Efficient Ordered Statistics Decoding of BCH Codes without Gaussian Elimination

Lijia Yang, *Student Member, IEEE,* Jianguo Zhao, *Student Member, IEEE,* Xihao Li, *Student Member, IEEE,* Li Chen, *Senior Member, IEEE,* Huazi Zhang, *Senior Member, IEEE,* and Jiajie Tong, *Member, IEEE*

***Abstract*—Ordered statistics decoding (OSD) can achieve near maximum likelihood (ML) decoding performance for BCH codes. However, Gaussian elimination (GE) that delivers the systematic generator matrix of the code has an uncompromised latency. Addressing this challenge, this paper proposes a low-latency OSD (LLOSD) for BCH codes. Since BCH codes are binary subcodes of Reed-Solomon (RS) codes, codeword candidates can be produced using the RS systematic generator matrix, whose entries can be generated in parallel. By eliminating the non-binary codeword candidates and identifying the ML codeword, the LLOSD yields a lower latency as well as complexity than the OSD. It is shown that the LLOSD can be interpreted as generating the codeoword candidates through systematic encoding of a punctured BCH codeword, explaining its low-complexity feature. Moreover, the segmented variant is proposed to further facilitate the LLOSD. In order to decode long BCH codes, a hybrid soft decoding (HSD) is finally proposed. It integrates the LLOSD and the algebraic Chase decoding that can effectively provide extra TEPs for the LLOSD, enhancing the decoding performance. Both the complexity and performance of the proposed decoding are analyzed, demonstrating their advantage over the relevant state-of-the-art decoding.**

***Index Terms*—Algebraic Chase decoding, BCH codes, basis reduction, ordered statistics decoding, subfield subcode**

I. Introduction

Competent short-to-medium length channel codes play a vital role in realizing ultra-reliable low-latency communications (URLLC). Bose-Chaudhuri-Hocquenghem (BCH) codes have a good algebraic structure and distance property, being one of the best performing short-to-medium length codes. Recent research [1]–[3] has shown that ordered statistics decoding (OSD) of BCH codes can achieve a performance close to the normal approximation (NA) bound [4].

This article was presented in part at the IEEE Information Theory Workshop (ITW), 2022.

Lijia Yang is with the School of Electronics and Communication Engineering, Sun Yat-sen University, Shenzhen 518107, China (e-mail: yanglj39@mail2.sysu.edu.cn).

Jianguo Zhao is with the School of System Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China (e-mail: zhaojg5@mail2.sysu.edu.cn).

Xihao Li is with the School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China (e-mail: lixh98@mail2.sysu.edu.cn).

Li Chen is with the School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China, and also with Guangdong Province Key Laboratory of Information Security Technology, Guangzhou 510006, China (e-mail: chenli55@mail.sysu.edu.cn).

Huazi Zhang and Jiajie Tong is with the Hangzhou Research Center, Huawei Technologies Company Ltd., Hangzhou, China (e-mail: zhanghuazi@huawei.com, tongjiajie@huawei.com).

The OSD was proposed as a near maximum likelihood (ML) decoding approach for linear block codes [5]. With the received symbols, the most reliable independent positions (MRIPs) are identified. The codeword candidates are estimated through re-encoding the test messages that are formed by superimposing the test error patterns (TEPs) onto the MRIP decisions. The re-encoding requires the systematic generator matrix of the code, which is yielded by Gaussian elimination (GE). In the systematic generator matrix, columns that correspond to the MRIPs form an identity submatrix. The decoding is parameterized by order τ, which indicates the maximum Hamming weight of the TEPs. For a BCH code with a length of n and a dimension of k, the OSD exhibits a complexity of $O(k^\tau + n\min\{k, n-k\}^2)$, where the first and second terms correspond to the complexity of re-encoding and GE, respectively. In order to reduce the decoding complexity, several skipping rules that help identify the unpromising TEPs were proposed [6] [7]. Meanwhile, the ML codeword can be identified during the decoding so that it can be terminated earlier [8] [9]. This approach has been widely adopted to facilitate the OSD. The OSD complexity can also be reduced by segmenting the TEPs, which eliminates the redundant re-encoding attempts [10]. Alternatively, the reduced decoding complexity can be exchanged by enhancing its decoding performance. E.g., the box-and-match algorithm (BMA) stores a band of the re-encoded symbols outside the MRIPs and their corresponding TEPs. They will be utilized in the search-and-match process to generate extra codeword candidates, enhancing the OSD performance [11]. Other OSD performance improvement approaches include the use of multiple MRIPs bases that expand the volume of TEPs [12]–[14]. The hard reliability-based OSD [15] is further proposed for scenarios in which the soft received information is unavailable, e.g., the McEliece public key cryptosystem. Despite the competent decoding performance, the latency of OSD remains challenging to overcome. This is primarily caused by GE, which is a sequential process. So far, it has only been addressed by pre-computing some systematic generator matrices [16]–[18]. In the schemes of [16] and [17], GE will only be incurred if the desired codeword cannot be produced by the pre-computed matrices. In [18], GE is not required as it only utilizes a pre-computed systematic generator matrix for re-encoding. The potential TEPs are enumerated according to weight orders defined by the guessing random additive noise decoding (GRAND) [19] methods. However, since the re-encoding is not based on the MRIPs, its decoding performance

degrades from the conventional OSD.

Since BCH codes are binary subcodes of Reed-Solomon (RS) codes, the RS decoding techniques can be applied for BCH codes. E.g., in practice, the Berlekamp-Massey (BM) algorithm [20] has been widely applied for decoding BCH codes. It can correct errors up to half of the code's minimum Hamming distance. The interpolation-based Guruswami-Sudan (GS) algorithm [21] can correct errors beyond this limit. Utilizing the soft information, its soft-decision variants include the Kötter-Vardy (KV) [22] and the low-complexity Chase (LCC) decoding algorithms [23]. In contrast to the OSD, Chase decoding utilizes the complementary reliability information, i.e., those of the least reliable positions (LRPs). There exist various approaches to facilitate LCC decoding [24] [25]. Their simplifications for decoding BCH codes were reported in [25] [26]. Interpolation of LCC decoding can be realized using the basis reduction (BR) technique, formulating the LCC-BR decoding [27]. It yields both complexity and latency advantages over the LCC decoding that is realized by Kötter's interpolation. The OSD and Chase decoding have been combined in [28] [29], where Chase decoding enables a reduced OSD order for yielding the near ML decoding performance. However, this remains a rigid integration in which Chase decoding and the OSD cannot exchange or share their computation.

Although the OSD can yield a near ML decoding performance for BCH codes, its decoding complexity remains exponential in its decoding order. Meanwhile, its uncompromised GE latency remains unsolved. This limits its applications, especially for decoding longer BCH codes. Addressing them, this paper proposes a low-latency OSD (LLOSD), which can effectively reduce the decoding latency. It is designed based on the property that BCH codes are binary subcodes of RS codes. Hence, the BCH codeword candidates can be produced through generating binary RS codewords, eliminating the need for GE. By identifying the ML codeword candidate and terminating the decoding earlier, this new decoding can also yield a low decoding complexity. It is shown that the proposed LLOSD can be interpreted as a serial concatenation between the parity-checker of a punctured BCH code and a systematic encoder of a binary linear block code. It helps explain the low-complexity feature of LLOSD. A segmented variant is further proposed to facilitate the LLOSD. The RS code algebra also enables its integration with the LCC-BR decoding, forming the hybrid soft decoding (HSD). It helps restrain the LLOSD order, being suitable for decoding long BCH codes. Major contributions of this work are summarized below:

- An LLOSD algorithm is proposed. It circumvents the latency-orienting GE. It is shown that the BCH codeword candidates can be produced through the RS systematic generator matrix. By identifying the most reliable positions (MRPs), the matrix can be constructed using the Lagrange interpolation polynomials in a fully parallel manner, enabling the low-latency feature of the decoding. By eliminating the redundant TEPs that produce non-binary codeword candidates and identifying the ML candidate, the LLOSD also yields a low decoding complexity.
- A concatenated interpretation is introduced for the LLOSD. It is shown that its re-encoding can be realized through concatenating parity-checker of a punctured BCH code and a systematic encoder of a binary linear block code. Such an interpretation can not only convert the non-binary re-encoding operations into binary, but also unveil that there are far fewer BCH codeword candidates than the decoding TEPs. The latter vindicates the low-complexity feature of LLOSD.
- A segmented variant of the LLOSD (SLLOSD) is also proposed to further reduce the decoding complexity. It is shown that by appropriately choosing decoding orders of the codeword segments, near ML decoding performance can be achieved. Our simulation results show the latency and complexity of LLOSD can be further reduced.
- In order to decode long BCH codes, an HSD algorithm is further proposed, which integrates the LLOSD and the LCC-BR decoding. This research shows that the latter can effectively provide extra TEPs for the LLOSD, enhancing its decoding performance with a limited decoding order. Note that both the LLOSD and the LCC-BR decoding are formulated based on the RS code algebra. Their computations share some common operations. Consequently, the supplementary LCC-BR decoding only incurs limited computational cost. Our simulation results indicate that the HSD algorithm can efficiently handle long BCH codes with rates ranging from 0.67 to 0.83. The HSD can also outperform several state-of-the-art decoding for BCH codes.

The rest of the paper is organized as follows. Section II introduces the preliminaries for the paper. Section III presents the LLOSD and its concatenated perspective. Section IV presents the segmented variant of the LLOSD. Section V further presents the HSD. Section VI analyzes their decoding complexity. Finally, Section VII concludes the paper.

II. Preliminaries

This section presents preliminaries of the paper, including the subfield subcode relationship between BCH and RS codes, and the OSD for BCH codes.

A. BCH Codes and RS Codes

Let $\mathbb{F}_2$ and $\mathbb{F}_{2^m}$ denote the binary field and its extension field, respectively, where m is a positive integer. Let α denote the primitive element of $\mathbb{F}_{2^m}$. Let $\mathbb{F}_2[x]$ and $\mathbb{F}_{2^m}[x]$ denote the univariate polynomial rings defined over $\mathbb{F}_2$ and $\mathbb{F}_{2^m}$, respectively. In our later introduction of HSD, the bivariate polynomial ring $\mathbb{F}_{2^m}[x, y]$ will also be needed. A binary primitive BCH code $\mathbb{C}_{\text{BCH}}$ of length $n = 2^m - 1$ and designed Hamming distance $d = 2t + 1$ is a cyclic code with the generator polynomial $g(x) \in \mathbb{F}_2[x]$ that satisfies

$$g(\alpha) = g(\alpha^2) = \cdots = g(\alpha^{2t}) = 0 \tag{1}$$

and $\deg g(x)$ shall be the minimum. Let k denote dimension of the BCH code. Given a message $\underline{\boldsymbol{f}} = (f_0, f_1, ..., f_{k-1}) \in \mathbb{F}_2^k$, the (n, k) BCH codeword $\underline{\boldsymbol{c}} = (c_0, c_1, ..., c_{n-1})$ can be generated by

$$c(x) = f(x)\, g(x), \tag{2}$$

where $c(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1} \in \mathbb{F}_2[x]$ and $f(x) = f_0 + f_1x + \cdots + f_{k-1}x^{k-1} \in \mathbb{F}_2[x]$ are the codeword polynomial and message polynomial, respectively. For any $\underline{\boldsymbol{c}} \in \mathbb{C}_{\text{BCH}}$, eqs. (1) and (2) imply that

$$c(\alpha) = c(\alpha^2) = \cdots = c(\alpha^{2t}) = 0. \tag{3}$$

An (n, k') RS code $\mathbb{C}_{\text{RS}}$ defined over $\mathbb{F}_{2^m}$ can be constructed by evaluating its message polynomials over the nonzero elements of $\mathbb{F}_{2^m}$, i.e., $\mathbb{F}_{2^m}\backslash\{0\}$. Given a message $\underline{\boldsymbol{u}} = (u_0, u_1, ..., u_{k'-1}) \in \mathbb{F}_{2^m}^{k'}$, the (n, k') RS codeword $\underline{\boldsymbol{v}} = (v_0, v_1, ..., v_{n-1})$ can be generated by

$$\underline{\boldsymbol{v}} = (u(1), u(\alpha), ..., u(\alpha^{n-1})), \tag{4}$$

where $u(x) = u_0 + u_1x + \cdots + u_{k'-1}x^{k'-1} \in \mathbb{F}_{2^m}[x]$ is the message polynomial, and $1, \alpha, \ldots, \alpha^{n-1}$ are the code locators [1]. The codeword polynomial is $v(x) = v_0 + v_1x + \cdots + v_{n-1}x^{n-1} \in \mathbb{F}_{2^m}[x]$. For any $\underline{\boldsymbol{v}} \in \mathbb{C}_{\text{RS}}$, it holds that

$$\begin{aligned} v(\alpha^i) &= u(1) + u(\alpha)\alpha^i + \cdots + u(\alpha^{n-1})(\alpha^i)^{n-1} \\ &= \sum_{j=0}^{k'-1} u_j + \sum_{j=0}^{k'-1} u_j\alpha^j \cdot \alpha^i \\ &\quad + \cdots + \sum_{j=0}^{k'-1} u_j(\alpha^{n-1})^j \cdot (\alpha^i)^{n-1} \\ &= \sum_{j=0}^{k'-1} u_j \left(1 + \alpha^{i+j} + \cdots + (\alpha^{i+j})^{n-1}\right) \end{aligned} \tag{5}$$

For $i = 1, 2, ..., n - k'$ and $j = 0, 1, ..., k' - 1$, $\alpha^{i+j} \neq 1$. Hence, $v(\alpha^i) = \sum_{j=0}^{k'-1} u_j \frac{1-\alpha^{n(i+j)}}{1-\alpha^{i+j}}$. Since $\alpha^n = \alpha^{2^m-1} = 1$,

$$v(\alpha) = v(\alpha^2) = \cdots = v(\alpha^{n-k'}) = 0. \tag{6}$$

Moreover, it has a minimum Hamming distance of $d' = n - k' + 1$. Hence, RS codes are maximum distance separable (MDS) codes.

Now it is sufficient to show that BCH codes are the binary subcodes of RS codes. For this, the subfield subcode definition needs to be introduced.

Definition I *([30]).* Given a linear code $\mathbb{C} \subseteq \mathbb{F}_{2^m}^n$, its binary subcode $\mathbb{C}'$ is defined as $\mathbb{C}' = \mathbb{C} \cap \mathbb{F}_2^n$.

Lemma 1 *([31]).* Given a BCH code $\mathbb{C}_{\text{BCH}} \subseteq \mathbb{F}_2^n$ with the designed Hamming distance d and an RS code $\mathbb{C}_{\text{RS}} \subseteq \mathbb{F}_{2^m}^n$ with the minimum Hamming distance d', if $d = d'$, the BCH code is the binary subcode of the RS code, i.e., $\mathbb{C}_{\text{BCH}} = \mathbb{C}_{\text{RS}} \cap \mathbb{F}_2^n$.

Proof. If $d = d'$, $n - k' = 2t$. For any BCH codeword $\underline{\boldsymbol{c}} \in \mathbb{C}_{\text{BCH}}$, let us define a polynomial $f'(x) = \sum_{j=0}^{n-1} f'_j x^j \in \mathbb{F}_{2^m}[x]$ such that $f'_j = c(\alpha^{-j})$. Then, for $i = 0, 1, ..., n - 1$,

$$\begin{aligned} f'(\alpha^i) &= c(1) + c(\alpha^{-1})\alpha^i + \cdots + c(\alpha^{1-n})(\alpha^i)^{n-1} \\ &= \sum_{j=0}^{n-1} c_j \left(1 + \alpha^{i-j} + \cdots + (\alpha^{i-j})^{n-1}\right). \end{aligned}$$

[1]Note that the codeword can be generated by evaluating the message polynomial at the code locators in an arbitrary order. For presentation convenience, this paper adopts the order shown in eq. (4).

Note that

$$1 + \alpha^{i-j} + \cdots + (\alpha^{i-j})^{n-1} = \begin{cases} 1, & \text{if } j = i; \\ \frac{1-\alpha^{n(i-j)}}{1-\alpha^{i-j}} = 0, & \text{otherwise.} \end{cases}$$

Hence, $f'(\alpha^i) = c_i$ for $i = 0, 1, ..., n-1$. Since $\alpha^{-j} = \alpha^{n-j}$, further based on eq. (3), $c(\alpha^{-j}) = 0$ for $j = k', k'+1, ..., n-1$ and $f'(x) = \sum_{j=0}^{k'-1} f'_j x^j$. Furthermore, based on eq. (4), $\underline{\boldsymbol{c}} \in \mathbb{C}_{\text{RS}}$. Hence, $\mathbb{C}_{\text{BCH}} \subseteq \mathbb{C}_{\text{RS}} \cap \mathbb{F}_2^n$.

For any binary RS codeword $\underline{\boldsymbol{v}} \in \mathbb{C}_{\text{RS}} \cap \mathbb{F}_2^n$, $v(x) \in \mathbb{F}_2[x]$ and it satisfies eq. (6). Since $g(x)$ is the minimal nonzero polynomial in $\mathbb{F}_2[x]$ that satisfies eq. (1), $v(x) \bmod g(x) = 0$. Furthermore, based on eq. (2), $\underline{\boldsymbol{v}} \in \mathbb{C}_{\text{BCH}}$. Hence, $\mathbb{C}_{\text{RS}} \cap \mathbb{F}_2^n \subseteq \mathbb{C}_{\text{BCH}}$. Therefore, $\mathbb{C}_{\text{BCH}} = \mathbb{C}_{\text{RS}} \cap \mathbb{F}_2^n$. ■

In the rest of this paper, let $d = d'$. Hence, the (n, k) BCH code is the binary subcode of the (n, k') RS code.

Remark 1. Since RS codes are MDS codes, the dimension of RS codes is always greater than that of their BCH subcodes, i.e., $k' > k$. Moreover, based on the Plotkin bound [30], binary BCH codes with dimension $k \geqslant m$ have a minimum distance of $d \leqslant 2^{m-1}$ and their corresponding RS codes have a dimension of $k' > \frac{n}{2}$.

B. OSD of BCH Codes

Assume that a BCH codeword $\underline{\boldsymbol{c}}$ is transmitted by the binary phase shift keying (BPSK) modulation as: $0 \mapsto 1$; $1 \mapsto -1$. The modulated symbol sequence is $\underline{\boldsymbol{\mathcal{X}}} = (\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_{n-1}) \in \{-1, 1\}^n$. After a memoryless channel, the received symbol sequence is $\underline{\boldsymbol{\mathcal{Y}}} = (\mathcal{Y}_0, \mathcal{Y}_1, \ldots, \mathcal{Y}_{n-1}) \in \mathbb{R}^n$. Let $\Pr(\mathcal{Y}_j \mid c_j = 0)$ and $\Pr(\mathcal{Y}_j \mid c_j = 1)$ denote channel transition probabilities of c_j, its log-likelihood ratio (LLR) can be determined by

$$L_j = \ln \frac{\Pr(\mathcal{Y}_j \mid c_j = 0)}{\Pr(\mathcal{Y}_j \mid c_j = 1)}. \tag{7}$$

Subsequently, the received LLR sequence can be obtained as $\underline{\boldsymbol{L}} = (L_0, L_1, \ldots, L_{n-1}) \in \mathbb{R}^n$. The hard-decision received word $\underline{\boldsymbol{r}} = (r_0, r_1, \ldots, r_{n-1}) \in \mathbb{F}_2^n$ can be further obtained based on $r_j = 0$ if $L_j > 0$, or $r_j = 1$ otherwise. Note that a greater $|L_j|$ indicates the received symbol r_j is more reliable. Hence, reliability of all received symbols can be sorted based on $|L_j|$, resulting in a refreshed symbol index sequence $j_0, j_1, \ldots, j_{n-1}$. It indicates $|L_{j_0}| > |L_{j_1}| > \cdots > |L_{j_{n-1}}|$. Let $\Lambda(\cdot)$ denote the permutation function that is a result of the above sorting, the sorted received word is

$$\Lambda\left(\underline{\boldsymbol{r}}\right) = \left(r_{j_0}, r_{j_1}, \ldots, r_{j_{n-1}}\right). \tag{8}$$

Applying the same permutation to the columns of the BCH generator matrix $\mathbf{G}$ yields $\Lambda(\mathbf{G})$. The systematic generator matrix $\mathbf{G}_{\text{BCH}}$ can be further obtained by performing the GE on $\Lambda(\mathbf{G})$, reducing the first k columns of $\Lambda(\mathbf{G})$ into a $k \times k$ identity submatrix. Note that if the first k columns of $\Lambda(\mathbf{G})$ are not linearly independent, a second permutation would be needed so that an identity submatrix can be formed. In this case, the sorted received word $\Lambda(\underline{\boldsymbol{r}})$ also needs to be updated accordingly. Without further mentioning, we assume that the first k columns of $\Lambda(\mathbf{G})$ have been ensured with this property through one or more permutations.

After ensuring the first k columns of $\Lambda(\mathbf{G})$ being linearly independent, the first k positions in $\Lambda(\underline{\boldsymbol{r}})$ are called the MRIPs, which are denoted as $\Upsilon = \{j_0, j_1, \ldots, j_{k-1}\}$. In this paper, given a vector $\underline{\boldsymbol{z}} = (z_0, z_1, \ldots, z_{n-1})$, we use $\underline{\boldsymbol{z}}_{[J]}$ to denote the subvector of $\underline{\boldsymbol{z}}$ with entries indexed by the set J. It is called the support of $\underline{\boldsymbol{z}}_{[J]}$, and denoted as $\mathrm{supp}(\underline{\boldsymbol{z}}_{[J]})$. That says $\mathrm{supp}(\underline{\boldsymbol{z}}_{[J]}) = J$. Consequently, the initial message can be generated as

$$\begin{aligned}\underline{\boldsymbol{f}} &= \underline{\boldsymbol{r}}_{[\Upsilon]} \\ &= \left(r_{j_0}, r_{j_1}, \ldots, r_{j_{k-1}}\right),\end{aligned} \tag{9}$$

where $\underline{\boldsymbol{f}} \in \mathbb{F}_2^k$. Let $\underline{\boldsymbol{e}}^{(\omega)} = (e_0^{(\omega)}, e_1^{(\omega)}, \ldots, e_{n-1}^{(\omega)}) \in \mathbb{F}_2^n$ denote an error pattern of length n. The OSD TEPs are the subvector of $\underline{\boldsymbol{e}}^{(\omega)}$ defined by the MRIPs, which are written as $\underline{\boldsymbol{e}}_{[\Upsilon]}^{(\omega)} = (e_{j_0}^{(\omega)}, e_{j_1}^{(\omega)}, \ldots, e_{j_{k-1}}^{(\omega)}) \in \mathbb{F}_2^k$. They are utilized to update $\underline{\boldsymbol{f}}$, yielding the test messages. The OSD is parameterized by its order τ that can be interpreted as the maximum Hamming weight of the TEPs. Let $\mathrm{wt}(\cdot)$ denote the Hamming weight function. It follows that $\mathrm{wt}(\underline{\boldsymbol{e}}_{[\Upsilon]}^{(\omega)}) \leq \tau$. Hence, with a decoding order τ, $\omega = 0, 1, \ldots, N_{\text{TEPs}} - 1$ and $N_{\text{TEPs}} = \sum_{\rho=0}^{\tau} \binom{k}{\rho}$ is the number of TEPs. With a TEP $\underline{\boldsymbol{e}}^{(\omega)}$, the test message is generated by

$$\underline{\boldsymbol{f}}^{(\omega)} = \underline{\boldsymbol{f}} + \underline{\boldsymbol{e}}_{[\Upsilon]}^{(\omega)}. \tag{10}$$

The BCH codeword $\hat{\underline{\boldsymbol{c}}}^{(\omega)} = (\hat{c}_0^{(\omega)}, \hat{c}_1^{(\omega)}, \ldots, \hat{c}_{n-1}^{(\omega)}) \in \mathbb{F}_2^n$ can be further generated by

$$\hat{\underline{\boldsymbol{c}}}^{(\omega)} = \Lambda^{-1}(\underline{\boldsymbol{f}}^{(\omega)} \cdot \mathbf{G}_{\text{BCH}}), \tag{11}$$

where $\Lambda^{-1}(\cdot)$ is inverse of the permutation function $\Lambda(\cdot)$.

It can be seen that an OSD with order τ produces $N_{\text{BCH}} = N_{\text{TEPs}} = \sum_{\rho=0}^{\tau} \binom{k}{\rho}$ codeword candidates, exhibiting a decoding complexity of $O(k^\tau)$. With the LLR sequence $\underline{\boldsymbol{L}}$, the correlation distance between $\underline{\boldsymbol{r}}$ and $\hat{\underline{\boldsymbol{c}}}^{(\omega)}$ is defined as

$$\mathcal{D}(\underline{\boldsymbol{r}}, \hat{\underline{\boldsymbol{c}}}^{(\omega)}) \triangleq \sum_{j: r_j \neq \hat{c}_j^{(\omega)}} |L_j|. \tag{12}$$

A codeword candidate that yields a smaller correlation distance to $\underline{\boldsymbol{r}}$ is more likely to be the transmitted codeword. Meanwhile, the OSD can be facilitated by identifying a codeword that satisfies the ML criterion [8] as below. Let $S_\omega = \{L_j | r_j = \hat{c}_j^{(\omega)}\}$, and its entries can be reordered as

$$|L_{\xi_0}| \leq |L_{\xi_1}| \leq \cdots \leq |L_{\xi_{n-d_\omega-1}}|, \tag{13}$$

where d_ω denotes the Hamming distance between $\underline{\boldsymbol{r}}$ and $\hat{\underline{\boldsymbol{c}}}^{(\omega)}$. If the estimated codeword $\hat{\underline{\boldsymbol{c}}}^{(\omega)}$ satisfies [8]

$$\mathcal{D}(\underline{\boldsymbol{r}}, \hat{\underline{\boldsymbol{c}}}^{(\omega)}) \leq \sum_{j=0}^{d-d_\omega-1} |L_{\xi_j}|, \tag{14}$$

it is the ML codeword. Once an ML codeword is identified, the decoding can be terminated. Otherwise, among the OSD output list, the one that yields the minimum correlation distance to $\underline{\boldsymbol{r}}$ will be selected as $\hat{\underline{\boldsymbol{c}}}_{\text{opt}}$.

The above description shows that the GE produces the systematic generator matrix $\mathbf{G}_{\text{BCH}}$ based on the MRIPs. It is indispensable for generating the codeword candidates. However, its sequential feature incurs an uncompromised decoding latency.

III. The LLOSD Algorithm

This section presents the LLOSD algorithm, in which the BCH codeword candidates are produced by using the RS systematic generator matrix. We first show the construction of this matrix, which exhibits a full parallelism, underpinning the low-latency feature of the proposed decoding. Afterwards, the concatenated interpretation of the LLOSD is introduced, also unveiling its low complexity features.

A. Construction of RS Systematic Generator Matrix

Given the sorted received word $\Lambda(\underline{\boldsymbol{r}})$ of eq. (8), its MRPs are denoted by $\Theta = \{j_0, j_1, \ldots, j_{k'-1}\}$. Its complementary set is $\Theta^c = \{j_{k'}, j_{k'+1}, \ldots, j_{n-1}\}$. The MDS property of RS codes ensures any k' columns of its generator matrix are linearly independent. With Θ, the initial message can be formed as

$$\begin{aligned}\underline{\boldsymbol{u}} &= \underline{\boldsymbol{r}}_{[\Theta]} \\ &= (r_{j_0}, r_{j_1}, \ldots, r_{j_{k'-1}}),\end{aligned} \tag{15}$$

where $\underline{\boldsymbol{u}} \in \mathbb{F}_2^{k'}$ and $\mathrm{supp}(\underline{\boldsymbol{u}}) = \Theta = \{j_0, j_1, \ldots, j_{k'-1}\}$. Subsequently, the systematic re-encoding message polynomial is defined as

$$H_{\underline{\boldsymbol{u}}}(x) = \sum_{j \in \mathrm{supp}(\underline{\boldsymbol{u}})} r_j T_j(x), \tag{16}$$

where

$$T_j(x) = \prod_{j' \in \mathrm{supp}(\underline{\boldsymbol{u}}), j' \neq j} \frac{x - \alpha^{j'}}{\alpha^j - \alpha^{j'}} \tag{17}$$

is the Lagrange interpolation polynomial w.r.t. code locator α^j. Note that for any $j \in \mathrm{supp}(\underline{\boldsymbol{u}})$, it enables $T_j(\alpha^j) = 1$, and $T_j(\alpha^{j'}) = 0$ where $j' \neq j$. With the code locators $1, \alpha, \ldots, \alpha^{n-1}$, the systematic RS codeword $\underline{\boldsymbol{v}} = (v_0, v_1, \ldots, v_{n-1}) \in \mathbb{F}_{2^m}^n$ can be generated by

$$\underline{\boldsymbol{v}} = (H_{\underline{\boldsymbol{u}}}(1), H_{\underline{\boldsymbol{u}}}(\alpha), \ldots, H_{\underline{\boldsymbol{u}}}(\alpha^{n-1})), \tag{18}$$

where $H_{\underline{\boldsymbol{u}}}(\alpha^j) = r_j$, $\forall j \in \mathrm{supp}(\underline{\boldsymbol{u}})$. They are the message symbols in $\underline{\boldsymbol{v}}$, while the remaining symbols indexed by Θ^c are the parity symbols.

In order to construct the RS systematic generator matrix, let us first define k' weight-1 messages as $\underline{\boldsymbol{u}}_{j_0} = (1, 0, \ldots, 0)$, $\underline{\boldsymbol{u}}_{j_1} = (0, 1, \ldots, 0)$, $\ldots$, $\underline{\boldsymbol{u}}_{j_{k'-1}} = (0, 0, \ldots, 1)$, respectively. They have the same support as $\underline{\boldsymbol{u}}$, i.e., $\mathrm{supp}(\underline{\boldsymbol{u}}_{j_0}) = \mathrm{supp}(\underline{\boldsymbol{u}}_{j_1}) = \cdots = \mathrm{supp}(\underline{\boldsymbol{u}}_{j_{k'-1}}) = \mathrm{supp}(\underline{\boldsymbol{u}}) = \Theta$. Consequently, a systematic generator matrix of the (n, k') RS code can be defined as

$$\mathbf{G}_{\text{RS}} = \begin{bmatrix} H_{\underline{\boldsymbol{u}}_{j_0}}(1) & H_{\underline{\boldsymbol{u}}_{j_0}}(\alpha) & \cdots & H_{\underline{\boldsymbol{u}}_{j_0}}(\alpha^{n-1}) \\ H_{\underline{\boldsymbol{u}}_{j_1}}(1) & H_{\underline{\boldsymbol{u}}_{j_1}}(\alpha) & \cdots & H_{\underline{\boldsymbol{u}}_{j_1}}(\alpha^{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ H_{\underline{\boldsymbol{u}}_{j_{k'-1}}}(1) & H_{\underline{\boldsymbol{u}}_{j_{k'-1}}}(\alpha) & \cdots & H_{\underline{\boldsymbol{u}}_{j_{k'-1}}}(\alpha^{n-1}) \end{bmatrix}, \tag{19}$$

where $\mathbf{G}_{\text{RS}} \in \mathbb{F}_{2^m}^{k' \times n}$ and each row of $\mathbf{G}_{\text{RS}}$ is a systematic RS codeword. They correspond to messages $\underline{\boldsymbol{u}}_{j_0}, \underline{\boldsymbol{u}}_{j_1}, \ldots, \underline{\boldsymbol{u}}_{j_{k'-1}}$, respectively. Since the k' messages are linearly independent, so are the k' codewords. Meanwhile, columns $j_0, j_1, \ldots, j_{k'-1}$ form a $k' \times k'$ identity submatrix. With $\underline{\boldsymbol{u}}_{j_0}, \underline{\boldsymbol{u}}_{j_1}, \ldots, \underline{\boldsymbol{u}}_{j_{k'-1}}$, entries of $\mathbf{G}_{\text{RS}}$ can be computed as

$$H_{\underline{\boldsymbol{u}}_i}(\alpha^j) = \begin{cases} 0, & \text{if } j \in \Theta, j \neq i; \\ 1, & \text{if } j \in \Theta, j = i; \\ T_i\left(\alpha^j\right), & \text{if } j \in \Theta^c, \end{cases} \tag{20}$$

where

$$\begin{aligned} T_i\left(\alpha^j\right) &= \prod_{j' \in \Theta, j' \neq i} \frac{\alpha^j - \alpha^{j'}}{\alpha^i - \alpha^{j'}} \\ &= \frac{\prod_{j' \in \Theta}\left(\alpha^j - \alpha^{j'}\right)}{\left(\alpha^j - \alpha^i\right)\prod_{j' \in \Theta, j' \neq i}\left(\alpha^i - \alpha^{j'}\right)}. \end{aligned} \tag{21}$$

Since $\prod_{j=0}^{n-1}(x - \alpha^j) = x^n - 1$ and $-1 = 1$ in $\mathbb{F}_{2^m}$, we have

$$\begin{aligned} \alpha^j \prod_{j'=0, j' \neq j}^{n-1}\left(\alpha^j - \alpha^{j'}\right) &= \prod_{j=0}^{n-1} \alpha^j \\ &= \prod_{j=0}^{n-1}(0 - \alpha^j) \\ &= 1. \end{aligned} \tag{22}$$

Eq. (21) can also be written as

$$T_i\left(\alpha^j\right) = \frac{\alpha^i \prod_{j' \in \Theta^c}\left(\alpha^i - \alpha^{j'}\right)}{\alpha^j\left(\alpha^j - \alpha^i\right)\prod_{j' \in \Theta^c, j' \neq j}\left(\alpha^j - \alpha^{j'}\right)}. \tag{23}$$

Remark 1 shows that the RS codes considered in this work would have a dimension $k' > \frac{n}{2}$. Hence, $|\Theta^c| < |\Theta|$. Computing the $\mathbf{G}_{\text{RS}}$ entries using eq. (23) requires less $\mathbb{F}_{2^m}$ operations than using eq. (21). Meanwhile, only the $k' \times (n - k')$ entries with $j \in \Theta^c$ need to be computed. This computation can be performed in a fully parallel manner, which yields the low-latency feature for the LLOSD.

B. Generation of BCH Codeword Candidates

After determining the RS systematic generator matrix $\mathbf{G}_{\text{RS}}$, BCH codeword candidates can be further generated through $\mathbf{G}_{\text{RS}}$. With the initial message $\underline{\boldsymbol{u}}$, an initial estimated RS codeword $\hat{\underline{\boldsymbol{v}}}^{(0)} = (\hat{v}_0^{(0)}, \hat{v}_1^{(0)}, \ldots, \hat{v}_{n-1}^{(0)}) \in \mathbb{F}_{2^m}^n$ can be generated by

$$\hat{\underline{\boldsymbol{v}}}^{(0)} = \underline{\boldsymbol{u}} \cdot \mathbf{G}_{\text{RS}}. \tag{24}$$

The systematic feature of $\mathbf{G}_{\text{RS}}$ enables $\hat{v}_j^{(0)} = r_j$, $\forall j \in \Theta$. The LLOSD TEPs can be further defined as

$$\underline{\boldsymbol{e}}_{[\Theta]}^{(\omega)} = (e_{j_0}^{(\omega)}, e_{j_1}^{(\omega)}, \ldots, e_{j_{k'-1}}^{(\omega)}) \in \mathbb{F}_2^{k'}. \tag{25}$$

Again, with a decoding order τ, $\omega = 0, 1, \ldots, N_{\text{TEPs}} - 1$ and $N_{\text{TEPs}} = \sum_{\rho=0}^{\tau}\binom{k'}{\rho}$. Subsequently, the test messages $\underline{\boldsymbol{u}}^{(\omega)}$ can be generated by

$$\underline{\boldsymbol{u}}^{(\omega)} = \underline{\boldsymbol{u}} + \underline{\boldsymbol{e}}_{[\Theta]}^{(\omega)}. \tag{26}$$

Note that eqs. (15) and (25) ensure that $\underline{\boldsymbol{u}}^{(\omega)} \in \mathbb{F}_2^{k'}$. RS codeword candidates $\hat{\underline{\boldsymbol{v}}}^{(\omega)} = (\hat{v}_0^{(\omega)}, \hat{v}_1^{(\omega)}, \ldots, \hat{v}_{n-1}^{(\omega)}) \in \mathbb{F}_{2^m}^n$ can be further generated by

$$\begin{aligned} \hat{\underline{\boldsymbol{v}}}^{(\omega)} &= \underline{\boldsymbol{u}}^{(\omega)} \cdot \mathbf{G}_{\text{RS}} \\ &= \hat{\underline{\boldsymbol{v}}}^{(0)} + \underline{\boldsymbol{e}}_{[\Theta]}^{(\omega)} \cdot \mathbf{G}_{\text{RS}}, \end{aligned} \tag{27}$$

Based on *Lemma 1*, if $\hat{\underline{\boldsymbol{v}}}^{(\omega)} \in \mathbb{F}_2^n$, it is also an (n, k) BCH codeword candidate. This binary property can be effectively assessed by the following theorem.

Theorem 2. During the re-encoding of eq. (27), for $j \in \Theta^c$, if there exists $\hat{v}_j^{(0)} + \sum_{i \in \Theta, e_i^{(\omega)} \neq 0} H_{\underline{\boldsymbol{u}}_i}(\alpha^j) \notin \mathbb{F}_2$, $\hat{\underline{\boldsymbol{v}}}^{(\omega)}$ is not a BCH codeword.

Proof. Since $\underline{\boldsymbol{u}}^{(\omega)} \in \mathbb{F}_2^{k'}$, codeword symbols $\hat{v}_j^{(\omega)} \in \mathbb{F}_2$, $\forall j \in \Theta$. For the remaining symbols that are indexed in Θ^c, we have

$$\hat{v}_j^{(\omega)} = \hat{v}_j^{(0)} + \sum_{i \in \Theta, e_i^{(\omega)} \neq 0} H_{\underline{\boldsymbol{u}}_i}(\alpha^j). \tag{28}$$

If they are also binary, $\hat{\underline{\boldsymbol{v}}}^{(\omega)} \in \mathbb{F}_2^n$. Based on *Lemma 1*, it is a BCH codeword. Otherwise, codeword $\hat{\underline{\boldsymbol{v}}}^{(\omega)}$ will be discarded as an invalid codeword candidate. ■

Note that with the assessment of *Theorem 2*, the proposed LLOSD can eliminate an invalid BCH codeword candidate once a codeword symbol generated by eq. (28) is non-binary. Subsequently, the following re-encoding can be skipped. This assessment can eliminate the redundant re-encoding if they are not yielding a BCH codeword. Our later analysis of Section III-C and numerical results of Section III-D will show that the number of BCH codeword candidates is far fewer than N_{TEPs}. Further incorporating with the decoding termination introduced in II-B, OSD requires fewer amount of $\mathbb{F}_{2^m}$ operations than that of $\mathbb{F}_2$ operations required by the OSD. Therefore, *Theorem 2* can effectively facilitate the LLOSD. Meanwhile, in Section IV, a TEP segmentation approach will also be introduced to further facilitate the LLOSD.

With an LLOSD output list denoted as $\{\hat{\underline{\boldsymbol{v}}}\}$ which contains N_{BCH} codeword candidates, the correlation distance between $\underline{\boldsymbol{r}}$ and the BCH codeword candidate $\hat{\underline{\boldsymbol{v}}}^{(\omega)}$ will be further

Algorithm 1 The LLOSD Algorithm

Require: $\underline{\boldsymbol{\mathcal{Y}}}$, τ;
Ensure: $\hat{\underline{\boldsymbol{v}}}_{\text{opt}}$;
1: Compute the LLRs as in eq. (7), and determine $\underline{\boldsymbol{r}}$;
2: Define the MRPs, $\underline{\boldsymbol{u}}$, and let $\mathcal{D}_{\min} = +\infty$;
3: Generate $\mathbf{G}_{\text{RS}}$ as in eqs. (20), (21) or (23);
4: Generate the initial codeword $\hat{\underline{\boldsymbol{v}}}^{(0)}$ as in eq. (24);
5: **For** each TEP $\underline{\boldsymbol{e}}_{[\Theta]}^{(\omega)}$, **do**
6: Test if the codeword $\hat{\underline{\boldsymbol{v}}}^{(\omega)}$ is binary as in eq. (28);
7: **If** $\hat{\underline{\boldsymbol{v}}}^{(\omega)}$ is binary
8: Determine $\mathcal{D}(\underline{\boldsymbol{r}}, \hat{\underline{\boldsymbol{v}}}^{(\omega)})$ as in eq. (12);
9: **If** $\mathcal{D}(\underline{\boldsymbol{r}}, \hat{\underline{\boldsymbol{v}}}^{(\omega)}) < \mathcal{D}_{\min}$
10: Update $\mathcal{D}_{\min} = \mathcal{D}(\underline{\boldsymbol{r}}, \hat{\underline{\boldsymbol{v}}}^{(\omega)})$ and $\hat{\underline{\boldsymbol{v}}}_{\text{opt}} = \hat{\underline{\boldsymbol{v}}}^{(\omega)}$;
11: **If** $\hat{\underline{\boldsymbol{v}}}^{(\omega)}$ satisfies the ML criterion as in eq. (14);
12: Terminate the decoding;
13: **End for**
14: Return $\hat{\underline{\boldsymbol{v}}}_{\text{opt}}$;

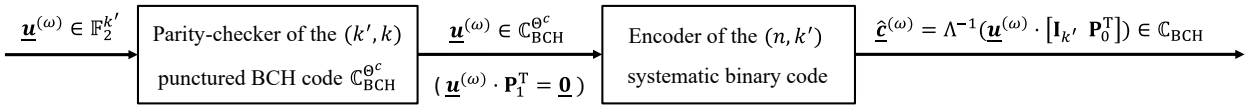


Fig. 1. Concatenated perspective of the LLOSD.

determined as in eq. (12). The one that yields the minimum correlation distance with $\underline{\boldsymbol{r}}$ will be selected as $\hat{\underline{\boldsymbol{v}}}_{\text{opt}}$. Meanwhile, the decoding can be facilitated by identifying the ML codeword. Once a codeword candidate $\hat{\underline{\boldsymbol{v}}}^{(\omega)}$ satisfies the ML criterion [8], it will be selected as the decoding output $\hat{\underline{\boldsymbol{v}}}_{\text{opt}}$ and decoding terminates. Summarizing the above description, the LLOSD algorithm is presented below as in *Algorithm 1*.

C. The Concatenated Perspective

This subsection provides another perspective onto the re-encoding of LLOSD. It can be re-interpreted as a concatenation between the parity-checker of a punctured BCH code and the systematic encoder of a binary linear block code. This concatenated perspective not only converts the non-binary re-encoding operations into binary, but also unveils why the LLOSD generates far fewer BCH codeword candidates than the number of decoding TEPs (as shown in Fig. 2), verifying its low-complexity feature.

Given the systematic generator matrix $\mathbf{G}_{\text{RS}}$ of the (n, k') RS code, its systematic parity-check matrix $\mathbf{H}_{\text{RS}}$ can be obtained based on $\mathbf{G}_{\text{RS}} \cdot \mathbf{H}_{\text{RS}}^{\mathsf{T}} = \mathbf{0}$, where $\mathbf{0}$ is an all zero matrix. Entries of $\mathbf{H}_{\text{RS}}$ are defined as

$$h_{i,j} = \begin{cases} 0, & \text{if } j \in \Theta^c, j \neq j_{k'+i}; \\ 1, & \text{if } j \in \Theta^c, j = j_{k'+i}; \\ T_j(\alpha^{j_{k'+i}}), & \text{if } j \in \Theta, \end{cases} \tag{29}$$

where $i = 0, 1, ..., n-k'-1$ and $j = 0, 1, ..., n-1$. Since the entries $h_{i,j} \in \mathbb{F}_{2^m}$, they can be represented as a vector in $\mathbb{F}_2^m$ through

$$\begin{aligned} h_{i,j} &= h_{i,j}^{(0)} + h_{i,j}^{(1)}\alpha + \cdots + h_{i,j}^{(m-1)}\alpha^{m-1} \\ &= (1, \alpha, ..., \alpha^{m-1}) \cdot (h_{i,j}^{(0)}, h_{i,j}^{(1)}, ..., h_{i,j}^{(m-1)})^{\mathsf{T}}, \end{aligned} \tag{30}$$

where $h_{i,j}^{(0)}, h_{i,j}^{(1)}, ..., h_{i,j}^{(m-1)} \in \mathbb{F}_2$. Based on *Lemma 1*, $\mathbf{H}_{\text{RS}}$ is also a parity-check matrix of the (n, k) BCH code. By representing the entries of $\mathbf{H}_{\text{RS}}$ as column vectors over $\mathbb{F}_2$, an $m(n-k') \times n$ binary parity-check matrix $\mathbf{H}_{\text{BCH}}$ for the BCH code can be obtained as

$$\mathbf{H}_{\text{BCH}} = \begin{bmatrix} h_{0,0}^{(0)} & h_{0,1}^{(0)} & \cdots & h_{0,n-1}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{0,0}^{(m-1)} & h_{0,1}^{(m-1)} & \cdots & h_{0,n-1}^{(m-1)} \\ h_{1,0}^{(0)} & h_{1,1}^{(0)} & \cdots & h_{1,n-1}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{1,0}^{(m-1)} & h_{1,1}^{(m-1)} & \cdots & h_{1,n-1}^{(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-k'-1,0}^{(0)} & h_{n-k'-1,1}^{(0)} & \cdots & h_{n-k'-1,n-1}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-k'-1,0}^{(m-1)} & h_{n-k'-1,1}^{(m-1)} & \cdots & h_{n-k'-1,n-1}^{(m-1)} \end{bmatrix}. \tag{31}$$

Note that

$$\Lambda(\mathbf{H}_{\text{RS}}) = \begin{bmatrix} h_{0,j_0} & h_{0,j_1} & \cdots & h_{0,j_{k'-1}} & 1 & 0 & \cdots & 0 \\ h_{1,j_0} & h_{1,j_1} & \cdots & h_{1,j_{k'-1}} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{n-k'-1,j_0} & h_{n-k'-1,j_1} & \cdots & h_{n-k'-1,j_{k'-1}} & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{32}$$

In conjunction with $1 = (1, \alpha, ..., \alpha^{m-1}) \cdot (1, 0, ..., 0)^{\mathsf{T}}$ and $0 = (1, \alpha, ..., \alpha^{m-1}) \cdot (0, 0, ..., 0)^{\mathsf{T}}$, as in (33), shown at the bottom of the page. Rows of $\Lambda(\mathbf{H}_{\text{BCH}})$ are further permuted such that an $(n-k') \times (n-k')$ identity submatrix $\mathbf{I}_{n-k'}$ appears in its top right corner as in (34), shown at the bottom of the page, where $\mathbf{P}_0$ and $\mathbf{P}_1$ are the $(n-k') \times k'$ and $(m-1)(n-k') \times k'$ submatrices, respectively. Therefore, it can be seen that any vector $\underline{\boldsymbol{r}} \in \mathbb{F}_2^n$ is a BCH codeword if and only if

$$\Lambda(\underline{\boldsymbol{r}}) \cdot [\mathbf{P}_0 \quad \mathbf{I}_{n-k'}]^{\mathsf{T}} = \underline{\mathbf{0}}, \tag{35}$$

and

$$\Lambda(\underline{\boldsymbol{r}}) \cdot [\mathbf{P}_1 \quad \mathbf{0}]^{\mathsf{T}} = \underline{\mathbf{0}}. \tag{36}$$

They imply $\underline{\boldsymbol{r}}_{[\Theta]} \cdot \mathbf{P}_0^{\mathsf{T}} = \underline{\boldsymbol{r}}_{[\Theta^c]}$ and $\underline{\boldsymbol{r}}_{[\Theta]} \cdot \mathbf{P}_1^{\mathsf{T}} = \underline{\mathbf{0}}$, respectively. Let $\mathbb{C}_{\text{BCH}}^{\Theta^c}$ denote the punctured BCH code that is obtained by puncturing symbols of the (n, k) BCH codeword at the positions of Θ^c, i.e.,

$$\mathbb{C}_{\text{BCH}}^{\Theta^c} = \{\underline{\boldsymbol{c}}_{[\Theta]} = (c_{j_0}, c_{j_1}, ..., c_{j_{k'-1}}) \mid \underline{\boldsymbol{c}} \in \mathbb{C}_{\text{BCH}}\}. \tag{37}$$

The following *Theorem 3* reveals that $\mathbf{P}_1$ is a parity-check matrix of $\mathbb{C}_{\text{BCH}}^{\Theta^c}$.

Theorem 3. The matrix $\mathbf{P}_1$ defined in eq. (34) is a parity-check matrix of the (k', k) punctured BCH code $\mathbb{C}_{\text{BCH}}^{\Theta^c}$.

Proof. For any codeword $\underline{\boldsymbol{c}} \in \mathbb{C}_{\text{BCH}}$, it satisfies eq. (36), i.e., $\underline{\boldsymbol{c}}_{[\Theta]} \cdot \mathbf{P}_1^{\mathsf{T}} = \underline{\mathbf{0}}$. Based on eq. (37), any codeword $\tilde{\underline{\boldsymbol{c}}} \in \mathbb{C}_{\text{BCH}}^{\Theta^c}$ satisfies $\tilde{\underline{\boldsymbol{c}}} \cdot \mathbf{P}_1^{\mathsf{T}} = \underline{\mathbf{0}}$. On the other hand, for any vector $\underline{\boldsymbol{a}} \in \mathbb{F}_2^{k'}$ such that $\underline{\boldsymbol{a}} \cdot \mathbf{P}_1^{\mathsf{T}} = \underline{\mathbf{0}}$, let $\underline{\boldsymbol{c}} = \Lambda^{-1}(\underline{\boldsymbol{a}} \cdot [\mathbf{I}_{k'} \quad \mathbf{P}_0^{\mathsf{T}}])$. It can be verified that $\underline{\boldsymbol{c}}$ satisfies both eqs. (35) and (36), which implies $\underline{\boldsymbol{c}} \in \mathbb{C}_{\text{BCH}}$. Since $\underline{\boldsymbol{a}} = \underline{\boldsymbol{c}}_{[\Theta]}$, based on eq. (37), $\underline{\boldsymbol{a}} \in \mathbb{C}_{\text{BCH}}^{\Theta^c}$. Therefore, for any vector $\underline{\boldsymbol{a}} \in \mathbb{F}_2^{k'}$, $\underline{\boldsymbol{a}} \in \mathbb{C}_{\text{BCH}}^{\Theta^c}$ if and only if $\underline{\boldsymbol{a}} \cdot \mathbf{P}_1^{\mathsf{T}} = \underline{\mathbf{0}}$, i.e., $\mathbf{P}_1$ is a parity-check matrix of the punctured BCH code $\mathbb{C}_{\text{BCH}}^{\Theta^c}$.

Furthermore, since the ranks of $\mathbf{H}_{\text{BCH}}$ and $[\mathbf{P}_0 \quad \mathbf{I}_{n-k'}]$ are $n-k$ and $n-k'$, respectively, the rank of $\mathbf{P}_1$ is $n-k-(n-k') = k'-k$. Hence, the punctured BCH code $\mathbb{C}_{\text{BCH}}^{\Theta^c}$ has a dimension of k. ■

Based on eqs. (35), (36) and *Theorem 3*, re-encoding of LLOSD can be reformulated as the follows. Given a test message $\underline{\boldsymbol{u}}^{(\omega)} \in \mathbb{F}_2^{k'}$, if it is a punctured BCH codeword, i.e.,

$$\underline{\boldsymbol{u}}^{(\omega)} \cdot \mathbf{P}_1^{\mathsf{T}} = \underline{\mathbf{0}}, \tag{38}$$

$$\Lambda(\mathbf{H}_{\text{BCH}}) = \begin{bmatrix} h_{0,j_0}^{(0)} & h_{0,j_1}^{(0)} & \dots & h_{0,j_{k'-1}}^{(0)} & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{0,j_0}^{(m-1)} & h_{0,j_1}^{(m-1)} & \dots & h_{0,j_{k'-1}}^{(m-1)} & 0 & 0 & \dots & 0 \\ h_{1,j_0}^{(0)} & h_{1,j_1}^{(0)} & \dots & h_{1,j_{k'-1}}^{(0)} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{1,j_0}^{(m-1)} & h_{1,j_1}^{(m-1)} & \dots & h_{1,j_{k'-1}}^{(m-1)} & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{n-k'-1,j_0}^{(0)} & h_{n-k'-1,j_1}^{(0)} & \dots & h_{n-k'-1,j_{k'-1}}^{(0)} & 0 & 0 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{n-k'-1,j_0}^{(m-1)} & h_{n-k'-1,j_1}^{(m-1)} & \dots & h_{n-k'-1,j_{k'-1}}^{(m-1)} & 0 & 0 & \dots & 0 \end{bmatrix} \tag{33}$$

$$\begin{aligned} \Lambda(\mathbf{H}'_{\text{BCH}}) &= \begin{bmatrix} h_{0,j_0}^{(0)} & h_{0,j_1}^{(0)} & \dots & h_{0,j_{k'-1}}^{(0)} & 1 & 0 & \dots & 0 \\ h_{1,j_0}^{(0)} & h_{1,j_1}^{(0)} & \dots & h_{1,j_{k'-1}}^{(0)} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{n-k'-1,j_0}^{(0)} & h_{n-k'-1,j_1}^{(0)} & \dots & h_{n-k'-1,j_{k'-1}}^{(0)} & 0 & 0 & \dots & 1 \\ h_{0,j_0}^{(1)} & h_{0,j_1}^{(1)} & \dots & h_{0,j_{k'-1}}^{(1)} & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{n-k'-1,j_0}^{(m-1)} & h_{n-k'-1,j_1}^{(m-1)} & \dots & h_{n-k'-1,j_{k'-1}}^{(m-1)} & 0 & 0 & \dots & 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{P}_0 & \mathbf{I}_{n-k'} \\ \mathbf{P}_1 & \mathbf{0} \end{bmatrix} \end{aligned} \tag{34}$$

a BCH codeword candidate can be generated by

$$\begin{aligned} \underline{\hat{\boldsymbol{c}}}^{(\omega)} &= \Lambda^{-1}(\underline{\boldsymbol{u}}^{(\omega)} \cdot [\mathbf{I}_{k'} \quad \mathbf{P}_0^{\mathsf{T}}]) \\ &= \Lambda^{-1}(\underline{\boldsymbol{u}}^{(\omega)}, \underline{\boldsymbol{u}}^{(\omega)} \cdot \mathbf{P}_0^{\mathsf{T}}). \end{aligned} \tag{39}$$

Therefore, the LLOSD re-encoding can be equivalently realized through the concatenation between a parity-checker of $\mathbb{C}_{\text{BCH}}^{\Theta^c}$ and an encoder of the (n, k') systematic binary code. The latter is defined by the generator matrix $\Lambda^{-1}([\mathbf{I}_{k'} \quad \mathbf{P}_0^{\mathsf{T}}])$. Fig. 1 illustrates this concatenated perspective for the LLOSD.

This concatenated perspective further converts the non-binary re-encoding operations into the binary, resulting in the LLOSD re-encoding being more friendly for hardware implementation. Our numerical results will verify that this concatenated realization of the LLOSD can substantially reduce the decoding latency. Moreover, based on eq. (26), the parity-check equation of (38) can be rewritten as

$$\begin{aligned} (\underline{\boldsymbol{u}} + \underline{\boldsymbol{e}}_{[\Theta]}^{(\omega)}) \cdot \mathbf{P}_1^{\mathsf{T}} &= \underline{\mathbf{0}} \\ \Leftrightarrow \underline{\boldsymbol{u}} \cdot \mathbf{P}_1^{\mathsf{T}} + \underline{\boldsymbol{e}}_{[\Theta]}^{(\omega)} \cdot \mathbf{P}_1^{\mathsf{T}} &= \underline{\mathbf{0}} \\ \Leftrightarrow \underline{\boldsymbol{s}} + \underline{\boldsymbol{e}}_{[\Theta]}^{(\omega)} \cdot \mathbf{P}_1^{\mathsf{T}} &= \underline{\mathbf{0}} \\ \Leftrightarrow \underline{\boldsymbol{e}}_{[\Theta]}^{(\omega)} \cdot \mathbf{P}_1^{\mathsf{T}} &= \underline{\boldsymbol{s}}, \end{aligned} \tag{40}$$

where $\underline{\boldsymbol{s}} = \underline{\boldsymbol{u}} \cdot \mathbf{P}_1^{\mathsf{T}}$ is the syndrome vector. Hence, by pre-computing the syndrome vector $\underline{\boldsymbol{s}}$ of the initial message $\underline{\boldsymbol{u}}$, the parity-check computation of eq. (38) can be simplified into eq. (40). Note that $\mathbf{P}_1$ can be further reduced by GE, resulting in an $(k' - k) \times k'$ matrix. However, applying GE contradicts the primary aim of this work. In practice, using a reduced $\mathbf{P}_1$ does not significantly improve efficiency of the above parity-check computation, since the computation can be terminated once a check imposed by $\mathbf{P}_1$ is violated. This often happens after checking the first few rows of $\mathbf{P}_1$.

Furthermore, it can also be seen that among all TEPs, only those that can produce a (k', k) punctured BCH codeword through eq. (26) are able to further yield an (n, k) BCH codeword candidate. This implies that the LLOSD will generate fewer BCH codeword candidates than the decoding TEPs. Based on eq. (40), a TEP $\underline{\boldsymbol{e}}_{[\Theta]}^{(\omega)}$ can produce a punctured BCH codeword if and only if $\underline{\boldsymbol{e}}_{[\Theta]}^{(\omega)}$ is an element of the following coset of $\mathbb{C}_{\text{BCH}}^{\Theta^c}$

$$\underline{\boldsymbol{u}} + \mathbb{C}_{\text{BCH}}^{\Theta^c} := \{\underline{\boldsymbol{u}} + \underline{\tilde{\boldsymbol{c}}} \mid \underline{\tilde{\boldsymbol{c}}} \in \mathbb{C}_{\text{BCH}}^{\Theta^c}\}. \tag{41}$$

Let $A_\rho(\cdot)$ denote the number of weight-ρ codewords in a coset (code). For an order-τ LLOSD, since the TEPs consist of all vectors in $\mathbb{F}_2^{k'}$ with a weight at most τ, the number of BCH codeword candidates is

$$N_{\text{BCH}} = \sum_{\rho=0}^{\tau} A_\rho(\underline{\boldsymbol{u}} + \mathbb{C}_{\text{BCH}}^{\Theta^c}). \tag{42}$$

Note that if τ is less than the minimum weight of the coset $\underline{\boldsymbol{u}} + \mathbb{C}_{\text{BCH}}^{\Theta^c}$, $N_{\text{BCH}} = 0$. Since this minimum weight varies

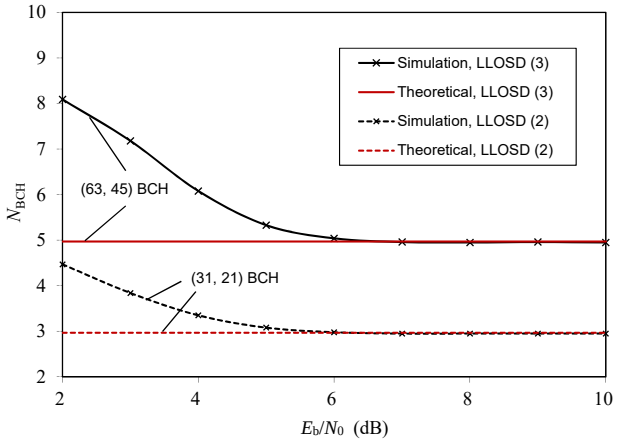


Fig. 2. Number of BCH codeword candidates.

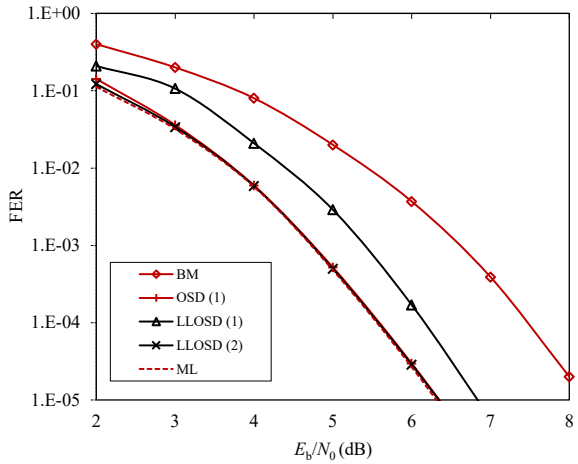


Fig. 3. Performance of the $(31,\ 21)$ BCH code.

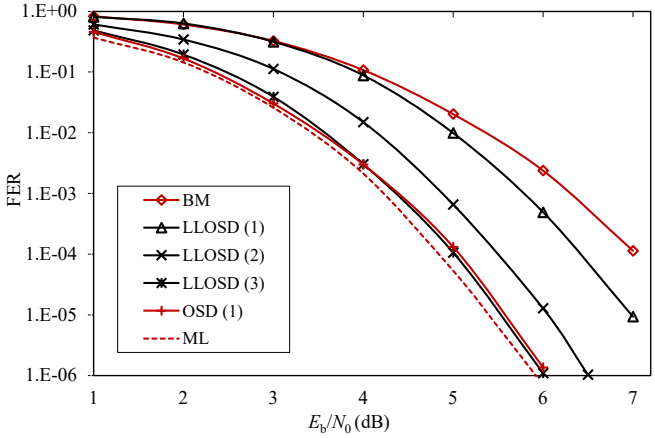


Fig. 4. Performance of the $(63,\ 45)$ BCH code.

with the channel conditions, the LLOSD does not guarantee the delivery of a valid codeword. Our research shows that when the received information is only mildly corrupted, it frequently occurs that $\underline{\boldsymbol{u}} = \underline{\boldsymbol{c}}^*_{[\Theta]}$, where $\underline{\boldsymbol{c}}^* \in \mathbb{C}_{\mathrm{BCH}}$ denotes the transmitted codeword. In this case, $N_{\mathrm{BCH}} = \sum_{\rho=0}^{\tau} A_\rho(\mathbb{C}_{\mathrm{BCH}}^{\Theta^c})$. It is much smaller than N_{TEPs}, where $N_{\mathrm{TEPs}} = \sum_{\rho=0}^{\tau} \binom{k'}{\rho}$.

Fig. 2 shows our numerical results on the number of BCH codeword candidates N_{BCH} generated by the LLOSD. In this paper, our numerical results are obtained over the additive white Gaussian noise (AWGN) channel using BPSK modulation. The signal-to-noise ratio (SNR) is measured as E_b/N_0, where E_b is the power per information bit and N_0 is the single side-band power spectrum density of the noise. The theoretical results are the average of $\sum_{\rho=0}^{\tau} A_\rho(\mathbb{C}_{\mathrm{BCH}}^{\Theta^c})$ that are obtained in $10,000$ trials of randomly puncturing $n - k'$ positions of the BCH code $\mathbb{C}_{\mathrm{BCH}}$. It can be seen that in both the LLOSD of the $(63,\ 45)$ BCH code with $\tau = 3$ and the $(31,\ 21)$ BCH code with $\tau = 2$, N_{BCH} converges to the theoretical estimations of five and three, respectively. Note that N_{TEPs} are 30914 and 379 for the two LLOSD scenarios, respectively.

D. Decoding Performances

Figs. 3 and 4 show the decoding frame error rate (FER) of the $(31,\ 21)$ and the $(63,\ 45)$ BCH codes, respectively. They are binary subcodes of the $(31,\ 27)$ and the $(63,\ 57)$ RS codes. Performances of the BM decoding [20], the OSD [5], and ML decoding [37] are also presented as benchmarks.

Based on the above subsection, non-binary re-encoding operations of the LLOSD can be simplified to binary ones. This simplified LLOSD is denoted as LLOSD-B. It exhibits the same decoding performance as the LLOSD, but results in less finite field operations and a smaller decoding latency. Our simulation results show that the LLOSD performance can approach that of the OSD, but requiring a larger decoding order. E.g., for the $(63,\ 45)$ BCH code, the LLOSD (3) performs the same as the OSD (1). This is due to the LLOSD functions under the RS paradigm, where its TEPs have a greater dimension than that of the OSD. Hence, to achieve the same decoding performance, the LLOSD inevitably exhibits a larger worst case (when all test messages are re-encoded) decoding complexity. However, in practice, early termination is usually applied. The decoding is terminated once a desired codeword is found, e.g., the one that satisfies the ML criterion of eq. (14). In this case, the LLOSD can show its complexity and latency advantages over the OSD and its enhanced variants. Please also note that Section III-C shows N_{BCH} is far smaller than N_{TEP}. Most of the re-encoding computation can be filtered, especially by the LLOSD-B.

To further verify the analysis, Table I-A also shows our numerical results on the decoding complexity for the $(63,\ 45)$ BCH code. The decoding will be terminated once a codeword that satisfies eq. (14) is produced. As the SNR increases, the complexity of the OSD, the LLOSD and the LLOSD-B reduce since the ML codeword can be produced earlier. It

TABLE I
NUMERICAL RESULTS IN DECODING THE $(63,\ 45)$ BCH CODE

A. Decoding Complexity.

Algorithms	E_b/N_0 (dB)	Complexity		
		$\mathbb{F}_2$ oper.	$\mathbb{F}_{64}$ oper.	Floating oper.
OSD (1)	4	2.78×10^4		81
	5	2.60×10^4		19
	6	2.56×10^4		8
LLOSD (3)	4		1.81×10^4	15
	5		5.21×10^3	8
	6		2.58×10^3	7
LLOSD-B (3)	4	3.13×10^4		15
	5	6.17×10^3	1.77×10^3	8
	6	1.56×10^3		7

B. Decoding Latency.

Algorithms	E_b/N_0 (dB)	Latency (μs)
OSD (1)	4	6.58×10^2
	5	5.34×10^2
	6	5.06×10^2
LLOSD (3)	4	1.99×10^3
	5	4.36×10^2
	6	1.32×10^2
LLOSD-B (3)	4	1.13×10^3
	5	2.04×10^2
	6	8.46×10^1

can be observed that with similar decoding performances, the amount of $\mathbb{F}_{64}$ operations required by the LLOSD (3) can be reduced to only a fifth of $\mathbb{F}_2$ operations required by the OSD (1). Our analysis in Section III-C shows the number of BCH codeword candidates is far fewer than N_{TEPs}. Hence, the re-encoding skipping rule in *Theorem* 2 can effectively reduce the LLOSD complexity. It also subsequently results in fewer floating-point operations that are required in calculating the correlation distance of eq. (14). In the LLOSD-B, the amount of $\mathbb{F}_{64}$ operations is unrelated to the SNR, as the $\mathbb{F}_{64}$ operations are only attributed to the calculation of $\mathbf{G}_{\text{RS}}$. In comparison with the OSD (1), the LLOSD-B (3) also requires fewer $\mathbb{F}_2$ operations as the SNR increases.

To evaluate the decoding latency of the proposed algorithms, all simulations in this paper are conducted using the Intel core i7-10710U CPU. It is measured as the average time in decoding a codeword. In all algorithms, the TEPs are processed sequentially. This provides a supplementary reference for assessing the decoding latency difference among the three algorithms. Table I-B compares the decoding latency of OSD (1), LLOSD (3), and LLOSD-B (3). Note that we assume in the LLOSD and the LLOSD-B, rows of $\mathbf{G}_{\text{RS}}$ are generated in parallel. It shows that both the LLOSD and LLOSD-B can significantly reduce the decoding latency over the OSD. This is primarily due to both the LLOSD and LLOSD-B are free from conducting GE.

IV. THE SEGMENTED VARIANT

The LLOSD generates BCH codeword candidates using the RS systematic generator (or parity-check) matrix that is obtained by the use of Lagrange interpolation polynomials. Computation of the matrix entries can be performed in a fully parallel manner, removing the latency-orienting GE in the conventional OSD. However, as *Remark* 1 points out, an RS code has a greater dimension than its BCH subcode. Consequently, the k' MRPs of Θ may contain more errors than the k MRIPs of Υ. The LLOSD requires a higher order to achieve the same decoding performance as the conventional OSD, which can be observed from our simulation results in Section III-D. Although the complexity of the LLOSD re-encoding has been reduced by discarding the non-binary codeword candidates (or equivalently the test messages that fail the parity-check of the punctured BCH code), decoding complexity can be further reduced through reducing the number of TEPs. This section proposes a segmented variant of the LLOSD, denoted as the SLLOSD. It partitions the TEP into two segments that correspond to Υ and $\Theta\backslash\Upsilon$, respectively. They are enumerated separately to generate the TEPs, reducing the number of TEPs.

A. TEP Segmentation

This research shows that for the LLOSD to yield a performance close to that of the OSD (τ), it is unnecessary to utilize TEPs with a Hamming weight greater than τ over the k MRIPs (as denoted by Υ). Hence, we partition the MRPs (Θ) into Υ and $\Theta\backslash\Upsilon$. The TEPs can be generated by enumerating error patterns over Υ and $\Theta\backslash\Upsilon$ separately. To match order-τ OSD, the TEPs only need to maintain a maximum weight of τ over Υ. For the weight-ρ TEPs over Υ, we assign an order θ_ρ to the TEP over the complementary positions in $\Theta\backslash\Upsilon$. That says they have a maximum weight of θ_ρ over $\Theta\backslash\Upsilon$. Thereby, the number of TEPs can be significantly reduced.

Let $\theta_0, \theta_1, ..., \theta_\tau$ be a series of orders that are assigned to the $k' - k$ complementary positions of $\Theta\backslash\Upsilon$. For $\rho = 0, 1, ..., \tau$, the particular sets of TEPs can be defined as

$$E_\rho(\theta_\rho) = \{\underline{\boldsymbol{e}} = (e_1, e_2, ..., e_n) \in \mathbb{F}_2^n \mid \text{wt}(\underline{\boldsymbol{e}}_{[\Upsilon]}) = \rho, \\ \text{wt}(\underline{\boldsymbol{e}}_{[\Theta\backslash\Upsilon]}) \leqslant \theta_\rho,\ \underline{\boldsymbol{e}}_{[\Theta^c]} = \underline{\mathbf{0}}\}. \tag{43}$$

Each $E_\rho(\theta_\rho)$ contains the TEPs with an exact weight of ρ over Υ and a maximum weight of θ_ρ over $\Theta\backslash\Upsilon$, where $|E_\rho(\theta_\rho)| = \binom{k}{\rho} \sum_{\rho'=0}^{\theta_\rho} \binom{k'-k}{\rho'}$ TEPs. Furthermore, let

$$E(\theta_0, \theta_1, ..., \theta_\tau) = E_0(\theta_0) \cup E_1(\theta_1) \cup \cdots \cup E_\tau(\theta_\tau) \tag{44}$$

be the set of all TEPs that are processed by the SLLOSD. The algorithm can be parameterized by $(\theta_0, \theta_1, ..., \theta_\tau)$, and denoted as SLLOSD $(\theta_0, \theta_1, ..., \theta_\tau)$. The number of TEPs is

$$\begin{aligned} N_{\text{TEPs}} &= |E(\theta_0, \theta_1, ..., \theta_\tau)| \\ &= \sum_{\rho=0}^{\tau} |E_\rho(\theta_\rho)| \\ &= \sum_{\rho=0}^{\tau} \binom{k}{\rho} \sum_{\rho'=0}^{\theta_\rho} \binom{k'-k}{\rho'}. \end{aligned} \tag{45}$$

The following example illustrates how the SLLOSD can further reduce the number of TEPs required by the LLOSD.

Example 1. Let us consider decoding the $(63,\ 45)$ BCH code with performances shown in Fig. 4. Applying the aforementioned TEP segmentation, the LLOSD (3) can be expressed as the SLLOSD $(3,\ 2,\ 1,\ 0)$, where $N_{\text{TEPs}} = 30914$. Since both the OSD (1) and the LLOSD (3) can approach near ML performance, it is reasonable to set expect that the SLLOSD $(3,\ 2)$ can approximate the LLOSD (3), reducing N_{TEPs} to 3854. Our later discussion will show such a reduction does not degrade the decoding performance.

If the actual error pattern is included in $E(\theta_0, \theta_1, ..., \theta_\tau)$, the transmitted codeword will be in the SLLOSD output list. Let $P_{\text{e,SLLOSD}}(\theta_0, \theta_1, ..., \theta_\tau)$ and $P_{\text{e,ML}}$ denote the error probabilities of the SLLOSD $(\theta_0, \theta_1, ..., \theta_\tau)$ and the ML decoding, respectively. Further, let $P_{\text{list}}(\theta_0, \theta_1, ..., \theta_\tau)$ denote the probability that the transmitted codeword is not included in the SLLOSD $(\theta_0, \theta_1, ..., \theta_\tau)$ output list. The SLLOSD $(\theta_0, \theta_1, ..., \theta_\tau)$ error probability is upper bounded by

$$P_{\text{e,SLLOSD}}(\theta_0, \theta_1, ..., \theta_\tau) \leqslant P_{\text{e,ML}} + P_{\text{list}}(\theta_0, \theta_1, ..., \theta_\tau). \quad (46)$$

Let ε_Υ and $\varepsilon_{\Theta\backslash\Upsilon}$ further denote the numbers of errors locating at the positions of Υ and $\Theta\backslash\Upsilon$, respectively. In a memoryless channel, it can be assumed that ε_Υ and $\varepsilon_{\Theta\backslash\Upsilon}$ are independent [7]. Hence, $P_{\text{list}}(\theta_0, \theta_1, ..., \theta_\tau)$ can be characterized as

$$\begin{aligned}
&P_{\text{list}}(\theta_0, \theta_1, ..., \theta_\tau) \\
&= \sum_{\rho=0}^{\tau} \Pr(\varepsilon_\Upsilon = \rho, \varepsilon_{\Theta\backslash\Upsilon} > \theta_\rho) + \Pr(\varepsilon_\Upsilon > \tau) \\
&= \sum_{\rho=0}^{\tau} \Pr(\varepsilon_{\Theta\backslash\Upsilon} > \theta_\rho \mid \varepsilon_\Upsilon = \rho) \Pr(\varepsilon_\Upsilon = \rho) + \Pr(\varepsilon_\Upsilon > \tau) \\
&= \sum_{\rho=0}^{\tau} \Pr(\varepsilon_{\Theta\backslash\Upsilon} > \theta_\rho) \Pr(\varepsilon_\Upsilon = \rho) + \Pr(\varepsilon_\Upsilon > \tau).
\end{aligned} \quad (47)$$

Based on the performance of Section V.E in [5], if $\tau \geqslant \min\{\lceil d/4 - 1 \rceil, k\}$, asymptotically $\Pr(\varepsilon_\Upsilon > \tau) < P_{\text{e,ML}}$ [2]. If $\theta_0, \theta_1, ..., \theta_\tau$ are sufficiently large such that $\Pr(\varepsilon_{\Theta\backslash\Upsilon} > \theta_\rho) < P_{\text{e,ML}}$ holds for all $\rho = 0, 1, ..., \tau$,

$$\begin{aligned}
P_{\text{list}}(\theta_0, \theta_1, ..., \theta_\tau) &< \sum_{\rho=0}^{\tau} P_{\text{e,ML}} \cdot \Pr(\varepsilon_\Upsilon = \rho) + P_{\text{e,ML}} \\
&< 2P_{\text{e,ML}}.
\end{aligned} \quad (48)$$

Therefore, if the orders $\theta_0, \theta_1, ..., \theta_\tau$ are sufficiently large, the SLLOSD can also achieve a near ML decoding performance if $\tau \geqslant \min\{\lceil d/4 - 1 \rceil, k\}$. However, we still cannot develop the theoretical criterion for selecting the orders $\theta_0, \theta_1, ..., \theta_\tau$. They can only be determined empirically, aiming to yield a good performance-complexity tradeoff.

B. Decoding Performances

Fig. 5 shows the decoding performance of the $(63,\ 45)$ BCH code. It can be seen that the SLLOSD $(3,\ 2)$ performs similarly as the LLOSD (3). Furthermore, we compare the performance of both the LLOSD and SLLOSD with other state-of-the-art OSD improvements that also aim to reduce the GE computational burden. They include the Yue-Shirvanimoghaddam-Vucetic-Li (YSVL) OSD [17] and the Choi-Jeong (CJ) OSD [16], in which their decoding order are set to one. Moreover, the CJ OSD utilizes three permuted generator matrices that are computed offline, each with a predefined decoding order of one. It can be seen that the performance of the YSVL OSD (1) can closely approach that of the OSD (1), while the performance of the CJ OSD (1) becomes inferior to that of the OSD (1) as the SNR increases.

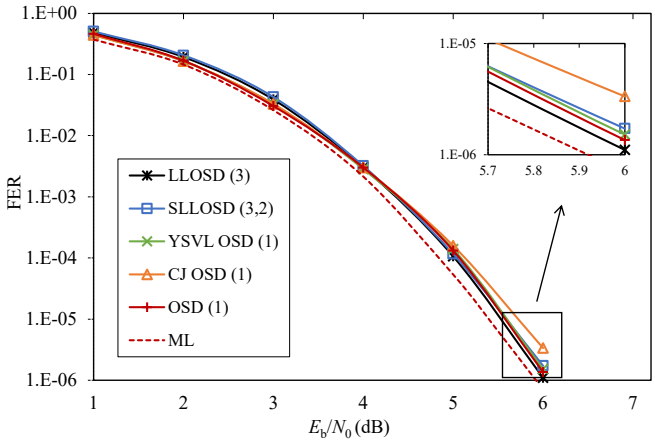


Fig. 5. Performance of the $(63,\ 45)$ BCH code.

Table II shows our numerical results of the decoding complexity and latency for the BCH code. Both the LLOSD and the SLLOSD are functioning in the concatenated manner introduced in Section III-C, and the facilitated SLLOSD is denoted as SLLOSD-B. It can be seen that the SLLOSD-B $(3,\ 2)$ requires fewer $\mathbb{F}_2$ operations and floating-point operations than the LLOSD-B (3). Note that the LLOSD-B (3) processes at most 30914 TEPs, while the SLLOSD-B $(3,\ 2)$ processes at most 3854 TEPs. This verifies the proposed TEP segmentation strategy. In comparison with the YSVL OSD and the CJ OSD, they require an identical level of non-binary operations for computing $\mathbf{G}_{\text{RS}}$. Note that both the YSVL OSD and the CJ OSD require fewer $\mathbb{F}_2$ operations than the conventional OSD which requires approximately 2.5×10^4 $\mathbb{F}_2$ operations. However, the YSVL OSD necessitates the calculation of probabilities that are needed for determining whether the GE can be skipped, triggering a substantial amount of floating-point operations. It can be observed that the SLLOSD-B $(3,\ 2)$ requires far fewer binary operations and floating-point operations than the YSVL OSD (1). At SNR of 6 dB, the SLLOSD-B $(3,\ 2)$ requires more $\mathbb{F}_2$ operations than the CJ OSD (1), due to the latter can bypass the GE in most decoding events. However, the SLLOSD-B $(3,\ 2)$ outperforms the CJ OSD (1) as demonstrated by Fig. 5. Table II-B also demonstrates that the SLLOSD-B $(3,\ 2)$ can reduce

[2]In this paper, the asymptoticity refers to the channel SNR approaches infinity

TABLE II
NUMERICAL RESULTS IN DECODING THE $(63,\ 45)$ BCH CODE

A. Decoding Complexity.

Algorithms	E_b/N_0 (dB)	Complexity		
		$\mathbb{F}_2$ oper.	$\mathbb{F}_{64}$ oper.	Floating oper.
LLOSD-B (3)	4	3.13×10^4	1.77×10^3	15
	5	6.17×10^3		8
	6	1.56×10^3		7
SLLOSD-B $(3,\ 2)$	4	5.19×10^3	1.77×10^3	8
	5	1.82×10^3		7
	6	1.20×10^3		7
YSVL OSD (1)	4	2.43×10^4		446
	5	2.41×10^4		385
	6	2.24×10^4		386
CJ OSD (1)	4	8.75×10^3		67
	5	3.37×10^3		12
	6	1.14×10^3		1

B. Decoding Latency.

Algorithms	E_b/N_0 (dB)	Latency (μs)
LLOSD-B (3)	4	1.13×10^3
	5	2.04×10^2
	6	8.46×10^1
SLLOSD-B $(3,\ 2)$	4	4.92×10^2
	5	8.60×10^1
	6	5.56×10^1
YSVL OSD (1)	4	3.57×10^2
	5	3.06×10^2
	6	2.85×10^2
CJ OSD (1)	4	2.85×10^2
	5	1.13×10^2
	6	5.44×10^1

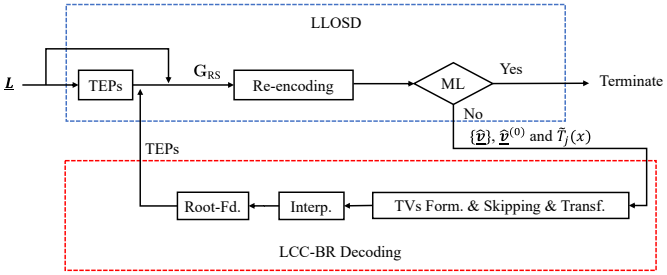


Fig. 6. Block diagram of the HSD.

the decoding latency over that of the LLOSD-B (3), especially in the low SNR regime. Again, this thanks to fewer TEPs are processed by the SLLOSD-B $(3,\ 2)$. It also exhibits a latency advantage over the YSVL OSD (1) and the CJ OSD (1).

V. THE HSD ALGORITHM

This section further proposes the HSD, which integrates the LLOSD and the LCC-BR decoding. The latter provides extra TEPs for the LLOSD, enhancing its decoding performance with a limited order. Since both the LLOSD and the LCC-BR decoding are designed based on RS code algebra, they share substantial common computations. The HSD only incurs a limited amount of extra computation over the LLOSD, making it particularly suitable for decoding longer BCH codes.

A. Block Diagram

Block diagram of the HSD is illustrated as in Fig. 6. In the HSD, the LLOSD is the primary decoding, which will be first deployed. If it fails to produce an intended codeword, e.g., one that satisfies the ML condition [8], the LCC-BR decoding will be deployed. In the LCC-BR decoding, each test-vector is decoded through the BR interpolation and the root-finding processes. For BCH codes, the root-finding can be simplified into a partial operation, namely the partial root-finding [26]. It provides extra TEPs for the LLOSD, enhancing its error-correction capability. With the Chase decoding test-vectors, the BR interpolation consists of their re-encoding transform and the formation of interpolation module basis isomorphism. They can be facilitated by utilizing the initial estimated RS codeword $\underline{\hat{\boldsymbol{v}}}^{(0)}$ and the Lagrange interpolation polynomials $\tilde{T}_j(x)$ that were generated in the LLOSD, respectively. Meanwhile, assuming the LLOSD preserves its list of codeword candidates $\{\underline{\hat{\boldsymbol{v}}}^{(0)}\}$, a skipping rule is introduced to eliminate the redundant test-vectors for the LCC-BR decoding. The following subsection first briefly reviews the LCC-BR decoding, and its integration with the LLOSD in formulating the HSD with be further introduced.

B. The LCC-BR Decoding

Given the sorted received word $\Lambda(\underline{\boldsymbol{r}})$ of eq. (8), its η ($\eta < n - k'$) least reliable positions (LRPs) can be denoted as $\Psi = \{j_{n-\eta}, j_{n-\eta+1}, \ldots, j_{n-1}\}$, with its complementary being $\Psi^c = \{j_0, j_1, \ldots, j_{n-\eta-1}\}$. For an error pattern $\underline{\boldsymbol{e}}^{(\omega)} = (e_0^{(\omega)}, e_1^{(\omega)}, \ldots, e_{n-1}^{(\omega)}) \in \mathbb{F}_2^n$, its entries are $e_j^{(\omega)} = 0$ or 1, if $j \in \Psi$; and $e_j^{(\omega)} = 0$, otherwise. Hence, 2^η test-vectors can be formulated as

$$\underline{\boldsymbol{r}}_\omega = \underline{\boldsymbol{r}} + \underline{\boldsymbol{e}}^{(\omega)} = (r_0^{(\omega)}, r_1^{(\omega)}, \ldots, r_{n-1}^{(\omega)}), \tag{49}$$

where $\omega = 0, 1, \ldots, 2^\eta - 1$ indexes the test-vectors. In order to specify an error pattern, we let $\omega = \sum_{i=0}^{\eta-1} 2^i e_{j_{n-i-1}}^{(\omega)}$. In particular, if $\omega = 0$, $\underline{\boldsymbol{e}}^{(0)} = \underline{\boldsymbol{0}}$. First, $\underline{\boldsymbol{r}}_\omega$ needs to be transformed by the initial estimated RS codeword $\underline{\hat{\boldsymbol{v}}}^{(0)}$ that was generated in the LLOSD as

$$\underline{\boldsymbol{r}}_\omega \mapsto \underline{\boldsymbol{z}}_\omega : z_j^{(\omega)} = r_j^{(\omega)} - \hat{v}_j^{(0)}, \quad \forall j. \tag{50}$$

This transform helps reduce the BR interpolation complexity, as will be demonstrated below. For the transformed test-vectors $\underline{\boldsymbol{z}}_\omega$, $z_j^{(\omega)} = 0, \forall j \in \Theta$. Meanwhile, the n interpolation points of $\underline{\boldsymbol{z}}_\omega$ can be written as

$$(1, z_0^{(\omega)}), (\alpha, z_1^{(\omega)}), \ldots, (\alpha^{n-1}, z_{n-1}^{(\omega)}). \tag{51}$$

The LCC-BR decoding of $\underline{\boldsymbol{z}}_\omega$ aims to construct the minimal polynomial $Q(x, y)$ that interpolates the above points with

a multiplicity of one. For this, it constructs a basis of the interpolation module that consists polynomials defined in $\mathbb{F}_{2^m}[x,y]$. The module basis is reduced into a Gröbner basis that contains the intended polynomial $Q(x,y)$.

Let

$$V(x)=\prod_{j\in\Theta}(x-\alpha^j). \tag{52}$$

It can be realized that $V(x)$ interpolates k' of the n points of eq. (51). They are $(\alpha^{j_0},z_{j_0}^{(\omega)}),(\alpha^{j_1},z_{j_1}^{(\omega)}),\cdots,(\alpha^{j_{k'-1}},z_{j_{k'-1}}^{(\omega)})$. This leads to the formulation of a basis of an isomorphic module $\mathcal{M}_\omega$. It is defined by a subset of the further transformed interpolation points. For this, the following mappings between points in $\{\alpha^j|j\in\Theta^c\}\times\mathbb{F}_{2^m}$ are needed

$$\begin{aligned}F&:(x,y)\mapsto\left(x,\frac{y}{V(x)}\right),\\ F^{-1}&:(x,y)\mapsto(x,yV(x)).\end{aligned} \tag{53}$$

Therefore, among the n interpolation points of eq. (51), those defined by Θ^c can be further transformed as

$$(\alpha^j,z_j^{(\omega)})\mapsto\left(\alpha^j,\frac{z_j^{(\omega)}}{V(\alpha^j)}\right),\forall j\in\Theta^c. \tag{54}$$

Let us define two seed polynomials as

$$\mathcal{G}(x)=\prod_{j\in\Theta^c}(x-\alpha^j) \tag{55}$$

and

$$R_\omega(x)=\sum_{j\in\Theta^c}z_j^{(\omega)}\tilde{T}_j(x), \tag{56}$$

where

$$\tilde{T}_j(x)=\frac{\prod_{j'\in\Theta^c,j'\neq j}(x-\alpha^{j'})}{\prod_{j'=0,j'\neq j}^{n-1}(\alpha^j-\alpha^{j'})}. \tag{57}$$

$\tilde{T}_j(x)$ is also the Lagrange interpolation polynomial w.r.t. code locator α^j. Subsequently, the basis $\mathcal{B}_\omega$ of the isomorphic module $\mathcal{M}_\omega$ can be formed by

$$P_{\omega,0}(x,y)=\mathcal{G}(x), \tag{58}$$

$$P_{\omega,1}(x,y)=y-R_\omega(x). \tag{59}$$

It can be seen that for the transformed interpolation points of eq. (54), $P_{\omega,0}\left(\alpha^j,\frac{z_j^{(\omega)}}{V(\alpha^j)}\right)=0$ and $P_{\omega,1}\left(\alpha^j,\frac{z_j^{(\omega)}}{V(\alpha^j)}\right)=0$, $\forall j\in\Theta^c$. A basis reduction algorithm, e.g., the Mulders-Storjohann (MS) algorithm [34] [35], can be applied to further reduce $\mathcal{B}_\omega$ into a Gröbner basis $\mathcal{B}'_\omega$. It contains polynomials $P'_{\omega,0}(x,y)$ and $P'_{\omega,1}(x,y)$. Among them, the minimal one [3] is denoted as

$$\begin{aligned}Q^*_\omega(x,y)&=Q^{*(0)}_\omega(x)+Q^{*(1)}_\omega(x)y\\&=\min\{P'_{\omega,0}(x,y),P'_{\omega,1}(x,y)\}.\end{aligned} \tag{60}$$

Performing the inverse mapping of $Q^*_\omega(x,y)$, the interpolation polynomial $Q_\omega(x,y)$ w.r.t. the transformed test-vectors $\underline{\boldsymbol{z}}_\omega$ can be restored as

$$Q_\omega(x,y)=V(x)Q^{*(0)}_\omega(x)+Q^{*(1)}_\omega(x)y. \tag{61}$$

The estimated message polynomial of test-vector $\underline{\boldsymbol{z}}_\omega$ can be obtained by finding its y-root as

$$\hat{u}_\omega(x)=-\frac{V(x)Q^{*(0)}_\omega(x)}{Q^{*(1)}_\omega(x)}. \tag{62}$$

The intended decoded codeword $\underline{\hat{\boldsymbol{v}}}_\omega=\left(\hat{v}_0^{(\omega)},\hat{v}_1^{(\omega)},\ldots,\hat{v}_{n-1}^{(\omega)}\right)$ can be further obtained by

$$\hat{v}_j^{(\omega)}=\hat{u}_\omega\left(\alpha^j\right)+v_j^{(0)},\forall j. \tag{63}$$

C. LLOSD and LCC-BR Integration

The above mentioned LCC-BR decoding is further integrated with the LLOSD, forming the HSD for BCH codes. The LLOSD plays a primary role in the decoding, while the LCC-BR decoding provides extra TEPs for the LLOSD. It enhances the error-correction capability of the LLOSD with a limited decoding order τ. In this integration, the redundant the LCC-BR decoding test-vectors can be eliminated based on the LLOSD output list.

As introduced in Section V-A, the LCC-BR decoding is deployed if the LLOSD fails to produce the intended codeword. For the LCC-BR decoding, the LLOSD output list can be utilized to identify its redundant test-vectors. For this, the following *Lemma* 4 needs to be introduced.

Lemma 4. Given an LLOSD codeword candidate $\underline{\hat{\boldsymbol{v}}}$, let $d_{\mathrm{H}}(\underline{\hat{\boldsymbol{v}}},\underline{\boldsymbol{r}}_\omega)$ denote the Hamming distance between $\underline{\hat{\boldsymbol{v}}}$ and test-vector $\underline{\boldsymbol{r}}_\omega$. If $d_{\mathrm{H}}(\underline{\hat{\boldsymbol{v}}},\underline{\boldsymbol{r}}_\omega)\leqslant t$, where t is the error-correction capability of the code, $\underline{\boldsymbol{r}}_\omega$ can be skipped.

Proof. Given a test-vector $\underline{\boldsymbol{r}}_\omega$, if $d_{\mathrm{H}}(\underline{\hat{\boldsymbol{v}}},\underline{\boldsymbol{r}}_\omega)\leqslant t$, the LCC-BR decoding will produce $\underline{\hat{\boldsymbol{v}}}$ because each Chase decoding event can correct at most t errors.[4] Therefore, the LCC-BR decoding of $\underline{\boldsymbol{r}}_\omega$ is redundant. ■

Hence, some of the 2^η test-vectors can be eliminated based on the LLOSD output list. The following *Lemma* shows that generating the isomorphic module basis in the LCC-BR decoding and the RS systematic generator matrix in the LLOSD share common computations.

Lemma 5. With the module seed polynomial $\mathcal{G}(x)$ of eq. (55) and the Lagrange interpolation polynomials $\tilde{T}_j(x)$ of eq. (57), the RS systematic generator matrix entries $H_{\underline{\boldsymbol{u}}_i}(\alpha^j)$, where $j\in\Theta^c$ can be computed by

$$H_{\underline{\boldsymbol{u}}_i}(\alpha^j)=\frac{\alpha^i\mathcal{G}(\alpha^i)}{(\alpha^j-\alpha^i)\tilde{T}_j(\alpha^j)}. \tag{64}$$

[3]Note that with the re-encoding transform, polynomials of $\mathbb{F}_{2^m}[x,y]$ are arranged in the $(1,-1)$-revlex order [33]. Given two bivariate monomials $x^{\mu_1}y^{\nu_1}$ and $x^{\mu_2}y^{\nu_2}$ with $(1,-1)$-weighted degree, the $(1,-1)$-revlex order is defined as $x^{\mu_1}y^{\nu_1}<x^{\mu_2}y^{\nu_2}$, if $\mu_1-\nu_1<\mu_2-\nu_2$, or $\mu_1-\nu_1=\mu_2-\nu_2$ and $\nu_1<\nu_2$. Given $Q(x,y)=\sum_{\mu,\nu}Q_{\mu\nu}x^\mu y^\nu\in\mathbb{F}_{2^m}[x,y]$, the leading monomial of Q is defined as the maximum monomial $x^\mu y^\nu$ with $Q_{\mu\nu}\neq 0$ and the $(1,-1)$-weighted degree of Q is $\mu-\nu$. Moreover, the $(1,-1)$-revlex order is imposed on the polynomials in $\mathbb{F}_{2^m}[x,y]$ by comparing their leading monomials.

[4]Note that we only consider an interpolation multiplicity of one. Further based on *Remark* 1, the RS codes considered in this paper have a rate greater than half. Hence, the error-correction capacity of each Chase decoding event is t.

Proof. Based on eqs. (20)−(23), matrix entries $H_{\underline{\boldsymbol{u}}_i}\left(\alpha^j\right)$, with $j \in \Theta^c$ can be computed as in eq. (23). Further based on eq. (22), the Lagrange interpolation polynomials $\tilde{T}_j(x)$ of eq. (57) can be simplified into

$$\tilde{T}_j(x) = \alpha^j \prod_{j' \in \Theta^c, j' \neq j} (x - \alpha^{j'}). \tag{65}$$

Based on eq. (55), entries $H_{\underline{\boldsymbol{u}}_i}(\alpha^j)$ can be written as

$$\begin{aligned} H_{\underline{\boldsymbol{u}}_i}\left(\alpha^j\right) &= \frac{\alpha^i \prod_{j' \in \Theta^c}\left(\alpha^i - \alpha^{j'}\right)}{\alpha^j\left(\alpha^j - \alpha^i\right)\prod_{j' \in \Theta^c, j' \neq j}\left(\alpha^j - \alpha^{j'}\right)} \\ &= \frac{\alpha^i \mathcal{G}(\alpha^i)}{(\alpha^j - \alpha^i)\tilde{T}_j(\alpha^j)}. \end{aligned} \tag{66}$$

■

Therefore, in HSD, both the module seed polynomial $\mathcal{G}(x)$ and the Lagrange interpolation polynomials $\tilde{T}_j(x)$ can be computed once. They are then utilized in computing the RS systematic generator matrix $\mathbf{G}_{\mathrm{RS}}$ and the isomorphic module basis $\mathcal{B}_\omega$.

By considering there is only one test-vector that yields the intended TEP for the LLOSD, the test-vectors in the LCC-BR decoding are processed in a sequential manner. This exchanges the decoding latency for complexity. Under such an LCC-BR decoding paradigm, when a new test-vector $\underline{\boldsymbol{z}}_\omega$ $(\omega > 0)$ is decoded, its isomorphic module basis can be constructed based on that of the first test vector $\underline{\boldsymbol{z}}_0$, as demonstrated by the following *Lemma* 6.

Lemma 6. With the first isomorphic module basis $\mathcal{B}_0$, the isomorphic module basis $\mathcal{B}_\omega$ w.r.t. test-vector $\underline{\boldsymbol{z}}_\omega$ can be constructed as

$$P_{\omega,0}(x,y) = P_{0,0}(x,y), \tag{67}$$

$$P_{\omega,1}(x,y) = P_{0,1}(x,y) - \Upsilon_\omega(x), \tag{68}$$

where

$$\Upsilon_\omega(x) = \sum_{j \in \Psi, e_j^{(\omega)} \neq 0} \tilde{T}_j(x) \tag{69}$$

denotes the difference between polynomials $P_{\omega,1}(x,y)$ and $P_{0,1}(x,y)$.

Proof. With $P_{\omega,0}(x,y) = P_{0,0}(x,y) = \mathcal{G}(x)$, it interpolates the $n-k'$ points of eq. (54). If $P_{\omega,1}(x,y) = y - R_\omega(x)$, it also interpolates the points. Therefore, polynomials $P_{\omega,0}(x,y)$ and $P_{\omega,1}(x,y)$ span the module $\mathcal{M}_\omega$ of $\underline{\boldsymbol{z}}_\omega$. Based on eqs. (49) and (50),

$$\begin{aligned} R_\omega(x) &= \sum_{j \in \Theta^c} z_j^{(\omega)} \tilde{T}_j(x) \\ &= \sum_{j \in \Theta^c} (z_j^{(0)} + e_j^{(\omega)}) \tilde{T}_j(x), \end{aligned}$$

where $e_j^{(\omega)} = 0, \forall j \in \Psi^c$ and $e_j^{(\omega)} = r_j^{(\omega)} - r_j^{(0)}$. Hence, $R_\omega(x) = R_0(x) + \sum_{j \in \Psi, e_j^{(\omega)} \neq 0} \tilde{T}_j(x)$ and $P_{\omega,1}(x,y) = P_{0,1}(x,y) - \Upsilon_\omega(x)$. ■

The following *Lemma* 7 further shows evaluation of the root-finding outcome of eq. (62) over the MRPs provides a TEP for the LLOSD.

Lemma 7. Evaluation of the root-finding outcome $\hat{u}_\omega(x)$ of eq. (62) over the MRPs, i.e. $\underline{\hat{\boldsymbol{e}}}_\omega = (\hat{u}_\omega(\alpha^{j_0}), \hat{u}_\omega(\alpha^{j_1}), \ldots, \hat{u}_\omega(\alpha^{j_{k'-1}}))$, constitutes a TEP for the LLOSD.

Proof. Based on eqs. (62)−(63), the estimated codeword is

$$\begin{aligned} \underline{\hat{\boldsymbol{v}}}_\omega &= \left(\hat{v}_0^{(\omega)}, \hat{v}_1^{(\omega)}, \ldots, \hat{v}_{n-1}^{(\omega)}\right) \\ &= (\hat{u}_\omega(1), \hat{u}_\omega(\alpha), \ldots, \hat{u}_\omega(\alpha^{n-1})) + \underline{\hat{\boldsymbol{v}}}^{(0)}. \end{aligned} \tag{70}$$

With the systematic property of the RS code, the estimated codeword $\underline{\hat{\boldsymbol{v}}}_\omega$ can also be generated as

$$\begin{aligned} \underline{\hat{\boldsymbol{v}}}_\omega &= (\hat{u}_\omega(\alpha^{j_0}), \hat{u}_\omega(\alpha^{j_1}), \ldots, \hat{u}_\omega(\alpha^{j_{k'-1}})) \cdot \mathbf{G}_{\mathrm{RS}} + \underline{\hat{\boldsymbol{v}}}^{(0)} \\ &= \underline{\hat{\boldsymbol{e}}}_\omega \cdot \mathbf{G}_{\mathrm{RS}} + \underline{\hat{\boldsymbol{v}}}^{(0)}. \end{aligned} \tag{71}$$

Pairing with eq. (27), it can be realized that $\underline{\hat{\boldsymbol{e}}}_\omega = (\hat{u}_\omega(\alpha^{j_0}), \hat{u}_\omega(\alpha^{j_1}), \ldots, \hat{u}_\omega(\alpha^{j_{k'-1}}))$ constitutes a TEP for the LLOSD. ■

Moreover, based on the binary nature of BCH codes, the root-finding outcome of eq. (62) can be simplified into a partial operation that is characterized in the following *Theorem* 8.

Theorem 8. Given an interpolation polynomial $Q_\omega(x,y) = V(x)Q_\omega^{*(0)}(x) + Q_\omega^{*(1)}(x)y$, the estimated TEP $\underline{\hat{\boldsymbol{e}}}_\omega = (\hat{u}_\omega(\alpha^{j_0}), \hat{u}_\omega(\alpha^{j_1}), \ldots, \hat{u}_\omega(\alpha^{j_{k'-1}}))$ can be calculated as

$$\hat{u}_\omega(\alpha^j) = \begin{cases} 1, & \text{if } Q_\omega^{*(1)}(\alpha^j) = 0; \\ 0, & \text{otherwise.} \end{cases} \tag{72}$$

Proof. Since we only consider binary error patterns, i.e., $\underline{\hat{\boldsymbol{e}}}_\omega \in \mathbb{F}_2^{k'}$, if the entry $\hat{u}_\omega(\alpha^j) \neq 0$, it indicates an error at the position j. Based on eq. (62), the error positions over the MRPs can be determined by finding the roots of $Q_\omega^{*(1)}(x)$. We first consider the case of $Q_\omega^{*(1)}(\alpha^j) = 0$, where $j \in \Theta$. If α^j is a single root of $Q_\omega^{*(1)}(x)$, it possesses a factor $x - \alpha^j$. Based on eq. (52), $V(x)$ also has such a factor. Hence, $x - \alpha^j$ becomes a common factor for the numerator and denominator of eq. (62), which can be canceled out. Since $x - \alpha^j$ is not a common factor between $Q_\omega^{*(0)}(x)$ and $Q_\omega^{*(1)}(x)$ [36], $Q_\omega^{*(0)}(\alpha^j) \neq 0$ and $\hat{u}_\omega(\alpha^j) \neq 0$. Furthermore, if α^j is a double root of $Q_\omega^{*(1)}(x)$, $Q_\omega^{*(1)}(x)$ possesses a factor of $(x - \alpha^j)^2$. Since in this case, $V(x)Q_\omega^{*(0)}(x)$ only has a factor of $x - \alpha^j$, $Q_\omega^{*(1)}(x)$ is not a factor of $V(x)Q_\omega^{*(0)}(x)$. Based on eq. (62), the estimated message polynomial $\hat{u}_\omega(x)$ cannot be recovered. Therefore, if $Q_\omega^{*(1)}(\alpha^j) = 0$, $\hat{u}_\omega(\alpha^j) \neq 0$. It can be assigned as 1, indicating an error at position j. ■

It should be noted that this partial root-finding approach and that of [26] are different. By exploiting the property that with $j \in \Theta$, α^j is a root of polynomial $V(x)$, this partial root-finding approach is considerably simpler than that of [26]. Based on the LLOSD output, the following *Corollary* 9 further provides a criterion for skipping the interpolation polynomial whose partial root-finding result had already been processed by the LLOSD.

Corollary 9. Given an interpolation polynomial $Q_\omega(x,y) = V(x)Q_\omega^{*(0)}(x) + Q_\omega^{*(1)}(x)y$, if $\deg Q_\omega^{*(1)}(x) \leqslant \tau$, the corresponding estimated TEP $\underline{\hat{\boldsymbol{e}}}_\omega$ had been processed by the LLOSD.

Proof. Based on *Theorem* 8, evaluation of $\hat{u}_\omega(x)$ over the

Algorithm 2 The HSD Algorithm

Require: $\underline{\boldsymbol{\mathcal{Y}}}$, τ, η;
Ensure: $\underline{\hat{\boldsymbol{v}}}_{\text{opt}}$;
Perform the order-τ LLOSD as in ***Algorithm 1***;
If it fails to find the ML codeword
  Construct 2^{η} test-vectors as in eq. (49);
  For each test-vector $\underline{\boldsymbol{r}}_{\omega}$, **do**
    If $\underline{\boldsymbol{r}}_{\omega}$ does not satisfy the skipping rule of *Lemma 4*;
      Transform $\underline{\boldsymbol{r}}_{\omega}$ to $\underline{\boldsymbol{z}}_{\omega}$ as in eq. (50);
      Construct basis $\mathcal{B}_{\omega}$ as in eqs. (67) and (68);
      Reduce $\mathcal{B}_{\omega}$ into a Gröbner basis $\mathcal{B}'_{\omega}$;
      Determine $Q_{\omega}(x, y)$ as in eq. (61) ;
      Perform partial root-finding to obtain $\underline{\hat{\boldsymbol{e}}}_{\omega}$ as in eq. (72);
      Determine the estimated codeword $\underline{\hat{\boldsymbol{v}}}_{\omega}$ as in *Steps 6-10* of ***Algorithm 1*** ;
      If $\underline{\hat{\boldsymbol{v}}}_{\omega}$ satisfies the ML criterion as in eq. (14)
        Terminate the decoding;
  End for
Return $\underline{\hat{\boldsymbol{v}}}_{\text{opt}}$;

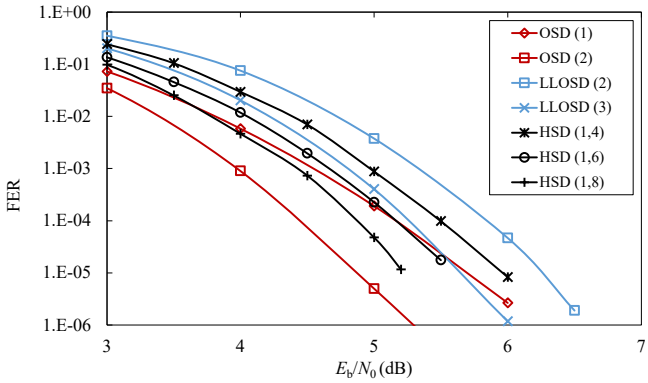


Fig. 7. Performance of the $(127,\ 99)$ BCH code.

MRPs are determined by $Q_{\omega}^{*(1)}(x)$, i.e., if $\hat{u}_{\omega}(\alpha^j) \neq 0$, where $j \in \Theta$, $Q_{\omega}^{*(1)}(\alpha^j) = 0$. Therefore, $\text{wt}(\underline{\hat{\boldsymbol{e}}}_{\omega}) \leqslant \deg Q_{\omega}^{*(1)}(x)$. Hence, if $\deg Q_{\omega}^{*(1)}(x) \leqslant \tau$, the estimated TEP $\underline{\hat{\boldsymbol{e}}}_{\omega}$ had been processed by the LLOSD. The partial root-finding of $Q_{\omega}(x, y)$ can be skipped. ■

After determining $\underline{\hat{\boldsymbol{e}}}_{\omega}$, the estimated codeword $\underline{\hat{\boldsymbol{v}}}_{\omega}$ can be further obtained as in eqs. (26) and (27). If the estimated codeword $\underline{\hat{\boldsymbol{v}}}_{\omega}$ satisfies the ML criterion [8], the HSD will be terminated and it yields $\underline{\hat{\boldsymbol{v}}}_{\omega}$ as the decoding output $\underline{\hat{\boldsymbol{v}}}_{\text{opt}}$. Otherwise, it will be added into the HSD output list. If the codeword that satisfies the ML criterion cannot be produced through the decoding, the codeword that yields the minimum correlation distance with $\underline{\boldsymbol{r}}$ will be selected as $\underline{\hat{\boldsymbol{v}}}_{\text{opt}}$. Summarizing the above description, the HSD algorithm is shown below as in *Algorithm 2*.

TABLE III
NUMERICAL RESULTS IN DECODING THE $(127,\ 99)$ BCH CODE

A. Decoding Complexity.

Algorithms	E_b/N_0 (dB)	Complexity		
		$\mathbb{F}_2$ oper.	$\mathbb{F}_{128}$ oper.	Floating oper.
OSD (1)	4	1.25×10^5		470
	5	1.24×10^5		78
	6	1.24×10^5		11
LLOSD (3)	4		2.64×10^5	9
	5		4.22×10^4	9
	6		6.71×10^3	9
HSD $(1,\ 8)$	4		9.12×10^4	27
	5		1.67×10^4	12
	6		5.65×10^3	9
HSD $(1,\ 6)$	4		2.52×10^4	13
	5		7.93×10^3	10
	6		5.44×10^3	9

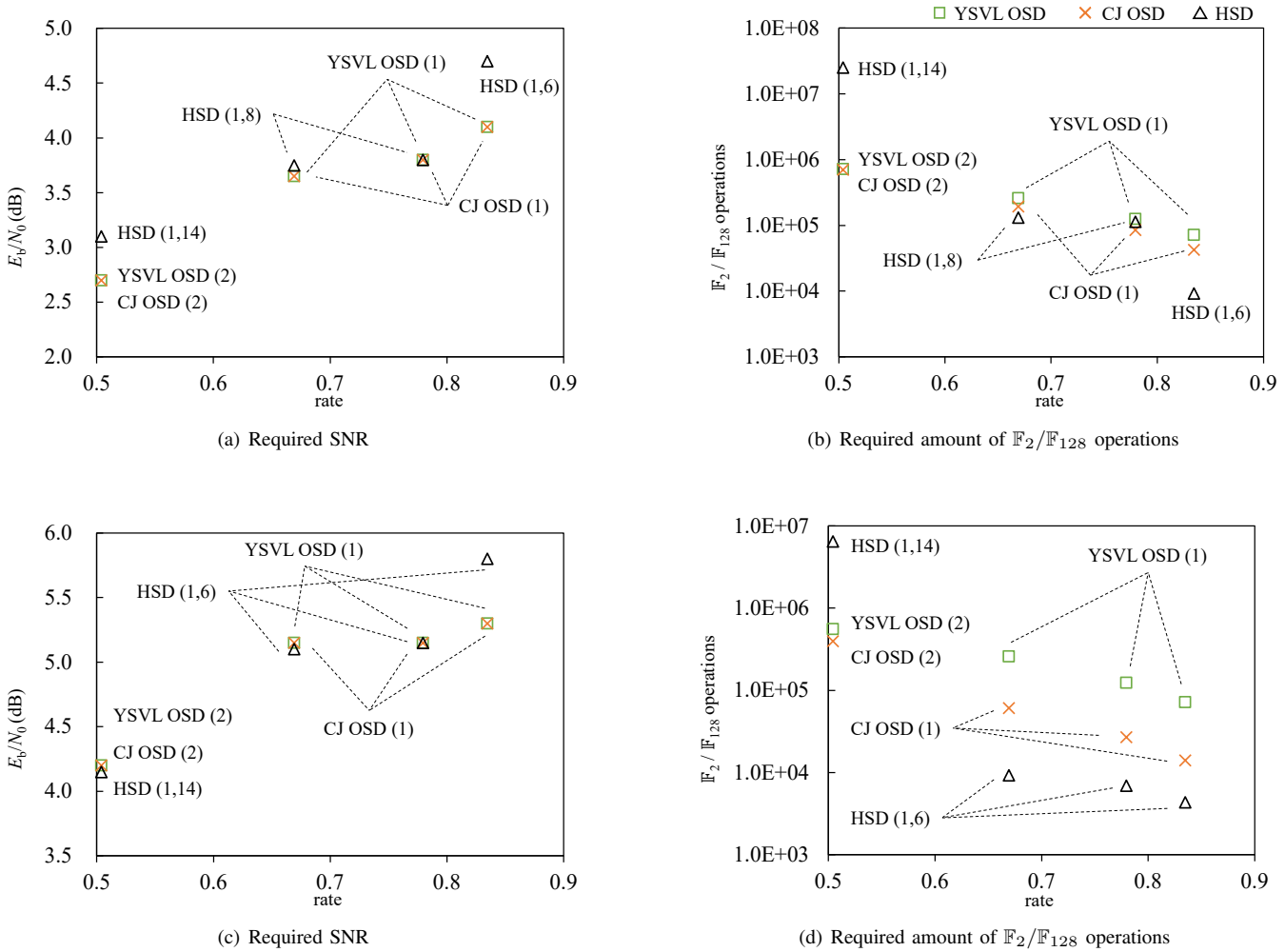
B. Decoding Latency.

Algorithms	E_b/N_0 (dB)	Latency (μs)
OSD (1)	4	9.90×10^1
	5	8.77×10^1
	6	8.64×10^1
LLOSD (3)	4	7.91×10^3
	5	1.15×10^3
	6	4.29×10^1
HSD $(1,\ 8)$	4	6.16×10^2
	5	9.42×10^1
	6	1.39×10^1
HSD $(1,\ 6)$	4	1.48×10^2
	5	3.15×10^1
	6	1.05×10^1

D. Decoding Performances

Fig. 7 shows the performance in decoding the $(127,\ 99)$ BCH code, where the HSD is parameterized by (τ, η). Note that for such a long BCH code, the LLOSD requires an impractical order to yield a high decoding performance. The decoding complexity becomes formidable. By integrating the LLOSD and the LCC-BR decoding, the HSD yields a high decoding performance with a low LLOSD order. E.g., both the HSD $(1,\ 6)$ and the HSD $(1,\ 8)$ can outperform the LLOSD (3) and the OSD (1).

Table III compares both the decoding complexity and latency of both the HSD, the LLOSD, and the OSD. It can be seen that the HSD $(1,\ 8)$ can not only outperform the LLOSD (3), but also being simpler. With the assistance of LCC-BR in providing TEPs, the HSD can yield a competent decoding performance with a smaller LLOSD order,leading to a reduced complexity. Meanwhile, both the HSD $(1,\ 6)$ and the HSD $(1,\ 8)$ require fewer $\mathbb{F}_{128}$ operations than the $\mathbb{F}_2$ operations required by the OSD (1). As the SNR increases, complexity of the HSD $(1,\ 6)$ and the HSD $(1,\ 8)$ converge. Our analysis in Section VI will show that as the SNR increases, the LCC-BR decoding is rarely deployed and $\overline{N}_{\text{TEPs}}$ converges to one. Their decoding complexities are only come from computing the RS



(a) Required SNR

(b) Required amount of $\mathbb{F}_2/\mathbb{F}_{128}$ operations

(c) Required SNR

(d) Required amount of $\mathbb{F}_2/\mathbb{F}_{128}$ operations

Fig. 8. Comparison of the YSVL OSD, the CJ OSD and the HSD for decoding the length-127 BCH codes in: (a) the required SNR in reaching a decoding FER of 10^{-2}; (b) the required amount of $\mathbb{F}_2/\mathbb{F}_{128}$ operations in reaching a decoding FER of 10^{-2}; (c) the required SNR in reaching a decoding FER of 10^{-4}; (d) the required amount of $\mathbb{F}_2/\mathbb{F}_{128}$ operations in reaching a decoding FER of 10^{-4}.

systematic generator matrix. Table III-B further compares the simulated decoding latency of the decoding algorithms. In both the HSD and the LLOSD, it is assumed that rows of $\mathbf{G}_{\text{RS}}$ are generated in parallel. It shows that both the HSD $(1,\ 6)$ and the HSD $(1,\ 8)$ can significantly reduce the decoding latency over the LLOSD.

To examine the applicable code rate range of the HSD, Figs. 8(a) and 8(b) compare the required SNR and complexity of the HSD, the YSVL OSD and the CJ OSD for decoding the length-127 BCH codes in reaching a decoding FER of 10^{-2}. Their decoding parameters are also indicated. At the rate 0.5, the HSD requires a larger SNR and a greater amount of $\mathbb{F}_{128}$ operations than the $\mathbb{F}_2$ operations that are required by the YSVL OSD and the CJ OSD. However, as the rate increases, the HSD starts to show its advantage over the YSVL OSD and the CJ OSD. When the rate reaches 0.83, the amount of $\mathbb{F}_{128}$ operations required by the HSD is only 22% and 13% of $\mathbb{F}_2$ operations required by the YSVL OSD and the CJ OSD, respectively. Figs. 8(c) and 8(d) further show the same measurements for the codes to reach a decoding FER of 10^{-4}. It can be seen that the required SNRs of these algorithms are close, except for the rate of 0.83. Similarly, as the code rate increases beyond 0.5, the HSD starts to show bigger advantage over other facilitated OSDs, which are demonstrated by the amount of $\mathbb{F}_{128}$ operations required by the HSD is far fewer than that of $\mathbb{F}_2$ operations required by the OSDs. Therefore, the HSD is more effective in decoding long and high rate BCH codes. In this case, the complexity of generating the systematic generator matrix becomes dominant, which is $O(n^2)$. It is asymptotically lower than the GE complexity of $O(n^3)$ in the OSD an its facilitated variants.

Finally, Fig. 9 shows the performance in decoding the $(255,\ 223)$ BCH code. The progressive LCC (PLCC) decoding [27] is also presented for comparison, which is parameterized by the number of the LRPs η. Their decoding complexity are also presented in Table IV. Note that both the HSD $(1,\ 6)$ and HSD $(1,\ 8)$ can outperform the OSD (1) and the LLOSD (2). They perform similarly as the PLCC (6) and PLCC (8), respectively. Table IV again shows that the HSD $(1,\ 8)$ requires fewer $\mathbb{F}_{256}$ operations than the $\mathbb{F}_2$ operations required by the OSD (1). It is also simpler than the PLCC (8) decoding. Meanwhile, the HSD $(1,\ 8)$ exhibits a

TABLE IV
NUMERICAL RESULTS OF COMPLEXITY IN DECODING THE $(255,\ 223)$ BCH CODE

Algorithms	E_b/N_0 (dB)	Complexity		
		$\mathbb{F}_2$ oper.	$\mathbb{F}_{256}$ oper.	Floating oper.
OSD (1)	4	3.40×10^5		2559
	5	2.91×10^5		621
	6	2.64×10^5		35
LLOSD (2)	4		5.60×10^4	6
	5		1.91×10^4	9
	6		1.15×10^4	9
PLCC (8)	4		1.58×10^6	60
	5		3.75×10^5	21
	6		4.71×10^4	10
HSD $(1,\ 8)$	4		4.16×10^5	59
	5		9.86×10^4	20
	6		1.33×10^4	10

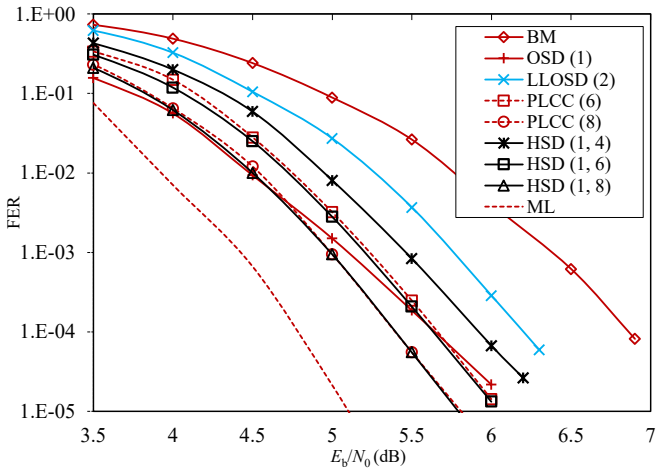


Fig. 9. Performance of the $(255,\ 223)$ BCH code.

higher complexity than the LLOSD (2), showing the cost of its decoding performance advantage. However, it should be noted that the HSD $(1,\ 8)$ and the PLCC (8) possess the same error-correction capability as the LLOSD (4). When $\eta = n - k' = 8$ and $t = 4$, the transmitted codeword will be included in the decoding output list if the number of errors in the MRPs is not greater than four. But the LLOSD (4) needs to process at most 1.5×10^8 TEPs, becoming impractical in decoding this long BCH code.

VI. DECODING COMPLEXITY

This section analyzes the complexity of both the LLOSD and the HSD algorithms. It is referred as the amount of finite field arithmetic operations in decoding a codeword. The theoretical characterization in part vindicates our complexity numerical results that have been presented so far. It also unveils the complexity insights of the proposals.

A. The LLOSD Complexity

The analysis of Section III-C shows that the LLOSD can only produce few BCH codeword candidates despite its decoding order. Hence, it only requires few floating point operations, which come from calculating the correlation distance of the estimated codeword and assessing whether they are the ML one. Therefore, the LLOSD complexity is dominated by performing $\mathbb{F}_{2^m}$ operations, which can also been demonstrated by our numerical results of Tables I-A and III-A. Let $\mathcal{C}_{\text{LLOSD}}$ denote the number of $\mathbb{F}_{2^m}$ operations required by the LLOSD. It comes from computing the RS systematic generator matrix and the re-encoding that finds the BCH codeword candidates, which are denoted as $\mathcal{C}_{\text{sys}}$ and $\mathcal{C}_{\text{rec}}$, respectively. The re-encoding involves the generation of the initial estimated RS codeword $\underline{\hat{\boldsymbol{v}}}^{(0)}$ and the reprocessing of the TEPs as in eq. (27), where their complexity are further denoted as $\mathcal{C}_{\text{ini}}$ and $\mathcal{C}_{\text{rep}}$, respectively. Therefore, the worst-case complexity of the LLOSD can be characterized as

$$\mathcal{C}_{\text{LLOSD}} = \mathcal{C}_{\text{sys}} + \mathcal{C}_{\text{ini}} + \mathcal{C}_{\text{rep}}. \tag{73}$$

Further let $\overline{N}_{\text{TEPs}}$ denote the average number of TEPs processed by the decoding, the average complexity of the LLOSD is

$$\mathcal{C}_{\text{LLOSD}} = \mathcal{C}_{\text{sys}} + \mathcal{C}_{\text{ini}} + \frac{\overline{N}_{\text{TEPs}}}{N_{\text{TEPs}}} \cdot \mathcal{C}_{\text{rep}}. \tag{74}$$

We now analyze $\mathcal{C}_{\text{sys}}$, $\mathcal{C}_{\text{ini}}$ and $\mathcal{C}_{\text{rep}}$ as follows.

Lemma 10. Complexity of computing the RS systematic generator matrix is $\mathcal{C}_{\text{sys}} = 2(n^2 - k'^2 + k')$.

Proof. The construction of matrix $\mathbf{G}_{\text{RS}}$ is realized as in eq. (23). Note that $\alpha^i \prod_{j' \in \Theta^c} \left(\alpha^i - \alpha^{j'}\right)$ and $\alpha^j \prod_{j' \in \Theta^c, j' \neq j} \left(\alpha^j - \alpha^{j'}\right)$ are the common factors for entries of each row and entries of each column, respectively. Hence, computing k' factors $\alpha^i \prod_{j' \in \Theta^c} \left(\alpha^i - \alpha^{j'}\right)$ with $i \in \Theta$ and $n - k'$ factors $\alpha^j \prod_{j' \in \Theta^c, j' \neq j} \left(\alpha^j - \alpha^{j'}\right)$ with $j \in \Theta^c$ require $2k'(n - k' + 1)$ and $2(n - k')^2$ $\mathbb{F}_{2^m}$ operations, respectively. In addition, $2k'(n - k')$ $\mathbb{F}_{2^m}$ operations are required to fully determine the $k'(n - k')$ entries. Hence, the complexity of computing $\mathbf{G}_{\text{RS}}$ is $\mathcal{C}_{\text{sys}} = 2(n^2 - k'^2 + k')$. ■

Note that in the LLOSD, the computation of $\mathbf{G}_{\text{RS}}$ replaces the GE in the OSD. The latter exhibits an $\mathbb{F}_2$ computational complexity of $O(n^3)$, while the above *Lemma* 10 shows that computing $\mathbf{G}_{\text{RS}}$ exhibits an $\mathbb{F}_{2^m}$ computational complexity of $O(n^2)$.

Lemma 11. Complexity of generating the initial estimated RS codeword and reprocessing of the TEPs are $\mathcal{C}_{\text{ini}} = k'(n-k')$ and $\mathcal{C}_{\text{rep}} \leq (n-k')\sum_{\rho=0}^{\tau}\rho\binom{k'}{\rho}$, respectively.

Proof. Computing the initial estimated RS codeword $\underline{\hat{\boldsymbol{v}}}^{(0)}$ as in eq. (24) requires $k'(n-k')$ $\mathbb{F}_{2^m}$ operations. With an LLOSD order τ, $N_{\text{TEPs}} = \sum_{\rho=0}^{\tau}\binom{k'}{\rho}$. Given a TEP $\underline{\boldsymbol{e}}^{(\omega)}_{[\Theta]}$ with $\text{wt}(\underline{\boldsymbol{e}}^{(\omega)}_{[\Theta]}) = \rho$, computing a codeword symbol requires ρ $\mathbb{F}_{2^m}$ operations. *Theorem 2* shows that the re-encoding process of a TEP can be terminated once a codeword symbol is identified as non-binary. Therefore, the complexity of reprocessing a TEP is upper bounded by the case when all re-encoded symbols are binary. After computing $n-k'$ codeword symbols in Θ^c, the complexity of reprocessing the TEPs is upper-bounded by

$$\mathcal{C}_{\text{rep}} \leq (n-k')\sum_{\rho=0}^{\tau}\rho\binom{k'}{\rho}.$$

■

Based on eq. (73), it can be seen that the order-τ LLOSD exhibits a worst-case complexity of $O(n^2+k'^{\tau})$. For the order-τ OSD, its worst-case complexity is $O(n^3+k^{\tau})$. Since $k' > k$ and a greater order is needed for the LLOSD to achieve a similar performance as the OSD, LLOSD exhibits a higher worst-case complexity. Both the LLOSD-B and the SLLOSD can also reduce the worst-case complexity. It should be noted that if the TEPs are processed sequentially, the LLOSD will be terminated once the ML codeword is found. Hence, the LLOSD does not need to process all TEPs. Fig.10 shows our numerical results of $\overline{N}_{\text{TEPs}}$ in decoding the $(31,\ 21)$ and the $(63,\ 45)$ BCH codes. The LLOSD is functioning with $\tau = 1$ and $\tau = 2$. It can be seen that as the SNR increases, the ML codeword can be found earlier through processing fewer TEPs. With $\overline{N}_{\text{TEPs}} \to 1$, the LLOSD complexity becomes only dominated by $\mathcal{C}_{\text{sys}}$ and $\mathcal{C}_{\text{ini}}$. In this case, the LLOSD exhibits a complexity of $O(n^2)$, which is lower than the asymptotic complexity $O(n^3)$ of the OSD.

B. The HSD Complexity

Besides the LLOSD complexity, the HSD complexity should further consider that of the BR interpolation and the partial root-finding, which are denoted as $\mathcal{C}_{\text{int}}$ and $\mathcal{C}_{\text{prf}}$, respectively. Let $\overline{N}_{\text{TVs}}$ denote the average number of test-vectors in a decoding event. The HSD complexity can be characterized as

$$\mathcal{C}_{\text{HSD}} = \mathcal{C}_{\text{LLOSD}} + \frac{\overline{N}_{\text{TVs}}}{2^{\eta}} \cdot (\mathcal{C}_{\text{int}} + \mathcal{C}_{\text{prf}}). \tag{75}$$

Lemma 12. Complexity of the BR interpolation is $\mathcal{C}_{\text{int}} = (n-k')^2 + 2^{\eta-1}\eta(n-k') + 2^{\eta+2}(n-k')(n-k'+1)$.

Proof. The BR interpolation includes the basis construction and reduction. The former is realized as in eqs. (67) and (68), in which $\mathcal{G}(x)$ and $\tilde{T}_j(x)$ are computed during the LLOSD.

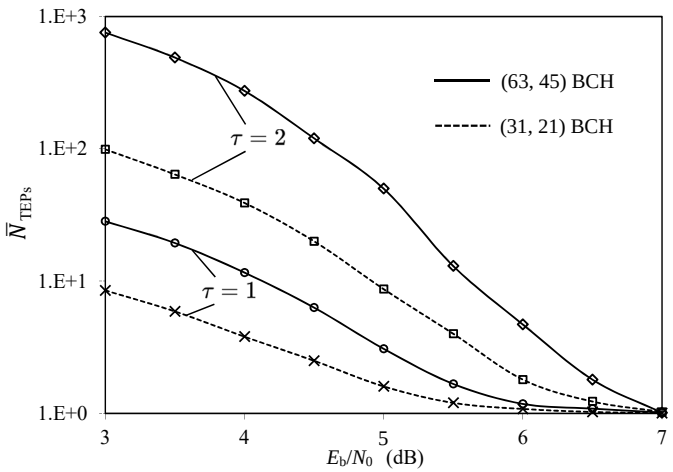


Fig. 10. $\overline{N}_{\text{TEPs}}$ in the LLOSD with $\tau = 1, 2$.

Since $\deg \mathcal{G}(x) = n-k'-1$ and $|\Theta^c| = n-k'$, computing $R_0(x)$ requires at most $(n-k')^2$ $\mathbb{F}_{2^m}$ operations. Eqs. (67) and (68) show that computing $R_{\omega}(x)$ with $\text{wt}(\underline{\boldsymbol{e}}^{(\omega)}) = i$ requires i linear combinations of $\tilde{T}_j(x)$ and $R_0(x)$. Hence, constructing bases for the $2^{\eta}-1$ isomorphic modules $\tilde{\mathcal{M}}_{\omega}$ also requires $(n-k')\cdot\sum_{i=1}^{\eta} i\binom{\eta}{i} = 2^{\eta-1}\eta(n-k')$ $\mathbb{F}_{2^m}$ operations. Furthermore, based on [34], the MS basis reduction requires at most $2^{\eta+2}(n-k')(n-k'+1)$ $\mathbb{F}_{2^m}$ operations. ■

Lemma 13. Complexity of the partial root-finding is $\mathcal{C}_{\text{prf}} = 2^{\eta}\cdot k'(n-k')$.

Proof. The partial root-finding is realized as in eq. (72). Based on [36], $\deg Q_{\omega}^{*(1)}(x) \leqslant \frac{n-k'}{2}$, computing $Q_{\omega}^{*(1)}(\alpha^j)$ with $j \in \Theta$ requires $(n-k')$ $\mathbb{F}_{2^m}$ operations. Hence, the complexity of partial root-finding is $\mathcal{C}_{\text{prf}} = 2^{\eta}\cdot k'(n-k')$. ■

It should be noted that if the LLOSD finds the BCH codeword that satisfies the ML criterion, the LCC-BR decoding will not be deployed, and $\overline{N}_{\text{TVs}} = 0$. Consequently, the HSD complexity converges to that of the LLOSD. This is often the case if the channel condition is sufficiently good. Fig. 11 shows our numerical results of $\overline{N}_{\text{TVs}}$ in decoding the $(63,\ 39)$ and the $(255,\ 223)$ BCH codes. In the HSD, the LLOSD is functioning with $\tau = 1$ and the LCC-BR decoding is functioning with $\eta = 4$ and 6. Note that it is assumed that in LCC-BR decoding, the test-vectors are decoded sequentially. It can be seen that as the SNR increases, fewer test-vectors are decoded by the LCC-BR decoding, leading to a lower HSD complexity. When SNR $= 6$ dB, $\overline{N}_{\text{TVs}}$ converges to zero. This implies most decoding events terminate without deploying the LCC-BR decoding. In this case, generating the RS systematic generator matrix dominates the decoding complexity.

VII. Conclusion

This paper has proposed the LLOSD and its enhanced variants for BCH codes. They are designed based on the property that BCH codes are binary subcodes of RS codes. Consequently, BCH codeword candidates can be produced through the RS systematic generator matrix whose entries can

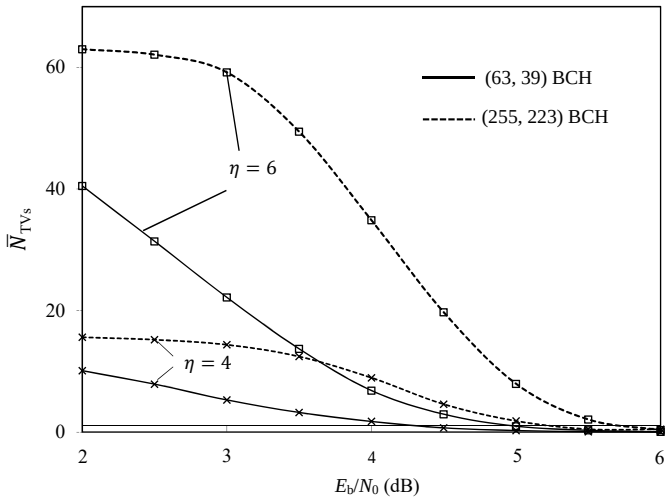


Fig. 11. $\overline{N}_{\mathrm{TVs}}$ in the HSD with $\tau = 1$ and $\eta = 4,\ 6$.

be computed in parallel. It replaces the latency-orienting GE in the conventional OSD. It has been shown that by eliminating the RS re-encoding that does not produce a binary codeword, the LLOSD requires far fewer $\mathbb{F}_{2^m}$ operations than the $\mathbb{F}_2$ operations required by the OSD. A concatenated perspective of the LLOSD has been presented, showing the LLOSD can be interpreted as a serial concatenation between the parity-checker of a punctured BCH code and a systematic encoder of a linear block code. Such a re-enterpretation unveils the low-complexity feature of the LLOSD. This concatenated perspective also enables the conversion of non-binary re-encoding operations in the LLOSD into binary operations, further facilitating the LLOSD. The segmented LLOSD has also been proposed to reduce the decoding complexity. By employing the segmented LLOSD in a concatenated manner, the proposed LLOSD variant has demonstrated advantages over other state-of-the-art OSD variants. In order to decode longer BCH codes, the HSD has been proposed. It integrates the LLOSD and the LCC-BR decoding. The latter provides extra TEPs for the LLOSD, achieving a high performance with a low decoding order. Complexity and performance of the proposed decoding have been analyzed. Our simulation results have demonstrated that the LLOSD yields both complexity and latency advantages over the OSD. The HSD is more effective in decoding longer BCH codes. They also exhibit the complexity advantage over several state-of-the-art decoding for BCH codes.

ACKNOWLEDGMENT

This work is sponsored in part by the National Natural Science Foundation of China (NSFC) with project IDs 62471503, 62071498, and in part by the Natural Science Foundation of Guangdong Province (NSFGP) with project ID 2024A1515010213.

REFERENCES

[1] G. Liva *et al.*, "Code design for short blocks: A survey," 2016, arXiv:1610.00873.

[2] J. Van Wonterghem *et al.*, "Performance comparison of short-length error-correcting codes," in *Proc. IEEE Symp. Commun. Veh. Technol. (SCVT)*, Mons, Belgium, Nov. 2016, pp. 1–6.

[3] Y. Polyanskiy, V. Poor, and S. Verdu, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359, 2010.

[4] M. Coşkun *et al.*, "Efficient error-correcting codes in the short block-length regime," *Phys. Commun.*, vol. 34, pp. 66–79, 2019.

[5] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inf. Theory*, vol. 41, no. 5, pp. 1379–1396, 1995.

[6] Y. Wu and C. Hadjicostis, "Soft-decision decoding using ordered recodings on the most reliable basis," *IEEE Trans. Inf. Theory*, vol. 53, no. 2, pp. 829–836, 2007.

[7] C. Yue *et al.*, "A revisit to ordered statistics decoding: distance distribution and decoding rules," *IEEE Trans. Inf. Theory*, vol. 67, no. 7, pp. 4288–4337, 2021.

[8] T, Kaneko *et al.*, "An efficient maximum-likelihood decoding algorithm for linear block codes with algebraic decoder," *IEEE Trans. Inf. Theory*, vol. 40, no. 2, pp. 320–327, 1994.

[9] W. Jin and M. Fossorier, "Probabilistic sufficient conditions on optimality for reliability based decoding of linear block codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Seattle, WA, USA, July 2006.

[10] S. Alnawayseh and P. Loskot, "Ordered statistics-based list decoding techniques for linear binary block codes," *EURASIP J. Wirel. Commun. Netw.*, vol. 2012, no. 1, pp. 1–12, Dec.2012.

[11] A. Valembois and M. Fossorier, "Box and Match techniques applied to soft-decision decoding," *IEEE Trans. Inf. Theory*, vol. 50, no. 5, pp. 796–810, 2004.

[12] M. Fossorier, "Reliability-based soft-decision decoding with iterative information set reduction," *IEEE Trans. Inf. Theory*, vol. 48, no. 12, pp. 3101–3106, 2002.

[13] W. Jin and M. Fossorier, "Reliability-based soft-decision decoding with multiple biases," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 105–120, 2007.

[14] Y. Wu and C. Hadjicostis, "Soft-decision decoding of linear block codes using preprocessing and diversification," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 378–393, Jan. 2007.

[15] S. Yu and Q. Huang, "Hard reliability-based ordered statistic decoding and its application to McEliece public key cryptosystem," *IEEE Commun. Lett.*, vol. 26, no. 3, pp. 490–494, 2022.

[16] C. Choi and J. Jeong, "Fast soft decision decoding algorithm for linear block codes using permuted generator matrices," *IEEE Commun. Lett.*, vol. 25, no. 12, pp. 3775–3779, 2021.

[17] C. Yue *et al.*, "Ordered statistics decoding with adaptive Gaussian elimination reduction for short codes". in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Rio de Janeiro, Brazil, 2022, pp. 492-497.

[18] M. Jalaleddine *et al.*, "Partial ordered statistics decoding with enhanced error patterns". in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Taipei, ROC, June. 2023.

[19] K. Duffy, J. Li, and M. Medard "Capacity-achieving guessing random additive noise decoding," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4023–4040, July 2019.

[20] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inf. Theory*, vol. IT-15, no. 1, pp. 122–127, Jan. 1969.

[21] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic geometric codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 1, pp. 1757–1767, Mar. 1999.

[22] R. Kötter and A. Vardy, "Algebraic soft-decision decoding of Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 49, no. 11, pp. 2809–2825, Nov. 2003.

[23] J. Bellorado and A. Kavčić, "Low-complexity soft-decoding algorithms for Reed-Solomon codes–Part I: An algebraic soft-in hard-out Chase decoder," *IEEE Trans. Inf. Theory*, vol. 56, no. 3, pp. 945-959, Mar. 2010.

[24] X. Zhang and Y. Zheng, "Systematically re-encoded algebraic soft-decision Reed-Solomon decoder, " *IEEE Trans. Cir. Syst.-II: Express Briefs*, vol. 59, no. 6, pp. 376–380, Jun. 2012.

[25] Y. Wu, "Fast Chase decoding algorithms and architectures for Reed-Solomon codes," *IEEE Trans. Inf. Theory*, vol. 58, no. 1, pp. 109–129, Jan. 2012.

[26] X. Zhang, "An efficient interpolation-based Chase BCH decoder," *IEEE Trans. Cir. Syst.-II: Express Briefs*, vol. 60, no. 4, pp. 212–216, Apr. 2013.

[27] J. Xing, L. Chen, and M. Bossert, "Low complexity Chase decoding of Reed-Solomon codes using module," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6012–6022, Oct. 2020.

[28] M. Fossorier and S. Lin, "Complementary reliability-based decodings of binary linear block codes," *IEEE Trans. Inf. Theory*, vol. 43, no. 5, pp. 1667–1672, Sep. 1997.
[29] M. Fossorier and S. Lin, "A unified method for evaluating the error-correction radius of reliability-based soft-decision algorithms for linear block codes," *IEEE Trans. Inf. Theory*, vol. 44, pp. 691–700, 1998.
[30] E. Berlekamp, "Algebraic coding theory," *New York, NY, USA:McGraw-Hill*, 1968.
[31] P. Delsarte, "On subfield subcodes of modified Reed-Solomon codes (Corresp.)," *IEEE Trans. Inf. Theory*, vol. 21, no. 5, pp. 575-576, Sep. 1975.
[32] F. MacWilliams and N. Sloane, The theory of error correcting codes. Elsevier, 1977, vol. 16.
[33] R. Kötter, J. Ma, and A Vardy, "The re-encoding transformation in algebraic listdecoding of Reed-Solomon codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 633–647, Feb. 2011.
[34] T. Mulders and A. Storjohann, "On lattice reduction for polynomial matrices," *J. Symbolic Comput.*, vol. 35, no. 4, pp. 377–401, Apr. 2003.
[35] J. Xing, L. Chen and M. Bossert, "Progressive algebraic soft decoding of Reed-Solomon codes using module minimization," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, USA, Jun. 2018. pp. 11-15.
[36] J. Zhu and X. Zhang, "Factorization-free low-complexity Chase soft decision decoding of Reed-Solomon codes," in *Proc. IEEE Int. Symp. Circuits Syst*., May 2009, pp. 2677–2680.
[37] M. Helmling *et al.*, "Database of channel codes and ML simulation results," www.uni-kl.de/channel-codes, 2019.

Lijia Yang (Student Member, IEEE) received the B.Sc. degree in information engineering from Jinan University, Guangzhou, China, in 2020, and the M.Sc. degree in communication and information system from Sun Yat-sen University, Shenzhen, China, in 2023. His research interests include channel coding and data communications. He joined OPPO Guangdong Mobile Communications Co., Ltd. in 2023.

Jianguo Zhao (Student Member, IEEE) received the B.Sc. degree in information engineering and the M.Sc. degree in information and communication engineering from Sun Yat-sen University, China, in 2021 and 2024, respectively. His research interests include information and coding theory.

Xihao Li (Student Member, IEEE) received the B.Sc. and the M.Sc. degree in information and communication engineering from Sun Yat-sen University, China, in 2022 and 2025, respectively. His research interests include information and coding theory.

Li Chen (Senior Member, IEEE) received the B.Sc. degree in applied physics from Jinan University, Guangzhou, China, in 2003, and the M.Sc. degree in communications and signal processing and the Ph.D. degree in communications engineering from Newcastle University, U.K., in 2004 and 2008, respectively. From 2007 to 2010, he was a Research Associate with Newcastle University. In 2010, he returned to China as a Lecturer with the School of Information Science and Technology, Sun Yat-sen University, Guangzhou. From 2011 and 2016, he became an Associate Professor and a Professor of the University, respectively. From 2013, he became the Associate Head of Department of Electronic and Communication Engineering (ECE). From Aug. 2017 to Mar. 2020, he was the Deputy Dean of the School of ECE of SYSU. He specializes in channel coding, in particular, algebraic coding theory and techniques. From Jul. 2015 to Jun. 2016, he took sabbatical visiting both Ulm University in Germany and University of Notre Dame in U.S. He has also visited the Institute of Network Coding, the Chinese University of Hong Kong for several occasions. He founded and chairs the IEEE Information Theory Society Guangzhou Chapter, which was awarded Chapter-of-the-Year by the Society in 2021. He was a member of the IEEE Information Theory Society Board of Governors and chairing the Conference Committee (2022 - 2024). He was awarded The Chinese Information Theory Young Researcher award by the Chinese society of Electronics in 2014. He is an Associate Editor (AE) of the IEEE Transactions on Information Theory, and was an AE of the IEEE Transactions on Communications (2018 - 2023). He has been organizing several international conferences and workshops, including the 2018 IEEE Information Theory Workshop (ITW) in Guangzhou and the 2022 IEEE East Asian School of Information Theory (EASIT) in Shenzhen, for which he was the General Co-chair. He was also the TPC Co-chair of the 2022 IEEE / CIC International Conference on Communications in China (ICCC) in Foshan. He is the General Co-chair of the IEEE International Symposium on Information Theory (ISIT) 2026 in Guangzhou. He likes music and literature.

Huazi Zhang (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees from the Institute of Information and Communication Engineering, Zhejiang University, in 2008 and 2013, respectively. From 2011 to 2013, he was a Visiting Researcher with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, USA. From 2013 to 2014, he was a Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. From 2014 to 2015, he was a Research Scholar with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA. He joined Huawei Technologies Company, Ltd. in 2015. Since then, he has engaged in research projects on advanced wireless communications involving channel coding and signal processing techniques, and engaged in multiple standardization activities. His contribution led to the adoption of many state of-the-art research results into 5G standards, as well as the subsequent commercial rollout. His research interests include channel coding, information theory, and signal processing for wireless communications, with focus on theoretical analysis, algorithm design, and hardware implementations for 5G and beyond.

Jiajie Tong (Member, IEEE) He is currently a Research Engineer with Huawei Technologies Co., Ltd. His current research interests include channel coding schemes with a focus on algorithm design and hardware implementations.