Partial-Chip Extensions of Single-Chip Erasure Decoding for Flexible Use of Redundancy

Gyuri Kim*, Taeuk Ha*, Jungrae Kim*, Chanki Kim†, Sang-Hyo Kim**

*Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, South Korea
†Department of Computer Science and Artificial Intelligence, Jeonbuk National University, Jeonju, South Korea
**iamshkim@skku.edu

***Abstract*—As DRAM density scales, multi-bit and burst errors increasingly threaten system reliability. Prior work introduced single-chip erasure decoding using Reed–Solomon (RS) codes, but it was limited to full-chip erasure and lacked quantitative failure analysis. In this paper, we extend this approach to partial-chip erasure, where only a subset of symbols within a chip is treated as erasures. Using RS(40,32), our simulations show that partial-chip erasure reduces decoding failure probability by up to 70% compared to the full-chip method. We further propose a chip-localized post-decoding filter to eliminate miscorrections and a parallel decoding framework that applies the scheme selectively to high-weight error patterns. Together, these methods lower failure probability and improve design flexibility under the same redundancy budget.**

***Index Terms*—Error correction codes (ECC), DRAM reliability, Reed–Solomon codes, Chipkill correction, Single-chip erasure decoding**

I. Introduction

The reliability of main memory has become a critical factor for overall system stability in modern computing environments [2]–[5]. In particular, DRAM errors are a major source of system failures and data corruption [2], [6]. Early DRAM errors were mainly single-bit faults caused by manufacturing defects or cell failures [7]–[9], which were effectively handled by single error correction (SEC) [10] or single error correction and double error detection (SEC-DED) [11], widely adopted in memory systems [12].

However, as DRAM scales, inter-cell interference and reduced charge storage intensify [13]–[17], leading to frequent multi-bit upsets (MBUs) and burst errors [18]. To address these challenges, rank-level ECC (RL-ECC) was introduced, leveraging additional DRAM chips at the rank level to correct multi-bit and burst errors. With proper design, RL-ECC can even achieve Chipkill-correction, tolerating all error patterns in a single chip [19]. Various techniques for achieving Chipkill-correction have been proposed [19]–[22], among which Reed–Solomon (RS) code-based RL-ECC has been widely adopted [23], [24]. However, such schemes typically require about twice the redundancy to correct a single-chip failure, motivating further research into more efficient alternatives.

A prior work [1] proposed burst erasure decoding that exploits the error-and-erasure correction capability of RS codes to tolerate a single-chip failure. However, it only considered the case where all symbols in a chip are erased (hereafter referred to as *full-chip erasure*) and did not provide a quantitative analysis of decoding failure probability.

This paper addresses these limitations by extending single-chip erasure decoding to *partial-chip erasure*, where only a subset of symbols in a chip are erased. We analyze how the erasure size affects correction capability and decoding failure probability. Compared to full-chip erasure, partial-chip erasure offers lower failure probability while maintaining the same redundancy, thereby providing greater design flexibility. Furthermore, to mitigate miscorrections under incorrect chip assumptions, we propose a chip-localized post-decoding filter. In addition, we design a parallel decoding framework that applies the scheme selectively to high-weight error patterns, thereby improving overall decoding efficiency.

The remainder of this paper is organized as follows. Section II reviews the background, including DRAM organization, Reed–Solomon codes, Bamboo ECC, and single-chip erasure decoding. Section III describes the proposed partial-chip erasure method with the system model and erasure policies. Section IV presents the simulation setup with RS(40,32) and evaluates the failure probability under different erasure sizes. Section V discusses the causes of decoding failures and corresponding mitigation strategies. Finally, Section VI concludes the paper and outlines future work.

II. Preliminaries

A. DRAM Organization

DRAM is a volatile memory widely used as the main memory in computer and server systems, with performance and capacity continuously improving alongside standard evolution [25], [26]. A DRAM chip communicates through multiple input/output (I/O) pins, where an $\times N$ configuration denotes N parallel data pins (DQs). For example, an $\times 4$ chip transfers data over 4 DQs, while an $\times 8$ chip uses 8 DQs. By combining multiple DRAM chips, a DIMM (Dual In-line Memory Module) provides both wider data width and higher storage capacity. DDR5 DIMMs support a burst

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2024-00398449), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (RS-2024-00343913), and Samsung Electronics Co., Ltd (IO201209-07889-01). *(Corresponding author: Sang-Hyo Kim.)*

length (BL) of 16, enabling transfer of a 64-byte cache line with a single command. Internally, this transfer is divided into two segments, each consisting of 8 transfers, handled by a sub-channel with 32 DQs. The DDR5 standard further improves reliability by adding 8 ECC-dedicated DQs per sub-channel [26]. Thus, an ECC-DIMM provides 40 DQs per sub-channel, which serves as the granularity for the ECC schemes considered in this paper.

B. Basics of RS Codes

Reed–Solomon (RS) codes are non-binary cyclic codes that operate at the symbol level and are robust against both random and burst errors. For a symbol size of m bits, RS codes are defined over $\mathrm{GF}(2^m)$, with code length $n = 2^m - 1$. An (n, k) RS code with message length k has error correction capability

$$t = \left\lfloor \frac{n-k}{2} \right\rfloor \tag{1}$$

guaranteeing the correction of up to t symbol errors. Beyond this limit, miscorrections may occur.

Since RS codes are a class of cyclic codes, they can be expressed as polynomials. A generator polynomial $g(x)$ with t-symbol correction capability has degree $2t$ and roots $\alpha, \alpha^2, \ldots, \alpha^{2t}$, where α is a primitive element of $\mathrm{GF}(2^m)$. Decoding begins by computing syndromes from the received polynomial $r(x) = c(x) + e(x)$. By evaluating $r(x)$ at the roots of $g(x)$, the syndrome vector $\mathbf{S} = (S_1, S_2, \ldots, S_{2t})$ is obtained, with each element given by

$$S_i = r(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i), \quad 1 \le i \le 2t. \tag{2}$$

Because the codeword polynomial $c(x)$ is a multiple of $g(x)$, $c(\alpha^i) = 0$ holds, so syndromes depend only on the error polynomial $e(x)$. If all S_i are zero, no errors are detected; otherwise, decoding continues. The error locator polynomial $\sigma(x)$ is then computed via the Berlekamp–Massey or Euclidean algorithm. Error locations are found using the Chien search, and error values are computed using the Forney algorithm to reconstruct the original codeword.

C. Bamboo ECC

Bamboo ECC is a Chipkill-correction scheme based on Reed–Solomon (RS) codes [24]. It maps 8 bits transmitted along a DRAM data pin (DQ) to a single RS symbol, thereby providing protection at both the pin and chip levels. Conventional RS-based Chipkill schemes employ horizontal symbol organization aligned with chip boundaries [23], whereas Bamboo ECC adopts vertical symbol organization to extend the codeword length and enhance error detection capability.

In this work, Bamboo ECC is applied to a DDR5 ECC-DIMM subchannel model. Specifically, we use a shortened $\mathrm{RS}(40, 32)$ code tailored to 32 data DQs and 8 ECC DQs. The 8 parity symbols provided by two redundancy chips enable correction of up to four symbol errors or a single chip failure. Although Bamboo ECC can also correct pin-level errors distributed across multiple chips, an optimization restricts the correction range to mitigate the risk of silent data corruption (SDC).

D. Decoding of Erasure and Error RS Codes

The decoding procedure of RS codes described in Section II-B determines both the locations and values of errors. When some error locations are known in advance, the corresponding symbols can be treated as erasures to enable more efficient decoding [1], [27]. Errors have unknown locations and values, whereas erasures have known locations, so only the symbol values at those positions need to be recovered.

For an RS code with error correction capability t, decoding remains possible in the presence of both errors and erasures if

$$v + \frac{\varepsilon}{2} \le t \tag{3}$$

where v is the number of erroneous symbols and ε is the number of erased symbols.

Let the error positions be $i_1, i_2, \ldots, i_v$ and the erasure positions $i'_1, i'_2, \ldots, i'_\varepsilon$. When erasures are present, the coefficients at the erased positions in the received polynomial $r(x)$ are unknown; thus, they are replaced with a fixed value (e.g., zero) to construct a modified received polynomial $r^*(x)$. From this, the syndrome polynomial $S(x)$ is computed and multiplied by the erasure locator polynomial

$$\Gamma(x) = \prod_{l=1}^{\varepsilon} \left(1 - \alpha^{i'_l} x\right) \tag{4}$$

which yields the modified syndrome polynomial

$$T(x) = \Gamma(x) S(x) \bmod x^{2t} \tag{5}$$

The subsequent steps follow standard RS decoding. Using the modified syndrome $T(x)$, the error locator polynomial $\sigma(x)$ is derived to identify error positions. The known erasures and identified errors are then used to compute their values, thereby reconstructing the original codeword.

E. Single-Chip Erasure Decoding

The burst erasure decoding scheme [1] extends the scope of correctable error patterns in conventional RS decoders by jointly considering both errors and erasures. In an RS code with v errors and ε erasures, each error requires two parity symbols for correction, whereas each erasure requires only one since its location is known. Accordingly, the correctability condition is expressed as $v + \varepsilon/2 \le t$. Leveraging this property, when the location of a faulty chip is known, all symbols in that chip can be treated as erasures, enabling single-chip correction with only half as many parity symbols as required if they were treated as errors.

In practical systems, however, the location of the faulty chip is unknown. Thus, the decoder sequentially assumes each chip to be faulty, treats its symbols as erasures, and attempts decoding:

- *Correct chip selection:* All symbols in the actual faulty chip are treated as erasures, and decoding succeeds since the condition remains within the correction capability.
- *Incorrect chip selection:* Additional unnecessary erasures are introduced, which may exceed the correction limit and cause decoding to fail.

For clarity, we label these cases as *chip selection*, while subsequent discussion refers to them as chip *assumptions*. In rare cases of miscorrection, both the actual faulty chip and a wrongly assumed chip may appear to decode successfully. In such cases, the faulty chip cannot be uniquely identified, and the overall decoding process results in failure.

In this paper, we generalize this procedure as *single-chip erasure decoding*, where each chip is sequentially assumed to be faulty and erased. Within this framework, the conventional burst erasure decoding corresponds to *full-chip erasure*, while our work extends it to *partial-chip erasure* configurations.

III. Proposed Scheme

A. System Model

The system considered in this paper is based on a DDR5 ECC-DIMM sub-channel configuration, where a shortened RS$(40, 32)$ code is applied to cover a total of 40 DQs (32 data and 8 ECC). Each DQ maps 8 consecutive bits into one symbol, so each component of the received vector $\mathbf{r} = (r_0, \ldots, r_{39})$ belongs to $\mathrm{GF}(2^8)$.

In a $\times 4$ DDR5 ECC-DIMM configuration, a sub-channel consists of ten $\times 4$ DRAM chips, each providing four DQ pins. Accordingly, the symbol indices of the received vector correspond one-to-one with the DQ pins of the sub-channel. For the chip index $j \in \{0, 1, \ldots, 9\}$, the mapping is defined as

$$r_{4j+i} \;\leftrightarrow\; \mathrm{DQ}_i \text{ of chip } j, \quad i \in \{0, 1, 2, 3\}$$

Hence, the symbol index set of chip j is given by $\mathcal{I}_j = \{4j,\, 4j+1,\, 4j+2,\, 4j+3\}$.

The error correction capability of RS$(40, 32)$ is given by $t = 4$. In the presence of both errors and erasures, correction is possible if $v + \frac{\varepsilon}{2} \leq 4$, where v and ε denote the number of error symbols and erasure symbols, respectively. In this work, we assume that errors are restricted to a single chip. Accordingly, the error location set is given by $V \subseteq \mathcal{I}_j,\ v = |V|,\ 0 \leq v \leq 4$.

B. Partial- and Full-Chip Erasure Policy

We categorize the erasure policies considered in this work into two types. The erasure set is defined as the set of global symbol indices (0–39) in the sub-channel, obtained from a local index subset S within chip j:

$$E_{j,S} = \{\, 4j + i \;:\; i \in S \,\} \tag{6}$$

Here, $S \subseteq \{0, 1, 2, 3\}$ denotes the local indices of the erased symbols within chip j, and $\varepsilon = |S|$ is the number of erased symbols.

- *Full-chip erasure*: This corresponds to the case $S = \{0, 1, 2, 3\}$, where all four symbols of a chip are erased. In this case, $E_{j,S} = \mathcal{I}_j$ and $\varepsilon = 4$.
- *Partial-chip erasure*: This corresponds to the case $S \subseteq \{0, 1, 2, 3\}$ with $S \neq \varnothing$ and $S \neq \{0, 1, 2, 3\}$, such that only a subset of symbols in a chip are erased. In this case, $E_{j,S} = \{\, 4j + i : i \in S \,\}$ and $\varepsilon = |S| \in \{1, 2, 3\}$.

Partial-chip erasure offers a fine-grained approach to full-chip erasure, enabling more flexible correction strategies under the same redundancy resources.

TABLE I
Decoding conditions by ε when $j \neq j'$.

ε	Decodable region for v
1	$v \leq 3$
2	$v \leq 3$
3	$v \leq 2$
4	$v \leq 2$

C. Application of Single-Chip Erasure Decoding

Decoding is performed using the single-chip erasure decoding method according to the erasure policies defined in Section III-B. An erasure location set S is first determined based on the selected policy. Then, each chip $j' \in \{0, \ldots, 9\}$ is sequentially assumed to be faulty, and the corresponding erasure set is applied:

$$E_{j',S} = \{\, 4j' + i \;:\; i \in S \,\} \tag{7}$$

followed by an attempt of RS decoding.

The error location set V denotes the indices of erroneous symbols within the actual faulty chip j, with size $v = |V| \leq 4$. For a chip assumption j' with a fixed erasure set S, the number of erroneous symbols covered by the erasure set is defined as

$$c(j', S) = |V \cap E_{j',S}| \tag{8}$$

and the number of remaining errors is

$$v_{\mathrm{out}}(j', S) = v - c(j', S) \tag{9}$$

Decoding succeeds if the RS error–erasure correction condition is satisfied:

$$v_{\mathrm{out}}(j', S) + \frac{\varepsilon}{2} \leq t, \tag{10}$$

where $t = 4$ for the RS$(40, 32)$ code.

- *Correct chip selection* ($j = j'$): Since a single chip contains only four symbols, $|V \cup E_{j',S}| \leq 4$ always holds. Consequently, the correction condition is automatically satisfied, and decoding under the correct chip assumption always succeeds.
- *Incorrect chip selection* ($j \neq j'$): In this case, $E_{j',S} \cap V = \emptyset$, which implies $c(j', S) = 0$ and hence $v_{\mathrm{out}}(j', S) = v$. The success condition is therefore simplified to $v + \varepsilon/2 \leq 4$. The corresponding conditions for each value of ε are summarized in Table I.

Let the set of successfully decoded chips under assumption j' be

$$A_{\mathrm{chip}}(S) = \left\{\, j' \in \{0, \ldots, 9\} \;:\; v_{\mathrm{out}}(j', S) + \frac{\varepsilon}{2} \leq t \,\right\} \tag{11}$$

If $|A_{\mathrm{chip}}| = 1$, the faulty chip is uniquely identified and decoding succeeds. However, as shown in Table I, an incorrect assumption ($j \neq j'$) may also satisfy the correction condition, leading to successful decoding for both the correct and incorrect chips. Even when the correction condition is exceeded, miscorrections may still occur, causing an incorrect assumption to be judged as valid. In either case, single-chip erasure decoding ultimately fails.

TABLE II
FAILURE PATTERN COUNTS AND RATIOS BY ERASURE SIZE ε AND ERROR WEIGHT v.

$\varepsilon = \lvert S \rvert$	Failure pattern count				Failure ratio [%]
	$v=1$	$v=2$	$v=3$	$v=4$	
4	$4 \cdot 255$	$\binom{4}{2} \cdot 255^2$	4,847,040	82,620	0.1239
3	$4 \cdot 255$	$\binom{4}{2} \cdot 255^2$	0	1,211,760	0.0373
2	$4 \cdot 255$	$\binom{4}{2} \cdot 255^2$	$\binom{4}{3} \cdot 255^3$	73,440	1.5551
1	$4 \cdot 255$	$\binom{4}{2} \cdot 255^2$	$\binom{4}{3} \cdot 255^3$	0	1.5534

IV. SIMULATION RESULTS

A. Simulation Setup

In this section, the decoding procedure described in Section III is applied to evaluate the decoding failure probability as a function of the number of erasures, $\varepsilon = |S|$. For each value of ε, simulations were performed across all possible numbers of erroneous symbols v, and the results were aggregated to obtain the overall decoding failure probability. The $\mathrm{RS}(40, 32)$ code was used, and the experiment exhaustively evaluated all possible single-chip error patterns $(256^4 - 1)$, assuming a specific chip j to be faulty.

B. Failure Pattern Distribution

Table II summarizes the distribution of decoding failure patterns for each number of erasures ε. The failure of single-chip erasure decoding arises when decoding also succeeds under an incorrect chip assumption, and the causes can be classified into two categories: (i) cases within the correction condition, and (ii) cases beyond the correction condition, where a miscorrection occurs. A detailed discussion of these causes is provided in Section V.

Based on these two causes, the failure patterns for each ε are analyzed as follows:

- $\varepsilon = 1$: Since $v + \varepsilon/2 \leq 4$ holds for all $v \leq 3$, all such patterns lead to failure, while $v = 4$ does not contribute to failure. The failure probability is approximately 1.5534%.
- $\varepsilon = 2$: All patterns with $v \leq 3$ fail, and an additional $73{,}440$ miscorrection patterns occur at $v = 4$. The failure probability is approximately 1.5551%.
- $\varepsilon = 3$: All patterns with $v \leq 2$ fail, with an additional $1{,}211{,}760$ miscorrection patterns at $v = 4$. The failure probability is approximately 0.0373%.
- $\varepsilon = 4$: All patterns with $v \leq 2$ fail, with an additional $4{,}847{,}040$ miscorrection patterns at $v = 3$ and $82{,}620$ at $v = 4$. The failure probability is approximately 0.1239%.

C. Failure Probability Trends

Synthesizing the results of the decoding failure pattern analysis for each erasure size ε, the differences in failure probability can be explained by two main factors. First, the difference between $(\varepsilon = 1, 2)$ and $(\varepsilon = 3, 4)$ stems from the range of error patterns that satisfy the correction condition $v + \varepsilon/2 \leq t$. In the former, all patterns with $v \leq 3$ fall within this range, whereas in the latter, only patterns with $v \leq 2$ do. Second, for $\varepsilon = 3$ or 4, the number of patterns satisfying the correction condition decreases, and the failure probability is primarily determined by whether miscorrections occur beyond this condition. In particular, when $\varepsilon = 3$, no miscorrections occur at $v = 3$, resulting in a very low failure probability. In contrast, when $\varepsilon = 4$, miscorrections arise even at $v = 3$, slightly increasing the failure probability. In conclusion, compared with the conventional full-chip erasure approach at $\varepsilon = 4$, the proposed partial-chip erasure configuration at $\varepsilon = 3$ offers an advantage in terms of failure probability.

V. DISCUSSION

A. Causes of Decoding Failure

In our simulations, each assumed faulty chip was regarded as successful if the RS decoder produced a codeword, and as a failure otherwise. Since this criterion depends only on whether a codeword is obtained rather than its correctness, it provides a limited assessment of decoder performance.

The analysis reveals two primary causes of decoding failures:

1) *Within the correction condition* $(v + \varepsilon/2 \leq t)$: Decoding always succeeds even under an incorrect chip assumption, resulting in multiple assumptions being simultaneously classified as successful. This failure is inherent to the correction capability of the RS code and is therefore unavoidable.
2) *Beyond the correction condition with miscorrection* $(v + \varepsilon/2 > t)$: Although the correction condition is exceeded, certain patterns may still lead to miscorrection, causing an incorrect chip assumption to be judged as successful. Unlike the first case, these failures occur only in specific patterns and can potentially be eliminated through additional detection techniques.

In the following subsections, we first address the second cause by introducing a filtering method to remove miscorrections, and then present a parallel decoding framework to mitigate the first cause.

B. Chip-localized Post-decoding Filtering

This subsection addresses the second cause of decoding failure, miscorrection, by introducing a *chip-localized post-decoding filter*. When the correct chip is assumed, all errors and erasures are confined to the four symbols within that chip, so the difference between the corrected codeword and the received vector also remains limited to the same chip. In contrast, under an incorrect chip assumption, a miscorrection incorrectly corrects symbols outside the assumed chip, causing the difference to extend beyond its boundary. Exploiting this property, the filter operates as follows: if the difference is confined within the assumed chip, the decoding result is accepted; if it extends outside, the result is identified as a miscorrection and discarded.

Before applying the filter, the $\varepsilon = 3$ partial-chip erasure yielded a lower failure probability than the $\varepsilon = 4$ full-chip erasure, due to differences in miscorrection occurrence. After applying the filter, all miscorrections with $(\varepsilon, v) = (4, 3)$,

$(4, 4)$, and $(3, 4)$ are removed, making the total failure rates of $\varepsilon = 3$ and $\varepsilon = 4$ identical. However, since $\varepsilon = 3$ provides a larger margin under the correction condition $v + \varepsilon/2 \leq t$, this additional margin is advantageous for extending ECC designs with enhanced error correction or detection capabilities.

C. Parallel Decoding Framework

While the chip-localized post-decoding filter effectively eliminates the second cause of failure, miscorrection, it cannot address the first cause. To mitigate this limitation, we propose employing the single-chip erasure decoding scheme as an auxiliary decoder within a parallel decoding architecture. In this framework, the primary decoder handles low-weight error patterns, and the proposed scheme is invoked only for high-weight patterns that the primary decoder fails to resolve. For example, when $\varepsilon \in 1, 2$, if error patterns with up to three symbols are corrected by the primary decoder, the corresponding cases associated with the first cause of failure can be excluded from the scope of the proposed scheme. Such parallel operation complements existing ECC architectures, improves reliability without additional parity, and provides practical advantages for system design.

VI. Conclusion

This paper proposed a partial-chip extension of single-chip erasure decoding for DDR5 ECC-DIMM systems. The proposed extension reduces decoding failure probability compared with conventional full-chip erasure, while maintaining the same redundancy and offering greater design flexibility. Simulation results demonstrate that, under certain configurations, the proposed scheme achieves up to a 70% reduction in failure probability. Building on the failure analysis, we further introduced a chip-localized post-decoding filter to prevent miscorrections and a parallel decoding framework that applies the scheme selectively to high-weight error patterns. Future work will examine system conditions for effective deployment and investigate the optimal utilization of the saved parity symbols for enhanced correction and detection.

References

[1] S. -L. Gong, J. Kim, S. Lym, M. Sullivan, H. David and M. Erez, "DUO: Exposing On-Chip Redundancy to Rank-Level ECC for High Reliability," *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 683-695, 2018.

[2] B. Schroeder and G. A. Gibson, "A Large-Scale Study of Failures in High-Performance Computing Systems," in *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 337-350, 2010.

[3] R. Nair, "Evolution of Memory Architecture," in *Proc. of the IEEE*, vol. 103, no. 8, pp. 1331-1345, 2015.

[4] S. Park, B. Choi and J. Kim, "C4ECC: Data Compression for Bandwidth Efficiency Under ECC Protection in GPUs," *2025 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1-4, 2025.

[5] S. Park and J. Kim, "Evaluating the Impact of In-band ECC on GPU Performance," *2024 21st International SoC Design Conference (ISOCC)*, pp. 320-321, 2024.

[6] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM errors in the wild: A large-scale field study," in *Proc. SIGMETRICS*, pp. 193–204, 2009.

[7] Sang-Chul Oh et al., "Automatic failure analysis system for high density DRAM," in *Proc. International Test Conference*, pp. 526-530, 1994.

[8] R. P. Cenker, D. G. Clemons, W. R. Huber, J. B. Petrizzi, F. J. Procyk and G. M. Trout, "A fault-tolerant 64K dynamic random-access memory," in *IEEE Transactions on Electron Devices*, vol. 26, no. 6, pp. 853-860, 1979.

[9] Y. Lim, D. Kim and J. Kim, "SELCC: Enhancing MLC Reliability and Endurance with Single-Cell Error Correction Codes," *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1-6, 2024.

[10] R. W. Hamming, "Error detecting and error correcting codes," in *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147-160, 1950.

[11] C. L. Chen and M. Y. Hsiao, "Error-Correcting Codes for Semiconductor Memory Applications: A State-of-the-Art Review," in *IBM Journal of Research and Development*, vol. 28, no. 2, pp. 124-134, 1984.

[12] H. M. Abdullah, U. U. Fayyaz, T. Mahmood and S. Hong, "Enhanced Positional SECDED: Achieving Maximal Double-Error Correction in Racetrack Memories," in *IEEE Transactions on Magnetics*, vol. 60, no. 3, pp. 1-10, 2024

[13] S. Cha et al., "Defect Analysis and Cost-Effective Resilience Architecture for Future DRAM Devices," *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 61-72, 2017.

[14] M. White, J. Qin and J. B. Bernstein, "A study of scaling effects on DRAM reliability," in *Proc. Annual Reliability and Maintainability Symposium*, pp. 1-6, 2011.

[15] J. Liu, B. Jaiyen, Y. Kim, and C. Wilkerson, "An experimental study of data retention behavior in modern DRAM devices: Implications for retention time profiling mechanisms," in *Proc. International Symposium on Computer Architecture (ISCA)*, pp. 60–71, 2013.

[16] G. Jung, H. J. Na, S. -H. Kim and J. Kim, "Dual-Axis ECC: Vertical and Horizontal Error Correction for Storage and Transfer Errors," *2024 IEEE 42nd International Conference on Computer Design (ICCD)*, pp. 409-417, 2024.

[17] Ju H, Kong D-H, Lee K, Lee M-K, Cho S, Kim S-H, "How to Use Redundancy for Memory Reliability: Replace or Code?," *Electronics*, vol. 14, no. 9, 2025.

[18] A. Makihara et al., "Analysis of single-ion multiple-bit upset in high-density DRAMs," in *IEEE Transactions on Nuclear Science*, vol. 47, no. 6, pp. 2400-2404, 2000.

[19] Dell, Timothy J. "A white paper on the benefits of chipkill-correct ECC for PC server main memory," *IBM Microelectronics division*, 1997.

[20] Blankenship, R.G. et al. Memory error detection and/or correction. US Patent 8,250,435, 2012.

[21] P. J. Nair, V. Sridharan and M. K. Qureshi, "XED: Exposing On-Die Error Detection Information for Strong Memory Reliability," in *Proc. 43rd International Symposium on Computer Architecture (ISCA)*, pp. 341-353, 2016.

[22] T. Ha, G. Kim, C. Kim, and S.-H. Kim, "Chipkill-Level ECC Using 4-Bit Symbol Reed-Solomon Codes for DDR5 DRAM," Proc. of the 2025 16th International Conference on Information and Communication Technology Convergence (ICTC), to appear.

[23] Advanced Micro Devices (AMD), *BIOS and Kernel Developer's Guide (BKDG) for AMD Family 15th Models 00h-0Fh Processors*, Jan. 2013.

[24] J. Kim, M. Sullivan and M. Erez, "Bamboo ECC: Strong, safe, and flexible codes for reliable computer memory," in *IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pp. 101-112, 2015.

[25] *DDR4 SDRAM STANDARD,JESD79-4*,Joint Electron Device Engineering Council, Sep. 2012.

[26] *DDR5 SDRAM STANDARD,JESD79-5*,Joint Electron Device Engineering Council, Sep. 2022.

[27] Jyh-Horng Jeng and Trieu-Kien Truong, "On decoding of both errors and erasures of a Reed-Solomon code using an inverse-free Berlekamp-Massey algorithm," in *IEEE Transactions on Communications*, vol. 47, no. 10, pp. 1488-1494, 1999.