Improved Miscorrection Detection for Generalized Integrated Interleaved BCH Codes

Zhenshan Xie and Xinmiao Zhang
Dept. of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210, USA
Email: {xie.855, zhang.8952}@osu.edu

***Abstract*—The generalized integrated interleaved (GII) codes can nest BCH sub-codewords to form more powerful BCH codewords. GII codes enable hyper-speed decoding and achieve excellent error-correction capability. They are among the best candidates for the new storage class memories (SCMs). However, SCMs require high code rate and short codeword length. In this case, the GII sub-codewords have small correction capability, and miscorrections on the sub-words lead to severe performance degradation. In previous work, higher-order nested syndromes are computed to detect and mitigate miscorrections in GII decoding. These computations cause long decoding latency, even though they can be implemented by sharing the hardware architecture for other decoding steps. This paper proposes three methods to optimize the miscorrection detection by investigating dominant error patterns leading to miscorrections. The first scheme is to skip the nested syndrome checking for cases that are less likely miscorrected. To make up for the performance loss caused by the first scheme, our second approach exploits 2-bit extended BCH codes to protect each sub-codeword. In addition, the third scheme is developed to protect all sub-codewords using extra parity bits while keeping the code rate loss negligible. Formulas are also derived to estimate the achievable performance. Applying the proposed optimizations, the average nested decoding latency is reduced by 43% for an example GII code with 3-error-correcting sub-codewords at input bit error rate 10^{-3}, while the performance loss and complexity overheads are negligible.**

***Index Terms*—BCH codes, Generalized integrated interleaved codes, Miscorrection, Storage class memories**

I. Introduction

Generalized integrated interleaved (GII) codes [1], [2] nest Reed-Solomon (RS) or BCH sub-codewords to form more powerful RS or BCH codewords. The GII decoding is divided into two stages: sub-word decoding and nested decoding, which may involve multiple rounds depending on the error number. In most cases, there are a small number of errors, and only the decoding of individual short sub-words is carried out. Hence GII decoding can achieve very high throughput with low complexity. Additionally, by utilizing the stronger nested codewords, extra errors are correctable. Therefore, GII codes have much better error-correcting performance than traditional RS or BCH codes that achieve similar decoding throughput.

The hyper throughput and excellent correction capability make GII codes among the best candidates for storage class memories (SCMs). The sensing latency of new SCMs is much shorter than that of Flash memories. Error-correcting codes with relative short length, *e.g.* several thousand bits, and high code rate, *e.g.* 90%, are required to achieve the speed potential of SCMs. For such applications, the individual sub-codewords of GII codes have small correction capability, such as $t_0 = 3$. In this case, a sub-codeword corrupted with extra errors may be decoded as another sub-codeword with higher probability. Such miscorrections degrade the GII decoding performance significantly. The reason is that if a received sub-word is miscorrected and not identified, then the more powerful nested codewords cannot be utilized to correct extra errors. From our previous study [3], the sub-word miscorrection causes several orders of magnitude degradation on the decoding frame error rate (FER) for an example code with $t_0 = 3$.

It was proposed in [4] to mitigate miscorrections by carrying out decoding on the nested words directly. However, the number of error patterns correctable by this scheme is very limited and the performance degradation is severe. In [5], the first suspicious sub-word whose number of errors is found to be t_0 from sub-word decoding is sent to the nested decoding. This scheme only locates a small portion of the miscorrections for GII codes with small t_0. Three methods for miscorrection mitigation were developed in [3] to bring the actual FER very close to the theoretical FER of GII codes. In detail, [3] uses 1-bit extended BCH (eBCH) code, degree of error locator polynomial, and higher-order nested syndromes to detect miscorrections. However, the computation of the nested syndromes requires many extra clock cycles and leads to significant decoding latency overhead.

This paper proposes three optimizations on miscorrection mitigation to significantly reduce the GII decoding latency with negligible degradation on the FER. Through analyzing the dominant error patterns leading to miscorrections, the first scheme is developed to skip the nested syndrome checking when miscorrections are unlikely to happen. This helps to substantially reduce the probability of activating the nested syndrome checking and hence the nested decoding latency. To cover the miscorrections that are not detected due to the first modification, our second and third approaches add more parities, which can be computed with no latency overhead and negligible complexity. The second approach exploits 2-bit eBCH codes to protect each individual sub-codeword. To further improve the performance with only very slight code rate loss, our third scheme adds multiple global parity bits to protect all sub-codewords. Moreover, formulas are derived to estimate the FER after applying the proposed schemes. For an example GII-BCH code over $GF(2^{10})$ with 3-error-correcting

This work was supported in part by Kioxia Corporation and by the National Science Foundation under Award No. 2011785.

BCH sub-codewords, the average nested decoding latency is reduced by 43% using the proposed schemes with negligible performance loss.

This paper is organized as follows. Section II introduces GII codes and miscorrections. Section III describes the three proposed optimization schemes for miscorrection mitigation. Section IV summarizes the proposed GII decoding procedure and conclusions follow in Section V.

II. GII-BCH Codes and Miscorrections

A GII-BCH $[m, v]$ code is constructed using $v+1$ BCH codes $\mathcal{C}_v \subseteq \mathcal{C}_{v-1} \subseteq \cdots \subseteq \mathcal{C}_1 \subset \mathcal{C}_0$. They are defined over $GF(2^q)$ with error-correction capabilities $t_v \geq t_{v-1} \geq \cdots \geq t_1 > t_0$. A GII codeword consists of m length-n sub-codewords $c_i(x)$ and v nested codewords $\tilde{c}_l(x) \in \mathcal{C}_{v-l}$ are formed by linear combinations of $c_i(x)$. A GII-BCH $[m, v]$ code is formally defined as [2]:

$$\begin{aligned}\mathcal{C} \triangleq \Big\{ & [c_0(x), c_1(x), \cdots, c_{m-1}(x)] : c_i(x) \in \mathcal{C}_0, \\ & \tilde{c}_l(x) = \sum_{i=0}^{m-1} \alpha^{il}(x) c_i(x) \in \mathcal{C}_{v-l}, 0 \leq l < v \Big\},\end{aligned} \tag{1}$$

where α is a primitive element of $GF(2^q)$ and $\alpha^{il}(x)$ is the polynomial form of the standard basis representation of α^{il}.

GII-BCH decoding has two stages. The first is the traditional t_0-error-correcting BCH decoding on individual received sub-word $y_i(x) = c_i(x) + e_i(x)$, where $e_i(x)$ is the error polynomial. $2t_0$ syndromes are first calculated as the evaluation values $S_j^{(i)} = y_i(\alpha^{j+1})$ $(0 \leq j < 2t_0)$. If all syndromes are zero, the sub-word is regarded as error-free. Otherwise, a key equation solver (KES), such as the Berlekamp-Massey (BM) algorithm, computes the error locator polynomial $\Lambda(x)$ using the $2t_0$ syndromes. Then, error locations can be found from the inverse roots of $\Lambda(x)$. Decoding success is declared if the number of distinct roots of $\Lambda(x)$ equals the degree of $\Lambda(x)$, which is denoted by $\deg(\Lambda(x))$. Otherwise, the sub-word decoding fails.

The second-stage nested decoding is activated when there are failures or miscorrections in sub-word decoding due to extra errors. $2t$ syndromes are needed to correct t errors. Extra higher-order syndromes can be derived from nested words $\tilde{y}_l(x) = \sum_{i=0}^{m-1} \alpha^{il}(x) y_i(x)$. Let the indices of the $b \leq v$ sub-words with extra errors be $i_0, i_1, \cdots, i_{b-1}$. From (1), the nested codewords $\tilde{c}_l(x)$ $(0 \leq l < b)$ are at least t_1-error-correcting. Hence their higher-order syndromes with indices $2t_0 \leq j < 2t_1$ are computed as $\tilde{S}_j^{(l)} = \tilde{y}_l(\alpha^{j+1})$. From the nesting in (1), for sub-words with extra errors, the higher-order syndromes with indices $2t_0 \leq j < 2t_1$ can be computed as

$$\left[S_j^{(i_0)}, S_j^{(i_1)}, \cdots, S_j^{(i_{b-1})}\right]^T = A^{-1}\left[\tilde{S}_j^{(0)}, \tilde{S}_j^{(1)}, \cdots, \tilde{S}_j^{(b-1)}\right]^T, \tag{2}$$

where the matrix entry $A_{u,w}$ equals $\alpha^{i_w u(j+1)}$ $(0 \leq u, w < b)$ [2]. After that, up to t_1 errors can be corrected for each of the b sub-words. If there are b' sub-words that remain to be corrected, then the first b' nested words, which are at least t_2-error-correcting, are utilized to compute the following $2(t_2 - t_1)$ higher-order syndromes in a similar way. This process is repeated for up to v rounds.

Sort the number of errors in the received sub-words as $\tau_0 \geq \tau_1 \geq \cdots \geq \tau_{m-1}$. The error pattern is correctable if $\tau_l \leq t_{v-l}$ for $0 \leq l \leq v$. Accordingly, the theoretical FER without miscorrections can be estimated by the formula in [2]. For SCM applications using around 256 parity bits to protect 2560 data bits, GII-BCH [4,3] codes over $GF(2^{10})$ with sub-codeword length $n = (2560 + 256)/4 = 704$ can be utilized to achieve a good tradeoff on error-correcting performance and decoding complexity. For this codeword length and redundancy, the best FER is achieved when $[t_0, t_1, t_2, t_3] = [3, 5, 6, 11]$.

To achieve the theoretical performance in actual GII decoding, the miscorrection issue needs to be handled. When a t-error-correcting BCH codeword is corrupted with more than t errors, its distance to another codeword may be smaller than t. In this case, it will be decoded as another codeword and miscorrection occurs. Miscorrections happen more likely for smaller t. If miscorrected sub-words are not identified and not sent to further nested decoding, it will cause decoding failure even if the error pattern is within the correction capability of the GII code. As a result, GII codes consisting of sub-codewords with small t_0, such as the [4,3] code with $t_0 = 3$, would have orders of magnitude performance degradation if miscorrections are not detected and mitigated [3].

Miscorrections can be detected by higher-order nested syndromes [3]. If any of them is nonzero, miscorrections are detected. This is because that the nested codewords are linear combinations of the sub-codewords and they have higher correction capabilities. Another two methods were also developed in [3] to better identify the miscorrected sub-words with negligible complexity overhead and no extra latency. The first utilizes 1-bit eBCH codes. If the bit-wise XOR result of the received sub-word is different from the parity of the number of roots of $\Lambda(x)$, then miscorrections are found. In the second approach, $\deg(\Lambda(x)) > t_i$ in nested decoding round i indicates miscorrections. Here, the first-stage individual sub-word decoding is considered as nested decoding round 0. Fig. 1 shows the nested decoding latency for the example code with $[t_0, t_1, t_2, t_3] = [3, 5, 6, 11]$ using the approaches from [3]. In the worst case, three nested decoding rounds are activated to handle three sub-words with extra errors. The nested decoding of each sub-word includes higher-order nested syndrome computation, syndrome conversion, nested KES, and Chien search [6]. The number of clock cycles for each step needed in the hardware implementation targeting high speed and efficiency is included in the parenthesis in Fig. 1. Unlike those for the nested decoding itself, the nested syndromes for miscorrection detection before each nested decoding round should be computed based on the previous decoding output. Although they can be calculated by sharing the same hardware, extra clock cycles are needed and they account for a significant portion of the overall nested decoding latency.

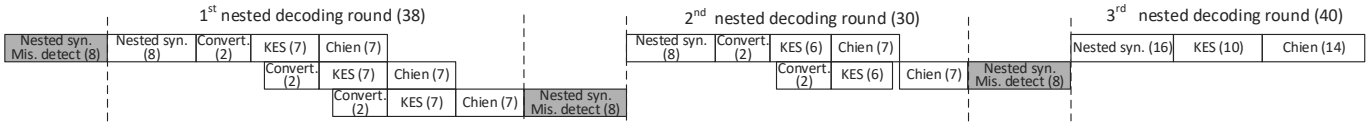

Fig. 1. Worst-case latency of the nested decoding incorporating miscorrection detection of the GII-BCH [4,3] code with $[t_0, t_1, t_2, t_3] = [3, 5, 6, 11]$.

III. Optimized Miscorrection Detection

This section proposes three optimization schemes for miscorrection detection to reduce the nested decoding latency with negligible performance loss and complexity overhead. The proposed methods include skipping the nested syndrome checking when all sub-words have small $\deg(\Lambda(x))$ and at the end of those nested decoding rounds with higher correction capabilities, exploiting 2-bit eBCH codes for each sub-codeword, and adopting multiple global parities to protect all sub-codewords. Formulas are also derived to estimate the resulted FERs.

A. Skipping Nested Syndrome Checking when Miscorrections are Less Likely to Happen

In [3], if any of the sub-words fails to be decoded or is detected to be miscorrected by using either the 1-bit eBCH code or checking if $\deg(\Lambda(x)) > t_i$, then up to $v - i$ sub-words are chosen and sent to nested decoding round $i + 1$ to correct extra errors. Otherwise, higher-order nested syndromes are computed to check for potential miscorrections. The nested syndromes tell if any of the sub-words are miscorrected. If every sub-word is less likely to be miscorrected, then the nested syndrome checking can be skipped to shorten the nested decoding latency with small degradation on the FER.

From our simulations, it is observed that the probability of a received sub-word with more than t errors being miscorrected decreases significantly with $\deg(\Lambda(x))$. For example, if a 3-error-correcting BCH sub-codeword with $n = 704$ is corrupted by 6 errors, the probabilities of miscorrections when $\deg(\Lambda(x)) = 3, 2, 1$ are $5.4 \times 10^{-2}, 2.5 \times 10^{-4}, 6 \times 10^{-7}$, respectively. From the above observation, this paper proposes to skip the nested syndrome checking if the $\deg(\Lambda(x))$ from the decoding of every sub-word does not exceed a threshold, th. Apparently, th should be less than t_0.

The FER degradation caused by the above skipped nested syndrome checking at lower input bit error rate (BER) cannot be simulated in practical time and can be analyzed as follows. If a sub-word has $\deg(\Lambda(x)) \leq th$ at the end of nested decoding round i, it can be corrupted by either up to th errors or more than t_i errors. From [3], in most of the cases that miscorrections ever cause degradation on the FER, there is only one sub-word miscorrected but not identified, and all the other sub-words are successfully decoded. Hence, the dominating error patterns leading to performance loss when the nested syndromes are not checked in the case of $\deg(\Lambda(x)) \leq th$ have one sub-word miscorrected with more than t_i errors and each of the other sub-words with up to th errors. As a result, the FER degradation compared to the case

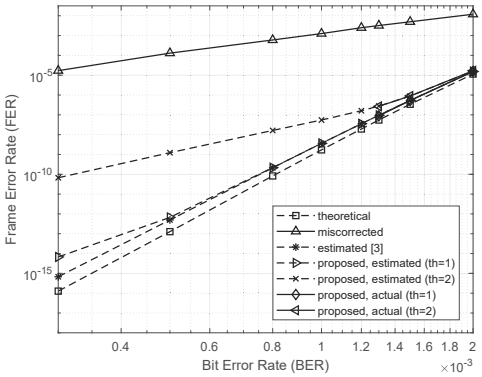

Fig. 2. FERs of GII-BCH [4,3] decoding with nested syndrome checking skipped when $\deg(\Lambda(x)) \leq th$.

of checking the nested syndromes regardless of $\deg(\Lambda(x))$ as in [3] after nested decoding round i can be estimated as

$$F_1^{(i)} = \binom{v-i}{1} \left(\sum_{w=t_i+1}^{t_v} \phi_w G_w^{(i)} \right) \left(\sum_{w=0}^{th} \phi_w \right)^{m-1}. \quad (3)$$

In (3), $\phi_w = \binom{n}{w} p_b^w (1-p_b)^{n-w}$ is the probability of a n-bit sub-codeword corrupted with w errors when the input BER is p_b. $G_w^{(i)}$ is the probability of a w-error-corrupted sub-word miscorrected with $\deg(\Lambda(x)) \leq th$ and not detected by the 1-bit eBCH scheme in nested decoding round i. Its value can be derived from simulations over a limited number of random samples, such as 10^8. Note that if the miscorrected sub-word has more than t_v errors, then the error pattern is not correctable by the GII code anyway and skipping the nested syndrome checking does not lead to further FER degradation. Hence, the upper bound of the summation in the middle of (3) is t_v. $F_1^{(0)}$, which corresponds to the nested syndrome checking after the first-stage sub-word decoding, is by far much larger than $F_1^{(i)}$ with $i > 0$, and can be used to estimate the overall FER degradation caused by skipping the nested syndrome checking when $\deg(\Lambda(x)) \leq th$.

The FERs of the example GII code estimated by using $F_1^{(0)}$ with different th values are plotted in Fig. 2. Simulations over the binary symmetric channel (BSC) at higher BERs, such as $\geq 1.3 \times 10^{-3}$, have also been carried out and the results overlap with the estimation in the figure. Compared to the methods in [3], the proposed scheme only has slight FER degradation when $th = 1$. However, increasing th to 2 makes the FER much higher. Fig. 3 shows the probabilities of activating the higher-order nested syndrome computation for miscorrection detection after the first-stage sub-codeword decoding. By setting a threshold to $\deg(\Lambda(x))$ as in the proposed approach, the nested syndromes are computed much less frequently for

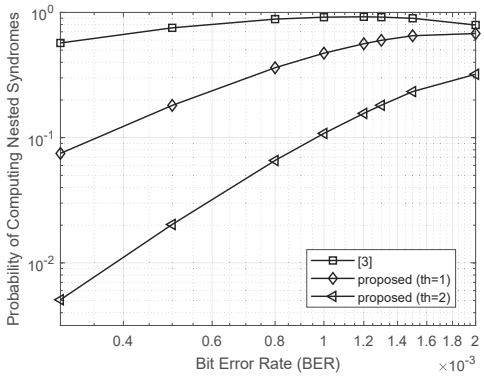

Fig. 3. Probabilities of computing higher-order nested syndromes for miscorrection detection after the first-stage sub-codeword decoding for the example GII-BCH [4,3] code.

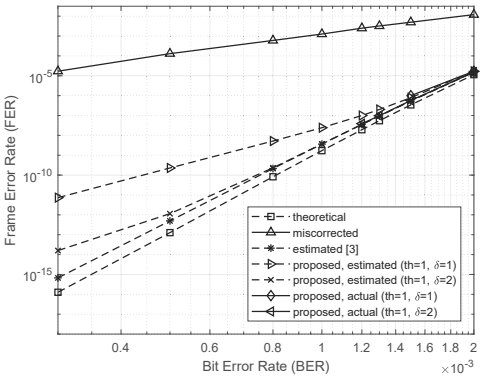

Fig. 4. FERs of GII-BCH [4,3] decoding with nested syndrome checking skipped.

miscorrection detection and hence the average nested decoding latency is reduced significantly, especially at lower BER. Although $th = 2$ leads to further reduction in the probability of activating the nested syndrome computation, it should not be used since it leads to substantial FER degradation as shown in Fig. 2.

The above approach reduces the average nested decoding latency. To reduce the worst-case nested decoding latency, this paper proposes to skip the nested syndrome checking after later nested decoding rounds, since miscorrections are less likely to happen for the corresponding BCH codes that have higher correction capabilities. In order to decide the nested decoding round δ, after which the nested syndrome checking is skipped, the corresponding FER degradation needs to be analyzed. The major cases leading to performance degradation are still that only one sub-word is miscorrected and all the other sub-words are decoded correctly [3]. Since the δ-th nested decoding round can correct at most t_δ errors, the major cases leading to FER increase after this round are that there is one miscorrected sub-word with more than t_δ errors and all the other sub-words have no more than t_δ errors. Similar to (3), the FER degradation caused by these miscorrections can be estimated as

$$F_2^{(\delta)} = \binom{v-\delta}{1} \left(\sum_{w=t_\delta+1}^{t_v} \phi_w G_w'^{(\delta)} \right) \left(\sum_{w=0}^{t_\delta} \phi_w \right)^{m-1}, \quad (4)$$

where $G_w'^{(\delta)}$ is the probability of a w-error-corrupted sub-word miscorrected with $\deg(\Lambda(x)) \leq t_\delta$ and not detected by the 1-bit eBCH scheme in nested decoding round δ. $G_w'^{(\delta)}$ can be derived by simulations over a limited number of random samples.

Combining (3) and (4), the FERs of GII decoding using the two proposed schemes for skipping the nested syndrome checking are plotted in Fig. 4 for the example [4,3] code with $[t_0, t_1, t_2, t_3] = [3, 5, 6, 11]$. It can be observed that skipping the nested syndrome checking after nested decoding round $\delta = 2$ with $t_2 = 6$ only brings small FER degradation. On the other hand, skipping the nested syndrome checking for earlier decoding rounds with smaller t_δ leads to significant performance loss. Simulations over BSC at high BERs have been carried out, and the results match the estimated FERs.

Although the two proposed schemes of skipping the nested syndrome checking reduce the nested decoding latency by a significant portion, they bring noticeable performance loss as shown in Fig. 4. In order to close the FER gap, two methods with negligible complexity and no latency overhead are developed next.

B. Exploiting 2-bit Extended BCH Codes

Miscorrections are detected by using 1-bit eBCH codes in [3]. The basic idea is to incorporate the factor $(x+1)$ into the generator polynomials of the involved BCH codes. Since now the Hamming weight of every sub-codeword is even, whether the number of errors in a received sub-word is even or odd can be determined by XORing all the bits in the sub-word. On the other hand, the number of distinct roots of $\Lambda(x)$ tells the number of errors found from the decoding process. If the parities of these two numbers do not match, then miscorrection is detected. However, this 1-bit eBCH scheme fails to detect miscorrections when the number of errors in the received sub-word and the number of distinct roots of $\Lambda(x)$ are different but are both even or odd.

To locate the miscorrections that are undetectable by 1-bit eBCH codes, this paper proposes to exploit 2-bit eBCH codes [7]. The generator polynomial of a 2-bit eBCH code is constructed by multiplying (x^2+1) to the generator polynomial of a BCH code. Hence, the 2-bit eBCH code is a sub-code of the 1-bit eBCH code and the Hamming weight of a 2-bit eBCH codeword is still even. Besides, a 2-bit eBCH codeword has the property that the Hamming weights of all even-index bits and all odd-index bits are both even. With this extra information, more miscorrections can be detected. Consider a case that the miscorrected sub-word has j errors and the corresponding $\Lambda(x)$ has k distinct roots, where j and k are both even. This is undetectable by the 1-bit eBCH code. Let the number of errors in the even-index bits of the sub-word be j'. Whether j' is even or odd can be determined by XORing all the even-index bits in the 2-bit eBCH sub-word. Assume that k' of the roots of $\Lambda(x)$ are in even-index locations. When one of j' and k' is even and the other is odd, miscorrection is detected. Of course, miscorrections are undetected when j' and k' are

TABLE I
VALUES OF $G_w^{(i)}$ IN (3) WHEN ζ GLOBAL PARITIES ARE UTILIZED FOR THE DECODING OF GII-BCH [4,3] CODE WITH $th = 1$

ζ	$G_7^{(0)}$	$G_9^{(0)}$	$G_{11}^{(0)}$
0	3.6×10^{-7}	2.4×10^{-7}	1.6×10^{-7}
2	1.4×10^{-7}	1.7×10^{-7}	7.0×10^{-8}
4	$<10^{-8}$	$<10^{-8}$	$<10^{-8}$
6	$<10^{-9}$	$<10^{-8}$	$<10^{-8}$

TABLE II
VALUES OF $G_w'^{(\delta)}$ IN (4) WHEN ζ GLOBAL PARITIES ARE UTILIZED FOR THE DECODING OF GII-BCH [4,3] CODE WITH $\delta = 2$

ζ	$G_7'^{(2)}$	$G_8'^{(2)}$	$G_9'^{(2)}$	$G_{10}'^{(2)}$	$G_{11}'^{(2)}$
0	5×10^{-9}	6.1×10^{-5}	1.3×10^{-6}	1.4×10^{-4}	7.0×10^{-7}
2	$<5\times10^{-9}$	3.2×10^{-5}	2.0×10^{-7}	4.3×10^{-5}	4.0×10^{-7}
4	$<5\times10^{-9}$	9.2×10^{-6}	$<10^{-7}$	1.3×10^{-5}	$< 10^{-7}$
6	$<5\times10^{-9}$	1.2×10^{-6}	$<10^{-7}$	2.2×10^{-6}	$<10^{-7}$

both even or odd. Similar analysis applies to the case when j and k are both odd. Accordingly, using 2-bit eBCH codes can further detect half of the miscorrection cases that are not found by the 1-bit eBCH scheme.

Applying the 2-bit eBCH scheme, the values of $G_w^{(i)}$ in (3) and $G_w'^{(\delta)}$ in (4) will be reduced by a half. Accordingly, the performance loss brought by skipping the nested syndrome computation as proposed in the previous subsection is also reduced by a half. The code rate loss resulted from utilizing the 2-bit eBCH scheme is negligible, because only two extra bits are needed for each individual sub-codeword.

C. Adopting Global Parities

The FER gap resulted from skipping the nested syndrome checking can be further closed by utilizing more parity bits. However, keep adding more parities to each sub-codeword will lead to noticeable code rate loss. On the other hand, in most of the cases, only one sub-word is miscorrected. To more efficiently utilize the parities and keep the code rate loss negligible, this paper proposes to adopt global parities that are XOR results of all sub-codewords for miscorrection detection. The global parities can effectively detect a single miscorrected sub-word when the other sub-words are corrected.

Assume that ζ extra global parities, $p_0, p_1, \cdots, p_{\zeta-1}$, are used. In our scheme, each individual sub-codeword is divided into ζ segments, and the i-th global parity bit is the XOR result of the i-th segment of every sub-codeword. Assume that $\zeta \mid n$. In other words,

$$p_i = \sum_{j=0}^{m-1} \sum_{k=0}^{n/\zeta-1} c_{j,i(n/\zeta)+k}, \tag{5}$$

where $c_{j,a}$ is the a-th bit of the j-th BCH sub-codeword, and the addition is the XOR. If $\zeta \nmid n$, then the n bits of each sub-word should be divided into ζ groups as evenly as possible.

The global parities are used to detect miscorrections as follows. Assume that there is no error on p_i at the receiver. Similar XOR computations as in (5) are carried out on the received sub-words to get $\hat{p}_i$. If p_i XOR $\hat{p}_i$ ='0', it means that the number of errors on the i-th segment of the sub-codewords is even. Otherwise, the number of errors is odd. Also, count the number of errors in the i-th segment found from the decoding for all sub-words. If the parity of this number does not match the conclusion drawn from p_i XOR $\hat{p}_i$, then miscorrections are detected. The probability that a global parity bit is erroneous is equal to the input BER. For lower BER, the global parities can help to effectively detect more miscorrections.

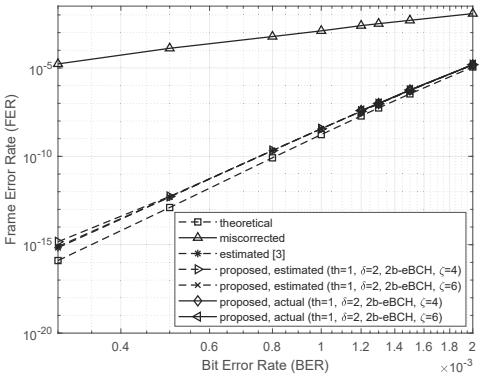

Fig. 5. FERs of GII-BCH [4,3] decoding with the three proposed miscorrection detection schemes combined.

Since the global parities are the XOR results of every sub-word, for a given number of global parities, there are a large number of possible error patterns among the m sub-words. Hence, it is difficult to develop a concise formula for the FER improvement resulted from using global parities. However, $G_w^{(i)}$ and $G_w'^{(\delta)}$ in (3) and (4), respectively, do not change with the input BER and the other components of the formulas can be calculated algebraically. Therefore, the FERs at lower BERs can still be estimated once $G_w^{(i)}$ and $G_w'^{(\delta)}$ are derived from simulations over a limited number of random samples. The values of $G_w^{(i)}$ and $G_w'^{(\delta)}$ for the example code with different numbers of global parities are shown in Table I and II, respectively. Since $F_1^{(0)}$ is much larger than the other $F_1^{(i)}$ with $i > 0$, only $G_w^{(0)}$ with $w \geq t_0 + 1$ need to be considered. Besides, when $th = 1$ and eBCH codes are used, only the miscorrections with odd number of errors are not detected. Additionally, since the minimum distance of the $t_0 = 3$ BCH code is $2t_0 + 1 = 7$, a 5-error-corrupted sub-word will not be miscorrected with $\deg(\Lambda(x)) = 1$. Therefore, only $G_7^{(0)}$, $G_9^{(0)}$, and $G_{11}^{(0)}$ need to be taken into account as shown in Table I. Also $\delta = 2$ is adopted for the example code. Hence Table II starts from $G_{t_2+1}'^{(2)}$. The estimated FERs of the proposed optimization schemes for latency reduction are plotted in Fig. 5. It can be observed that the FER of our new schemes becomes almost the same as that of the approach in [3] when 6 global parities are used. Simulation results at higher BER also confirm our analysis.

IV. MODIFIED GII DECODING AND LATENCY ANALYSES

The GII-BCH decoding with the proposed optimized miscorrection detection schemes for latency reduction is summarized in Algorithm 1. The set I includes the sub-words that remain to be corrected. I^t consists of the sub-words that are declared decoding success, and $I^c = \{0, 1, \cdots, m-1\} \backslash I$. If

Algorithm 1: GII-BCH Dec. with Prop. Miscorrection Detection

Input: received sub-words $y_i(x)$ $(0 \le i < m)$; t_i $(0 \le i \le v)$
Initialization: $I \Leftarrow \{0, 1, \cdots, m-1\}$; $I^c \Leftarrow \emptyset$; $I^t \Leftarrow \emptyset$
$f_i \Leftarrow$XOR of all even bits of $y_i(x)$
$g_i \Leftarrow$XOR of all odd bits of $y_i(x)$
for $l=0, 1, \cdots, v$ **do**
  if (l=0): compute $2t_0$ syndromes for each $y_i(x) \in I$
    if (all are zero): declare decoding success; stop. **end**
  else: derive $2t_l - 2t_{l-1}$ higher-order syn. for each $y_i(x) \in I$
  end
  for each $i \in I$ **do**
    carry out KES on $y_i(x)$
    $d_i \Leftarrow$number of even inverse roots of $\Lambda_i(x)$
    $q_i \Leftarrow$number of odd inverse roots of $\Lambda_i(x)$
    if ($\deg(\Lambda_i(x)) \le t_l$) & ($d_i \bmod 2 = f_i$) & ($q_i \bmod 2 = g_i$)
      & ($d_i + q_i = \deg(\Lambda_i(x))$):
        $I^t \Leftarrow I^t \cup i$; $I \Leftarrow I \backslash i$
    end
  end
  if ($|I| > v - l$): declare decoding failure; stop. **end**
  $h \Leftarrow$true
  if ($|I|$=0):
    use global parities to detect miscorrections
    if (miscorrections not found from global parities):
      if (all $\deg(\Lambda_i(x)) \le th$): $h \Leftarrow$false
      elseif ($l < \delta$): compute higher-order nested syndromes
          if (all nested syndromes are zero)
            $h \Leftarrow$false
          end
      else: $h \Leftarrow$false
      end
    end
  end
  if (h=false):
    $I^c \Leftarrow I^c \cup I^t$; use $\Lambda_i(x)$ to correct each $y_i(x) \in I^c$
    declare decoding success; stop.
  end
  while $|I| < v - l$ **do**
    find smallest $i \in I^t$ with max $\deg(\Lambda(x))$; $I \Leftarrow I \cup i$; $I^t \Leftarrow I^t \backslash i$
  end
  $I^c \Leftarrow I^c \cup I^t$; $I^t \Leftarrow \emptyset$
end

$|I|$ exceeds the number of sub-words that can be corrected by the GII code, overall decoding is terminated with failure. If the decoding of every sub-word has been declared successful, whose decision is contributed by the parities of the 2-bit eBCH scheme, miscorrection detection is carried out. A flag, h, is used to indicate if further decoding is needed. It is initially set to true and is changed to false if miscorrections are not found by using global parities and i) all sub-words have $\deg(\Lambda(x)) \le th$, ii) no miscorrection is found from nested syndrome checking, or iii) it is already after nested decoding round δ. When h becomes false, the GII decoding is successfully completed.

For the example GII-BCH [4,3] code with $[t_0, t_1, t_2, t_3] = [3, 5, 6, 11]$, from simulations, the probabilities of activating the nested decoding round 1, 2, and 3 are $pn_1 = 2.3 \times 10^{-2}$, $pn_2 = 3.6 \times 10^{-4}$, and $pn_3 = 4.4 \times 10^{-5}$, respectively, at BER=$10^{-3}$. Let the probabilities of activating the nested syndrome computation for miscorrection detection after decoding round i be pm_i. The average nested decoding latency of the design in [3] is $L_{orig} = 8pm_0 + pn_1(38 + 8pm_1) + pn_2(30 + 8pm_2) + 40pn_3$ according to Fig. 1. From simulations, $pm_0 = 0.92$, and pm_1 and pm_2 are much smaller. It can be computed that $L_{orig} = 8.3$. In the proposed schemes, adopting $th = 1$ and $\delta = 2$ with 2-bit eBCH codes and 6 global parities leads to negligible performance loss. In this setting, pm_0 is reduced to 0.47 at BER=10^{-3} from Fig. 3 and pm_2 becomes zero. Accordingly, the average nested decoding latency of the proposed scheme is $L_{prop} = 4.7$, which is 43% shorter. The average latency reduction is more significant at lower BER due to the smaller pm_0. The proposed schemes also reduce the worst-case latency from 132 in Fig. 1 to 124 since the nested syndrome checking after the second nested decoding round is eliminated. For GII-BCH codes with larger t_i, such as a [4,2] code with $[t_0, t_1, t_2] = [4, 6, 11]$, eliminating the nested syndrome checking from earlier decoding rounds results in more significant worst-case latency improvement.

The proposed schemes have negligible overheads in hardware complexity and redundancy. To use 2-bit eBCH codes, only one more tap is needed in the linear feedback shift registers for the encoder. The XOR computations for the extra parities require negligible silicon area compared to that of the overall GII decoder. Besides, the additional parities only bring (4+6)/704/4=0.35% code rate loss for the example code compared to the approaches in [3].

V. Conclusions

This paper proposed three miscorrection mitigation optimizations to reduce the nested decoding latency with negligible complexity overhead. Nested syndrome checking is skipped when miscorrections are unlikely to happen. To make up for the resulted small FER degradation, 2-bit eBCH codes and global parities are utilized to detect more miscorrections. Overall, the proposed schemes can achieve almost the same error-correcting performance as prior designs with significantly shorter latency. Formulas and methodologies for estimating the FER at lower input BER are also provided. Future research will investigate the optimization of the other GII decoding steps.

References

[1] X. Tang and R. Koetter, "A novel method for combining algebraic decoding and iterative processing," in *Proc. IEEE Int. Symp. Inf. Theory*, Seattle, WA, USA, Jul. 2006, pp. 474-478.

[2] Y. Wu, "Generalized integrated interleaved codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 2, pp. 1102-1119, Feb. 2017.

[3] Z. Xie and X. Zhang, "Miscorrection mitigation for generalized integrated interleaved BCH codes," *IEEE Commun. Letters*, vol. 25, no. 7, pp. 2118-2122, Apr. 2021.

[4] X. Tang and R. Koetter, "On the performance of integrated interleaving coding schemes," in *Proc. IEEE Int. Symp. Inf. Theory*, Chicago, IL, USA, 2004, pp. 329-329.

[5] W. Li, J. Lin, and Z. Wang, "A 124-Gb/s decoder for generalized integrated interleaved codes," *IEEE Trans. Circuits and Syst. I: Regular Papers*, vol. 66, no. 8, pp. 3174-3187, Aug. 2019.

[6] X. Zhang and Z. Xie, "Efficient architectures for generalized integrated interleaved decoder," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 66, no. 10, pp. 4018-4031, Oct. 2019.

[7] C. Häger and H. D. Pfister, "Approaching miscorrection-free performance of product codes with anchor decoding," *IEEE Trans. Commun.*, vol. 66, no. 7, pp. 2797-2808, Jul. 2018.