UDC 621.391.(075.8)

A HARD-DECISION ITERATIVE DECODING METHOD FOR 2D SEC-DED CODES

X.H. REN, Y.M. CHEN, V.K. KANAPELKA

Belarusian State University of Informatics and Radioelectronics, Republic of Belarus

Received March 13, 2023

Abstract. Product codes are well known to have very promising correction potential and the ability to deal with burst errors and can correct random errors in which the number of errors is above half the minimum distance. Two-dimensional single-error correcting and double-error detecting (2D SEC-DED) codes are types of product codes. Decoding methods for product codes can roughly be divided into hard-decision decoding and soft-decision decoding. The hard-decision decoding method, also called the iterative decoding method, which is easy to implement and has low computational complexity. However, the error correcting capability of the existing hard-decision iterative decoding methods for 2D SEC-DED is much less than half the minimum distance of the code. In this paper, we propose an improved hard-decision iterative decoding method for 2D SEC-DED codes to overcome this defect.

Keywords: hard-decision, 2D SEC-DED codes, iterative decoding.

Introduction

A 2D SEC-DED code based on two extended Hamming codes $C_1(n_1,k_1,4)$ and $C_2(n_2,k_2,4)$ can be constructed by performing the following two steps. Suppose that *k* bits of the message can be resized to a rectangular array with k_1 rows and k_2 columns. We first encode the message in the row direction by using the encoding rule of extended Hamming code C_2 and obtain a k_1 row and n_2 column code C_{mid}. Then, we encode C_{mid} in the column direction by using the encoding rule of the extended Hamming code C_1 and obtain the final two-dimensional code C_{2d} with parameters $(n_1n_2,k_1k_2,4)$. We know that the maximum correction capability of a 2D SEC-DED code based on extended Hamming codes is seven. Therefore, a proper iterative hard-decision decoding method for 2D SEC-DED codes should satisfy the following two requirements. First, it should be able to correct all error patterns in which the number of error bits is less than or equal to seven. Second, the iterative hard-decision decoding method should correct as many error patterns as possible in which the number of error bits is greater than seven [1–5].

Decoding methods for 2D SEC-DED codes

The former decoding methods usually adopt an iterative approach, as first introduced by Elias [1]. There are two major types of decoding methods for product codes: hard-decision decoding and soft-decision decoding. Iterative decoding methods are easy to implement and have low computational complexity [2–5]. In the past decade, many efforts have been made and many improved methods have been proposed to improve the performance of hard-decision iterative decoding [6–12]. In comparison, decoding methods based on the soft-decision approach perform better in terms of correctness capability than hard-decision decoding, but they also have high complexity and require extra information to indicate the reliability of each piece of input data. The maximum correction capability of iterative decoding methods is up to half the minimum distance of the code. However, the limitation of iterative decoding is that it is unable to correct certain special error patterns, named stall patterns, for which the weight is within half the minimum distance [11].

Two-dimensional single-error correcting and double-error detecting (2D SEC-DED) codes are types of product codes, which are widely adopted in many applications, and the corresponding encoding and decoding procedures are relatively simple. One SEC-DED code is the extended Hamming code, which can be obtained by adding one extra parity bit to the original Hamming code. Many distinct methods have been proposed in the past. However, the error correction capabilities of most of them are insufficient to correct error patterns with the number of errors up to half the minimum distance of the code. In our previous work [13], we designed an iterative decoding method for standard Hamming product codes that can correct all stall patterns with four errors and thus can correct all errors up to half the minimum distance. However, this method is not suitable for 2D SEC-DED codes because the component codes are different, and the minimum distance correspondingly increases from the original value of four to seven. To overcome this challenge, in this paper, we follow the idea of our previous work and design an improved hard-decision iterative decoding method for 2D SEC-DED codes based on extended Hamming codes; this approach can be used to correct errors up to half the minimum distance of the code.

The simplest iterative hard-decision decoding method is the two-step row-column method [3]. In the first step, the syndromes of all columns of the received code are computed in accordance with the decoding method corresponding to the encoding method, based on which the decoder locates all possible positions of single errors (correctable errors) and rectifies them in place. The decoder will not attempt to fix any double errors (uncorrectable errors). Then, the decoding result of the first step is passed to the second step. In the second step, a similar decoding operation is performed again but in the other direction. The two-step decoding method is very efficient and can correct many error patterns with the number of errors above half the minimum distance of the code. However, it fails to correct some stall patterns, such as 2-by-2 error patterns, since the decoder takes no action for double errors.

The proposed decoding method

The proposed method consists of two procedures: a preprocessing procedure and a decoding procedure.

1. Preprocessing procedure.

In the preprocessing procedure, in addition to the registers for the received code, four additional registers are required to record the error status: the row existing-error register (REER), the row double-error register (RDER), the column existing-error register (CEER) and the column double-error register (CDER). The *i*-th bit of the REER/CEER will be set to one when errors are detected in the *i*-th row/column based on the syndrome. Otherwise, this bit should be set to 0. Similarly, the *i*-th bit of the RDER/CDER will be set to one only when a double error is detected in the *i*-th row/column according to the syndrome. Otherwise, this bit should be set to 0.

The preprocessing procedure has two functions: determining the initial decoding direction and applying a pre-erasure process to reduce the number of errors when necessary. The initial direction of decoding is determined by comparing the estimated numbers of errors from rows ($\mathrm{RN}_{\mathrm{error}}$) with the estimated numbers of errors from columns ($\mathrm{CN}_{\mathrm{error}}$), which can be computed according to formulas (1) and (2), respectively. If $\mathrm{RN}_{\mathrm{error}}$ is greater than $\mathrm{CN}_{\mathrm{error}}$, then the initial decoding direction remains the row direction (the default direction); in contrast, if $\mathrm{RN}_{\mathrm{error}}$ is less than $\mathrm{CN}_{\mathrm{error}}$, then the initial decoding direction is changed to the column direction by transposing the received code (a flag will be set to indicate whether transposition is conducted; if the flag is true, then at the end of the decoding procedure, another transposition is conducted after the whole decoding process is complete). The reason for this is that we contend that decoding from the side from which more errors are estimated initially will introduce fewer errors during the decoding procedure.

$$\mathrm{RN}_{\mathrm{error}} = \sum_{i=1}^{n_2} \mathrm{REER}_i + \sum_{i=1}^{n_2} \mathrm{RDER}_i \tag{1}$$

$$CN_{error} = \sum_{i=1}^{n_1} CEER_i + \sum_{i=1}^{n_1} CDER_i \quad (2)$$

The pre-erasure process is implemented when the following three conditions are satisfied: RN_{error} is equal to CN_{error}; the numbers of instances of 1 in both the REER and CEER are equal; and the product of the numbers of instances of 1 in the REER and CEER is less than the sum of RN_{error} and CN_{error}. The positions for erasure are determined by the values of 1 in the REER and in CEER. The idea behind this is intuitive: performing the erasure process on a small region in which the number of error bits is greater than the number of correct bits can reduce the number of error bits. The flow chart of the preprocessing procedure is presented in Figure 1.

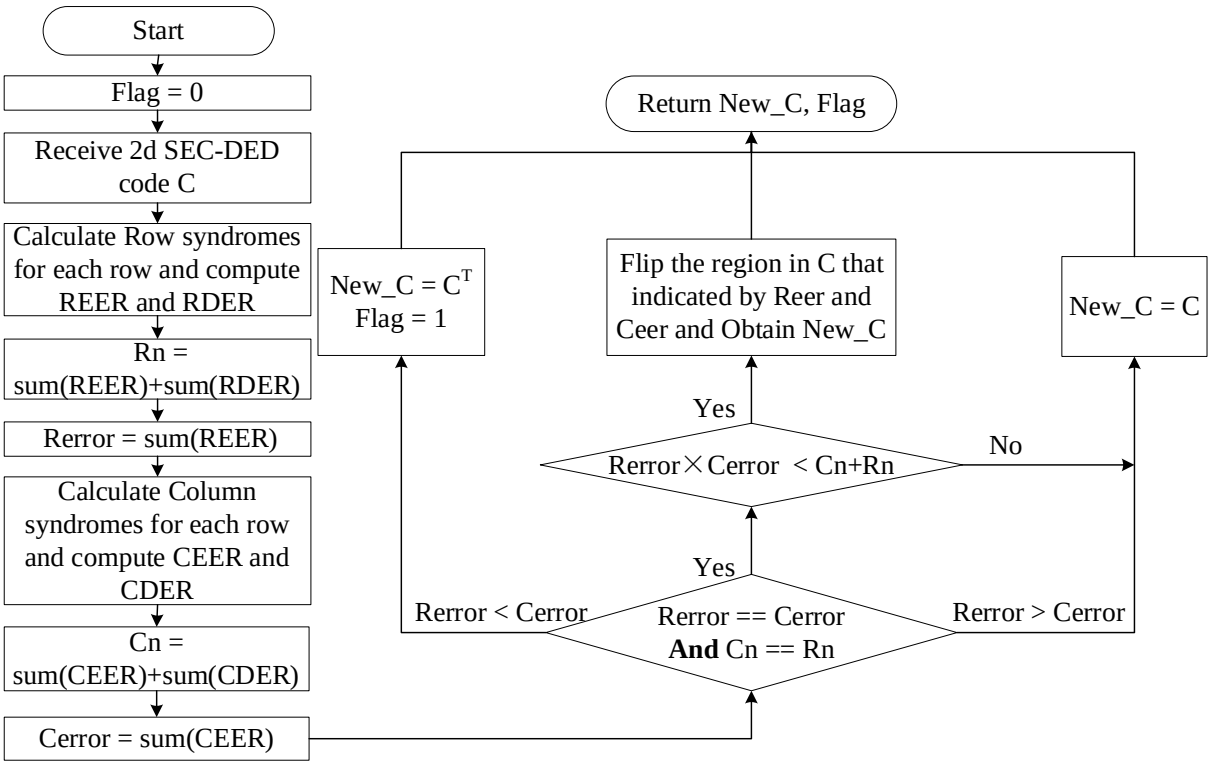

Figure 1. Flow chart of the preprocessing procedure

2. Decoding procedure.

The four registers introduced in the previous procedure are also used in this procedure. However, since the proposed decoding procedure is a modified version of Bao's three-step iterative decoding method [5], the usage of the CDER and REER is changed to that of the row status vector and column status vector used in Bao's method.

The proposed iterative decoding procedure is a three-step decoding method.

In the first step, the row syndromes for each row are calculated, and on this basis, the REER and RCDR are updated; then, row decoding is conducted. All the correctable single errors are flipped in accordance with the syndromes.

In the second step, the column syndromes for each column are calculated, and the CEER and CDER are updated. If the number of 1 value in the RDER is equal to 3 and the number of 1 value in the CEER is equal to 2, then erasure is conducted at the coordinates indicated by the RDER and CEER. Otherwise, column decoding is conducted based on the column syndromes, followed by row decoding, in which the syndromes for each row are recalculated. Then, single errors are corrected based on these updated row syndromes, and double errors are flipped based on the CDER.

In the last step, the column decoding process is repeated to correct the remaining single errors. Then, the corrected code may be transposed in accordance with the flag generated in the previous procedure. The flow chart of the decoding procedure is presented in Figure 2.

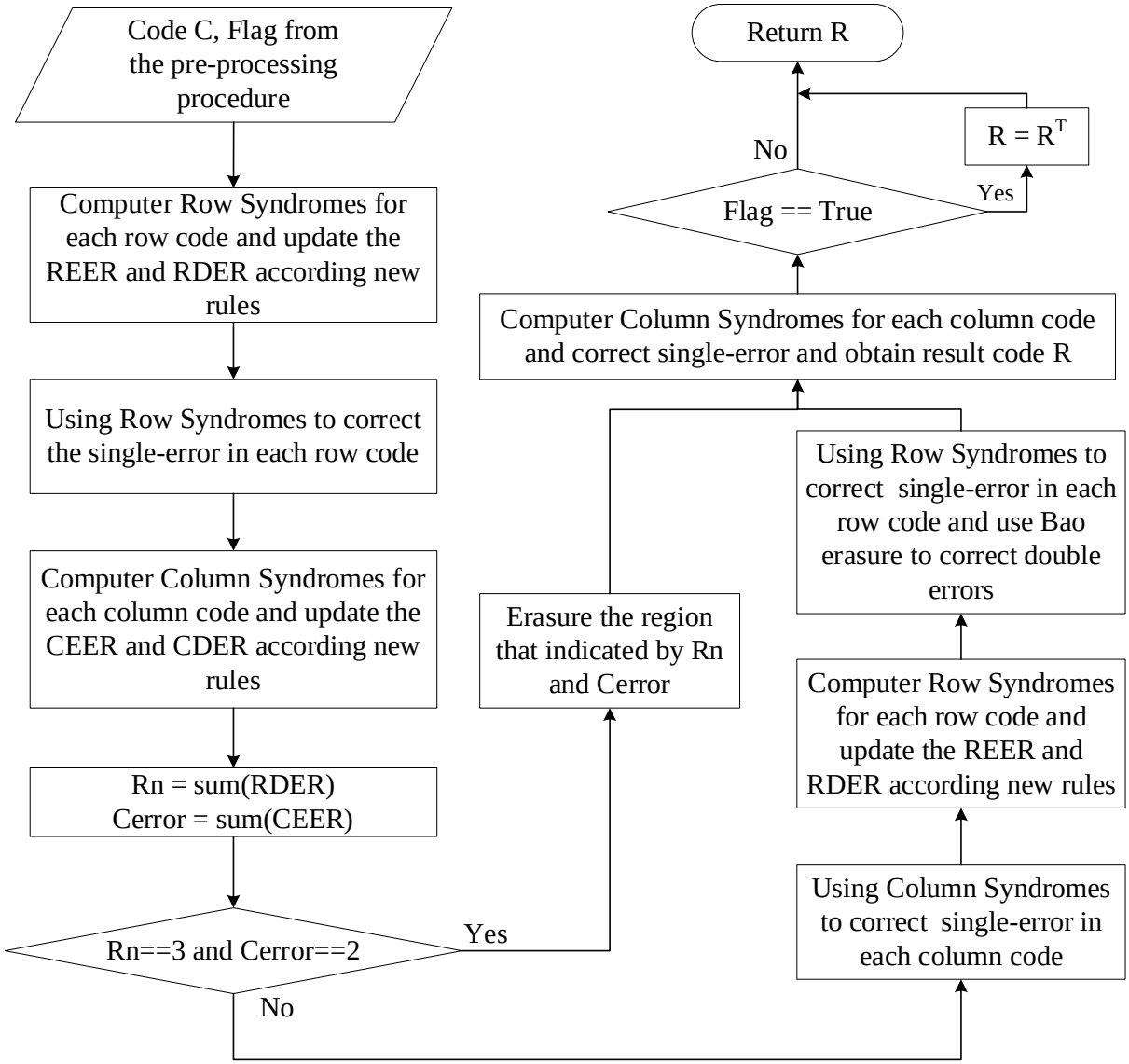

Figure 2. Flow chart of the decoding procedure

Experiment based on error patterns

To explore the potential of all implemented decoding methods, experiments based on error patterns were conducted. The number of error bits was manually set, but the error positions among the error patterns were randomly generated. It is noted that the size of the error patterns was kept the same as the size of the original (64, 16, 16) code obtained by encoding a random 16-bit binary message. The number of error bits in the error patterns was gradually increased from one error to twelve errors. For the error patterns with no more than 5 errors, we generated all possible error patterns and then added them to the codeword separately and attempted to correct these errors using each implemented decoding method. For the error patterns with more than 5 errors, since generating all the error patterns would be very difficult and time consuming, we randomly selected one million samples from all error patterns with a given number of errors for the decoding experiments. Table 1 shows the numbers of error patterns used in the current experiments for different given numbers of error bits.

Table 1. **Numbers of error patterns and error bits**

Number of Error Bits	Number of Error Patterns
1	64
2	2016
3	41664
4	635376
5	7624512
6	1000000
7~9	1000000

Table 2 and Table 3 summarize the numbers of word errors and bit errors made with the different decoding methods under various numbers of error bits. Table 4 and Table 5 were obtained by normalizing the data shown in Table 2 and Table 3, respectively. Figure 3 and Figure 4 are visualizations of Table 4 and Table 5, respectively.

Table 2. **Numbers of word decoding errors under a given number of error bits**

Given Number of Error Bits	Decoding Method			
	Two-Step Method [5]	Kreshchuk's Method [10]	Bao's Method [8]	Proposed Method
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	10192	0	0	0
5	58208	18816	0	0
6	191112	24974	383	0
7	369766	63753	5569	0
8	578553	138496	32585	3229
9	770939	254869	116425	25084

Table 3. **Numbers of bit decoding errors under a given number of error bits**

Given Number of Error Bits	Decoding Method			
	Two-Step Method [5]	Kreshchuk's Method [10]	Bao's Method [8]	Proposed Method
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	21952	0	0	0
5	1229312	75264	0	0
6	445510	159180	1655	0
7	949792	745824	23714	0
8	1714958	2171274	140275	17828
9	2765042	5035850	524295	141308

Table 4. **Normalization of the word errors produced by the decoding methods under a given number of errors bits**

Given Number of Error Bits	Decoding Method			
	Two-Step Method [5]	Kreshchuk's Method [10]	Bao's Method [8]	Proposed Method
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0,01604	0	0	0
5	0,00763	0,00246	0	0
6	0,19111	0,02497	0,00038	0
7	0,36976	0,06375	0,00556	0
8	0,57855	0,13849	0,03258	0,00322
9	0,77093	0,25486	0,11642	0,02508

Table 5. **Normalization of the bit errors produced by a decoding method under a given number of errors bits**

Given Number of Error Bits	Decoding Method			
	Two-Step Method [5]	Kreshchuk's Method [10]	Bao's Method [8]	Proposed Method
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0,00053	0	0	0
5	0,00251	0,00015	0	0
6	0,00696	0,00248	0,00002	0
7	0,01484	0,01165	0,00037	0
8	0,02679	0,03392	0,00219	0,00027
9	0,04320	0,07868	0,00819	0,00221

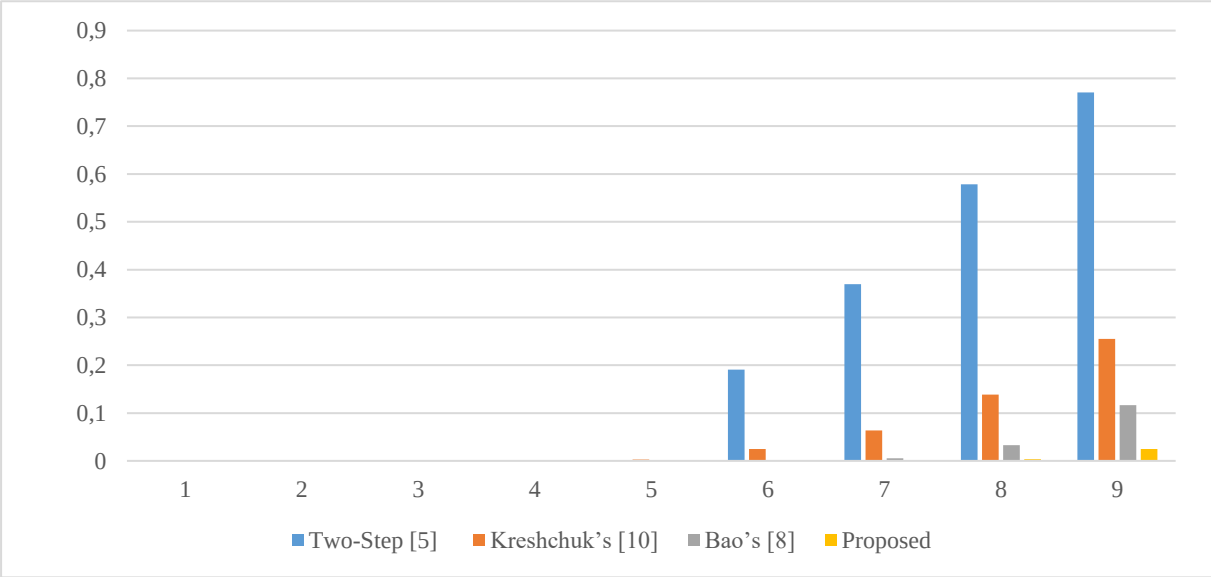

Figure 3. Histogram of the normalized word errors produced by the different decoding methods

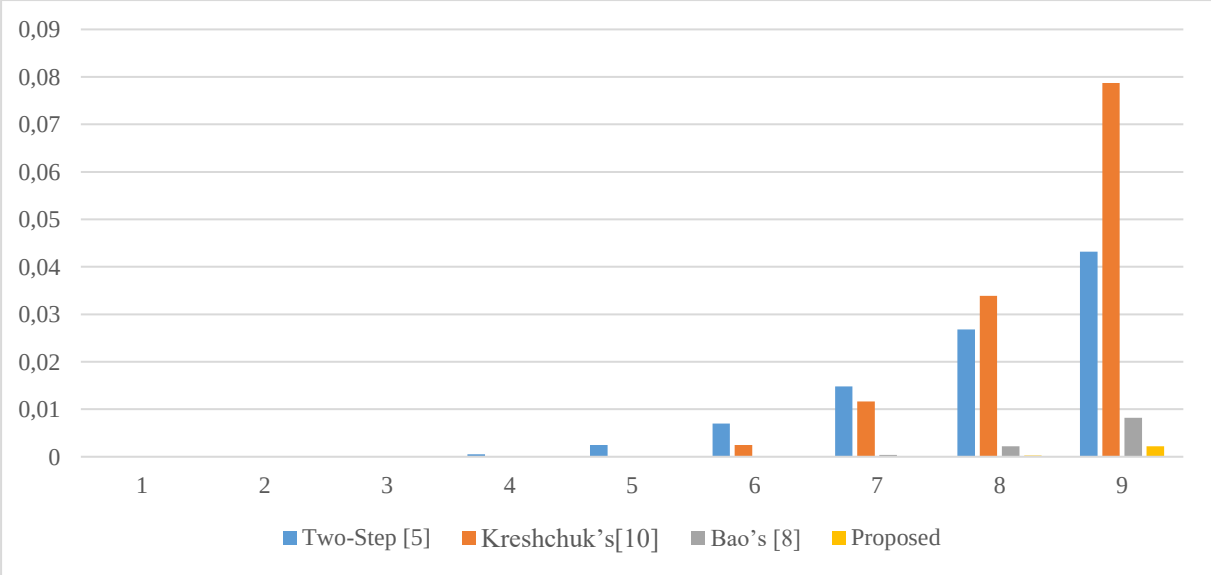

Figure 4. Histogram of the normalized bit errors produced by the different decoding methods

From the above tables, the proposed method displays the best performance in terms of the error correction capability within the range of half the minimum distance of the code, and it can properly correct more error patterns than the other three decoding methods. Exhaustive experiments have proven

that the proposed method can correct all error patterns with the number of errors below 5. For error patterns with 6 or 7 errors, the proposed method properly decodes all sampled error patterns and corrects all the error bits; therefore, there is a high probability that the proposed method also can correct all error patterns in which the number of errors is not above seven, i.e., half the minimum distance of the current code. In contrast, the two-step method, Kreshchuk's method, and Bao's method begin to produce decoding errors when the given number of errors is four, five and six, respectively.

Moreover, when the given number of errors is greater than half the minimum distance of the code, the proposed method still provides more powerful rectification ability than the other methods. For example, when the given number of errors is 9, the proposed method produces errors for only approximately 2,5 % of words and 0,02 % of bits. In contrast, the two-step method generates a 77 % word error and a 4,3 % bit error, Kreshchuk's method generates a 25 % word error and a 7,8 % bit error, and Bao's method generates an 11 % word error and a 0,8 % bit error.

Conclusion

In this paper, we have proposed an improved hard-decision iterative decoding method for 2D SEC-DED codes. The error correction capability of the proposed method is very close to half the minimum distance of the code. The decoding experiment based on a given number of errors indicated that the proposed method achieves better performance than the other decoding methods in the sense that it can correct more error patterns.

References

1. Elias P. // IEEE Trans. on Information Theory. 1954. Vol. 4. P. 29–37.
2. Forney G. // IEEE Trans. on Information Theory. 1966. Vol. 12. P. 125–131.
3. Abramson N. // IEEE Trans. on Communication Technology. 1968. Vol. 16. P. 398–402.
4. Reddy S., Robinson J. // IEEE Trans. on Information Theory. 1972. Vol. 18. P. 182–185.
5. Lin. S., Costello D.J. // Error Control Coding. Lebanon, 2001.
6. Bo F., Ampadu P. // IEEE SOC Conference. 2008. P. 59–62.
7. Bo F., Ampadu P. // VLSI Design. 2008. P. 1–14.
8. Bo F., Ampadu P. // IEEE Trans. on Circuits and Systems. 2009. Vol. 56. P. 2042–2054.
9. Kim J., Jee Y. // Proc. of International Conf. on Computer Technology and Development. 2010. P. 611–615.
10. Alexey K., Victor Z., Eygene R. // Proc.of International Workshop on Algebraic and Combinational Coding Theory. 2014. P. 211–214.
11. Blomqvist F. // Applicable Algebra in Engineering, Communication and Computing. 2021. P. 1–18
12. Chlaab A.K., Flayyih W.N., Rokhani F.Z. // Bulletin of Electrical Engineering and Informatics. 2020. Vol. 9. P.1979–1989.
13. Ren X.H., Ma J., Tsviatkou V.Yu., Kanapelka V.K. // Engineering Letters. 2022. Vol. 30. P. 948–954.