e-ISSN: 2564-7954 CUSJE 18(2): 125-132 (2021) Research Article

Çankaya University

Journal of Science and Engineering

https://dergipark.org.tr/cankujse

CUJSE

Fast Calculation of Polar Code Bits and Frozen-Bit Locations

Fatih Genç[1], Orhan Gazi[1*]

[1] Department of Electronics and Communication Engineering, Çankaya University, Ankara, Turkey

Keywords

Polar Codes,
Encoding Algorithm,
Tree Structure,
Generator matrix transformation,
FPGA.

Abstract

In this paper, we show that encoding operation for the polar codes can be achieved without the employment of the generator matrix, and all the polar code bits can be generated at the same time using a number of tree-encoding structures running in parallel. Since encoding matrix is not used in the implementation of the polar encoders in digital electronic devices, hardware space is saved, and low complexity hardware applications are achieved. Besides, we also proposed a method for the calculation of split channel parameters, such as Bhattacharyya bounds or average-bit-error probabilities of the transmitted bits using a tree-based structure. Moreover, the proposed structure enables to calculate the probability of bit-error values of all the transmitted bits at the same time in a parallel manner and decide the locations of data and frozen bits very rapidly.

1. Introduction

Polar codes are a class of channel codes designed in a non- trivial manner [1]. Polar codes are the first mathematically provable channel codes available in the channel coding world, and this can be considered as a major breakthrough in coding society. Polar codes have low performance at moderate and low codeword lengths. To improve the low performance of polar codes, the list and stack decoding algorithms are proposed [2], [3], [4], [5].

Decoding algorithms show better performance than that of the classical successive cancelation method [6], they involve much more computations regarding the classical successive cancelation algorithm. Polar codes can also be decoded using a belief propagation algorithm [7]. In A. A. Andi, O. Gazi work [9], polar codes are concatenated with CRCs to prevent the performance degrading effect of error propagation.

One other challenge on the implementation of these algorithms for future communication systems is the requirement of comprehensive digital electronic devices where hardware programming such as FPGAs stands as a strong candidate. The potential of FPGAs depends on the usage its resources efficiently. The encoding operation of the polar codes can be generated as the following formula,

$$x = uG$$

where u is the information word, x is code-word and G is the generator matrix. For $N = 1024$, and for full rate encoding operation, a binary generator matrix of size 1024×1024 should be stored in the memory units of the FPGA devices. Besides, if different frame lengths are used for the transmissions such as 64, 128, 256, 1024, 2048 etc., a generator matrix dedicated for each should be stored separately with different sizes in the memory units for efficient implementations. In this paper, we propose a method for the encoding of polar codes without the need for employment of generator matrix. The proposed method utilizes an n-bit binary counter, where $n =$

* Corresponding Author: o.gazi@cankaya.edu.tr
Received: August 22, 2021, Accepted: October 06, 2021

log2(N), and a tree structure [9] involving n levels. Since the proposed method avoids the use of generator matrix, a smaller memory space in FPGA is required which reduces the hardware complexity of the polar encoders. Using the proposed method, it is also possible to calculate the code bits in parallel at the same instant. Split channel parameters such as Bhattacharyya bounds or average bit error probabilities are used for the determination indices of data and frozen bits. We proposed a tree-based approach for the calculation of split channel parameters in an efficient manner. Using the proposed approach, it is possible to calculate the all-split channel parameters such as Bhattacharyya parameters of all the bits which can be transmitted at the same time in a parallel manner.

The outline of the manuscript is as follows. In Section II, polar encoding without the use of generator matrix, i.e., proposed encoding approach, is explained. Section III describes the proposed technique used for the calculation of split channel parameters in a fast and efficient manner. Hardware implementation results are given in Section IV, and finally conclusions are drawn in Section V.

2. Polar Encoding

2.1. Generator Matrix

In this section, we review the construction of the generator matrix, and polar encoding operation with generator matrix. Let's denote the N -bit information vector by

$$u_0^{N-1} = (u_0, u_1, \cdots, u_{N-1},) \tag{1}$$

The Polar code-word for the data vector u^{N-1} is obtained using

$$x_0^{N-1} = u_0^{N-1}\, G \tag{2}$$

where the generator matric G_N is calculated via

$$G_N = B_N F^{\otimes n} \tag{3}$$

in which $N = 2^n$, $F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ and B_N is found using

$$B_N = R_N\big(I_2 \otimes B_{N/2}\big) \tag{4}$$

where the initial value of B_N, i.e., B_2 is I_2, and R_N denotes the $N \times N$ reverse shuffle permutation matrix whose operation is explained in

$$(s_0, s_1, s_2, \cdots, s_{N-1})\, R_N = (s_0, s_2, s_4, \cdots, s_{N-2})(s_3 \cdots, s_{N-1}) \tag{5}$$

The Kronecker product of two matrices $A = [\cdot]_{m\times n}$ and $B = [\cdot]_{k\times l}$ is obtained as

$$A \otimes B = \begin{bmatrix} A_{11}B & \cdots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \cdots & A_{mn}B \end{bmatrix} \tag{6}$$

and the Kronecker power is defined as

$$A^{\otimes n} = A \otimes A^{\otimes(n-1)}. \tag{7}$$

2.2. Polar Encoding Without the Employment of Generator Matrix

The encoding operation for $N = 4$ to obtain the code bit x_0 and its tree representation are graphically illustrated in Fig. 1.

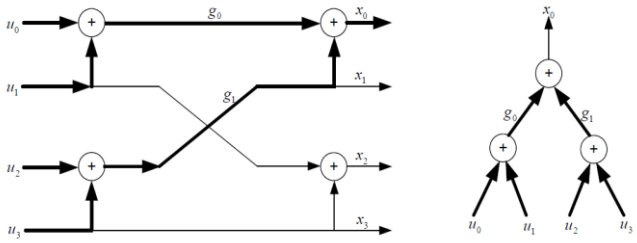

Figure 1. Encoding path and tree structure of the codeword symbol x_0.

Inspecting the encoding operations in Fig. 1 and its trellis representations, we propose a polar encoding operation without the employment of the generator matrix as follows. Pass-nodes in the encoding tree are defined as the nodes which only take the right incoming input bit from the lower layer. Sum-nodes in encoding tree are defined as the nodes which produce the mod-2 sum of the left and right incoming input bits from the lower layer. An n-bit counter is used for the encoding operation. The '1's in the n-bit counter indicate the levels which include pass nodes, and 0's in the n-bit counter indicates the levels which include sum-nodes. The n-bit counter is initialized to all zero tuples, and it is incremented at the calculation of each succeeding code-bit. The least significant bit of the counter indicates the top-level in the tree structure, and the most significant bit of the counter indicates the bottom level of the tree structure. As an example, for $N = 8$ in Fig. 2, the generation of the code bit x_6 is illustrated. The counter has the value of 110 and, the '1's in the counter indicates the levels 1 and 2 where the nodes are pass-nodes. The propagating signals are shown using bold arrows.

For $N = 1024$, we need an $n = log2(1024) \rightarrow n = 10 - bit$ counter, and there are 10 levels in the tree structure, and using the $n = 10 - bit$, and tree structure the encoding operation can be performed in a fast manner without requiring the generator matrix. The encoding algorithm using tree structure and counter is defined below.

2.3. Proposed Encoding Algorithm:

Initialize n-bit, where $n = log2(N)$ counter to all zeros and set $k = 0$.

The code-bit x_k, $k = 0 \ldots N - 1$, is to be generated, and n-bit counter has the binary equivalent of k.

Decide the levels corresponding to the positions of '1's in the counter such that the least significant bit of the counter points to the top level, i.e., level-0. The nodes of those levels corresponding to the positions of '1's of the counter is labeled as the pass-nodes, and the others are labeled as sum-nodes.

Starting from the lowest level above the ground level, OR the bit pairs coming from the predecessor level if the level has label '0', otherwise, just pass the right incoming bit to the upper-level and repeat this process till the top-most level and obtain the code bit x_k.

If $k = N - 1$ terminate, otherwise, increment the k value and go to step-1.

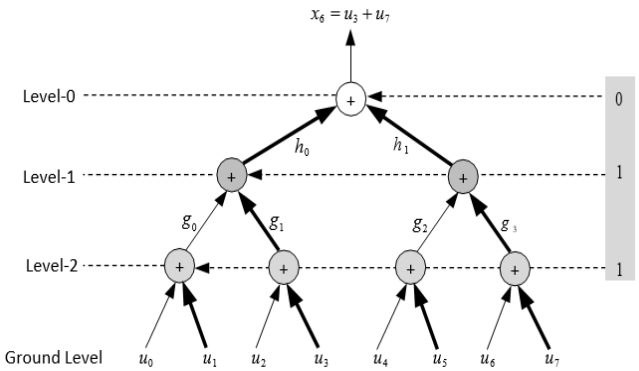

Figure 2. Proposed method encoding structure of x_6.

Example: For $N = 16$, the generation of the code bit x_{13} is illustrated in Fig. 3, where the 4-bit binary counter has the binary value 1101 whose decimal equivalent is 13.

As it is seen from Fig. 3 that for code bits with odd indices, only the right-hand side of the tree can be used, and this further decreases the complexity of hardware implementation.

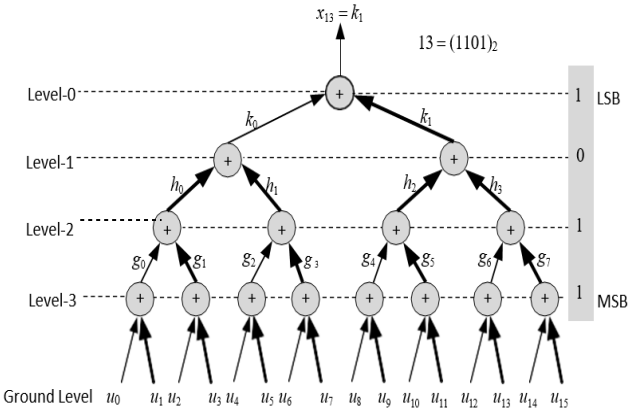

Figure 3. Example for tree-encoding operation.

It is also possible to generate all the code bits at the same time using the proposed tree-encoding structures in parallel. Since, the counter values are known and many of the tree- encoding structures can run in parallel, encoding operation can be completed by all parallel units at the same time. This approach reduces the encoding latency significantly, however, the hardware complexity increases.

3. Calculation of Maximum/Average Bit-Error Probability Using Three Structure

In polar codes, the location of the frozen bits, i.e., parity bits, are decided using the split channel capacities which are calculated before the encoding operation. Large capacity channels are used for the transmission of data bits whereas low-capacity channels are used for frozen bits. Location of frozen and data bits should be written into memory units to be used during the transmission and if the environment changes such as in the wireless communication, these split channel capacities should be re-calculated and written into memory locations again. For binary erasure channels, it is possible to calculate the split channel capacities exactly. However, for other channels, especially for wireless channels, the calculation of channel capacities is not a straightforward process and for most of them, explicit methods are not defined in the literature. The capacity of a split channel is closely related to the average- bit-error probability of the transmission performed through the channel under concern,

and Bhattacharyya parameters corresponding to bounds of maximum probability of bit errors can be used for this purpose. For polar codes, the Bhattacharyya parameters corresponding to maximum probability of bit errors are calculated in a recursive manner as in

$$Z\left(W_{2N}^{2i-1}\right) \leq 2Z\left(W_N^i\right) - Z^2\left(W_N^i\right) \quad i = 0, \dots, N-1$$
$$Z\left(W_{2N}^{2i}\right) = Z^2\left(W_N^i\right). \tag{8}$$

The locations corresponding to large and small Bhattacharyya values are chosen for frozen and information bits, respectively. It is seen from (8) that Bhattacharyya parameters are calculated in a serial manner. Considering the fact that reprogrammable hardware devices such as FPGAs will be idly used in the future communication technologies, split channel capacities or Bhattacharyya parameters can be calculated by such digital devices.

In this part, we propose a method to calculate Bhattacharyya parameters or maximum/average bit-error probabilities of the transmitted bits using tree structure in a parallel manner. With the proposed method, Bhattacharyya parameters or maximum/average bit-error probabilities can be calculated in an efficiently and a decision can be made whether the bit corresponds to a frozen or data bit.

Using the proposed method, the calculation of the maximum or average bit error probabilities for all split channels can also be performed at the same time in a parallel manner, and defining a threshold pte for the transmission error probabilities, those bits having maximum/average transmission error probabilities below a threshold value can be transmitted which can provide an uninterrupted transmission. For instance, if we take the threshold value $pte = 0.4$ then for a new channel, the information bit ui is transmitted if its calculated maximum/average bit error probability, related to the split channel capacity, is smaller than pte, i.e., if $pe < pte$.

3.1. Proposed Method

In this subsection we explain the proposed method with an algorithm for the determination of bit error probabilities of split channels:

Initialize the counter to all zeros and set $k = 0$.

The maximum or average bit-error probability is to be calculated for the bit uk, $k = 0 \dots N - 1$ and n-bit counter has the binary equivalent of k.

Decide the levels corresponding to the positions of ’0’s and ’1’s in the counter and assign the counter bits to the tree levels such that the least significant bit of the counter points to the top level, i.e., level-0.

Starting from the level above the ground level, combine the α terms where α terms can be Bhattacharyya parameters or they can be average bit-error probabilities, and use $f(x) = 2x - x^2$ if the level label is zero or use $g(x) = x^2$ is the level label is 1, and repeat this till the top most level and obtain the maximum/average probability of error bit uk.

If $k = N - 1$ terminate, otherwise, increment the k value and go to step-3.

The graphical illustration of the proposed approach for the calculation of maximum/average bit error probability for u_{13} is depicted in Fig. 4 where g(·) function is employed for the nodes belonging to the levels whose bit label is ’1’, whereas f(·) function is employed for the nodes belonging to the levels whose bit label is ’0’. In Fig. 4, except for the nodes in level-1, g(·) function is utilized for all the other nodes. A numerical example for the calculation of Bhattacharyya value for u_{13} for binary erasure channels with erasure probability $\alpha = 0.5$ is depicted in Fig. 5 where it is seen that the output of the top node is 0.015 defined as the maximum probability of bit error for the bit u_{13}. Here, due to such a small maximum bit error probability, the bit u_{13} can be chosen as information bit, i.e., bit location 13 can be reserved for information bits.

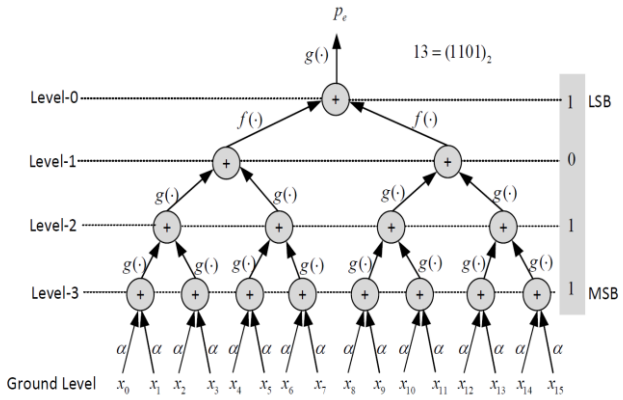

Figure 4. Maximum/average bit error probability calculation for u_{13}.

The tree structure shown in Figure 5 can be constructed for all the data bits, since the counter values can be known enabling parallel calculation of all the split channel parameters.

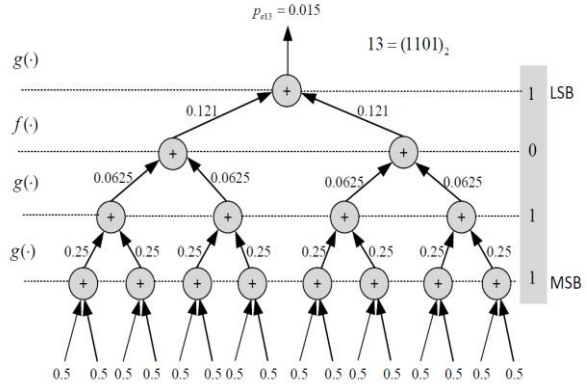

Figure 5. Calculation of Bhattacharyya value for u_{13} for BEC with α= 0.5.

4. Implementation Results

In this section, we provide information about the hardware space consumption of the proposed techniques and make a comparison with the classical approaches. The polar encoding using the generator matrix and proposed algorithm are compared in Table-1 in terms of the digital resources used. For the classical polar encoding, we used the formula $x = uG$, and the rows of G matrix corresponding to the positions of '1's in u vector are XORed. Both algorithms are implemented by using an Nexys-3 spartan-6 FPGA board. It is seen from the table that as the frame length increases the hardware requirement of the proposed method favors significantly over the classical one where for frame length N = 1024, the proposed approach uses three times less hardware resources.

Table 1. Hardware Consumption (Number of Slice registers- NOSR)

Frame Length	Generic Method- NOSR	Proposed Method - NOSR
N=8	11	10
N=16	23	19
N=32	46	36
N=64	112	69
N=128	261	135
N=256	586	266
N=512	1312	524
N=1024	3136	1038

The consumed hardware space gain of the proposed approach is depicted in Fig. 6 where it is seen that for frame length $N = 1024$ the proposed method gains 70% hardware space, i.e., if the classical approach consumes 100 hardware resources, the proposed method consumes only 30 hardware resources.

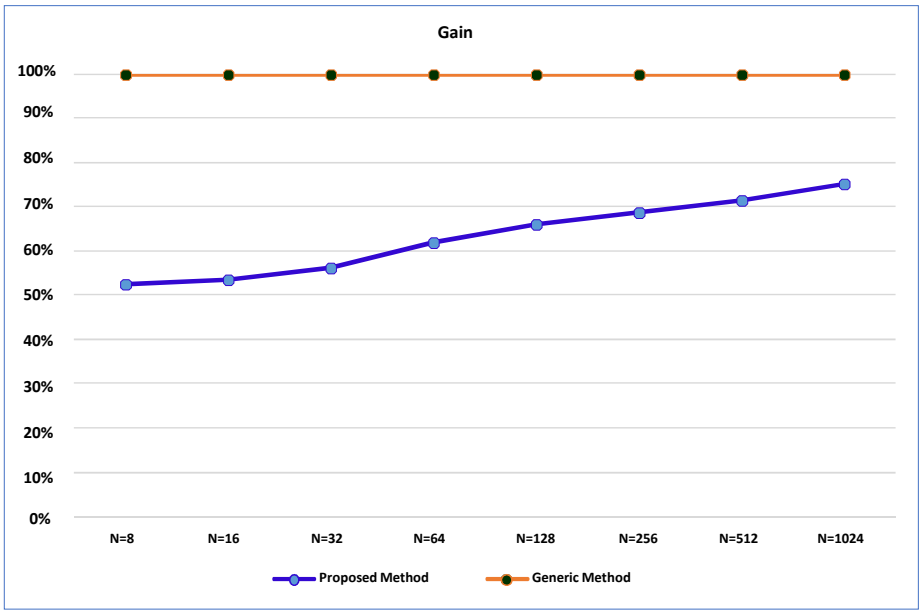

Figure 6. Comparison of hardware space consumption gain for the proposed and classical encoding approaches.

5. Conclusions

In this manuscript, we proposed tree-based structures for the encoding of polar codes without the use of a generator matrix, and for the calculation of split channel parameters which are used for the classification of bits to be transmitted as data or frozen bits. The suggested structures are suitable for sequential and parallel processing operations. It is also shown that using the proposed methods, it is possible to calculate the polar code bits, and split channel parameters at the same instant in a parallel manner. When the suggested structures are implemented in hardware using digital electronic devices, they consume less hardware space and work faster.

Declaration of Competing Interest

The authors declare that there is no competing financial interests or personal relationships that influence the work in this paper.

Authorship Contribution Statement

Fatih GENÇ: Data Preparation, Simulations, Reviewing, Methodology

Orhan GAZI: Writing, Reviewing, Methodology, Supervision

References

[1] O. Gazi, Polar Codes: A Non-Trivial Approach to Channel Coding, Springer Verlag, 2019.

[2] I. Tal and A. Vardy, " List decoding of polar codes," *IEEE International Symposium on Information Theory - Proceedings*, pp. 15, 2011.

[3] K. Niu and K. Chen, “Stack decoding of polar codes,” *Electronics Letters*, vol. 48, no. 12, pp. 695697, 2012.

[4] K. Chen, K. Niu, and J. Lin, “Improved successive cancellation decoding of polar codes,” *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 31003107, 2013.

[5] B. Li, H. Shen, and D. Tse, “ An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check,” *IEEE Communications Letters*, vol. 16, no. 12, 2012.

[6] E. Arikan, “ Channel polarization: A method for constructing capacity achieving codes for symmetric binary-input memoryless channels,” *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 30513073, 2009.

[7] A. Çağrı Arlı, O. Gazi, “ Noise-aided Belief Propagation List Decoding of Polar Codes,” *IEEE Communications Letters*, pp: 1285-288, 2019.

[8] K. Niu and K. Chen, “ CRC-aided decoding of polar codes,” *IEEE Communications Letters*, vol. 16, no. 10, pp. 16681671, 2012.

[9] A. A. Andi, O. Gazi, “ Fast Decoding of Polar Codes Using Tree Structure,” *IET Communications*, vol. 13, no. 14, pp. 2063-2068, 2019.