Improved Hard-Decision Iterative Decoding Method for 2D SEC-DED Codes

Xunhuan Ren, Jun Ma, Viktar Yurevich Tsviatkou, Valery Kanstantinavich Kanapelka

Abstract—**Two-dimensional single-error correcting and double-error detecting (2D SEC-DED) codes are types of product codes. Product codes are well known to have very promising correction potential and the ability to deal with burst errors, i.e., multiple errors that consecutively appear, and they can also correct some error patterns in which the number of errors is above half the minimum distance. Product codes can be decoded with a hard-decision iterative 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. Experiments prove that this decoding method outperforms other decoding methods in terms of its correction capability; it can correct nearly all errors up to half the minimum distance of the 2D SEC-DED code and can correct more error patterns in which the number of errors is beyond half the minimum distance of the code.**

Index Terms— **2D SEC-DED codes, iterative decoding, hard-decision, stall pattern, correction capability of code**

I. Introduction

Two-dimensional code, also named product code in some literature, was initially developed by Elias [1] in 1954 by using simple error-correcting and detecting codes. Product code is a powerful type of code that has many valuable properties [2]. The most important of these properties is that it can correct burst errors; notably, the product code construction procedure includes interleaving as an inherent feature. Another important property is that the coverage radius of a two-dimensional code is larger than half the minimum distance of the code, which means that when a proper decoding method is adopted, it has the potential to correct error patterns in which the number of errors is greater than half the minimum distance of the code [3,4]. As a result, two-dimensional codes have been applied in many fields to mitigate the effects of errors, such as correcting the errors in wireless sensor networks [5-7], correcting on-chip interconnection errors to improve the reliability of on-chip communication [8] and maintaining steady satellite communication [9].

There are two major types of decoding methods for two-dimensional codes: hard-decision decoding and soft-decision decoding. The former decoding methods usually adopt an iterative approach, as first introduced by Elias [1] and then comprehensively presented in [10]. Iterative decoding methods are easy to implement and have low computational complexity. 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]. In the past decade, many efforts have been made and many improved methods have been proposed [11-17] to improve the performance of hard-decision iterative decoding. 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. Methods following this concept were proposed in [11,18,19].

Two-dimensional single-error correcting and double-error detecting (2D SEC-DED) codes are widely adopted in many applications since they are constructed from SEC-DED codes, and the corresponding encoding and decoding procedures are relatively simple [5]. One SEC-DED code is the extended Hamming code [20], which can be obtained by adding one extra parity bit to the original Hamming code. As a result, 2D SEC-DED codes based on extended Hamming codes are attractive options, and the decoding of these codes with hard-decision iterative methods has become popular. 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 a number of errors up to half the minimum distance of the code.

In our previous work [21], we designed an iterative decoding method for standard Hamming product codes that can correct all stall patterns with four errors and thus has the ability to 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

Manuscript received Sep. 03, 2022; revised Jan. 13, 2023.

Xunhuan Ren is a postgraduate student at Belarusian State University of Informatics and Electronics, Minsk, Belarus. e-mail: rxh1549417024@gmail.com).

Jun Ma is a postgraduate student at Belarusian State University of Informatics and Electronics, Minsk, Belarus. (e-mail: majun1313@hotmail.com).

Viktar Yurevich Tsviatkou is a Professor in the Department of Info-communication Technology, Belarusian State University of Informatics and Electronics, Minsk, Belarus. (e-mail: vtsvet@bsuir.by).

Valery Kanstantinavich Kanapelka is a Professor in the Department of Info-communication Technology, Belarusian State University of Informatics and Electronics, Minsk, Belarus. (e-mail: volos@bsuir.by).

the code.

The remainder of this paper is organized as follows. Some basic terms and some existing related works are reviewed in section 2. Next, in section 3, our improved iterative hard-decision method is presented. Then, in section 4, we discuss the results of the conducted decoding experiments. Finally, the conclusions and future work are presented in section 5.

II. Basic Terms And Related Works

A. Basic Terms

A linear block code *C* is denoted by (n,k,d), where *n*, *k*, and *d* are the number of bits of the codeword, the number of bits of the message and the minimum Hamming distance, respectively. The minimum Hamming distance of a code can be obtained by using the following formula:

$$d = min\left\{d_H\left(C_i,C_j\right)i,j = 1,\ldots,k),i \neq j\right\} \quad (1)$$

$$d_H\left(C_i,C_j\right) = sum\left(mod\left(C_{im}+C_{jm},2\right)\right) \quad (2)$$

For a given linear block code *C*, the minimum Hamming distance directly influences the code's correction and detection capabilities. The relationship between these capabilities is described in formulas (3) and (4):

$$t_d = d-1 \quad (3)$$

$$t_r = floor\left(\left(d-1\right)/2\right) \quad (4)$$

where t_d is the maximum number of error bits that can be detected and t_r is the maximum number of error bits that can be corrected.

For an extended Hamming code with a minimum Hamming distance of 4, it is easy to find that its t_d and t_r are 3 and 1, respectively. Accordingly, a decoder can reliably detect 3-bit errors when it does not try to correct any errors, but when the decoder does attempt to correct errors, some triple errors will be mistakenly regarded as single errors, and the decoder will consequently introduce new errors. The decoder can also detect double errors and will consider them uncorrectable. This is why the extended Hamming code is considered an SEC-DED code.

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 n2 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 $(n_1n_2,k_1k_2,16)$. This construction procedure is illustrated in Fig. 1.

According to formulas (3) and (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.

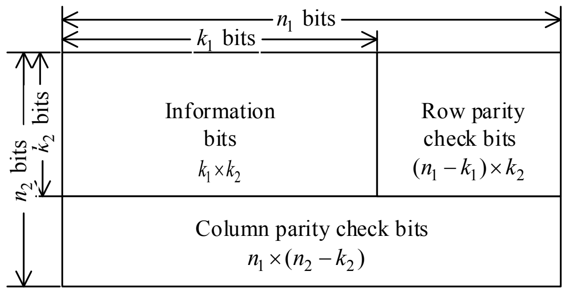
n_1 bits
k_1 bits
k_2 bits
n_2 bits
Information bits $k_1 \times k_2$
Row parity check bits $(n_1-k_1)\times k_2$
Column parity check bits $n_1 \times (n_2-k_2)$

Fig. 1. Construction of a two-dimensional code

B. Related Works

The simplest iterative hard-decision decoding method is the two-step row–column method [22]. 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 a 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.

To overcome the drawbacks mentioned above, Kreshchuk proposed a new iterative method based on the two-step decoding method [12], in which erasures are added. To determine the regions that require an erasure operation, the decoder records the row and column of each mistake into two registers during the row decoding procedure and the column decoding procedure, respectively, when the errors can be detected by the decoder. This decoding method can also be regarded as a postprocessing technique [11]. However, this decoding method can only properly correct all error patterns with a number of errors below 4, which is far from the minimum distance of the code.

Another interesting method to overcome the problem of stall patterns, as proposed by Bao [15], is a three-step (row–column–row) decoding method. In his method, similar to Kreshchuk's method, the erasure operation is adopted, and the position information for erasure steps is saved in two status vectors. The difference between Bao's method and Kreshchuk's method is that the region selection rules for erasures are different, as briefly described below.

During the first stage, if the decoder detects errors in the *i*-th row, then the *i*-th position in the row status vector is set to 1; otherwise, it is set to 0.

During the second stage, the *j*-th position in the column status vector is set to 1 if a corresponding error is detectable but not correctable (meaning that it is a double error) or if this error is correctable but the corresponding element in the row vector where the error occurs has a status value of 0; otherwise, it will be set to 0.

Bao's method can correct all error patterns with up to five errors, which is better than the capability of Kreshchuk's method, but there is still room for improvement.

Therefore, we design an improved hard-decision iterative decoding method that can correct nearly all error patterns with up to seven errors.

III. Proposed Method

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

A. 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 on the basis of 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. In Fig. 2, a real example of the calculation of these registers based on a (64, 16, 16) 2D SEC-DED code is provided.

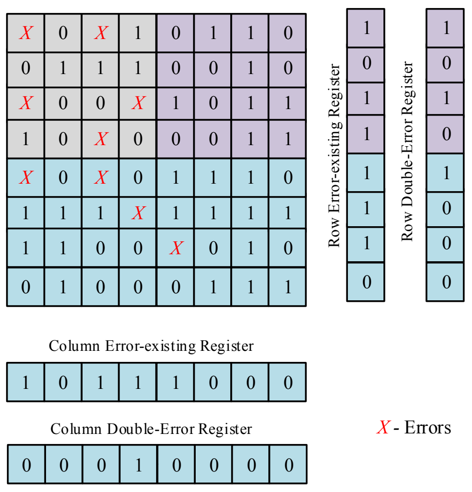

Fig. 2. Example of generating four registers from code.

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 (RN_{error}) with the estimated numbers of errors from columns (CN_{error}), which can be computed according to formulas (5) and (6), respectively. If RN_{error} is greater than CN_{error}, then the initial decoding direction remains the row direction (the default direction); in contrast, if RN_{error} is less than CN_{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.

$$RN_{\text{error}} = \sum_{i=1}^{n_2} REER_i + \sum_{i=1}^{n_2} RDER_i \tag{5}$$

$$CN_{\text{error}} = \sum_{i=1}^{n_1} CEER_i + \sum_{i=1}^{n_1} CDER_i \tag{6}$$

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 Fig. 3.

B. 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 [15], 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 values in the RDER is equal to 3 and the number of 1 values 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 Fig. 4.

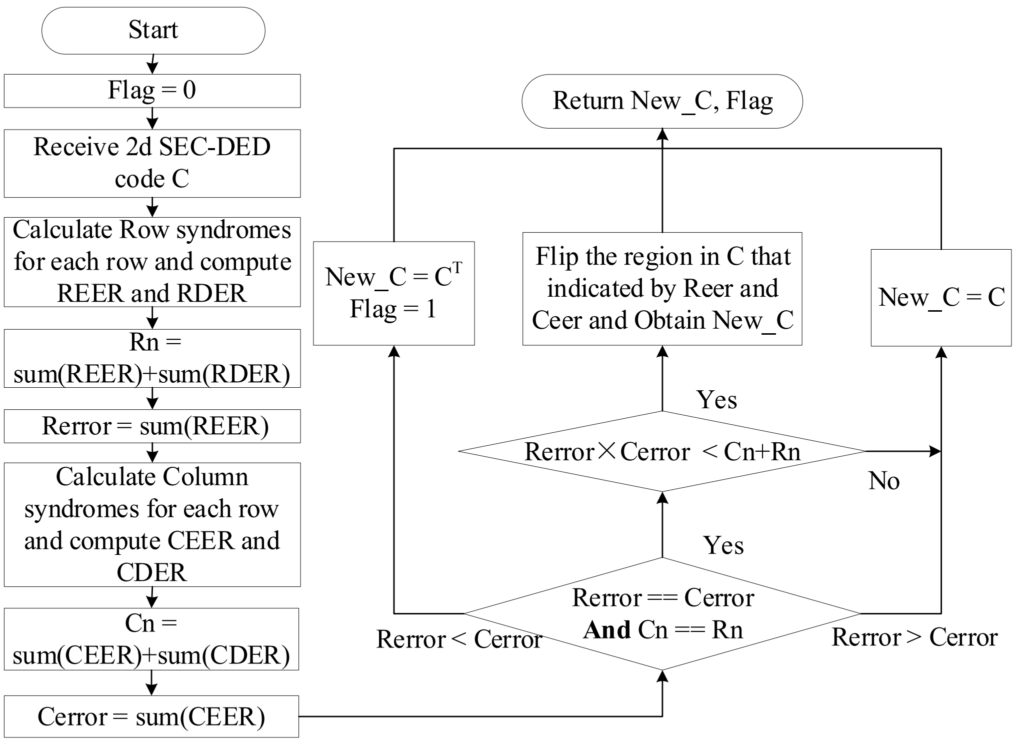
Start
Flag = 0
Receive 2d SEC-DED code C
Calculate Row syndromes for each row and compute REER and RDER
Rn = sum(REER)+sum(RDER)
Rerror = sum(REER)
Calculate Column syndromes for each row and compute CEER and CDER
Cn = sum(CEER)+sum(CDER)
Cerror = sum(CEER)
Rerror == Cerror And Cn == Rn
Rerror < Cerror
Rerror > Cerror
Yes
Rerror×Cerror < Cn+Rn
Yes
No
New_C = C^T Flag = 1
Flip the region in C that indicated by Reer and Ceer and Obtain New_C
New_C = C
Return New_C, Flag

Fig. 3. Flow chart of the preprocessing procedure.

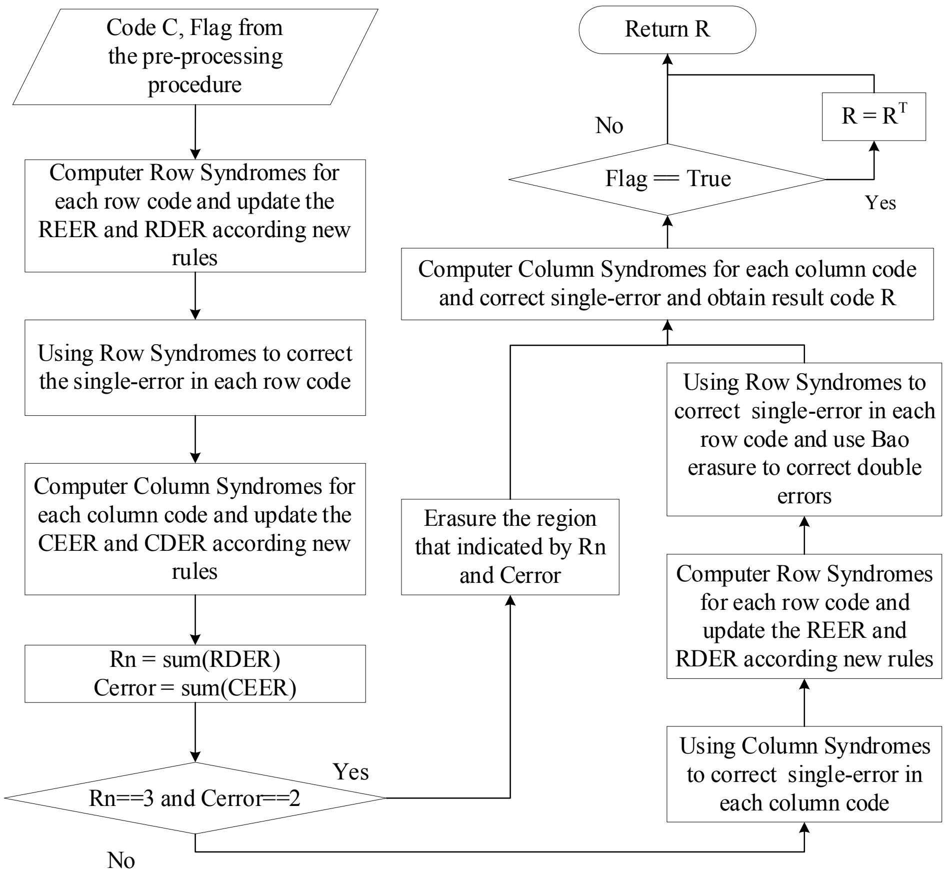
Code C, Flag from the pre-processing procedure
Computer Row Syndromes for each row code and update the REER and RDER according new rules
Using Row Syndromes to correct the single-error in each row code
Computer Column Syndromes for each column code and update the CEER and CDER according new rules
Rn = sum(RDER) Cerror = sum(CEER)
Rn==3 and Cerror==2
Yes
No
Erasure the region that indicated by Rn and Cerror
Using Column Syndromes to correct single-error in each column code
Computer Row Syndromes for each row code and update the REER and RDER according new rules
Using Row Syndromes to correct single-error in each row code and use Bao erasure to correct double errors
Computer Column Syndromes for each column code and correct single-error and obtain result code R
Flag == True
Yes
No
R = R^T
Return R

Fig. 4. Flow chart of the decoding procedure.

IV. Experiments and Results

A. Experimental Platform

All experiments were conducted on a standard PC laptop with an Intel i7-4720HQ (2.6 GHz) processor and 16 GB of DDR3 memory. The simulation software used was MATLAB R2021b.

B. Brief Description of the Experiments

The decoding object was a two-dimensional code with parameters (64, 16, 16) that was constructed from two extended Hamming product codes. For experimental comparisons, in addition to the proposed method, several well-known iterative decoding methods were implemented in MATLAB, including the two-step method [22], Kreshchuk's method [11] and Bao's method [15]. Three different decoding experiments were performed to evaluate the performance of all implemented decoding methods. The first experiment was conducted on a Gaussian channel with additive white Gaussian noise (AWGN). The second experiment was performed on a binary symmetric channel (BSC) with manually set error probabilities. In the third experiment, the number of errors was manually set, based on which the error correction capabilities of the different decoding methods were explored.

In the first two experiments, the bit error rate (BER) [23,24] and word error rate (WER) were used to evaluate the correction capabilities of the different decoding methods under different levels of noise. The level of noise is described by the signal-to-noise ratio (SNR) in the Gaussian channel and by the error probabilities in the BSC. However, in the third experiment, we conducted a statistical analysis of the numbers of instances of decoding failure when each decoding method was used for different given numbers of errors, and comparisons were performed.

C. Experiments based on the Gaussian Channel

In these experiments, we performed a total of 5 million independent encoding–decoding experiments under ten different SNRs with values from one to ten. Accordingly, under each SNR, 500 thousand repeated experiments were conducted.

In each independent encoding–decoding experiment based on the Gaussian channel, a 16-bit binary message was first randomly generated, which was then encoded into the original two-dimensional code using the method introduced in section two, yielding an 8×8 matrix. Afterward, binary phase-shift keying (BPSK) modulation [25] was applied, and the generated signal obtained from the original code was sent to the Gaussian channel, where AWGN with a certain SNR was added to the signal to obtain the received signal. The received code was then obtained by performing BPSK demodulation on the received signal. Then, the received code was decoded, and the bit errors were corrected using each implemented iterative decoding method separately. By comparing the decoded code against the original code, we could easily determine whether each decoding process had failed. We counted the failures in the experiments and calculated the corresponding statistics, based on which we could obtain the BER and WER. The pseudocode of the experiments with the Gaussian channel is presented in Fig. 5, and the experimental results obtained for the Gaussian channel are presented in Fig. 6 and Fig. 7.

```
Experiment 1 Experiment on Guassian Channel
Input: Experiment Repeat times $N$, Signal-to-Noise Rate SNR, decoding method $D_m$
Output: Decoding Failure times $N_e$
1: $N_e = 0$
2: for $(n = 0; n <= N; n++)$ do
3:     Msg = Random generate 16-bit binary number
4:     $Code_{sent}$ = encoding the Msg to 2D code
5:     $Signal_{sent}$ = Using BPSK to modulation the Code
6:     $Signal_{Rec}$ = Generate white Gaussian noise according to given SNR and add it to the $Signal_{sent}$
7:     $Code_{Rec}$ = demodulation the $Signal_{Rec}$ using BPSK
8:     $Code_{est}$ = Using given decoding method $D_m$ to correct errors in $Code_{Rec}$
9:     if $(Code_{sent} != Code_{est})$ then
10:        $N_e$++
11:    end if
12: end for
13: return $N_e$
```

Fig. 5. Pseudocode for the experiments based on the Gaussian channel.

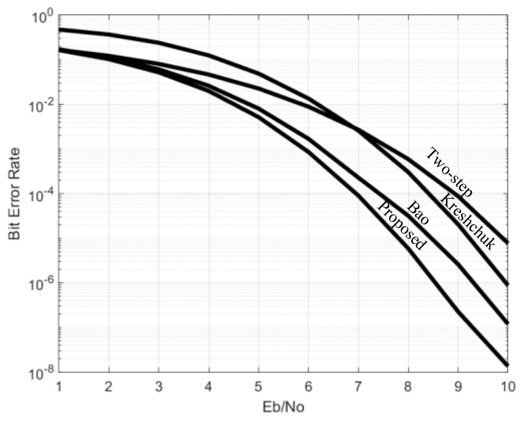

Fig. 6. BERs for the (64, 16, 16) code and the Gaussian channel.

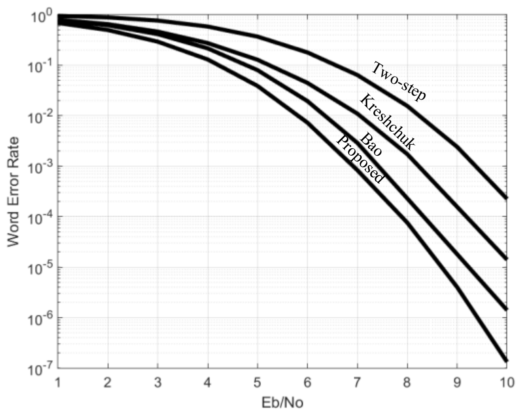

Fig. 7. WERs for the (64, 16, 16) code and the Gaussian channel.

Fig. 6 and Fig. 7 show that the proposed decoding method outperforms other decoding methods. To achieve a 10^{-4} BER, the proposed decoding method requires a 7 dB SNR; in contrast, Bao's decoding method, Kreshchuk's decoding method and the two-step decoding method require SNRs of approximately 7.3 dB, 8.4 dB and 9 dB, respectively. Moreover, when the SNR is 8 dB, the WER of the proposed method can reach 5.6×10^{-5}, lower than those of Bao's method, Kreshchuk's method and the two-step method, with values of 2.3×10^{-4}, 1.7×10^{-3} and 1.5×10^{-2}, respectively. Another interesting finding is that, as shown in Fig. 6, the BER of Kreshchuk's method is initially larger than that of the two-step method, but with increasing SNR, the BER of Kreshchuk's method decreases faster than that of the two-step method. When the SNR is seven, the BERs of these methods are equal, and when the SNR further increases, the BER of Kreshchuk's method is less than that of the two-step method. However, as shown in Fig. 7, the WER of Kreshchuk's method is always less than that of the two-step method. The reason for this phenomenon is that although Kreshchuk's method introduces erasure to properly address stall patterns with four errors and can therefore correct more error patterns than the two-step method (which is why its WER is always better than that of the two-step method), when it is faced with patterns that it is unable to correct, the erasure operation may introduce more error bits (which is why its BER may be higher than that of the two-step method).

D. Experiments based on the BSC

Based on the BSC, 7.5 million encoding–decoding experiments were independently repeated under 15 different levels of noise (with the error probability increasing from 0.01 to 0.15). Accordingly, for each specified level of noise, 500 thousand experiments were again conducted.

In each independent encoding–decoding experiment based on the BSC, the message and two-dimensional code were generated using a method similar to that introduced in the Gaussian channel experiments. Then, rather than applying modulation, the code was directly sent to the channel, and some bits were probabilistically flipped in accordance with the given error probability to obtain the received code. Next, all of the implemented decoding methods were used separately to decode the received code and correct error bits. After comparing the decoded code with the original code, the numbers of instances of decoding failure was counted, and the BER and WER were computed. The pseudocode for these experiments is shown in Fig. 8, and the results of the experiments are presented in Fig. 9 and Fig. 10.

```
**Experiment 2** Experiment on Binary symmetric Channel
**Input:** Experiment Repeat times $N$, Error Probability $P$, decoding method $D_m$
**Output:** Decoding Failure times $N_e$
  $N_e = 0$
  **for** $(n = 0; n <= N; n++)$ **do**
      Msg = Random generate 16-bit binary number
      $Code_{sent}$ = encoding the Msg to 2D code
      $Code_{Rec}$ = Flipped the bits in $Code_{sent}$ according to the $P$
      $Code_{est}$ = Using given decoding method $D_m$ to correct errors in $Code_{Rec}$
      **if** $(Code_{sent} != Code_{est})$ **then**
          $N_e$++
      **end if**
  **end for**
  **return** $N_e$
```

Fig. 8. Pseudocode for the experiments based on the BSC.

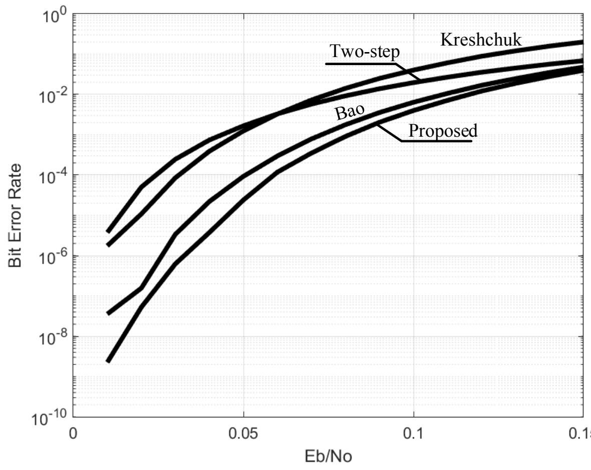

Fig. 9. BERs for the (64, 16, 16) code and the BSC.

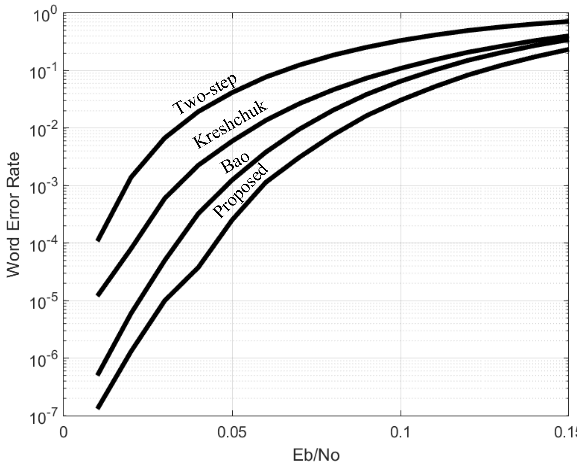

Fig. 10. WERs for the (64, 16, 16) code and the BSC.

Fig. 9 and Fig. 10 show that the proposed method has more powerful correction capabilities than the other three decoding methods. For example, when the error probability is 0.1, the BER and WER of the proposed method are approximately 3.9×10^{-3} and 3.05×10^{-2}, respectively. By comparison, the BER and WER of Bao's method, which are approximately 6.3×10^{-3} and 6.5×10^{-2}, respectively, are nearly double those of the proposed method. For the Kreshchuk and two-step methods, the BERs are 1.9×10^{-2} and 4×10^{-2}, respectively, and the WERs are 3.3×10^{-1} and 1.1×10^{-1}, respectively.

E. Experiments 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 I shows the numbers of error patterns used in the current experiments for different given numbers of error bits. Table II and Table III summarize the numbers of word errors and bit errors made with the different decoding methods under various numbers of error bits. Table IV and Table V were obtained by normalizing the data shown in Table II and Table III, respectively. Fig. 11 and Fig. 12 are visualizations of Table IV and Table V, respectively.

TABLE I
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
6~12	1000000

TABLE II
NUMBERS OF WORD DECODING ERRORS UNDER A GIVEN NUMBER OF ERROR BITS.

Given Number of Error Bits	Decoding Method			
	Two-Step Method	Kreshchuk's Method	Bao's Method	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
10	904245	409110	292349	95403
11	971570	590287	546901	263837
12	994317	762954	789497	532980

TABLE III
NUMBERS OF BIT DECODING ERRORS UNDER A GIVEN NUMBER OF ERROR BITS.

Given Number of Error Bits	Decoding Method			
	Two-Step Method	Kreshchuk's Method	Bao's Method	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
10	4082410	10131348	1445489	593974
11	5646670	17847788	3153695	1867220
12	7389060	26818870	5661903	4366654

TABLE IV
NORMALIZATION OF THE WORD ERRORS PRODUCED BY THE DECODING METHODS UNDER A GIVEN NUMBER OF ERROR BITS.

Given Number of Error Bits	Decoding Method			
	Two-Step Method	Kreshchuk's Method	Bao's Method	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
10	0.90424	0.40911	0.29234	0.09540
11	0.97157	0.59028	0.54690	0.26383
12	0.99431	0.76295	0.78949	0.53298

TABLE V
NORMALIZATION OF THE BIT ERRORS PRODUCED BY THE DECODING METHODS UNDER A GIVEN NUMBER OF ERROR BITS.

Given Number of Error Bits	Decoding Method			
	Two-Step Method	Kreshchuk's Method	Bao's Method	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
10	0.06378	0.15830	0.02258	0.00928
11	0.08822	0.27887	0.04927	0.02917
12	0.11545	0.41904	0.08846	0.06822

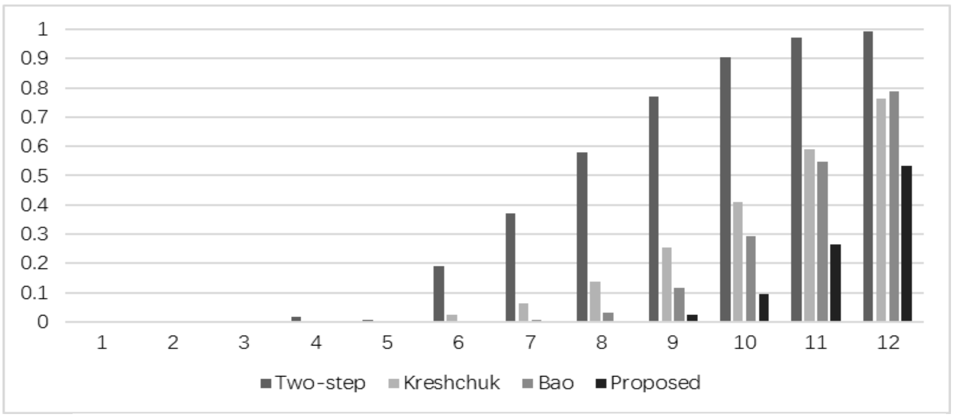

FIG. 11. HISTOGRAMS OF THE NORMALIZED WORD ERRORS PRODUCED BY THE DIFFERENT DECODING METHODS.

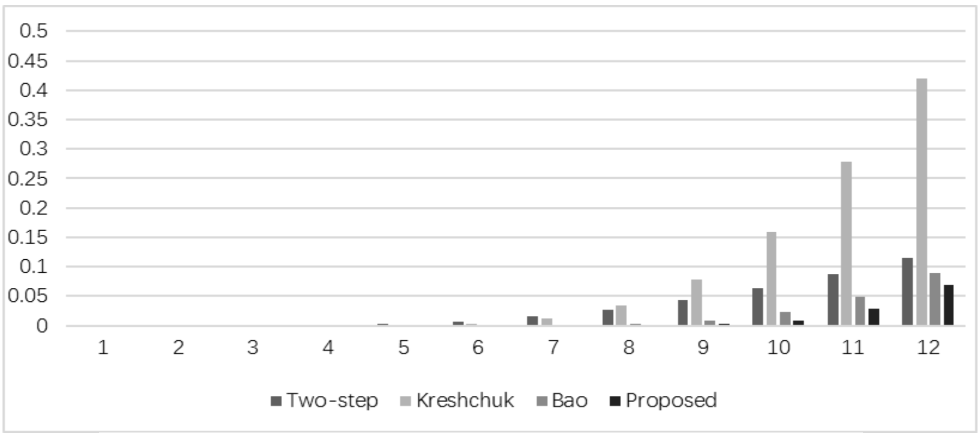

FIG. 12. HISTOGRAMS OF THE NORMALIZED BIT ERRORS PRODUCED BY THE DIFFERENT DECODING METHODS.

From the above tables, it is clear that 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 a 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 of the error bits; therefore, there is a high probability that the proposed method also has the ability to 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.

V. CONCLUSION AND FUTURE WORK

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. Three sets of decoding experiments performed based on a Gaussian channel, for a BSC and for 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.

In the future, we would like to use the proposed method in various fields to mitigate the effects of errors, such as correcting errors in wireless sensor networks and correcting on-chip interconnection errors to improve the reliability of on-chip communication.

REFERENCES

[1] P. Elias, "Error-free Coding," *IEEE Transactions on Information Theory*, vol. 4, no. 4, pp. 29-37, Sep. 1954.

[2] O. Al-Askary, "Iterative decoding of product codes," in *Signaler*, sensorer och system, 2003.

[3] G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein, "Covering codes," in *North-Holland Mathmatical Library Elsevier*, 1997.

[4] G. Cohen, M. Karpovsky, H. Mattson Jr., and J. Schatz, "Covering radius–survey and recent results," *IEEE Transactions on Information Theory*, vol. 31, May. 1985.

[5] G. S. Nikolic, M. K. Stojcev, T. R. Nikolic, B. D. Petrovic, and G. S. Jovanovic, "Reliable data transfer Rendezvous protocol in wireless sensor networks using 2D-SEC-DED encoding technique," *Microelectronics Reliability*, vol. 65, pp. 289–309, 2016, doi: 10.1016/j.microrel.2016.08.017.

[6] G. S. Nikolic, M. K. Stojcev, T. R. Nikolic, B. D. Petrovic, G. S. Jovanovic, and B. R. Dimitrijevic, "Implementation and evaluation of 2D SEC-DED forward error correction scheme in wireless sensor networks," *Microelectronics Reliability*, vol. 78, pp. 161–180, 2017, doi: 10.1016/j.microrel.2017.08.010.

[7] P. Chandrasekaran and R. Balasubramanyam, "An MCS with Iterative Multi-dimensional Hamming Product Codes," *IETE Journal Research*, vol. 66, no. 5, pp. 711–719, 2020, doi: 10.1080/03772063.2018.1527257.

[8] A. Kadum, W. N. Flayyih, and F. Z. Rokhani, “Reliability Analysis of Multibit Error Correcting Coding and Comparison to Hamming Product Code for On-Chip Interconnect,” *Journal of Engineering*, vol. 26, no. 6, pp. 94–106, 2020, doi: 10.31026/j.eng.2020.06.08.

[9] T. Hatamori and M. Kasahara, “Generalized product codes and their performance,” *Electronics and Communications in Japan (Part I Communications)*, vol. 77, no. 12, pp. 1–9, 1994, doi: 10.1002/ecja.4410771201.

[10] N. Abramson, “Cascade Decoding of Cyclic Product Codes,” *IEEE Transactions on Communication Technology*, vol. 16, no. 3, pp. 398-402, Jun. 1968.

[11] F. Blomqvist, “On hard–decision decoding of product codes,” *Applicable Algebra in Engineering*, Communication and Computing, pp. 1-18, 2021.

[12] K. Alexey, Z. Victor, and R. Eygene, “A new iterative decoder for product codes,” *Proceedings of the International Workshop on Algebraic and Combinational Coding Theory*, pp. 211-214, 2014.

[13] Fu Bo and P. Ampadu, “A multi-wire error correction scheme for reliable and energy efficient SoC links using Hamming product codes,” in SOC Conference, IEEE, pp. 59-62, 2008.

[14] Fu Bo and P. Ampadu, “An energy-efficient multiwire error control scheme for reliable on-chip interconnects using Hamming product codes,” VLSI Design, pp. 1-14, Dec. 2008.

[15] Fu Bo and P. Ampadu, “On hamming product codes with type-II hybrid ARQ for on-chip interconnects,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 56, no. 9, pp. 2042-2054, Sep. 2009.

[16] J. Kim and Y. Jee, “Hamming product code with iterative process for NAND flash memory controller,” in 2010 2nd International Conference on Computer Technology and Development, IEEE, pp. 611-615, Nov. 2010.

[17] A. K. Chlaab, W. N. Flayyih, and F. Z. Rokhani, “Lightweight hamming product code based multiple bit error correction coding scheme using shared resources for on chip interconnects,” *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 5, pp.1979-1989, Oct. 2020.

[18] G. Forney, “Generalized Minimum Distance decoding,” *IEEE Transactions on Information Theory*, vol. 12, no. 2, pp. 125-131, Apr. 1966.

[19] S. Reddy and J. Robinson, “Random error and burst correction by iterated codes,” *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 182-185, Jan. 1972.

[20] A. Sánchez-Macián, P. Reviriego and J. A. Maestro, "Hamming SEC-DAED and Extended Hamming SEC-DED-TAED Codes Through Selective Shortening and Bit Placement," in IEEE Transactions on Device and Materials Reliability, vol. 14, no. 1, pp. 574-576, March 2014, doi: 10.1109/TDMR.2012.2204753.

[21] X.H. Ren, J. Ma, V.Y. Tsviatkou and V. K. Kanapelka, "A New Hard-decision Iterative Decoding Method for Hamming Product Codes," *Engineering Letters*, vol. 30, no.3, pp 948-954, 2022.

[22] S. Lin and D. J. Costello, “Error control coding,” in *Scarborough: Prentice hall*, 2001.

[23] M. M. Karbassian and Ghafouri-Shiraz. Hooshang, "Phase-Modulations Analyses in Coherent Homodyne Optical CDMA Network Using a Novel Prime Code Family," Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2007, WCE 2007, 2-4 July, 2007, London, U.K., pp 358-362.

[24] A. Idris, D. Kaharudin, and S. Y. SK, Idris, A. K. Dimyati, and S. S. Yusof, "Performance of Linear Maximum Likelihood Alamouti Decoder with Diversity Techniques," Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2011, WCE 2011, 6-8 July, 2011, London, U.K., pp 1728-1731.

[25] H. Sandhu and D. Chadha, "Terrestrial free space LDPC coded MIMO optical link," Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2009, WCECS 2009, 20-22 October, 2009, San Francisco, USA, pp 372-375.