*entropy*

MDPI

*Article*

# A Reduced-Complexity Iterative Bounded Distance Decoder with Random Flipping for Product Codes

**Guoming Song, Dongming Pi and Shancheng Zhao ***

College of Information Science and Technology, Jinan University, Guangzhou 510632, China; guomingsong@stu2024.jnu.edu.cn (G.S.); dongmingpi@stu2023.jnu.edu.cn (D.P.)
* Correspondence: shanchengzhao@jnu.edu.cn

**Abstract**

Product codes (PCs) are widely used in high-speed communication systems due to their attractive trade-off between error-correction performance and complexity. To further meet the rapidly growing demand for higher data rates, soft-aided hard-decision decoders (SA-HDDs) have been developed. In this paper, we present a reduced-complexity iterative bounded distance decoder with random flipping (RC-iBDD-RF) for PCs, an SA-HDD that improves the decoding performance while preserving low decoding complexity. RC-iBDD-RF introduces an enhanced reliability metric that incorporates channel log-likelihood ratios (LLRs) and memory from previous iterations to guide random flipping. In addition, an error-and-erasure decoding (EaED) module employing fixed filling patterns, rather than randomly generated ones, is used for post-processing to further reduce the residual error rate. The comparisons for BCH-based PCs show that RC-iBDD-RF achieves both performance gain and complexity reduction with only a slight increase in memory requirement. Moreover, the proposed complexity-storage weighted SNR metric confirms its superior complexity-performance trade-off over iBDD-RF.

**Keywords:** product code; soft-aided hard-decision decoding; iterative bounded distance decoding; error-and-erasure decoding; bit flipping

## 1. Introduction

Product codes (PCs) and other product-like constructions are widely used in high-speed communication systems due to their attractive trade-off between implementation cost and error-correction performance [1,2]. For systems such as optical transport networks (OTNs), where high throughput and low decoding latency are critical [3], hard-decision decoders remain particularly attractive since they significantly reduce data flow and enable simpler hardware implementations compared with soft-decision decoding schemes [4–6].

Extensive research efforts have been devoted to the construction of advanced product-like codes. To improve the performance of classical PCs, staircase codes (SCs) were proposed, in which neighboring blocks are spatially coupled to enhance decoding performance [7]. Hu et al. designed a high-throughput SC encoder achieving an approximate throughput of 119.5 Gbps, demonstrating the feasibility of SCs for real-world OTNs [8]. In [9], an experimental demonstration of the performance advantages of the multi-chain coupled staircase code in a 240-Gbps optical coherent transmission system was presented. In [10], a novel framework termed zipper code was introduced to describe several product-like codes. Within this framework, several high-performance constructions, including



Academic Editor: Chi Wan Sung

Received: 11 June 2026
Revised: 13 July 2026
Accepted: 15 July 2026
Published: 20 July 2026

multi-chain coupled, Reed–Solomon (RS)-based zipper codes and high-order staircase codes, were proposed [11–15]. These constructions offer increased design flexibility in terms of coupling dimensions, component codes, and decoding schedules, while maintaining compatibility with hard-decision decoding. In addition to product-like code constructions over BCH and RS component codes, recent progress in algebraic coding theory has also explored code constructions and decoding algorithms over richer algebraic structures, such as Gaussian integer rings, Eisenstein integers, quaternion/Hurwitz integers, and octonion integers [16–18]. These studies broaden the algebraic foundations of coding theory and may provide useful insights for future code design and decoding algorithms.

However, the low-complexity decoders for product codes remain a critical performance bottleneck. In existing product-like code constructions, Bose–Chaudhuri–Hocquenghem (BCH) or RS codes are commonly employed as component codes [19]. Their low-complexity decoders typically rely on iterative bounded-distance decoding (iBDD), which, however, suffers from inherently limited performance gains [20]. In particular, when the number of errors within a received sequence exceeds the error-correcting capability, a bounded-distance decoder (BDD) may produce miscorrections [21], which can severely degrade the overall decoding performance and even lead to error propagation across iterations. As a result, the error-correction performances of iBDD-based decoders are often significantly inferior to those of the soft-decision decoders, particularly in the moderate-to-high signal-to-noise ratio (SNR) region. To address these limitations while preserving the low complexity of hard-decision decoding, extensive research focuses on the design of soft-aided hard-decision decoders (SA-HDDs).

A variety of SA-HDDs have been proposed in the literature by exploiting soft reliabilities. Log-likelihood ratios (LLRs) were incorporated into the decoding process of iBDD, leading to iBDD with scaled reliability (iBDD-SR) that significantly improves the decoding performance [22]. The parameters of iBDD-SR were optimized in [23]. Building upon iBDD-SR, Liga et al. introduced reliability-aware bit marking into BDD, achieving approximately 0.8 dB performance gain over conventional iBDD [24]. Further advancement was made by Liva et al. through the use of generalized minimum-distance decoding, which substantially reduced the performance gap between traditional iBDD and the Chase–Pyndiah decoder [1], albeit at the cost of increased decoding complexity [25]. Lei et al. proposed an SA-HDD by classifying bits based on their LLRs and selectively flipping those with smaller reliabilities to suppress miscorrections [26]. More recently, refined reliability metrics have been developed in [27]. Other low-complexity SA-HDDs can be found in [28,29]. To further improve decoding performance, iterative bounded-distance decoding with random flipping (iBDD-RF) [30] was proposed, resulting in a remarkable performance gain.

However, a significant performance gap remains between existing HDDs and the Chase–Pyndiah decoder [1] under comparable complexity constraints. In this paper, we propose a reduced-complexity iterative bounded-distance decoder with random flipping (RC-iBDD-RF), a soft-aided binary message-passing decoding scheme with refined bit reliability to enable low-cost random flipping. Specifically, RC-iBDD-RF augments the conventional weighted combination of the BDD output and channel LLR with a memory term that retains useful information from previous iterations. Specifically, unlike the existing iBDD-RF [30] that relies mainly on the current BDD output and channel LLR, RC-iBDD-RF augments its conventional weighted combination with a memory term to retain useful information from previous iterations. This enhanced reliability reduces erroneous flips and improves the overall error-correction capability.

The main contributions of this paper are summarized as follows. First, we develop an enhanced reliability metric that incorporates past decoding states into the soft reliability of each bit, enabling more informed bit-flipping decisions. Second, we incorporate the

error-and-erasure decoder (EaED) with fixed filling patterns as a post-processing module to reduce the residual errors. Third, we provide a detailed analysis of the memory and computational complexities of the proposed decoder. Finally, simulation results demonstrate that RC-iBDD-RF reduces the decoding complexity while achieving performance gains.

## 2. Preliminaries

### *2.1. Product Codes*

Product codes are a class of algebraic block codes constructed by the Cartesian product of two or more shorter component codes. Since their introduction by Elias in 1954 [31], product codes have been extensively investigated and have found widespread use in high-speed communication systems due to their favorable trade-off between decoding complexity and error-correction performance.

A PC can be represented by an $n_2 \times n_1$ matrix with $n_2$ rows and $n_1$ columns, where each row and each column corresponds to a codeword of a binary linear block code. Specifically, the row component code is denoted by $\mathscr{C}_1[n_1, k_1, t_1]$, while the column component code is denoted by $\mathscr{C}_2[n_2, k_2, t_2]$. Here, $n_1$ and $n_2$ denote the code lengths, $k_1$ and $k_2$ denote information dimensions, and $t_1$ and $t_2$ denote the error-correcting capabilities of the binary linear block codes $\mathscr{C}_1$ and $\mathscr{C}_2$, respectively.

The structure of a PC consists of four sub-matrices: an information-bit matrix of size $k_1 \times k_2$; a row-parity matrix of size $(n_1 - k_1) \times k_2$; a column-parity matrix of size $(n_2 - k_2) \times k_1$; and a parity-of-parity matrix of size $(n_1 - k_1) \times (n_2 - k_2)$. Accordingly, the code rate of the PC can be expressed as

$$R = R_1 \cdot R_2 = \frac{k_1 k_2}{n_1 n_2}, \tag{1}$$

where $R_1$ and $R_2$ denote the code rates of the component codes $\mathscr{C}_1$ and $\mathscr{C}_2$, respectively. When both rows and columns employ the same binary linear block code $\mathscr{C}[n, k, t]$, the code rate of the resulting PC is

$$R = \frac{k^2}{n^2}. \tag{2}$$

### *2.2. The Conventional iBDD-RF*

The iBDD-RF is a novel binary message-passing decoder that randomly flips bits with low reliability [30]. For completeness, we present in Figure 1 the block diagram of iBDD-RF. We use $\boldsymbol{\psi}^{(l)} = (\psi_1^{(l)}, \cdots, \psi_n^{(l)})$ and $\boldsymbol{\varphi}^{(l)} = (\varphi_1^{(l)}, \cdots, \varphi_n^{(l)})$ to represent the input sequence and output sequence of a row or a column of the iBDD-RF in the $l$-th iteration, respectively.

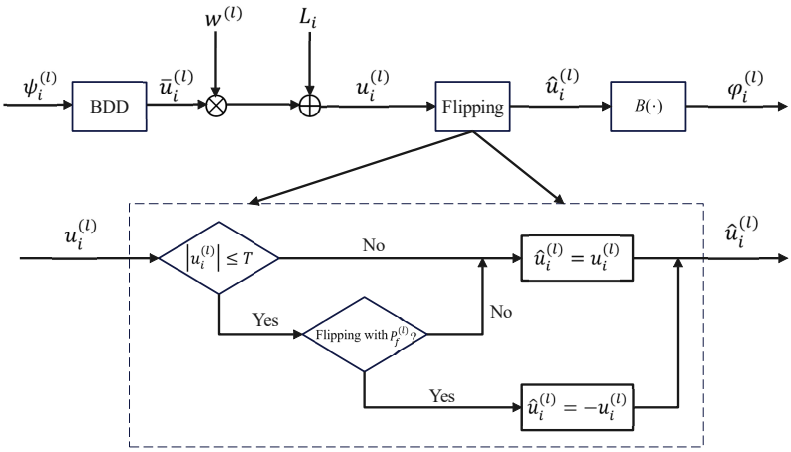



**Figure 1.** The decoding scheme of iBDD-RF.

Given the input $\boldsymbol{\psi}^{(l)} = (\psi_1^{(l)}, \cdots, \psi_n^{(l)})$, let $\bar{\boldsymbol{c}}^{(l)} = (\bar{c}_1^{(l)}, \cdots, \bar{c}_n^{(l)})$ denote the decoding output of the BDD at the $l$-th iteration. If the BDD declares a success, we have $\bar{u}_i^{(l)} = (-1)^{\bar{c}_i^{(l)}}$. Otherwise, we have $\bar{u}_i^{(l)} = 0$. Similar to iBDD-SR, the soft reliability $u_i^{(l)}$ of the $i$-th code bit $c_i$ is given as

$$u_i^{(l)} = w^{(l)} \bar{u}_i^{(l)} + L_i, \tag{3}$$

where $w^{(l)}$ is the scaling factor at the $l$-th iteration, and $L_i$ is the channel LLR of the $i$-th code bit. Let $T$ be a given threshold. If $|u_i^{(l)}| < T$, $u_i^{(l)}$ is flipped with a probability of $P_f^{(l)}$. The output after flipping is denoted as $\hat{u}_i^{(l)}$. That is, we have

$$\hat{u}_i^{(l)} = \begin{cases} u_i^{(l)} & \text{if } |u_i^{(l)}| > T \\ u_i^{(l)} & \text{if } |u_i^{(l)}| \leq T, \text{with probability } 1 - P_f^{(l)} \\ -u_i^{(l)} & \text{if } |u_i^{(l)}| \leq T, \text{with probability } P_f^{(l)}. \end{cases} \tag{4}$$

Finally, $\hat{\boldsymbol{u}}^{(l)} = (\hat{u}_1^{(l)}, \hat{u}_2^{(l)}, \cdots, \hat{u}_n^{(l)})$ is taken as the input to the hard-decision operator $B(\cdot)$ to obtain the output $\boldsymbol{\varphi}^{(l)}$ as $\boldsymbol{\varphi}^{(l)} = B(\hat{\boldsymbol{u}}^{(l)}) = (B(\hat{u}_1^{(l)}), B(\hat{u}_2^{(l)}), \cdots, B(\hat{u}_n^{(l)}))$, where

$$B(x) = \begin{cases} 0, & x > 0 \\ 1, & x \leq 0. \end{cases} \tag{5}$$

### *2.3. Error and Erasure Decoder*

The EaED performs two BDDs, denoted by $\mathcal{D}_1$ and $\mathcal{D}_2$, and then selects the more reliable result as the final output. The input sequence to EaED is given by

$$\boldsymbol{y} \in \{0, ?, 1\}^n, \tag{6}$$

where $n$ denotes the code length, the symbol "?" denotes an erased bit. We assume that the number of erased bits in the input sequence is $E$, and that the minimum Hamming distance of the component code is $d_{\min}$. The output sequence of EaED is denoted by $\boldsymbol{w}$.

If $E \geq d_{\min}$, EaED does not perform decoding and keeps the input sequence unchanged. We then have $\boldsymbol{w} = \boldsymbol{y}$.

If $E < d_{\min}$, two new input sequences $\boldsymbol{y}^{(1)}$ and $\boldsymbol{y}^{(2)}$ are constructed from $\boldsymbol{y}$ and fed into $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. The constructions of $\boldsymbol{y}^{(1)}$ and $\boldsymbol{y}^{(2)}$ are described in the following. For the non-erased positions, the two sequences are identical to the input sequence. For the erased positions, two filling patterns, $\boldsymbol{p}^{(1)}$ and $\boldsymbol{p}^{(2)}$, are used to replace the erased bits, where $\boldsymbol{p}^{(1)}$ and $\boldsymbol{p}^{(2)}$ are complementary to each other. In the iterative decoder of [32], the filling patterns $\boldsymbol{p}^{(1)}$ and $\boldsymbol{p}^{(2)}$ are randomly generated for each execution of EaED. The decoding output of EaED is given below:

- If both $\mathcal{D}_1$ and $\mathcal{D}_2$ fail to decode, the input sequence is retained, and we have $\boldsymbol{w} = \boldsymbol{y}$.
- If one of the two decoders succeeds while the other fails, the decoding result of the successful decoder is selected as the output $\boldsymbol{w}$.
- If both $\mathcal{D}_1$ and $\mathcal{D}_2$ decode successfully, the output is determined as follows. Let $\boldsymbol{r}^{(1)}$ and $\boldsymbol{r}^{(2)}$ denote the two decoding outputs. The Hamming distances of $\boldsymbol{r}^{(1)}$ and $\boldsymbol{r}^{(2)}$ to $\boldsymbol{y}$ are denoted by $\delta_1$ and $\delta_2$, respectively. Based on $\delta_1$ and $\delta_2$, the output $\boldsymbol{w}$ is given by

$$\boldsymbol{w} = \begin{cases} \boldsymbol{r}^{(1)}, & \text{if } \delta_1 < \delta_2 \\ \boldsymbol{r}^{(2)}, & \text{if } \delta_1 > \delta_2 \\ \boldsymbol{r}^{(1)} \text{ or } \boldsymbol{r}^{(2)}, & \text{if } \delta_1 = \delta_2. \end{cases} \tag{7}$$

## 3. The Proposed Reduced-Complexity Iterative Bounded Distance Decoder with Random Flipping

In this paper, we propose the following binary message passing decoder, which is a modification of iBDD-RF [30]. For ease of presentation, the number of iterations is indexed according to the number of half-iterations, i.e., the row decoding and column decoding at the $l$-th iteration are represented as the $(2l-1)$-th half-iteration and the $2l$-th half-iteration.

### *3.1. Enhanced Soft Reliability*

We use the notations of iBDD-RF [30] in this paper to present the proposed reduced-complexity iBDD-RF (RC-iBDD-RF). For completeness, we present the block diagram of RC-iBDD-RF in Figure 2. Let $\boldsymbol{\varphi}_i^{(l)} = (\varphi_{i,1}^{(l)}, \cdots, \varphi_{i,n}^{(l)})$ denote the output sequence of the $i$-th row at the $l$-th half-iteration. For the $l$-th half-iteration, the input of BDD of the $i$-th row is $\boldsymbol{\psi}_i^{(l)} = (\psi_{i,1}^{(l)}, \cdots, \psi_{i,n}^{(l)})$, and the decoding output of BDD is $\bar{\boldsymbol{c}}_i = (\bar{c}_{i,1}, \cdots, \bar{c}_{i,n})$. In RC-iBDD-RF, the decoding reliability $\bar{u}_{i,j}^{(l)}$ of the BDD for the $j$-th position of the $i$-th row at the $l$-th half-iteration is given as

$$\bar{u}_{i,j}^{(l)} = \begin{cases} 0 & \text{if BDD fails} \\ (-1)^{\bar{c}_{i,j}} & \text{if BDD succeeds.} \end{cases} \tag{8}$$

In iBDD-RF, the soft reliability $u_{i,j}^{(l)}$ is computed as $u_{i,j}^{(l)} = w^{(l)}\bar{u}_{i,j}^{(l)} + L_{i,j}$, where $w^{(l)}$ is the scaling factor, and $L_{i,j}$ is the channel LLR. It can be observed that, in iBDD-RF, the soft reliability $u_{i,j}^{(l)}$ does not account for either the number of bits corrected by BDD or the outcomes of the most recent decoding half-iteration. First, the probability of miscorrection in BDD increases with the number of corrected bits. Second, once a decoding success is declared in the most recent decoding half-iteration, the corresponding decision should not be easily reversed. These observations motivate the following enhanced soft reliability metric.

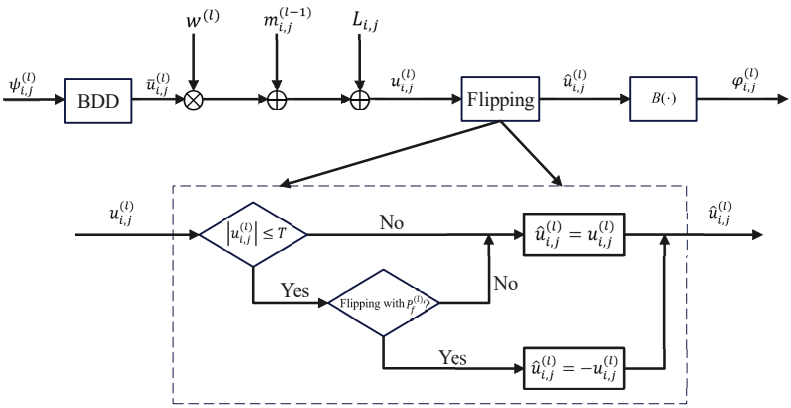



**Figure 2.** The decoding scheme of RC-iBDD-RF.

Let $s_i^{(l-1)}$ denote the status of the $i$-th BDD decoder at the $(l-1)$-th half-iteration. We have

$$s_i^{(l-1)} = \begin{cases} 0 & \text{if BDD fails} \\ t+1-\delta & \text{if BDD succeeds,} \end{cases} \tag{9}$$

where $t$ is the error correcting capability of the component code, and $\delta$ denotes the number of bits corrected by BDD. Accordingly, we define the *historical reliability* $m_{i,j}^{(l-1)}$ of the $(i,j)$ bit at the $(l-1)$-th half-iteration as

$$m_{i,j}^{(l-1)} = \beta^{(l-1)}(-1)^{\varphi_{i,j}^{(l-1)}} s_j^{(l-1)}, \tag{10}$$

when $l$ is even, and as

$$m_{i,j}^{(l-1)} = \beta^{(l-1)}(-1)^{\varphi_{i,j}^{(l-1)}} s_i^{(l-1)}, \tag{11}$$

when $l$ is odd. Here, $\beta^{(l)}$ is a given scaling factor, $\varphi_{i,j}^{(l-1)}$ is the output of last half-iteration. The soft reliability $u_{i,j}^{(l)}$ in RC-iBDD-RF is given as

$$u_{i,j}^{(l)} = w^{(l)} \bar{u}_{i,j}^{(l)} + m_{i,j}^{(l-1)} + L_{i,j}. \tag{12}$$

Given the soft reliability $u_{i,j}^{(l)}$, we attempt to flip bits randomly. Let $T$ be a given threshold. If $|u_{i,j}^{(l)}| < T$, $u_{i,j}^{(l)}$ is flipped with a probability of $P_f^{(l)}$. The output result after flipping is denoted as $\hat{u}_{i,j}^{(l)}$. That is, we have

$$\hat{u}_{i,j}^{(l)} = \begin{cases} u_{i,j}^{(l)} & \text{if } |u_{i,j}^{(l)}| > T \\ u_{i,j}^{(l)} & \text{if } |u_{i,j}^{(l)}| \le T, \text{with probability } 1 - P_f^{(l)} \\ -u_{i,j}^{(l)} & \text{if } |u_{i,j}^{(l)}| \le T, \text{with probability } P_f^{(l)}. \end{cases} \tag{13}$$

Finally, $\hat{\boldsymbol{u}}_i^{(l)}$ is taken as the input to the hard-decision operator $B(\cdot)$. The output $\boldsymbol{\varphi}_i^{(l)}$ of the RC-iBDD-RF at the $l$-th half-iteration is $\boldsymbol{\varphi}_i^{(l)} = B(\hat{\boldsymbol{u}}_i^{(l)}) = (B(\hat{u}_{i,1}^{(l)}), B(\hat{u}_{i,2}^{(l)}), \cdots, B(\hat{u}_{i,n}^{(l)}))$. For completeness, Algorithm 1 presents the procedure of a half-iteration of RC-iBDD-RF.

Remark: Unlike conventional SA-iBDD and reliability-combining methods that mainly rely on channel reliability $L_{i,j}$ or current decoding output $\varphi_{i,j}^{(l)}$, the proposed metric further incorporates the past decoding information $m_{i,j}^{(l-1)}$ of each bit. Hence, bits with similar channel reliabilities can be assigned different flipping priorities according to their decoding histories. This memory-aided design allows RC-iBDD-RF to focus random flipping on persistently unreliable bits.

### *3.2. Post-Processing by EaED*

However, similar to iBDD-RF, the decoding output of RC-iBDD-RF is not guaranteed to be a codeword, which may lead to a high error floor. To further improve the performance of RC-iBDD-RF, we propose to replace the last two half-iterations of BDD with EaED, where the enhanced reliability is used as the erasure criterion. First, to effectively control the decoding complexity, we apply erasure for EaED only once, i.e., at the penultimate half-iteration of RC-iBDD-RF. Second, in the proposed decoder, the two filling patterns $\boldsymbol{p}^{(1)}$ and $\boldsymbol{p}^{(2)}$ in EaED are fixed as the all-zero vector $\mathbf{0}$ and the all-one vector $\mathbf{1}$.

The details of the last two half-iterations with EaED are given as follows:

- Bits whose absolute reliabilities $|u_{i,j}^{(l)}|$ are smaller than a predefined threshold $T_{\text{era}}$ are erased.
- If an erased bit is successfully recovered during the penultimate half-iteration, then in the subsequent half-iteration, this bit is no longer in the erased state.
- If an erased bit is not recovered during the penultimate half-iteration, it remains in the erased state in the subsequent half-iteration.

**Algorithm 1:** The Decoding Procedure of a Half-Iteration of RC-iBDD-RF

**Input:** $\boldsymbol{\psi}_i^{(l)}$
**Output:** $\boldsymbol{\varphi}_i^{(l)}$
$\mathrm{BDD}(\boldsymbol{\psi}_i^{(l)}) \rightarrow \bar{\boldsymbol{c}}_i^{(l)}$;
**if** *BDD decoding is successful with* $\delta$ *bits corrected* **then**
  $s_i^{(l)} = t + 1 - \delta$;
**else**
  $s_i^{(l)} = 0$;
**while** $j \leq n$ **do**
  **if** *the BDD decoding is successful* **then**
    $\bar{u}_{i,j}^{(l)} = (-1)^{\bar{c}_{i,j}}$;
  **else**
    $\bar{u}_{i,j}^{(l)} = 0$;
  **if** $l$ *is even* **then**
    $u_{i,j}^{(l)} = w^{(l)} \bar{u}_{i,j}^{(l)} + \beta^{(l-1)} (-1)^{\varphi_{i,j}^{(l-1)}} s_j^{(l-1)} + L_{i,j}$;
  **else**
    $u_{i,j}^{(l)} = w^{(l)} \bar{u}_{i,j}^{(l)} + \beta^{(l-1)} (-1)^{\varphi_{i,j}^{(l-1)}} s_i^{(l-1)} + L_{i,j}$;
  **if** $|u_{i,j}^{(l)}| \leq T$ **then**
    Flip the bit with a flipping probability of $P_f^{(l)}$;
  **else**
    $\hat{u}_{i,j}^{(l)} = u_{i,j}^{(l)}$;
  $\boldsymbol{\varphi}_i^{(l)} = B(\hat{\boldsymbol{u}}_i^{(l)})$;

In [32], the two filling patterns are randomly generated for each execution of EaED, which incurs a higher complexity than the proposed scheme. We chose these two fixed filling patterns since erasing is executed only once in the proposed decoder.

In the following, we analyze the error-correcting capability of the ideal EaED (iEaED) with fixed filling patterns **0** and **1** using a genie-aided method. Since miscorrection is not allowed in iEaED, its decoding rule can be expressed as

$$\mathrm{iEaED}(\boldsymbol{y}) = \begin{cases} \boldsymbol{x} & \text{if } \mathrm{EaED}(\boldsymbol{y}) = \boldsymbol{x} \\ \boldsymbol{y} & \text{otherwise.} \end{cases} \tag{14}$$

where $\boldsymbol{x}$ denotes the transmitted codeword. Although such an ideal decoder is not implementable in practice, anchor decoders enable the decoding performance to approach that of an ideal miscorrection-free decoder. Moreover, the miscorrection-free assumption enables a quantitative mathematical analysis of the probability that EaED successfully decodes a component code.

Following [32], let $P_s(D, E)$ denote the probability that iEaED successfully decodes when $E$ bits are erased and $D$ bits are erroneous. Let $t$ denote the error-correcting capability of the component code and $e$ denote the number of positions in $\boldsymbol{p}^{(1)} = \boldsymbol{0}$ that differ from the transmitted codeword $\boldsymbol{x}$. Following [32], the decoding success probability of iEaED with fixed filling patterns is given by

$$P_s(D,E) = \begin{cases} 1 & 2D + E < d_{\min} \\ 0 & E \geq d_{\min} \text{ or } D > t \\ 2^{1-E} \sum_{e=0}^{t-D} \binom{E}{e} & \text{otherwise.} \end{cases} \tag{15}$$

It can be seen that when $2D + E < d_{\min}$, EaED always decodes successfully. If all erroneous bits are erased, the EaED can correct up to $d_{\min} - 1$ errors, which is significantly higher than the error-correction capability $t$ of BDD.

The above analysis shows the potential advantages of EaED in error correction. Hence, we propose to incorporate EaED into RC-iBDD-RF to obtain additional performance gain. Specifically, EaED is employed in the last two half-iterations of RC-iBDD-RF as a post-processing procedure.

## 4. The Analysis of Complexity

In [30], a detailed complexity comparison showed the advantage of iBDD-RF in terms of computational complexity when compared with existing SA-HDDs. Therefore, in the following, we mainly focus on comparing the complexity of RC-iBDD-RF with that of iBDD-RF. We consider memory storage and computational complexity in this paper and briefly discuss the practical implementation implications at the end of this section.

### *4.1. Memory Storage Analysis*

For RC-iBDD-RF, it is required to store $w^{(l)}$, $\beta^{(l)}$, the probability of flipping $p_f^{(l)}$, decoding status $s^{(l)}$, and the channel LLR. We consider a BCH component code $\mathscr{C}[n,k,t]$ with $q$-bit quantization. Let $l_{\text{RC}}$ denote the number of iterations for the proposed RC-iBDD-RF. The number of decoding iterations is set as $l_{\text{RC}} = l_{\text{RF}} = l_{\text{iBDD}} = 12$, following the common practice for HDD-based product-code decoders. This setting is also verified by our preliminary convergence tests, which show that further iterations provide only negligible performance improvement while increasing decoding complexity.

- For a PC, a complete iteration consists of a row half-iteration and a column half-iteration. Therefore, the memory required to store $w^{(l)}$, $\beta^{(l)}$, and $p_f^{(l)}$ is $2ql_{\text{RF}}$.
- The maximum possible number of statuses is $t+1$. Since the error correction capability $t$ of the BCH component code is typically smaller than four, three bits are enough for storing the status of a row or a column. As a result, the total memory requirement of the statuses is $3n$.
- For the channel LLR, its memory requirement is $n^2q$.

Thus, the additional memory requirement of RC-iBDD-RF, when compared with iBDD, is

$$M_{\text{RC-iBDD-RF}} = q(6l_{\text{RC}} + n^2) + 3n. \tag{16}$$

It was shown in [30] that the additional memory requirement of iBDD-RF, when compared with iBDD, is $M_{\text{iBDD-RF}} = q(4l_{\text{RF}} + n^2)$.

- For $\mathscr{C}_1[255,231,3]$, $n = 255$, $l_{\text{RC}} = l_{\text{RF}} = 12$, the memory requirement of RC-iBDD-RF is calculated as $65,097q + 765$ bits, while that of iBDD-RF is $65,073q$ bits, where $q$ is the quantization bit width of the channel LLR. Therefore, the extra memory of RC-iBDD-RF over iBDD-RF is only $24q + 765$ bits.
- For $\mathscr{C}_2[256,239,2]$, $n = 256$, $l_{\text{RC}} = l_{\text{RF}} = 12$, the corresponding memory requirements are $65,608q + 768$ bits and $65,584q$ bits, respectively, and the extra memory is $24q + 768$ bits.

In practical implementations, the quantization bit width $q$ is typically no more than 10 bits. These results show that the additional memory requirement of RC-iBDD-RF is limited.

### *4.2. Computational Complexity Analysis*

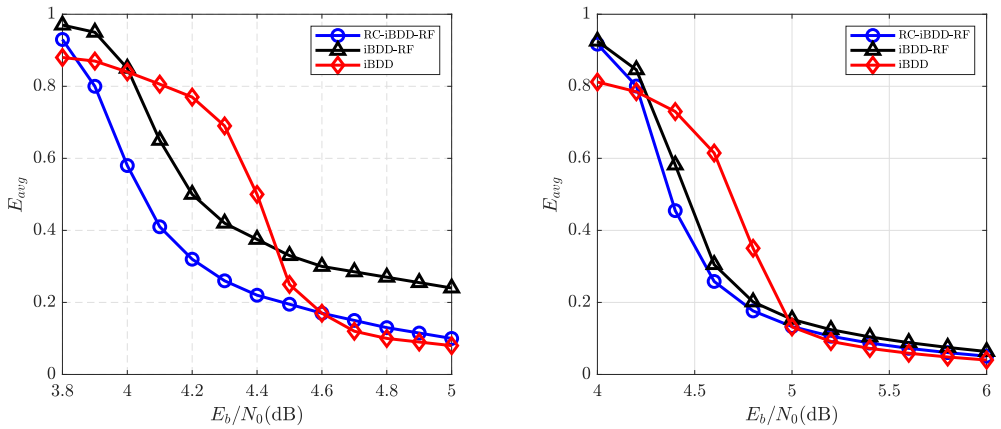
In this subsection, we analyze the computational complexity of RC-iBDD-RF. Similar to iBDD-RF, the main computational complexity of RC-iBDD-RF arises from the execution of the Berlekamp–Massey (BM) algorithm in BDD and the generation of random numbers.

The computational complexity of a single execution of the BM algorithm is $\mathcal{O}(n^2)$. Let $E_{\text{avg}}$ denote the average execution number of the BM algorithm per codeword in a single decoding iteration, which is defined as

$$E_{\text{avg}} = \frac{E_{\text{total}}}{2l_{\text{RC}}n}, \tag{17}$$

where $E_{\text{total}}$ is the total execution number of BDD during decoding. Since each iteration involves decoding all codewords in both row and column directions, the overall computational complexity of the BM algorithm in RC-iBDD-RF can be written as $\mathcal{O}(2l_{\text{RC}}E_{\text{avg}}n^3)$. It can be observed that this expression is consistent with that of iBDD-RF. Hence, to compare the computational complexities of RC-iBDD-RF and iBDD-RF, we only need to compare their $E_{\text{avg}}$. Figure 3a,b, we present the values of $E_{\text{avg}}$ for the RC-iBDD-RF. We select the BCH codes $\mathscr{C}_1[255, 231, 3]$ and $\mathscr{C}_2[256, 239, 2]$ as the component codes. The average execution numbers of the BM algorithm of iBDD-RF are also given in Figure 3a,b. Based on these two figures, we can make the following observations:

- The average execution number of BM in RC-iBDD-RF is lower than that of iBDD-RF.
- The average execution number of BM in RC-iBDD-RF is comparable to that of iBDD.




(**a**) Based on $\mathscr{C}_1$ (**b**) Based on $\mathscr{C}_2$

**Figure 3.** The average execution numbers $E_{\text{avg}}$ of BDD for RC-iBDD-RF and iBDD-RF.

We then consider the complexity of random number generation in RC-iBDD-RF. Since the filling patterns $\boldsymbol{p}^{(1)}$ and $\boldsymbol{p}^{(2)}$ are chosen as the all-zero and all-one sequences, random number generation in RC-iBDD-RF is only required for bit flipping. We use $N_{\text{avg}}$ to denote the average number of random numbers required for each codeword in a single iteration. Thus, the computational complexity associated with random number generation in RC-iBDD-RF is $\mathcal{O}(2l_{\text{RC}}nN_{\text{avg}})$. In Figure 4, we present the values of $N_{\text{avg}}$ for RC-iBDD-RF and iBDD-RF, where the code $\mathscr{C}_1$ is selected as the component code. It can be seen that the two decoders admit comparable average numbers for random number generation. Hence, the complexities of generating random numbers in RC-iBDD-RF and iBDD-RF are comparable.

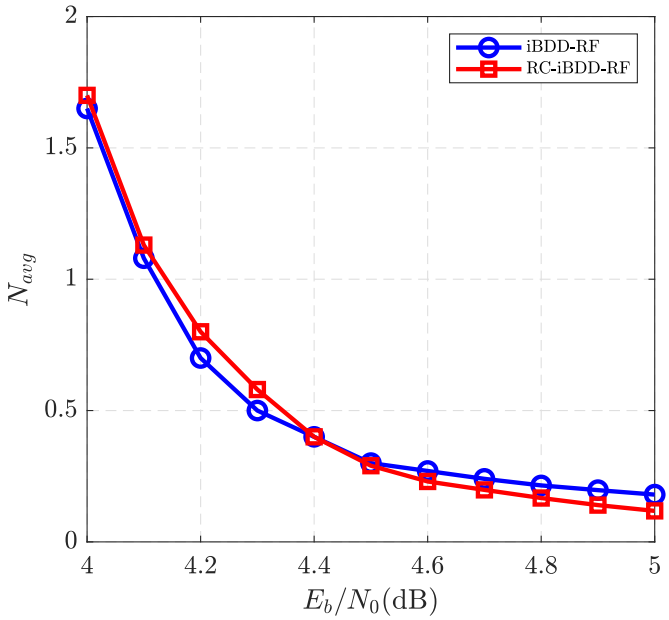



**Figure 4.** The $N_{\text{avg}}$ for RC-iBDD-RF and iBDD-RF, where $\mathscr{C}_1$ is selected as the component code.

### *4.3. Complexity Comparison*

In this subsection, we provide a detailed comparison of the additional memory requirements relative to iBDD and the computational complexities of RC-iBDD-RF, iBDD-RF, and iBDD. For clarity, Table 1 summarizes their additional memory overheads and the corresponding computational complexities, where $l_{\text{iBDD}}$ denotes the number of iterations for iBDD.

**Table 1.** Complexity comparison of RC-iBDD-RF, iBDD-RF, and iBDD.

| Algorithm | Additional Memory Required | Computational Complexity |
|---|---|---|
| RC-iBDD-RF | $(6l_{\text{RC}} + n^2)q + 3n$ | $\mathcal{O}(2l_{\text{RC}}E_{\text{avg}}n^3) + \mathcal{O}(2l_{\text{RC}}nN_{\text{avg}})$ |
| iBDD-RF | $(4l_{\text{RF}} + n^2)q$ | $\mathcal{O}(2l_{\text{RF}}E_{\text{avg}}n^3) + \mathcal{O}(2l_{\text{RF}}nN_{\text{avg}})$ |
| iBDD | 0 | $\mathcal{O}(2l_{\text{iBDD}}E_{\text{avg}}n^3)$ |

In the following, we illustrate the complexity advantage of RC-iBDD-RF through a practical example. The code $\mathscr{C}_1$ is selected as the component code. We consider the SNR $E_b/N_0 = 5.0$ dB. The number of decoding iterations is set as $l_{\text{RC}} = l_{\text{RF}} = l_{\text{iBDD}} = 12$. Similar to [30], BDD is performed at the last two iterations of iBDD-RF.

- From Figures 3a and 4, we have $E_{\text{avg}} = 0.105$ and $N_{\text{avg}} = 0.155$. Substituting these values into the expression in Table 1, the computational complexity of RC-iBDD-RF can be estimated as $\mathcal{O}(2.52n^3) + \mathcal{O}(3.72n)$.
- From Figures 3a and 4, we have $E_{\text{avg}} = 0.243$ and $N_{\text{avg}} = 0.193$. Substituting these values into the expression in Table 1, the computational complexity of iBDD-RF can be estimated as $\mathcal{O}(5.832n^3) + \mathcal{O}(3.86n)$.
- From Figure 3a, we have $E_{\text{avg}} = 0.081$. Substituting this value into the expressions in Table 1, the computational complexity of iBDD can be estimated as $\mathcal{O}(1.944n^3)$.

It can be seen from this example that the proposed algorithm achieves a 56.79% reduction in computational complexity when compared with iBDD-RF. Compared with iBDD, the complexity of RC-iBDD-RF increases by about 29.63%. From an implementation perspective, RC-iBDD-RF still preserves the standard row-column BDD decoding structure

of iBDD-based decoders. Therefore, the main BDD modules can be reused. Meanwhile, the extra memory access is mainly related to the quantized channel LLRs and the stored decoding states, which can be scheduled together with the row and column decoding process. Moreover, since the EaED module uses fixed filling patterns and is applied only once as post-processing, it does not introduce additional global decoding iterations. Hence, RC-iBDD-RF maintains an implementation-friendly structure for high-speed communication systems.

## 5. Performance Comparison

Extensive simulation results are presented to demonstrate the performance advantages of RC-iBDD-RF.

### *5.1. Parameter Optimization*

The scaling factors and the flipping probability $p_f^{(l)}$ in RC-iBDD-RF are all related to the number of iterations, which makes the parameter optimization of RC-iBDD-RF complex. However, no suitable theoretical method is currently available for parameter optimization. Therefore, following the dynamic reliability score decoder (DRSD) [32] and iBDD-RF [30], we employ the hyperparameter optimization tool *optuna* [33] to optimize the parameters of RC-iBDD-RF.

The parameters of RC-iBDD-RF include $w^{(l)}$, $\beta^{(l)}$, $p_f^{(l)}$, $T$, and $T_{\text{era}}$. We set the objective function as the required $E_b/N_0$ at a target bit error rate (BER) of $10^{-3}$. Specifically, the parameter set suggested by *Optuna* is used as the input, and the corresponding $E_b/N_0$ achieving the target BER is obtained as the output. This value is then fed back to *Optuna*, which iteratively samples and searches for the optimal hyperparameter combination based on previous results.

In this paper, we set the maximum number of decoding iterations to $l_{\max} = 12$. The search ranges of the optimized parameters are $w^{(l)} \in [3, 10]$, $\beta^{(l)} \in (0, 1)$, $p_f^{(l)} \in (0, 1)$, $T \in [0, 5]$, and $T_{\text{era}} \in [0, 5]$. The optimization is performed with a maximum budget of 100 trials and terminates when the predefined trial budget is exhausted, after which the final optimized parameter set is obtained. As an example, for the product code based on $\mathscr{C}_1[255, 231, 3]$, the optimized parameter set obtained by *Optuna* is given as follows:

$$\begin{aligned}
\boldsymbol{w} = \big[&3.4790,\ 9.0176,\ 4.2010,\ 5.2347,\ 7.8923,\ 3.9892,\\
&4.0141,\ 5.7246,\ 7.3568,\ 5.9619,\ 4.3605,\ 4.8770\big],\\
\boldsymbol{\beta} = \big[&0.7302,\ 0.4019,\ 0.8285,\ 0.4976,\ 0.7970,\ 0.6120,\\
&0.6074,\ 0.6438,\ 0.8120,\ 0.7579,\ 0.9121,\ 0.4449\big],\\
\boldsymbol{p}_f = \big[&0.5329,\ 0.4829,\ 0.0301,\ 0.3982,\ 0.9523,\ 0.4490,\\
&0.2613,\ 0.1214,\ 0.7795,\ 0.2944,\ 0.4397,\ 0.1018\big].
\end{aligned}$$

with

$$T = 0.9336, \qquad T_{\text{era}} = 1.2495.$$

### *5.2. The Influence of EaED*

In this subsection, we investigate the impact of EaED on the performance of RC-iBDD-RF, as well as on the execution number of BDD. We select the BCH code $\mathscr{C}_1[255, 231, 3]$ as the component code and employ the hyperparameter optimization tool *optuna* to obtain all parameters of RC-iBDD-RF. The number of iterations is set to 12.

In Figure 5, we present the performance of RC-iBDD-RF with and without EaED. It can be seen that at BER $= 10^{-6}$, RC-iBDD-RF with EaED achieves an approximately 0.03 dB gain over RC-iBDD-RF without EaED. To further investigate its impact on complexity, we

present simulation-based comparisons of the average execution numbers of BDD for the RC-iBDD-RF with and without EaED in the following. In Figure 6, we present $E_{\text{avg}}$ for RC-iBDD-RF without EaED and RC-iBDD-RF with EaED. The BCH code $\mathscr{C}_1[255, 231, 3]$ is selected as the component code. From Figure 6, we observe that RC-iBDD-RF without EaED and RC-iBDD-RF with EaED admit almost the same average execution numbers of BDD. This is due to the fact that, in RC-iBDD-RF, bit erasures are applied only once and are controlled by the threshold $T_{\text{era}}$. These results indicate that RC-iBDD-RF with EaED has almost the same decoding complexity as the RC-iBDD-RF without EaED, while achieving superior performance.

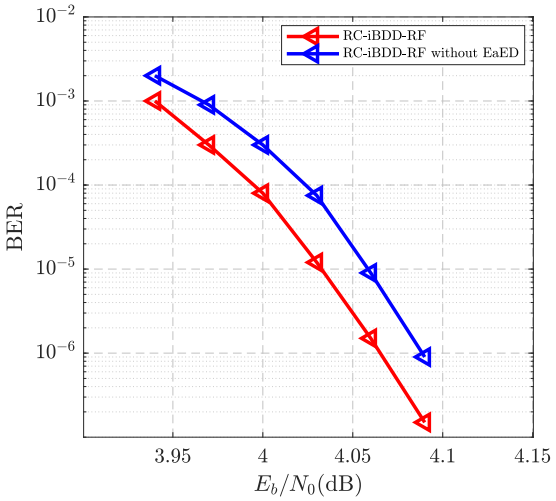



**Figure 5.** The performances of RC-iBDD-RF with and without EaED. The code $\mathscr{C}_1$ is selected as the component code.

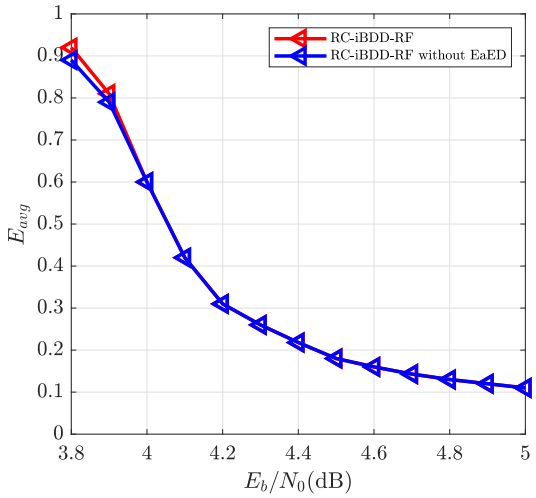



**Figure 6.** The average execution numbers of BDD for RC-iBDD-RF with and without EaED. The code $\mathscr{C}_1$ is selected as the component code.

We point out that the EaED post-processing contributes only a small additional gain, about 0.03 dB at BER $= 10^{-6}$ in the considered simulation. Therefore, EaED should be regarded as an optional residual-error cleanup module rather than the main source of the coding gain. Since the fixed filling patterns are applied only once after the iterative decoding process, this module does not introduce extra global decoding iterations or random pattern search. Thus, it may be enabled in low-BER-oriented applications where a small deterministic post-processing cost is acceptable, while it can be disabled in systems with more stringent latency or complexity constraints.

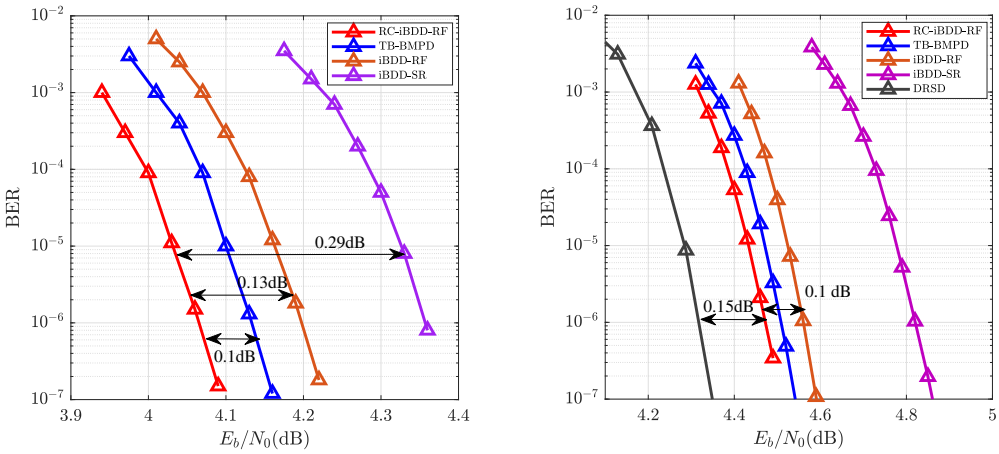
### *5.3. Performance Comparison for RC-iBDD-RF*

In this subsection, we present the performance of RC-iBDD-RF over the AWGN channel and compare it with several existing SA-HDDs, including iBDD-RF, iBDD-SR, the threshold-based binary messaging-passing decoder (TB-BMPD), and DRSD.

We select the representative high-rate BCH codes $\mathscr{C}_1[255, 231, 3]$ and $\mathscr{C}_2[256, 239, 2]$ [31,34,35] as component codes and set the number of iterations to 12. For TB-BMPD, in addition to ten algorithm iterations, two extra BDD iterations are required. For DRSD, only 10 iterations are performed, and the component codes are chosen as $\mathscr{C}_1$ and $\mathscr{C}_3[255, 238, 2]$. We point out that, since the code rates of $\mathscr{C}_2$ and $\mathscr{C}_3$ are very close, the difference can be neglected. The performances of various SA-HDDs are shown in Figure 7a,b. Based on these two figures, we have the following observations.

- For the PC based on $\mathscr{C}_1$, RC-iBDD-RF outperforms the iBDD-SR, iBDD-RF, and TB-BMPD. Specifically, its performance is approximately 0.29 dB better when compared with iBDD-SR, 0.13 dB better when compared with iBDD-RF, and 0.1 dB better when compared with TB-BMPD.
- For the PC based on $\mathscr{C}_2$, RC-iBDD-RF achieves about 0.1 dB performance gain over iBDD-RF, while it is only 0.15 dB away from DRSD. Note that it was shown in [30] that the decoding complexity of DRSD is higher than that of iBDD-RF. This indicates that the decoding complexity of RC-iBDD-RF is much lower than that of DRSD.

Within the simulated BER range in Figure 7, no obvious error floor is observed for RC-iBDD-RF. The low-BER gain suggests that the proposed reliability metric and the EaED-based post-processing can reduce part of the residual errors after iterative BDD. However, as with other iBDD-based decoders, rare decoding failures may still occur at very low BER due to stall patterns or miscorrection-induced residual patterns. The fixed filling patterns in EaED provide deterministic low-complexity post-processing and avoid random pattern search, but they may not cover all possible rare events in the deep error-floor region. A complete error-floor characterization would require further importance-sampling simulations or analytical stall-pattern analysis, which is an interesting topic for future research.




(**a**) Based on $\mathscr{C}_1$ (**b**) Based on $\mathscr{C}_2$

**Figure 7.** Performance comparison between RC-iBDD-RF and various SA-HDDs.

### *5.4. Complexity-Performance Trade-Off Analysis*

To further provide a clearer and fairer comparison among different decoders, we introduce a complexity-storage-weighted SNR metric inspired by the latency-weighted decoding metric in [36]. In [36], the decoding performance is evaluated by jointly considering the error-rate performance and the normalized decoding delay. Since this work focuses on algorithm-level decoding complexity and memory storage rather than hardware-

measured latency, we replace the normalized decoding delay with two normalized cost factors: normalized computational complexity and normalized storage complexity.

For decoder $d$, the normalized computational complexity is defined as

$$C_d^{\text{norm}} = \frac{C_d}{C_{\text{iBDD-}RF}}, \tag{18}$$

where $C_d$ denotes the computational complexity of decoder $d$, and $C_{\text{iBDD-}RF}$ denotes the computational complexity of iBDD-RF. Similarly, the normalized storage complexity is defined as

$$M_d^{\text{norm}} = \frac{M_d}{M_{\text{iBDD-}RF}}, \tag{19}$$

where $M_d$ denotes the memory requirement of decoder $d$.

The normalized required SNR at a target $\text{BER}_0$ is defined as

$$\Gamma_d(\text{BER}_0) = \frac{10^{\gamma_d(\text{BER}_0)/10}}{10^{\gamma_{\text{iBDD-}RF}(\text{BER}_0)/10}} = 10^{(\gamma_d(\text{BER}_0) - \gamma_{\text{iBDD-}RF}(\text{BER}_0))/10}, \tag{20}$$

where

$$\gamma_d(\text{BER}_0) = \left.\frac{E_b}{N_0}\right|_{\text{BER}=\text{BER}_0} \tag{21}$$

denotes the required $E_b/N_0$ for decoder $d$ to achieve the target $\text{BER}_0$.

The complexity-storage-weighted SNR is defined as

$$\Omega_d(\text{BER}_0) = \Gamma_d(\text{BER}_0) \cdot C_d^{\text{norm}} \cdot M_d^{\text{norm}}. \tag{22}$$

A smaller $\Omega_d(\text{BER}_0)$ indicates a better trade-off between decoding performance, computational complexity, and memory storage.

For the product code based on $\mathscr{C}_1[255, 231, 3]$, the memory requirements of RC-iBDD-RF and iBDD-RF are $65,097q + 765$ bits and $65,073q$ bits, respectively. Without loss of generality, we set $q = 5$. The computational complexity of RC-iBDD-RF and iBDD-RF are $\mathcal{O}(2.52n^3) + \mathcal{O}(3.72n)$ and $\mathcal{O}(5.832n^3) + \mathcal{O}(3.86n)$, respectively. At the target BER $\text{BER}_0 = 10^{-6}$, the required $E_b/N_0$ of RC-iBDD-RF and iBDD-RF are 4.06 dB and 4.20 dB, respectively, according to Figure 7a.

Accordingly, the complexity-storage-weighted SNR is calculated as

$$\begin{aligned}\Omega_{\text{RC-iBDD-RF}}(10^{-6}) &= \Gamma_{\text{RC-iBDD-RF}} \cdot C_{\text{RC-iBDD-RF}}^{\text{norm}} \cdot M_{\text{RC-iBDD-RF}}^{\text{norm}} \\ &= 0.4195.\end{aligned} \tag{23}$$

Since the metric is normalized to iBDD-RF, the corresponding value of iBDD-RF is 1. RC-iBDD-RF achieves a value smaller than 1, which indicates a better trade-off between decoding performance, computational complexity, and memory storage.

## 6. Conclusions

In this paper, we proposed the RC-iBDD-RF, a reduced-complexity random flipping decoder for product codes aided by erasure. We first presented an enhanced reliability metric. We then proposed to use EaED as a post-processing procedure to handle the residual errors. The complexity of the proposed decoder was investigated in terms of both memory storage and computational complexity. The analysis showed that the computational complexity of RC-iBDD-RF is lower than that of iBDD-RF. We also presented extensive numerical results to show the performance advantages of RC-iBDD-RF over iBDD-RF. Moreover, the proposed complexity-storage-weighted SNR metric further demonstrates that RC-iBDD-RF achieves a more favorable trade-off between decoding performance, computational

complexity, and memory storage. These results confirm the potential of RC-iBDD-RF for high-speed communication systems.

**Author Contributions:** Conceptualization, G.S., D.P. and S.Z.; methodology, G.S., D.P. and S.Z.; soft ware, G.S. and D.P.; validation, G.S. and D.P.; formal analysis, G.S. and D.P.; investigation, G.S.; resources, G.S., D.P., and S.Z.; data curation, G.S.; writing—original draft preparation, G.S. and D.P.; writing—review and editing, G.S., D.P., and S.Z.; visualization, G.S. and D.P.; supervision, S.Z.; project administration, S.Z.; funding acquisition, S.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2025B1515020027 and in part by the NSF of China under Grant 62571217 and Grant 62271233.

**Data Availability Statement:** The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Pyndiah, R. Near-optimum decoding of product codes: Block turbo codes. *IEEE Trans. Commun.* **1998**, *46*, 1003–1010. [CrossRef]
2. Qiu, M.; Yang, L.; Xie, Y.; Yuan, J. Terminated Staircase Codes for NAND Flash Memories. *IEEE Trans. Commun.* **2018**, *66*, 5861–5875. [CrossRef]
3. Chang, F.; Onohara, K.; Mizuochi, T. Forward error correction for 100 G transport networks. *IEEE Commun. Mag.* **2010**, *48*, S48–S55. [CrossRef]
4. Jose, R.; Pe, A. Analysis of hard decision and soft decision decoding algorithms of LDPC codes in AWGN. In Proceedings of the 2015 IEEE International Advance Computing Conference, Bangalore, India, 12–13 June 2015; Volume 60, pp. 430–435. [CrossRef]
5. Anguita, J.A.; Cisternas, J.E. Algorithmic decoding of dense OAM signal constellations for optical communications in turbulence. *Opt. Express* **2022**, *30*, 13540–13555. [CrossRef] [PubMed]
6. Wu, K.; Wang, H.; Ji, Y. Channel characteristics estimation based on a secure optical transmission system with deep neural networks. *Opt. Express* **2022**, *30*, 32391–32410. [CrossRef] [PubMed]
7. Smith, B.P.; Farhood, A.; Hunt, A.; Kschischang, F.R.; Lodge, J. Staircase Codes: FEC for 100 Gb/s OTN. *J. Light. Technol.* **2012**, *30*, 110–117. [CrossRef]
8. Hu, G.; Sha, J.; Wang, Z. Beyond 100Gbps Encoder Design for Staircase Codes. In Proceedings of the 2016 IEEE International Workshop on Signal Processing Systems, Dallas, TX, USA, 26–28 October 2016; Volume 4, pp. 154–158. [CrossRef]
9. Li, Y.; Yan, C.; Tian, F.; Xin, X.; Zhou, J.; Zhang, Q.; Gao, R.; Shi, H.; Yao, H.; Tian, Q.; et al. Experimental demonstration of the performance of net-coupled staircase code in 240-Gbps optical coherent transmission system. *Opt. Express* **2025**, *33*, 53345–53357. [CrossRef] [PubMed]
10. Sukmadji, A.Y.; Martínez-Peñas, U.; Kschischang, F.R. Zipper Codes. *J. Light. Technol.* **2022**, *40*, 6397–6407. [CrossRef]
11. Zhao, X.; Zhao, S.; Ma, X. A Class of Tiled Diagonal Zipper Codes with Multiple Chains. *IEEE Trans. Commun.* **2022**, *70*, 5004–5017. [CrossRef]
12. Li, M.; Zhou, C.; Yuan, R.; Xu, R.; Zhu, M.; Bai, B. Nonbinary Zipper Codes Based on Reed-Solomon Codes. In Proceedings of the 2023 International Conference on Wireless Communications and Signal Processing, Hangzhou, China, 2–4 November 2023; pp. 311–316. [CrossRef]
13. Sukmadji, A.Y.; Kschischang, F.R.; Shehadeh, M. Generalized Spatially-Coupled Product-Like Codes Using Zipper Codes with Irregular Degree. In Proceedings of the 2023 IEEE Globecom Workshops, Kuala Lumpur, Malaysia, 4–8 December 2023; pp. 1560–1565. [CrossRef]
14. Shehadeh, M.; Kschischang, F.R.; Sukmadji, A.Y. Generalized Staircase Codes with Arbitrary Bit Degree. In Proceedings of the Optical Fiber Communication Conference 2024, San Diego, CA, USA, 24–28 March 2024; p. W4C.2. [CrossRef]
15. Shehadeh, M.; Kschischang, F.R. Higher-Order Staircase Codes: A Unified Generalization of High-Throughput Coding Techniques. In Proceedings of the 2025 IEEE International Symposium on Information Theory, Ann Arbor, MI, USA, 22–27 June 2025; pp. 1–6. [CrossRef]
16. Hadi, A.; Isnaini, U.; Wijayanti, I.E.; Frederic Ezerman, M. On Signal Constellations Over Eisenstein Integers. *IEEE Trans. Inf. Theory* **2025**, *71*, 6801–6819. [CrossRef]
17. Souza, J.G.F.; Costa, S.I.R.; Ling, C. Multilevel Lattice Codes From Hurwitz Quaternion Integers. *IEEE Trans. Inf. Theory* **2025**, *71*, 8280–8293. [CrossRef]

18. Sajjad, M.; Shah, T.; Serna, R.J.; Suárez Aguilar, Z.E.; Delgado, O.S. Fundamental Results of Cyclic Codes over Octonion Integers and Their Decoding Algorithm. *Computation* **2022**, *10*, 219. [CrossRef]
19. Justesen, J.; Larsen, K.J.; Pedersen, L.A. Error correcting coding for OTN. *IEEE Commun. Mag.* **2010**, *48*, 70–75. [CrossRef]
20. Sheikh, A.; Graell i Amat, A.; Liva, G.; Hager, C.; Pfister, H.D. On Low-Complexity Decoding of Product Codes for High-Throughput Fiber-Optic Systems. In Proceedings of the 2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing, Hong Kong, China, 3–7 December 2018; Volume 73, pp. 1–5. [CrossRef]
21. Chase, D. Class of algorithms for decoding block codes with channel measurement information. *IEEE Trans. Inf. Theory* **1972**, *18*, 170–182. [CrossRef]
22. Sheikh, A.; Graell i Amat, A.; Liva, G. Iterative Bounded Distance Decoding of Product Codes with Scaled Reliability. In Proceedings of the 2018 European Conference on Optical Communication, Rome, Italy, 23–27 September 2018; pp. 1–3. [CrossRef]
23. Sheikh, A.; Graell i Amat, A.; Liva, G.; Alvarado, A. On parameter optimization of product codes for iterative bounded distance decoding with scaled reliability. In Proceedings of the 45th European Conference on Optical Communication, Dublin, Ireland, 22–26 September 2019; pp. 1–4. [CrossRef]
24. Liga, G.; Sheikh, A.; Alvarado, A. A novel soft-aided bit-marking decoder for product codes. In Proceedings of the 45th European Conference on Optical Communication, Dublin, Ireland, 22–26 September 2019; pp. 1–4. [CrossRef]
25. Sheikh, A.; Graell i Amat, A.; Liva, G. Binary Message Passing Decoding of Product-Like Codes. *IEEE Trans. Commun.* **2019**, *67*, 8167–8178. [CrossRef]
26. Lei, Y.; Chen, B.; Liga, G.; Balatsoukas-Stimming, A.; Sun, K.; Alvarado, A. A Soft-Aided Staircase Decoder Using Three-Level Channel Reliabilities. *J. Light. Technol.* **2021**, *39*, 6191–6203. [CrossRef]
27. Sheikh, A.; Graell i Amat, A.; Liva, G.; Alvarado, A. Refined Reliability Combining for Binary Message Passing Decoding of Product Codes. *J. Light. Technol.* **2021**, *39*, 4958–4973. [CrossRef]
28. Zhao, X.; Zhao, S.; Li, Z. Enhanced Anchor Decoder for Staircase Codes With Hard Reliability Scores. *IEEE Commun. Lett.* **2022**, *26*, 2826–2830. [CrossRef]
29. Xu, Y.; Jiang, M.; Zhu, M.; Hu, N.; Zhao, C.; Wang, J. Reduced-Complexity Decoding of 3D Product Codes for Satellite Communications. *Space Sci. Technol.* **2024**, *4*, 0096. [CrossRef]
30. Li, G.; Wang, S.; Zhao, S. Iterative Bounded Distance Decoding With Random Flipping for Product-Like Codes. *IEEE Trans. Commun.* **2025**, *73*, 2864–2875. [CrossRef]
31. Elias, P. Error-free Coding. *Trans. IRE Prof. Group Inf. Theory* **1954**, *4*, 29–37. [CrossRef]
32. Miao, S.; Rapp, L.; Schmalen, L. Improved soft-aided decoding of product codes with dynamic reliability scores. *J. Light. Technol.* **2022**, *40*, 7279–7288. [CrossRef]
33. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A Next-generation Hyperparameter Optimization Framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 2623–2631. [CrossRef]
34. Häger, C.; Pfister, H.D. Approaching Miscorrection-Free Performance of Product Codes With Anchor Decoding. *IEEE Trans. Commun.* **2018**, *66*, 2797–2808. [CrossRef]
35. Rapp, L.; Schmalen, L. Error-and-Erasure Decoding of Product and Staircase Codes. *IEEE Trans. Commun.* **2022**, *70*, 32–44. [CrossRef]
36. Miuccio, L.; Panno, D.; Riolo, S. A Flexible Encoding/Decoding Procedure for 6G SCMA Wireless Networks via Adversarial Machine Learning Techniques. *IEEE Trans. Veh. Technol.* **2023**, *72*, 3288–3303. [CrossRef]