^{-1}G_{AAc} = [1, 0, 0; 0, 1, 0; 0, 0, 1; 1, 1, 1; 1, 1, 1]$. x_{Ac} has the elements of x_0, x_2, and x_4 as outputs. These outputs are the same as the results from Eqs. (12), (13), and (14). Therefore, the characteristics of the generator matrix can be considered to use in the process of our encoding optimization algorithm for a general case.

4 The theory of OEA

Before discussing the characteristics of the generator matrix, we first divide the distribution of information bits [15] into two groups based on the preset value. Then, we divide the distribution of information bits into two areas based on the bit channel index. Figures 3, 4, 5, and 6 show the capacity of binary erasure channels (BEC) when the code length N is 128, 256, 1024, and 2048, respectively. We can map the bit channel index to the set of A and the set of Ac according to the capacity value larger than or smaller than the preset value. For the areas of all bit channel index in A or all bit channel index in Ac, we define them as non-hybrid areas. For the area of the bit channel index belonging to both A and Ac, we define it as a hybrid area. For example, in Fig. 3, we can set up 0.7 as a preset capacity value. When the capacity value is larger than 0.7, the bit channel index belongs to the set of A. When the capacity value is smaller than 0.7, the bit channel index belongs to the set of Ac. For the set of A, we select the lowest index in the area with capacity value larger than 0.7. We draw a line 1 across the lowest index and denote the left side of line 1 as a non-hybrid area. Similarly, for the set of Ac, we select the highest index in Ac to the area with capacity value smaller than 0.7. We draw a line 2 across the highest index. We call the right side of line 2 a non-hybrid area. We denote the area

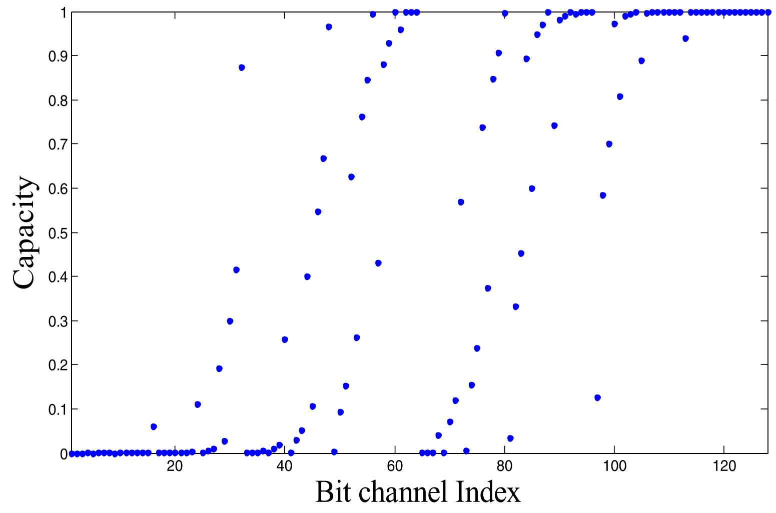

Fig. 3 The capacity of bit channel at N = 128: The capacity of binary erasure channels (BEC) when the code length N is 128

between the line 1 and the line 2 as a hybrid area. There are both frozen bits and information bits distributed in this hybrid area. For Figs. 4, 5, and 6, we can use the same approach to divide the areas into a hybrid or a non-hybrid area and map the distribution of information bits into the set of A and Ac.

The following variables are defined to describe the highest index value in Ac, the lowest index value in A, the number of information bits, and the number of frozen bits in the hybrid and non-hybrid areas. The defined variables are listed in the Table 1.

For the non-hybrid area, the index value of p_{fi} represents the number of the frozen bits, and $N - p_{lf} - 1$ represents the number of the information bits, where N has been defined as the code length previously. The value of N is the total number of all frozen bits and information bits. The width of the hybrid area in the generator matrix can be represented by$\Delta = p_{lf} - p_{fi} + 1$. The value of Δ also represents the sum of the frozen bits and the information bits in this hybrid area. Figure 7 shows these defined variables of p_{fi}, Δ, $N - p_{lf} - 1$ when the code length N is 16.

In Section 2, we have denoted G_N as a lower triangular with all ones across the diagonal and G_N is invertible. In binary GF (2), the invertible matrix is equal to itself, $G_N^{-1} = G_N$. $G_{i,j}$ represents an element in the matrix of G_N. $i, j \in N$, where $N = \{0,1,...,N-1\}$. When $i = j$, $G_{i,j} = 1$; when $i < j$, $G_{i,j} = 0$.

We have following discrete function definitions. These defined functions are used to show the lower triangle

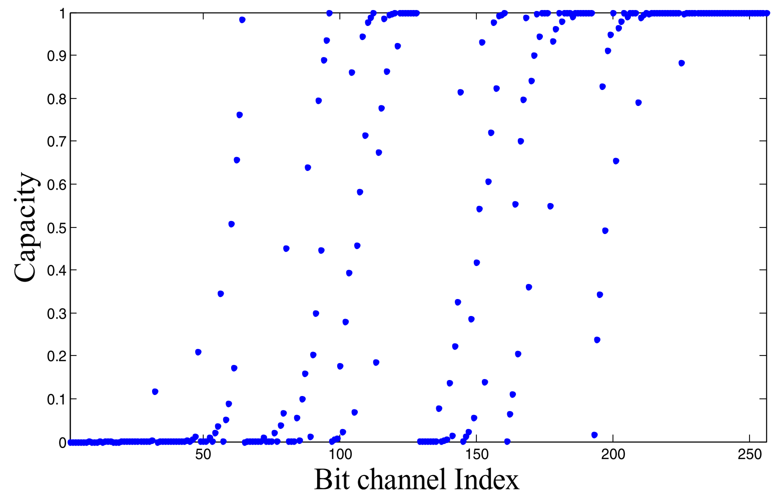

Fig. 4 The capacity of bit channel at N = 256: The capacity of binary erasure channels (BEC) when the code length N is 256

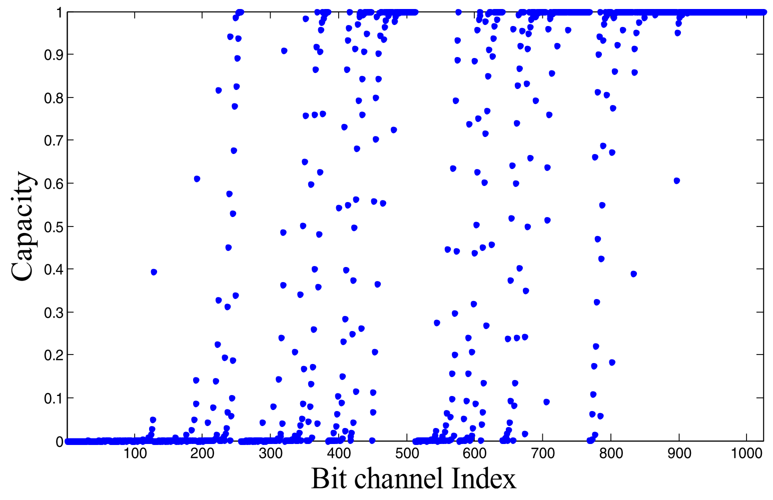

Fig. 5 The capacity of bit channel at N = 1024: The capacity of binary erasure channels (BEC) when the code length N is 1024

structure property of the generator matrix G_N and its submatrices of G_{AA} and G_{AAc}.

Discrete function definition:

$f_N(x) = x$, where a discrete variable x of the function is $x \in N$. The discrete function is $N, f_N(x) \subset N$.

$f_A(x) = A(x)$, where a discrete variable x of the function is $x \in \{1, 2, ..., K\}$, K is the length of information bits. The discrete function is $A(x)$, $f_A(x) \subset A$, and $f_A(1) = A(1) = p_{fi}$. $f_A(x)$ is a monotone increasing function.

$f_c(x) = Ac(x)$, where the discrete variable x of the function is $x \in \{1, 2, ..., N - K\}$, N is the code length and $N - K$ is the length of frozen bits. The discrete function is $Ac(x)$, $f_c(x) \subset Ac$, and $f_c(N - K) = Ac(N - K)$. $f_c(x)$ is a monotone increasing function.

Since both matrix G_N and G_{AA} are square matrices, for the defined function $f_N(x) = x$, in which x represents the xth row of G_N as well as the xth column of G_N, $f_N(x)$ can represent the $f_N(x)$th row of G_N as well as the $f_N(x)$ th column of G_N. For $f_A(x) = A(x)$, x represents the xth row of G_{AA} and G_{AAc}, as well as xth column of G_{AA}. $f_A(x)$ represents $A(x)$th row in G_N as well as $A(x)$th column of G_N. For $f_c(x) = Ac(x)$, x represents the xth column of G_{AAc} and $f_c(x)$ represents the $Ac(x)$th row of G_N as well as $Ac(x)$th column of G_N.

Lemma 1: *G_{AA} is a lower triangular with all ones across the diagonal.*

Proof: For G_{AA}, when $x = y$, $x,y \in \{1,2,... ,K\}$, $f_A(x) = f_A(y)$. So $f_N(f_A(x)) = f_N(f_A(y))$. Since $f_A(x), f_A(y) \in A \subset N$, and

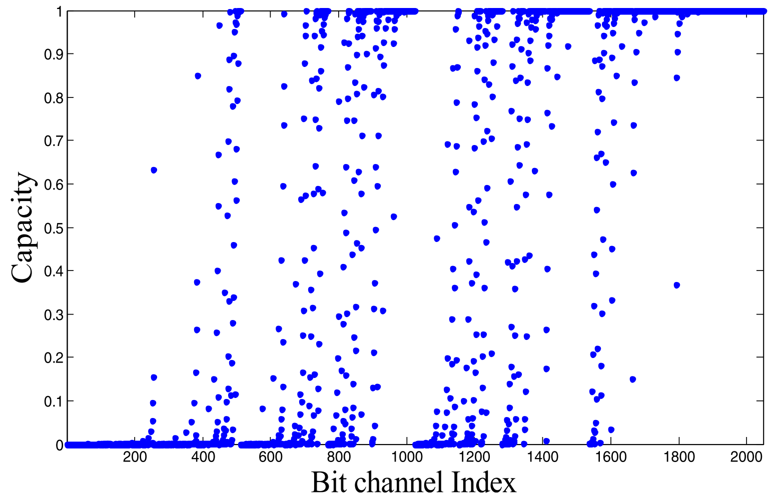

Fig. 6 The capacity of bit channel at N = 2048: The capacity of binary erasure channels (BEC) when the code length N is 2048

Table 1 Defined variables

Variable	Definition
p_{fi}	The first information bit index
p_{lf}	The last frozen bit index
Δ	The width of the hybrid area
Δi	The number of information bits in hybrid area
Δf	The number of frozen bits in hybrid area

$G_{fA(x),fA(y)} = 1$, $(G_{AA})_{x,y} = 1$. When $x < y$, $x,y \in \{1,2,\ldots,K\}$, due $f_A(x)$ is a monotone increasing discrete function, so $f_A(x) < f_A(y)$. Hence $f_N(f_A(x)) < f_N(f_A(y))$. Because of $f_A(x)$, $f_A(y) \in A \subset N$, we obtain that $G_{fA(x),fA(y)} = 0$. Hence, $(G_{AA})_{x,y} = 0$. Given the property that the inverted matrix has the lower triangular structure if the original one has the lower triangular matrix, G_{AA} is a lower triangular; therefore, the inverted $G_{AA}{}^{-1}$ has a lower triangular matrix.

Lemma 2: *A submatrix of G_{AAc} has the lower triangular structure.*

Proof: For $f_A(x_1)$, when $x_1 \in \{1,2,\ldots,\Delta i\}$, $f_A(x_1) \in \{A(1), A(2), \ldots, A(\Delta i)\}$, $f_A(1) = A(1) = p_{fi}$. For $f_c(x_2)$, in which $x_2 \in \{N - K - \Delta f, N - K - \Delta f{+}1, \ldots, N{-}K\}$, $f_c(x_2) \in \{Ac(N - K - \Delta f), Ac(N - K - \Delta f{+}1), \ldots, Ac(N - K)\}$. $f_c(N - K) = Ac(N - K) = p_{lf}$. In the hybrid area, we know that $A(1) < Ac(N - K - \Delta f)$ and $A(\Delta i) < Ac(N - K)$. As we know from the above analysis, each information bit index is smaller than the previous frozen bit index. Due to $f_A(x)$ and $f_c(x)$ are monotone increasing discrete functions, so $f_A(x_1) < f_c(x_2)$, $G_{fA(x1),fc(x2)} = 0$, and $(G_{AAc})_{x1,x2} = 0$. The property of a submatrix G_{AAc} with lower triangular structure exists if $f_A(x_1) < f_c(x_2)$, where x_1 and x_2 are in different discrete sets, $x_1 \in \{1,2,\ldots,\Delta i\}$ and $x_2 \in \{N - K - \Delta f, N - K - \Delta f{+}1, \ldots, N - K\}$.

5 A case study of OEA

For the case of code length of $N = 16$, suppose the code rate is 1/2, we will have $K = 8$, $A = \{3, 5, 7, 9, 11, 13, 14, 15\}$, and $Ac = \{0, 1, 2, 4, 6, 8, 10, 12\}$. For the generator matrix of G_{16} in Fig. 8a, the rows of information bits indices can be extracted to form $(G_{16})_A$ shown in Fig. 8a, b, c$_1$, and c$_2$ which illustrate the detailed procedures of the matrix transformation.

The row elements in the solid line box in Fig. 8a form a new matrix $(G_{16})_A$ shown in Fig. 8b. In Fig. 8b, $f_A(1) = A(1) = 3$ and $f_c(N - K) = Ac(N - K) = 12$, so the third column represents the first information bit p_{fi} and the 12th column represents the last frozen bit p_{lf} . We can form $(G_{16})_{AA}$ in Fig. 8c$_2$ by extracting the columns from the dashed box in Fig. 8b. To form $(G_{16})_{AAc}$ shown in Fig. 8c$_1$, we can use the remaining columns from the Fig. 8b. By the Lemma 1, $(G_{16})_{AA}$ is a lower triangular matrix with all ones across the diagonal, $(G_{16})_{AA}{}^{-1} = (G_{16})_{AA}$, $G_{AA}{}^{-1}$ is a lower triangular matrix. By the Lemma 2, $(G_{16})_{AAc}$ has a lower triangular structure. For the general case of $N = 2^n$, $n \geq 1$, the dimension of the lower triangular matrix in $(G_{16})_{AAc}$ is $N - K - p_{fi} + 1$. The dimension of $(G_{16})_{AAc}$ is $N - K$, minus the first part of the matrix, so the remaining matrix size is $d = N - K - p_{fi} + 1$. $G_{AA}{}^{-1}$ is the inverse of itself, and it is a lower triangular matrix with the dimension of K.

The matrix $(G_{16})_{AA}{}^{-1}$and $(G_{16})_{AAc}$ in Fig. 9a and b are partitioned to obtain a block of a zero submatrix. To further discuss about general case, Fig. 9a can be

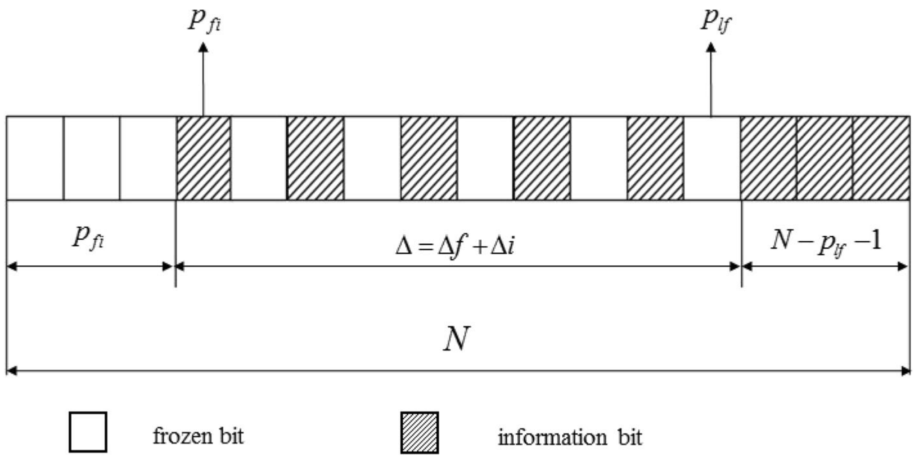

Fig. 7 The information bits and frozen bits distribution at $N = 16$: For the non-hybrid area, the index value of p_{fi} represents the number of the frozen bits, and $N - p_{lf} - 1$ represents the number of the information bits, where N has been defined as the code length previously. The value of N is the total number of all frozen bits and information bits. The width of the hybrid area in the generator matrix can be represented by$\Delta = p_{lf} - p_{fi} + 1$. The value of Δ also represents the sum of the frozen bits and the information bits in this hybrid area. These defined variables of p_{fi}, Δ, $N - p_{lf} - 1$ when the code length N is 16

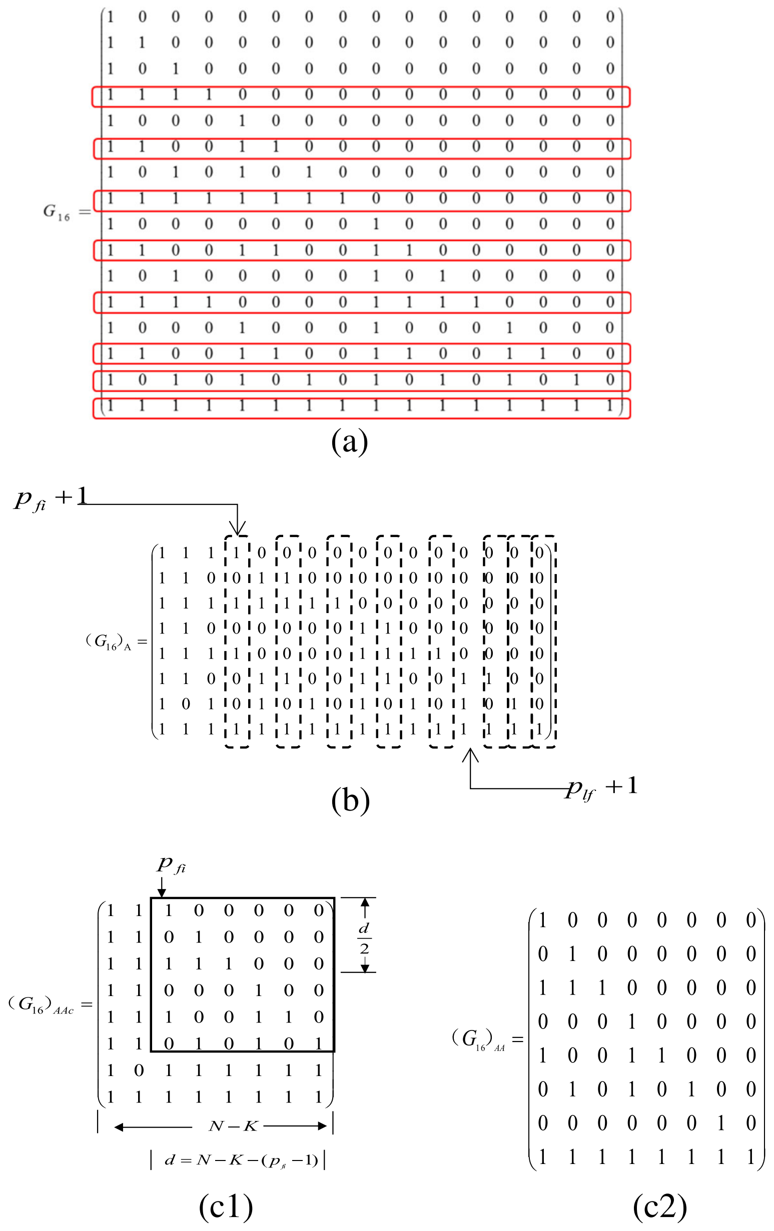

Fig. 8 Matrix extraction process: For the generator matrix of G_{16} in a, the rows of information bits indices can be extracted to form $(G_{16})_A$ shown in **b**. **a**–**c$_2$** illustrate the detailed process of matrix transformation. The row elements in the solid line box in a are formed into a new matrix $(G_{16})_A$ shown in **b**

partitioned and written as several submatrices of g_1, g_2, g_3, and g_4 in Fig. 10a. The submatrix g_2 is a zero matrix. Figure 9b can be partitioned into submatrices of c_1, c_2, c_3, and c_4 in Fig. 10b. The submatrix c_2 is a zero matrix.

6 The block matrix in Fig. 10a and b can be written as Eqs. (17) and (18), respectively

$$G_{AA}^{-1}=\begin{pmatrix} g_1 & g_2 \\ g_3 & g_4 \end{pmatrix} \tag{17}$$

$$G_{AAc}=\begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix} \tag{18}$$

When d is even, the sizes of the submatrices of g_1, g_2, g_3, and g_4 are $\frac{d}{2}\times\frac{d}{2}$,$\frac{d}{2}\times(K-\frac{d}{2})$,$(K-\frac{d}{2})\times\frac{d}{2}$, and $(K-\frac{d}{2})\times(K-\frac{d}{2})$, respectively. And the sizes of submatrices of c_1, c_2, c_3, and c_4 are $\frac{d}{2}\times(N-K-\frac{d}{2})$,$\frac{d}{2}\times\frac{d}{2}$,$(K-\frac{d}{2})\times(N-K-\frac{d}{2})$, and $(K-\frac{d}{2})\times\frac{d}{2}$, respectively. Due to

$$(G_{16})_{AA}^{-1} = \left(\begin{array}{ccc:ccccc}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\1&1&1&0&0&0&0&0\\ \hdashline 0&0&0&1&0&0&0&0\\1&0&0&1&1&0&0&0\\0&1&0&1&0&1&0&0\\0&0&0&0&0&0&1&0\\1&1&1&1&1&1&1&1\end{array}\right)$$

(a)

$$(G_{16})_{AAc} = \left(\begin{array}{ccccc:ccc}1&1&1&0&0&0&0&0\\1&1&0&1&0&0&0&0\\1&1&1&1&1&0&0&0\\ \hdashline 1&1&0&0&0&1&0&0\\1&1&1&0&0&1&1&0\\1&1&0&1&0&1&0&1\\1&0&1&1&1&1&1&1\\1&1&1&1&1&1&1&1\end{array}\right)$$

(b)

Fig. 9 Matrix segmentation at $N = 16$: The matrix $(G_{16})_{AA}^{-1}$and $(G_{16})_{AAc}$ in **a** and **b** can be divided to obtain the submatrix with all elements of zeros, respectively. To further discussion general case, **a** can be segmented into submatrices of g_1, g_2, g_3, and g_4 shown in Fig. 10a. The submatrix g_2 is a zero matrix

the zero property of g_2 and c_2, Eq. (7) can be rewritten as

$$x_{Ac} = x_A G_{AA}^{-1} G_{AAc} = x_A \begin{pmatrix} g_1 c_1 & 0 \\ g_3 c_1 + g_4 c_3 & g_4 c_4 \end{pmatrix} \tag{19}$$

For Eq. (19), when d is even, the dimension of the zero submatrix in $G_{AA}{}^{-1}G_{AAc}$ is $\frac{d}{2} \times \frac{d}{2}$; when d is odd, the size of the zero submatrix in $G_{AA}{}^{-1}G_{AAc}$ is $\frac{d-1}{2} \times \frac{d-1}{2}$. The computing procedures are omitted when multiplying such zero submatrix during an encoding process. Therefore, we can save the computing resources after applying the proposed OEA. The OEA is universal for general cases. The pseudocodes of our OEA are listed in the algorithm for the proposed OEA. We can clearly understand the characteristic of the proposed algorithm and the difference between the encoding algorithm in [11].

Algorithm for the proposed OEA

```
Input:   The code length $N$
         The message vector $u$
         The index vector of the information bit $A$
         The index vector of the frozen bit $Ac$
Output:  The encoding codewords $x$
1: $n = \log_2(N)$ and $G_N = F^{\otimes n}$ is an $N \times N$ binary matrix.
2: Set: $G_{AA}=(G_{i,j}: i\in A, j\in A)$, $G_{AcA}=(G_{i,j}: i\in Ac, j\in A)$, $G_{AAc}=(G_{i,j}: i\in A, j\in Ac)$, $G_{AcAc}=(G_{i,j}: i\in Ac, j\in Ac)$, $x_A=(x_i: i\in A)$, $x_{Ac}=(x_j: j\in Ac)$, $u_A=(u_i: i\in A)$, $u_{Ac}=(u_j: j\in Ac)$.
3: $x_A = u_A G_{AA} + u_{Ac} G_{AcA}$ and $x_{Ac} = x_A G_{AA}^{-1} G_{AAc}$.
4: $x = x_A + x_{Ac}$.
```

7 Result analysis

Table 2 discusses the dimension d of lower triangular submatrix of $GAAc$ at different code lengths of N and

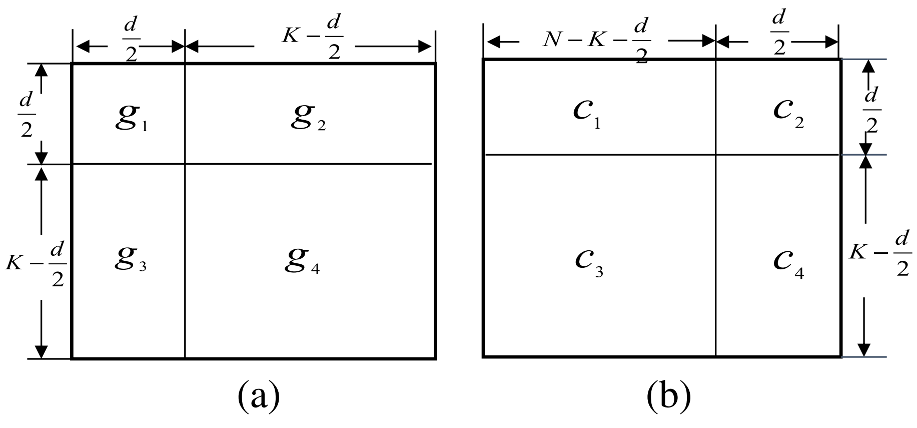

Fig. 10 Matrix segmentation at $N = 2^n$: Fig. 9a can be partitioned and written as several submatrices of g_1, g_2, g_3, and g_4 shown in **a**. Fig. 9b can segmented into submatrices of c_1, c_2, c_3, and c_4 shown in **b**. The submatrix c_2 is a zero matrix

Table 2 Detailed data of *d*

N	*R*	*K*	p_{fi}	*d*
1024	0.5	512	16	497
	0.75	768	4	253
	0.9	922	4	99
2048	0.5	1024	16	1009
	0.75	1536	8	505
	0.9	1843	4	202
4096	0.5	2048	32	2017
	0.75	3072	8	1017
	0.9	3686	1	410

different code rates of R, where $d = N - K - p_{fi} + 1$. Detailed data of three different code lengths and three different code rates for each of them are listed in this table.

Figure 11 shows the ordinate in a linear plot. The relation between d and R is in a linear relationship for each of code length N. The value changes of p_{fi} can be negligible to a code length.

Figure 12 shows the variation trend of d with the code length N. At different code rates (for example, R = 0.5, 0.75, and 0.9), d increases when the code length N becomes short. Especially, when $R = K/N = 0.5$, the growth trend is faster than others. In other words, the smaller the code rate is, the greater the value of d will be.

After applying the proposed OEA to the transformed matrix, the percentages of zero elements of the lower triangular part are shown in Table 3. For SPC and NSPC, the percentages of zero elements shown in Table 3 are calculated by Eqs. (20) and (21), respectively.

$$P_{OEA-NSPC} = \frac{(d^2 - d)/2}{N^2} \tag{20}$$

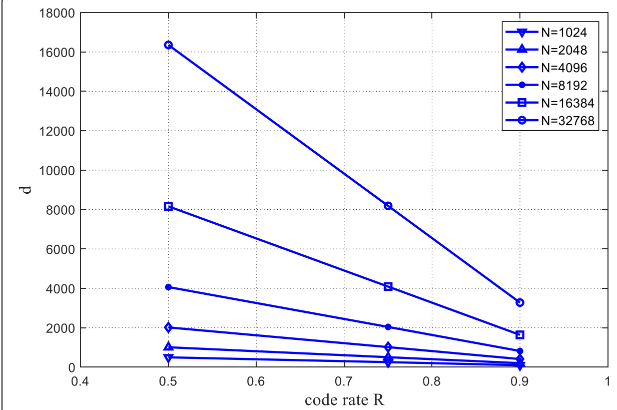

Fig. 11 Variation trend of *d* with code rate *R*: The ordinate in a linear form. The relation between *d* and *R* is in linear for each of code length *N*. The value changes of p_{fi} can be negligible to a code length

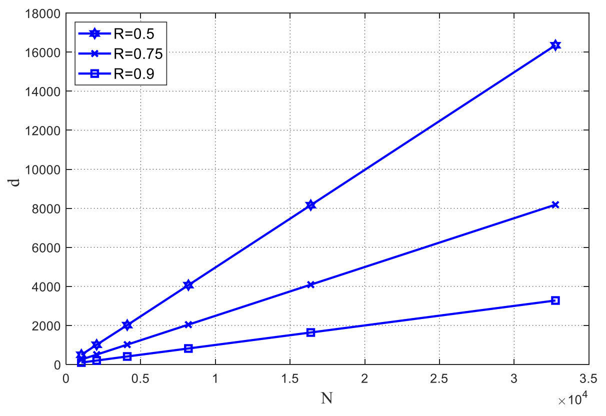

Fig. 12 Variation trend of *d* with code length log_2N for a different *R*: The variation trend of *d* with code length *N*. And it shows a semi-logarithmic relationship. At different code rates, *d* increases with the code length *N* for *R* = 0.5, 0.75, and 0.9, respectively

$$P_{OEA-SPC} = \frac{(d^2 - d)/2}{N \times (N - K)} \tag{21}$$

For NSPC, 0.47% of the computing resources can be saved when N = 1024 and R = 0.9. However, when N = 1024 and R = 0.5, only 11.8% of the computing resources can be saved. For SPC, 58.5% of the resource consumption can be saved when code rate R = 0.5 and the code length N = 2048. When the code rate increases, the percentage of zero elements in the lower triangle decreases. However, the percentage is almost invariable at different code lengths for SPC and NSPC at the same code rate.

The comparison of systematic polar encoders is shown in Table 4. Unlike [10, 11], we use the total times of XORs to represent the computational complexity in this paper. For the Encoder A in [10] and the SPC in [11], the times of XORs can be approximated as $(N^2 + N)(n^2 + n)/2 > \frac{n^2}{2} \cdot N^2 > N^2$ and $N(N^2 + N)/2 \geq \frac{N^3}{2}$, respectively. However, for the proposed OEA, the computational complexity can be decreased to NK due to $NK - K^2 - N + K = (N - K)(K - 1) < NK$. Therefore, the computational complexity of the OEA is lower than that of Encoder A in [10] and SPC in [11]. As for the Encoder B, Encoder C, and NSPC in [10], the times of XORs can be

Table 3 Percentages of zero elements in the transformed matrix for our OEA algorithm

R	$N = 2^{10} = 1024$		$N = 2^{11} = 2048$		$N = 2^{12} = 4096$	
OEA	NSPC (%)	SPC (%)	NSPC (%)	SPC (%)	NSPC (%)	SPC (%)
0.5	11.8	47.0	10.9	58.5	12.3	49.2
0.75	3.0	16.2	3.0	16.2	3.1	16.4
0.9	0.47	5.2	0.48	5.4	0.49	5.5

Table 4 Comparison of systematic polar encoders

Algorithm	Recursion	Memory	XORs (times)	Hardware implementation
Encoder A [10]	No	$N(1 + log_2N)$	$(N^2 + N)(n^2 + n)/2$	Yes
Encoder B [10]	Yes	$2N - 1$	$N(1 + log_2N)$	No
Encoder C [10]	Yes	N	$N(1 + 2log_2N)$	No
NSPC [10]	Yes/No	$2N$	$(Nlog_2N)/2$	No
SPC [11]	No	N	$N(N^2 + N)/2$	Yes
Our OEA	No	N	$NK - K^2 - N + K$	Yes

approximated as $N(1 + \log_2 N) > o(N)$, $N(1 + 2\log_2 N) > o(N)$, and $\frac{N}{2}\log_2 N > o(N)$, respectively. Compared with the recursive algorithms Encoder B, Encoder C, and NSPC in [10], the advantage of the times of XORs of the proposed OEA is not obvious, but it is beneficial to hardware implementation. For the proposed OEA, the operation of the matrix segmentation and transformation can be completed in the preprocessing stage, followed by multiplication of matrix and vector, which can be realized by XOR gate. There is no reverse transmission process of signal, which is beneficial to the timing and pipeline design of the hardware. The Encoder B, Encoder C, and NSPC in [10] are implemented by the recursive algorithm based on divide-and-conquer method in the software design, and the signal is passed from the first level to the last level, and then passed back to the first level. This process of bidirectional transmission of signals is not conducive to hardware implementation, especially in high-speed pipeline structures.

8 Conclusions and prospect

In this paper, we propose an optimization algorithm OEA for SPC. The number of zero elements in the transformed generator matrix and their locations can be determined in the proposed OEA. In the case of code rate reaching 0.5, half of the number of XOR computing units can be omitted to save computation resources due to the large number of zero elements found in the submatrix. For other code rates higher than 0.5, a smaller number of the computation units is saved. The proposed OEA not only reduces the number of XOR computing units compared with the existing non-recursive algorithms, but also is beneficial to hardware implementation compared with the existing recursive algorithms.

Abbreviations

BEC: Binary erasure channels; NSPC: Non-systematic polar codes; OEA: Optimized encoding algorithm; SPC: Systematic polar codes; XOR: Exclusive Or

Acknowledgements

The authors are grateful for the valuable comments and suggestions provided by the reviewers.

Authors' contributions

JL, XW, and ZZ performed the experiments and wrote the paper. YW, HC, ZL, and LS helped revise and improve the whole paper. All authors read and approve the final manuscript.

Authors' information

Jun Li was born on February 1977. He received the Ph.D degree in Communication and Information System from South China University of Technology, Guangzhou, China, in 2005. He worked in ZTE Corporation in 2005. And he has been an associate professor since 2007, and now he is in Binjiang College, Nanjing University of Information Science and Technology. His research interests include channel coding, signal detection, LTE physical layer standard, and wireless resource optimization.

Zhihong Zhang is a Master's student in College of Information Engineering in China Jiliang University. She received the B.E. degree in Communication Engineering from Yantai University, in 2015. Her research interests include signal and information processing.

Xiumin Wang was born in April 1, 1963. She received the B.E. degree in Communication and Electronic System from Dalian University of Technology. She is now a professor, Associate Dean of College of Information Engineering in China Jiliang University. Her research interests include signal and information processing.

Prof. Yu Wang earned her Ph.D. degree in Electrical Engineering from the Graduate Center of the City University of New York. Dr. Wang joined the Computer Engineering Technology Department of New York City College of Technology in 2009. Her primary area of interest includes LTE networks, real-time systems, network protocols, and embedded systems.

Haiyan Cao received the Ph.D degree in information engineering from South China University of Technology, Guangzhou, China, in 2006. She is currently an associate professor of College of Communication Engineering in Hangzhou Dianzi University. Her research interests include Massive MIMO, OFDM, LDPC codes, and resource allocation.

Zhengquan Li received the B.S. degree from the Jilin University of Technology in 1998, the M.S. degree from the University of Shanghai for Science and Technology in 2000, and the Ph.D. degree in circuit and system from Shanghai Jiaotong University in 2003. He is currently a Professor with Jiangnan University. He is also a Postdoctoral researcher with the National Mobile Communications Research Laboratory, Southeast University. His current research interests include space time coding and cooperative communications and massive MIMO.

Liang Shan received the B.S. degree from SouthEast University of Radio Engineering in 2001, the M.S. degree from South East University of Signal and Information processing in 2004. She is currently an associate professor with China Jiliang University. Her current research interests include signal processing and measurement.

Funding

This work was supported in part by the National Natural Science Foundation of China, under Grant No. 51874264, 61571108, 61501158, and 61379027; in part by the Open Foundation of State key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, under Grant SKLNST-2016-2-14; in part by the Jiangsu Postdoctoral Foundation, under Grant No. 1302059B.

Competing interests

The authors declare that they have no competing interests.

Author details
[1]College of Information Engineering, China Jiliang University, Hangzhou 310018, China. [2]Binjiang College Nanjing University of Information Science and Technology, Wuxi 214105, China. [3]Department of computer engineering technology, New York City College of Technology, New York 11201, USA. [4]College of Communication Engineering, Hangzhou Dianzi University, Hangzhou 310018, China. [5]School of IoT Engineering, Jiangnan University, Wuxi 214122, China.

Received: 20 June 2018 Accepted: 10 June 2019
Published online: 01 August 2019

References
1. E. Arikan, Channel Polarization: A method for constructing capacity achieving codes for symmetric Serbian-input memoryless channels. IEEE Trans. Inf. Theory **55**, 3051–3073 (2009)
2. E. Arikan, Channel polarization: A method for constructing capacity-achieving codes, in *IEEE International Symposium on Information Theory* (Toronto), pp. 1173–1177 (2008).
3. S. Zhao, P. Shi and B. Wang, Designs of Bhattacharyya Parameter in the Construction of Polar Codes, in *7th International Conference on Wireless Communications, Networking and Mobile Computing* (Wuhan), pp. 1–4 (2011).
4. R. Mori, T. Tanaka, Performance of polar codes with the construction using density evolution, IEEE Commun. Lett. vol. 13, no. 7, pp. 519–521 (2009)
5. A. Arpure, S. Gugulothu, FPGA implementation of polar codes based encoder architecture, in *International Conference on Communication and Signal Processing* (India), April 6-8, pp. 0691–0695 (2016)
6. X. Shih, P. Huang and Y. Chen, High-speed low-area-cost VLSI design of polar codes encoder architecture using radix-k processing engines, in *IEEE 5th Global Conference on Consumer Electronics* (Kyoto), pp. 1–2 (2016)
7. C. Zhang, J. Yang, X. You and S. Xu, Pipelined implementations of polar encoder and feed-back part for SC polar decoder, in *IEEE International Symposium on Circuits and Systems* (Lisbon), pp. 3032–3035 (2015)
8. H. Yoo, I.-C. Park, Partially parallel encoder architecture for long polar codes. IEEE Transactions on Circuits and Systems II: Express Briefs **62**(3), 306–310 (2015)
9. E. Arikan, Systematic polar coding. IEEE Commun Lett **15**(8), 860–862 (2011)
10. H. Vangala, Y. Hong, E. Viterbo, Efficient algorithms for systematic polar encoding. IEEE Commun Lett **20**(1), 17–20 (2016)
11. G. TaiChen, Z. Zhaoyang, Z. Caijun, Z. Liang, A low complexity encoding algorithm for systematic polar codes. IEEE Commun Lett **20**(7), 1277–1280 (2016)
12. G. Sarkis, P. Giard, A. Vardy, C. Thibeault, W.J. Gross, Fast polar decoders: Algorithm and implementation. IEEE J Sel Areas Commun **32**(5), 946–957 (2014)
13. G. Sarkis, I. Tal, P. Giard, A. Vardy, Flexible and low-complexity encoding and decoding of systematic polar codes. IEEE Trans Commun **64**(7), 2732–2745 (2016)
14. X. Wang, T. Ge, J. Li, C. Su, F. Hong, Efficient multi-rate encoder of QC-LDPC codes based on FPGA for WIMAX standard. Chinese Journal of Electronics **26**(2), 250–255 (2017)
15. X. Wang, W. Cao, J. Li, et al., Improved min-sum algorithm based on density evolution for low-density parity check codes. IET communications **11**(10), 1582–1586 (2017)
16. X. Wang, J. Li, Y. Wang, et al., Design of dual-mode decoder based on LDPC/turbo code. IET communications **11**(8), 1325–1329 (2017)
17. H. Vangala, E. Viterbo, Y. Hong, A comparative study of polar code constructions for the AWGN channel. Mathematics **1**, 1–7 (2015)

9 Publisher's Note