Generalized Integrated Interleaved Codes for High-Density DRAMs

Yok Jye Tang and Xinmiao Zhang
The Ohio State University, Columbus, OH 43210, USA

***Abstract*—As the density of dynamic random-access memories (DRAMs) keeps increasing, which results in higher error rates, the conventional single error correction and double error detection codes are no longer sufficient. Generalized integrated interleaved (GII) codes based on Reed-Solomon (RS) codes are among the best error-correcting codes for high-density DRAMs due to their hyper-speed decoding and good correction capability. However, the very short codeword length required by DRAMs leads to miscorrections that substantially degrade the error-correcting performance of GII-RS codes if untreated. Previous miscorrection mitigation schemes for longer GII-BCH codes lead to additional code rate loss when applied to very short GII-RS codes and hence affect the cost of DRAMs. This paper presents new miscorrection mitigation schemes with improved code rates. A small number of parity bits are allocated in an optimized manner and decoding trials are carried out to close the performance gap. Moreover, low-latency hardware implementation architectures have been developed for the proposed GII-RS decoder. For the example code considered for DRAMs, the proposed decoder reduces the worst-case latency by 45% with small area overhead while keeping the same average latency and critical path compared to the best possible alternative design.**

I. Introduction

The continuous rise in the density of dynamic random-access memories (DRAMs) for high-performance systems results in increased error rates. Low-redundancy error-correcting codes with hyper-speed decoding and good correction capability are needed to replace the traditional single error correction and double error detection codes. Generalized integrated interleaved (GII) codes [1], [2] that nest short Reed-Solomon (RS) sub-codewords to form codewords of more powerful RS codes are the best candidate. Their decoding consists of two stages. The first stage is the hyper-speed conventional RS decoding over individual short sub-words. The second-stage nested decoding is activated when there are extra errors.

The current single symbol correction CHIPKILL scheme used by the AMD and Sun UltraSPARC for DRAMs employs four un-nested 1-error-correcting (18,16) RS codewords over $GF(2^8)$ [3]. To keep a similar codeword format, each GII codeword has four 1-error-correcting RS sub-codewords with 16 data symbols. One 3-error-correcting RS codeword can be produced by nesting those four sub-codewords to correct more errors with simple decoders. Such a GII code can achieve orders of magnitude lower decoding failure rate compared to individual (18,16) RS codes. However, miscorrections on the 1-error-correcting sub-words lead to severe degradation on the correction performance if untreated.

This material is based upon work supported by the National Science Foundation under Award No. 2011785.

For longer GII codes based on BCH codes, nested syndromes are checked and extended BCH codes are adopted to identify and mitigate miscorrections in [4], [5]. However, for very short GII-RS codes, the extra parities of extended RS codes lead to additional code rate loss. Code rate decides the capacity and is a key factor affecting the cost of DRAM devices. Hence it needs to be increased as much as possible.

In this paper, two low-redundancy miscorrection mitigation schemes are proposed for GII-RS codes with 1-error-correcting sub-codewords. Instead of using extended RS codes with extra parity symbols, our first method uses a few parity bits produced by XOR operations. The parity bits allocated to sub-words are optimized to improve the probability of identifying miscorrections by incorporating the nested syndrome checking. The performance degradation caused by miscorrections is further reduced by carrying out multiple nested decoding trials in our second method. Additionally, low-latency hardware implementation architectures are developed for each decoding step. For the example code considered for DRAMs, the proposed decoder reduces the worst-case latency by 45% with small area overhead while having the same average latency and critical path compared to the best possible alternative design.

This paper is organized as follows. Section II introduces GII-RS codes and previous miscorrection mitigation schemes. The proposed low-redundancy miscorrection mitigation schemes are detailed in Section III. Low-latency hardware implementation architectures are developed for the proposed GII-RS decoder in Section IV. Section V is conclusions.

II. GII-RS codes

A $[m, v]$ GII-RS code can be constructed using $v + 1$ RS codes, $\mathcal{C}_v \subseteq \mathcal{C}_{v-1} \subseteq \cdots \mathcal{C}_1 \subset \mathcal{C}_0$, with dimension k and error-correcting capabilities $t_v \geq \cdots \geq t_1 > t_0$ as [1], [2]

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

where β is a primitive element of $GF(2^q)$. In (1), $c_i(x)$ $(0 \leq i < m)$ and $\tilde{c}_l(x)$ $(0 \leq l < v)$ are the sub-codewords and nested codewords, respectively. Let n_i denote the length of $c_i(x)$. For systematic GII codeword, n_i equals $k + 2t_{v-i}$ for $0 \leq i < v$ and $k+2t_0$ for $v \leq i < m$. Alternatively, GII codes can be constructed such that the sub-codewords have the same length but different numbers of data symbols.

Let $y_i(x) = c_i(x) + e_i(x)(0 \leq i < m)$ be one of the m received sub-words, where $e_i(x)$ is the error polynomial. GII decoding consists of two stages. First, traditional RS decoding is carried out on individual sub-words. Their syndromes are computed as $S_j^{(i)} = y_i(\beta^{j+1}) = e_i(\beta^{j+1})(0 \leq j < 2t_0)$. If all $2t_0$ syndromes are zero, it has no error. Otherwise, the key-equation solver (KES) is carried out to compute the error-locator polynomial, $\Lambda(x)$. If the root number of $\Lambda(x)$ is the same as its degree, the decoding is considered successful. Its inverse roots are the error locations. Then the error magnitudes can be computed by the Forney's formula using the error-locator and evaluator polynomials. In sub-word decoding, up to t_0 errors can be corrected in each $y_i(x)$.

The second-stage nested decoding is activated when at least one of the m sub-words has more than t_0 errors. It has up to v rounds and the η-th $(1 \leq \eta \leq v)$ nested decoding round can correct up to t_η errors in at most $b_\eta \leq v + 1 - \eta$ sub-words. Let $I = \{i_0, i_1, \cdots, i_{b-1}\}$ be the set of the indices of the b_η sub-words with extra errors and I^C be the set of the indices of the other sub-words that have been corrected in previous decoding rounds. First, the $2(t_\eta - t_{\eta-1})$ higher-order nested syndromes of the first b_η nested words are computed as $\tilde{S}_j^{(l)} = \tilde{y}_l(\beta^{j+1})(0 \leq l < b_\eta, 2t_{\eta-1} \leq j < 2t_\eta)$, where $\tilde{y}_l(x) = \sum_{i\in I} \beta^{il} y_i(x) + \sum_{i\in I^C} \beta^{il} c_i(x)$. Then the higher-order syndromes for those b_η sub-words with extra errors are derived as $[S_j^{(i_0)}, S_j^{(i_1)}, \cdots, S_j^{(i_{b-1})}] = A^{-1}[\tilde{S}_j^{(0)}, \tilde{S}_j^{(1)}, \cdots, \tilde{S}_j^{(b-1)}]^T$, where '$T$' denotes transpose and the entry of matrix A in the z-th row and u-th column is $\beta^{i_u z}$. After that, the KES computes the error-locator polynomial according to the $2t_\eta$ syndromes for each sub-word. Then the error magnitudes are computed when the root number of the error-locator polynomial equals its degree. If there are still some sub-words that remain to be corrected, they will be passed to the next nested decoding round and a similar process is repeated.

To match the codeword format of the AMD's and Sun UltraSPARC's CHIPKILL schemes that use four $(18, 16)$ RS codewords for DRAMs [3], the GII-RS $[4, 1]$ code with $k = 16$ and $[t_0, t_1] = [1, 3]$ over $GF(2^8)$ is considered. Such GII-RS code achieves good trade-off on the error-correcting capability and decoding complexity. Fig. 1 shows that this code can theoretically achieve orders of magnitude lower decoding frame error rate (FER) compared to the four un-nested (18, 16) RS codes used in CHIPKILL [3]. However, in sub-word decoding, a sub-word may be decoded to another valid codeword when the number of errors is larger than t_0. This is referred to as miscorrection. As shown in Fig. 1, the actual FER of this GII-RS code taking into account miscorrections is much higher. This is because, if the miscorrected sub-words are not identified, then they are not sent to the nested decoding to correct extra errors.

Three miscorrection mitigation schemes have been proposed for $t_0 = 3$ GII-BCH codes in [4]. First, miscorrections can be detected if any higher-order nested syndrome is non-zero. To identify which sub-words are miscorrected, the second method utilizes extended BCH codes. In the third scheme, a miscorrected sub-word can be found when the degree of its error-locator polynomial is higher than t_0.

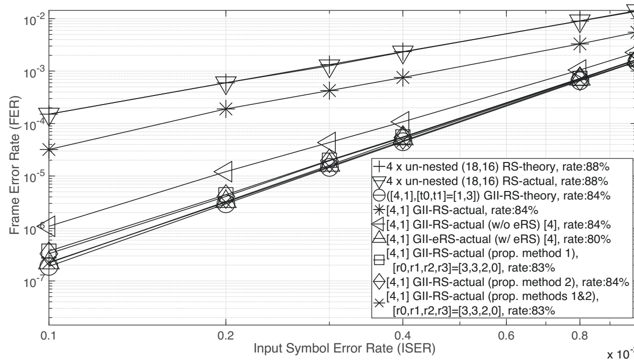

Fig. 1. FERs of [4, 1] GII-RS decoding with $[t_0, t_1] = [1, 3]$.

III. Miscorrection mitigation schemes for GII-RS codes with 1-error-correcting sub-codewords

In the case of GII-RS codes, extended RS (eRS) codes can be employed for miscorrection mitigation by borrowing the idea from [4]. A sub-word is miscorrected if its symbol-wise XOR result is non-zero. Combined with the other two schemes in [4], the actual FER of the $[4, 1]$ GII-RS code becomes closer to the theoretical FER as shown in Fig. 1. However, if eRS codes or doubly eRS codes as proposed in [5] are utilized for detecting miscorrections, there will be 4% or 8%, respectively, code rate loss. On the other hand, the FER increases significantly without them. This section presents two low-redundancy miscorrection mitigation schemes to improve the performance of short GII-RS codes.

A. *Parity bits for miscorrection detection*

Instead of using parity symbols, a small number of parity bits generated by XOR operations can be employed to detect miscorrections with lower redundancy. Assume that $r_i(r_i < q)$ parity bits are used for detecting miscorrections in sub-word $c_i(0 \leq i < m)$. Let $p_{i,j}(0 \leq j < r_i)$ denote those parity bits. Each codeword symbol over $GF(2^q)$ has q bits. Assuming nq is divisible by r_i, each parity bit can be computed as

$$p_{i,j} = c_{i,j} + c_{i,j+r_i} + \cdots + c_{i,nq-r_i+j+1}, \tag{2}$$

where $c_{i,l}$ $(0 \leq l < nq)$ is the l-th bit of c_i. When $(nq) \nmid r_i$, those $c_{i,l}$ for $l \geq nq$ in (2) are set to zero. The parity bits are calculated in the same way after the decoding and miscorrection is declared if any parity bit does not match.

The proposed scheme can detect more miscorrected sub-words by increasing the number of parity bits. The formula of the probability that the r_i parity bits do not detect miscorrections in a sub-word is very complicated. However, it is around $1/2^{r_i}$ from simulations. Although using eRS codes can detect miscorrections with higher probability, they require at least q extra parity bits for each sub-word and hence significantly decrease the code rate for short GII-RS codes. On the other hand, our proposed scheme provides a trade-off between the error-correcting performance and the code rate of short GII-RS codes. By setting $r_i = q$, the probability of detecting miscorrected sub-words in our scheme is similar to that of previous methods [4], [5] with eRS codes.

The probability of successfully detecting miscorrections in a sub-word increases with the number of parity bits allocated to that sub-word. Since the nested syndrome checking can tell if there are miscorrections among the m sub-words and only one sub-word can be sent to the nested decoding process for the $[4, 1]$ GII-RS code, the available parity bits are allocated to the first $m-1$ sub-words as evenly as possible to improve miscorrection detection. If some nested syndromes are nonzero and miscorrections are not detected for the first $m-1$ sub-words, then the last sub-word is miscorrected with a high probability and is sent to the nested decoding. The total number of parity bits should be a multiple of q to simplify the storage. For the $[4, 1]$ GII-RS code, allocating $[r_0, r_1, r_2, r_3] = [3, 3, 2, 0]$ parity bits can substantially reduce the performance gap as shown in Fig. 1 with only 1% code rate loss.

B. Nested decoding trials for miscorrection mitigation

If the miscorrected sub-word is considered as correctly decoded and an actually corrected sub-word is selected for nested decoding, then the higher-order syndromes for the KES are incorrect and the nested decoding will most likely fail. Let a be the index of the miscorrected sub-word. If the actually corrected sub-word b ($a \neq b$) is selected for the nested decoding, $y_b(x)$ is used to calculate the nested word, $\tilde{y}_0(x)$, and accordingly the higher-order nested syndrome. For $[4, 1]$ GII-RS codes, higher-order syndromes for sub-word b equal the higher-order nested syndromes as

$$S_j^{(b)} = \tilde{S}_j^{(0)} = \tilde{y}_0(\beta^{j+1}) = e_a(\beta^{j+1}) + e_b(\beta^{j+1}), \qquad (3)$$

where $e_a(x)$ and $e_b(x)$ are the error polynomials in the miscorrected sub-word a and $y_b(x)$, respectively. As shown in (3), the errors that contribute to the higher-order syndromes of sub-word b are not only from $y_b(x)$. Hence, carrying out the KES on those higher-order syndromes most likely generates an invalid error-locator polynomial whose root number is not equal to its degree and hence decoding failure is declared. This conclusion has also been verified from simulations. Therefore, whether the nested decoding fails helps to tell if the miscorrected sub-word has been chosen for nested decoding.

When the nested syndrome checking says there are miscorrections, our second proposed scheme carries out nested decoding trials on the sub-words until the decoding is successful. With high probability, the sub-words whose lowest $2t_0 = 2$ syndromes are both zero do not have errors. Hence, the nested decoding trials on those sub-words are skipped in our design. If none of the nested decoding trials is successful, GII decoding failure is declared. Fig. 1 shows that the actual FER of the $[4, 1]$ GII-RS code is significantly reduced by adopting our second scheme. Different from previous miscorrection mitigation schemes, our second proposed method does not require additional redundancy. Moreover, the nested decoding trials can be implemented with only a few clock cycles of latency by using pipelinable nested decoder architectures. By combining our two proposed schemes, the decoding FER becomes very similar to that of GII codes with eRS codes that have a much lower code rate as shown in Fig. 1.

IV. Low-latency decoder architectures for short GII-RS codes

Moderns DRAMs require error-correcting codes with very short decoding latency. While GII decoder architectures have

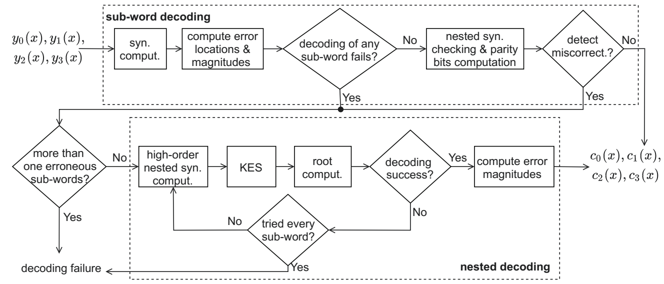

Fig. 2. $[4, 1]$ GII-RS decoding with $[t_0, t_1] = [1, 3]$ that employs the proposed miscorrection mitigation schemes.

been explored in [6]–[9], applying these designs to the proposed GII-RS decoder results in many clock cycles of decoding latency that does not meet the requirements of DRAMs. In this section, low-latency architectures are developed for the $[4, 1]$ GII-RS decoding with $[t_0, t_1] = [1, 3]$.

A. Decoding process and architectures

Fig. 2 shows the decoding process of the $[4, 1]$ GII-RS code that utilizes the proposed miscorrection mitigation schemes. In sub-word decoding, $m = 4$ single-error-correcting RS decoder architectures are employed to decode the 4 received sub-words in parallel. Since $t_0 = 1$, after the syndromes are computed, the error location and magnitude of each sub-word can be directly calculated from the syndromes by using two multipliers and two inverters [11]. If the decoding of more than one sub-word fails, GII decoding failure is declared. The nested decoding is activated when the decoding of only one sub-word fails since the $[4, 1]$ GII-RS code can only correct one sub-word with extra errors in the nested decoding, or when there is no sub-word decoding failure but miscorrections have been detected from the higher-order nested syndromes. In the latter case, nested decoding trials are carried out on those sub-words whose lowest $2t_0 = 2$ syndromes are not both zero. The parity bits allocated to sub-words 0-2 are calculated and compared. It is possible that a sub-word is correctly decoded in the sub-word decoding but the received parity bits for miscorrection checking themselves have errors. If multiple sub-words have mismatched parity bits, they are sent to the nested decoding one by one, starting from the sub-word with the lowest index. If none of the sub-words have mismatched parities, sub-word 3 is sent to the nested decoding and is followed by the others in the decoding trials. To achieve shorter decoding latency, the syndrome computation and nested syndrome checking can be implemented by using fully-parallel syndrome computation architectures, such as those in [7].

In the nested decoding, higher-order nested syndromes are first computed by sharing the same architecture used for the nested syndrome checking. For GII codes with $v = 1$, higher-order syndromes of the sub-word equal the higher-order nested syndromes. In the nested decoding of $t_1 = 3$, the KES calculates the error-locator and evaluator polynomials and their roots are computed in the next step. If the degree of the error-locator polynomial equals its root number, the decoding is considered successful and the error magnitudes are computed.

The KES architectures for the nested decoding have been studied in [6]–[9]. These designs originate from the reformulated inversionless Berlekamp-Massey (riBM) algorithm [10] and require at least $2(t_1 - t_0) + 1 = 5$ clock cycles for the $[4, 1]$ GII-RS code with $[t_0, t_1] = [1, 3]$. In our proposed miscorrection mitigation schemes, the nested decoding may be carried out in multiple trials. To reduce the worst-case latency with low complexity, a fully-pipelinable KES architecture is desirable. Fortunately, for $t_1 = 3$, the error-locator polynomial of a sub-word can be derived based on matrix inversion, which is implementable by an architecture free of feedback loops [11]. The data path of this architecture can be cut into 4 pipelining stages with a critical path consisting of one multiplier and 3 adders. Then the evaluator polynomials for computing the error magnitudes can be derived by multiplying the error-locator polynomial with the syndrome polynomial in the next clock cycle. As a result, using the pipelined architecture, the latency of the nested KES is 5 clock cycles. Compared to the KES architecture in [6]–[9], the pipelined architecture only needs 1 instead of 5 clock cycles for each additional nested decoding trial.

The roots of the error-locator and evaluator polynomials can be computed by using the Chien search architecture in [7]. The complexity of the Chien search is linear to the codeword length. Since the length of the sub-words involved in the $[4, 1]$ GII-RS code is at most $\max(n_0, n_1, n_2, n_3) = 22$, the fully-parallel Chien search architecture has lower complexity compared to the architectures in [11], [12] that are based on direct root computation. More importantly, the fully-parallel Chien search can be completed in one clock cycle. On the other hand, the designs in [11], [12] have more than 10 clock cycles of latency due to the pipelining of the long data path.

B. Joint nested syndrome computation architectures

To further reduce the decoding latency, nested syndrome computations for miscorrection checking and nested decoding can be combined. The nested syndromes for miscorrection checking are computed based on the sub-word decoding results. On the other hand, the received sub-word for the uncorrected sub-word is used to calculate the syndromes for nested decoding. However, instead of computing these syndromes separately, the intermediate results of computing the syndromes for miscorrection checking can be utilized to derive the syndromes for nested decoding at the same clock cycle with reduced complexity.

Let b be the index of the sub-word sent to the nested decoding and $f_b(x)$ be the error polynomial generated from the sub-word decoding. Denote the sum of the sub-word decoding results by $\tilde{y}'_0(x)$, which is used to compute the higher-order syndromes for miscorrection checking, $\tilde{S}'_j = \tilde{y}'_0(\beta^{j+1})$ $(2t_0 \leq j < 2t_1)$. Then $\tilde{y}_0(x) = \tilde{y}'_0(x) + f_b(x)$ is utilized to calculate the higher-order syndromes for the nested decoding. It can be derived that $\tilde{S}_j = \tilde{S}'_j + f_b(\beta^{j+1})$. For the $[4, 1]$ GII-RS decoding with $t_0 = 1$, $f_b(x)$ can only have one error and is in the format of $f_b(x) = f_{b,z}x^z (0 \leq z < n_i)$, where z and $f_{b,z}$ are the error location and magnitude, respectively, given by the sub-word decoding. Accordingly, $f_b(\beta^{j+1}) = f_{b,z}\beta^{(j+1)z}$.

Fig. 3 shows the proposed joint nested syndrome computation architecture. $\beta^{3z}, \beta^{4z}, \cdots, \beta^{6z}$ for different z can be

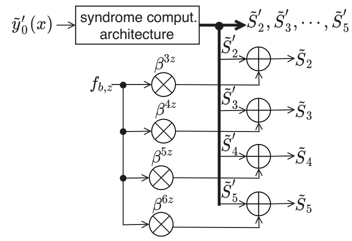

Fig. 3. Joint nested syndrome computation architectures for $[4, 1]$ GII-RS decoder with $[t_0, t_1] = [1, 3]$.

TABLE I
COMPLEXITY AND LATENCY COMPARISONS OF $[4, 1]$ GII-RS DECODERS WITH $[t_0, t_1] = [1, 3]$ AND 83% CODE RATE OVER $GF(2^8)$ USING THE TWO PROPOSED MISCORRECTION MITIGATION SCHEMES

	total complex. (# XORs)	critical path (# gates)	worst-case latency(# clks)	avg. latency (# clks)
alternat. design [6]	24.98k	10	29	4
alternat. design [9]	26.76k	10	45	4
prop. GII-RS dec.	26.52k	10	16	4

pre-computed and stored in a lookup table. For the $[4, 1]$ GII-RS code considered, only $4 \cdot \max(n_0, n_1, n_2, n_3) = 4 \cdot 22 = 88$ different constant elements need to be stored. By adopting a fully-parallel syndrome computation unit, the architecture in Fig. 3 computes the higher-order syndromes for both miscorrection checking and nested decoding in one clock cycle.

C. Complexity and latency analyses

Table I shows the complexity of the proposed $[4, 1]$ GII-RS decoder architecture with $[t_0, t_1] = [1, 3]$ over $GF(2^8)$ that employs the two proposed miscorrection mitigation schemes. The complexities are estimated in terms of the number of XOR gates. It is assumed that a register requires 3 times the area of an XOR gate and the area of a 2-to-1 multiplexer is similar to that of an XOR gate. A $GF(2^8)$ general multiplier can be implemented with an area similar to 98 XOR gates, while a constant multiplier carried out as a binary constant matrix multiplication has an average of 25 XOR gates [7]. The area of a $GF(2^8)$ inverter is similar to that of 175 XOR gates [7].

In the sub-word decoding, the critical path in each architecture that computes both the error location and magnitude has one inverter and one general multiplier. Their data paths are 14 and 6 gates [7], respectively. These computations are pipelined into two stages such that the critical path of the architectures is reduced to 10 gates. The complexity of the pipelining registers is included in Table I. Besides, each of the sub-word syndrome computation and nested syndrome checking can be done in one clock cycle using fully-parallel syndrome computation architectures. As a result, the latency of the sub-word decoding in our design is $1+2+1=4$ clock cycles.

By employing the proposed joint nested syndrome computation architecture in Fig. 3, the higher-order syndromes for the nested decoding are computed at the same time as the nested syndrome checking. Besides, the latency of the KES implemented by using the pipelined architecture in [11] is 5 clock cycles. After that, the fully-parallel Chien search takes 1 clock cycle to compute the roots of the error-locator and evaluator polynomials. For the error magnitude computation, multiplexers are needed to select $t_1 = 3$ out of 22 possible roots followed by inversions and multiplications, and the overall architecture is pipelined into 3 stages such that the critical data path does not exceed 10 gates. As a result, the latency of one nested decoding trial is $5+1+3=9$ clock cycles. Each additional nested decoding trial requires only one more clock cycle due to the fully pipelined architecture.

In the worst case, the proposed GII-RS decoder needs to carry out a total of four nested decoding trials. Therefore, the worst-case latency of the proposed design is $4+9+3=16$ clock cycles. Since the error rate of DRAMs is low, the probability of activating the nested decoding is very small. Accordingly, the average latency of the proposed design is around 4 clock cycle, which is mainly decided by the sub-word decoding. Since the sub-word and nested decoders are implemented by separate hardware units and they are fully pipelined, one GII-RS codeword is processed in each clock cycle as long as the nested decoding trial is carried out once.

Among existing nested KES architectures [6]–[9], the one in [6] is the most efficient in the case of small t_1. Its latency is $2(t_1-t_0)+1=5$ clock cycles for the example GII-RS code. For comparison, an alternative GII decoder design that utilizes the nested KES architecture in [6] is considered. Without using the proposed joint nested syndrome computation architecture, nested syndromes for the miscorrection checking and those for the nested decoding are computed at two different clock cycles. Hence, the latency of one nested decoding trial is $1+5+1+3=10$ clock cycles. Besides, the KES architecture from [6] is based on an iterative algorithm and hence 5 more clock cycles are needed for each additional decoding trial. Therefore, the worst-case latency of the alternative decoder is $4+10+3\times 5=29$ clock cycles when 4 nested decoding trials are carried out. Since the proposed design only requires one additional clock cycle for each nested decoding trial, it reduces the worst-case latency by $1-(16/29)\times 100=45\%$. The area of pipelinable KES architecture [11] adopted in the proposed design is only slightly larger than that of the nested KES architecture in [6] for $t_1=3$. Besides, the extra units to implement the joint nested syndrome computation architecture in Fig. 3 accounts for a small portion of the overall decoder. As a result, our proposed decoder is only 6% larger than the alterative design while keeping the same average latency and critical path. The design in [9] is more efficient than those in [7], [8]. However, they are not optimized for short codes and the latencies are even longer as shown in Table I.

V. Conclusions

GII codes based on RS codes are one of the best candidates for high-density DRAMs due to their good error-correcting capability and hyper-speed decoding. However, the very short codeword length of DRAMs brings new challenges to miscorrection mitigation, without which the error-correction performance of the GII-RS codes will be significantly degraded. This paper presents two low-redundancy miscorrection mitigation schemes for short GII-RS codes that allocate a small number of parity bits in an optimized manner and carry out decoding trials. Also low-latency hardware implementation architectures are developed for the proposed schemes. For the GII-RS code considered for DRAMs, our proposed decoder architecture has much shorter worst-case decoding latency with small area overhead while having the same average latency and critical path compared to the alternative design.

References

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

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

[3] Yeleswarapu, Ravikiran and Somani, Arun K., "Addressing multiple bit/symbol errors in DRAM subsystem," *PeerJ Comp. Science*, Feb. 2019.

[4] 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.

[5] Z. Xie and X. Zhang, "Improved miscorrection detection for generalized integrated interleaved BCH codes," *Proc. IEEE Int. Conf. Commun.*, 2022.

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

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

[8] Z. Xie and X. Zhang, "Reduced-complexity key equation solvers for generalized integrated interleaved BCH decoders," *IEEE Trans. on Circuits and Syst.-I*, vol. 67, no. 12, pp. 5520-5529, Dec. 2020.

[9] Z. Xie and X. Zhang, "Fast nested key equation solvers for generalized integrated interleaved decoder," *IEEE Trans. on Circuits and Syst-I*, vol. 68, no. 1, pp. 483-495, Jan. 2021.

[10] D. V. Sarwate and N. R. Shanbhag, "High-speed architectures for Reed-Solomon decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 5, pp. 641–655, Oct. 2001.

[11] Z. Yan, J. Lin and Z. Wang, "A low-complexity RS decoder for triple-error-correcting RS codes," *Proc. IEEE Computer Society Annual Symp. on VLSI*, 2019, pp. 489-494.

[12] X. Zhang and Z. Wang, "A low-complexity three-error-correcting BCH decoder for optical transport network," *IEEE Trans. on Circuits and Syst.-II*, vol. 59, no. 10, pp. 663-667, Oct. 2012.