Chipkill-Level ECC Using 4-Bit Symbol Reed–Solomon Codes for DDR5 DRAM

Taeuk Ha*, Gyuri 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*—In this paper, we introduce a DRAM Error Correction Code (ECC) providing chipkill-level protection for x4 DDR5 DIMMs using 4-bit symbol Reed–Solomon (RS) codes. Since ECC was introduced to address increasing DRAM error rates caused by technology scaling, RS codes have been widely employed in various chipkill ECC solutions because of their robust burst error correction capability. Due to the limited codeword length of 4-bit symbol RS codes and mismatches with DDR4 DIMM configurations, 8-bit symbol RS codes have traditionally been preferred. However, the sub-channel configuration of DDR5 DIMMs is well-suited for 4-bit symbol RS codes, offering advantages such as reduced computational complexity and smaller lookup table (LUT) sizes owing to the decreased Galois field size. We propose a DRAM ECC scheme based on (10,8) 4-bit symbol RS codes and present comparative results evaluating its error correction and detection capabilities against existing ECC methods.**

***Index Terms*—DRAM ECC, Reed-Solomon codes, Rank-Level ECC, chipkill-level correction**

I. Introduction

Ensuring reliability has become a primary challenge for DRAM, due to the increased error rates introduced by tech scaling. In early DRAMs, most errors were caused by defective cells resulting from manufacturing faults, and the overall error rates were sufficiently low that reliability could be maintained by replacing defective regions using spare rows and columns [1]–[3]. However, the increased error rate caused by sustained technological scaling in DRAM has exceeded the coverage of traditional sparing techniques [4]–[6].

To address this issue, Error Correction Codes (ECC) were introduced into DRAM systems [7]–[10]. Two primary approaches have been proposed: one integrates ECC directly within the DRAM chip, a technique referred to as On-Die ECC (OD-ECC) [11], [12], and the other corrects errors at the DIMM level using additional redundancy chips, commonly known as Rank-Level ECC (RL-ECC) [13].

RL-ECC has commonly employed either Single Error Correction and Double Error Detection (SEC-DED) codes or chipkill-level correction schemes [5]. SEC-DED codes have been widely used due to its simple structure and reasonable correction capability [14]. However, as cell density increased, multi-bit errors—particularly burst errors occurring in adjacent cells within the same chip—became more frequent. To address such faults, more robust chipkill-level correction has been proposed in high-reliability systems [13], [15].

There are various approaches to implementing chipkill-level correction, which refers to the ability to correct all possible error patterns within a single chip. Among them, a commonly used method leverages the symbol-level error correction capability of RS codes to tolerate chip-level faults [16]–[19].

Existing RS code-based chipkill schemes typically employ 8-bit symbols, as the codeword length of 4-bit symbol RS codes has not been compatible with the 72-bit bandwidth configuration of x4 DDR4 ECC DIMMs. However, organized into two 40-bit sub-channels, DDR5 ECC DIMMs now make x4 chipkill-level correction using 4-bit symbol RS codes a practical and compatible option.

In this paper, we propose a novel x4 chipkill correction scheme based on RS codes. The proposed RS code employs 4-bit symbols aligned with chip boundaries, which allows the use of a smaller Galois field and offers several advantages compared to conventional 8-bit symbol chipkill implementations. The remainder of this paper is organized as follows. Section II gives the preliminaries on DRAM organization and related works. In Section III, we propose our x4 chipkill correction scheme. The numerical results are given in Section IV, and we conclude the paper in Section V.

II. Preliminaries

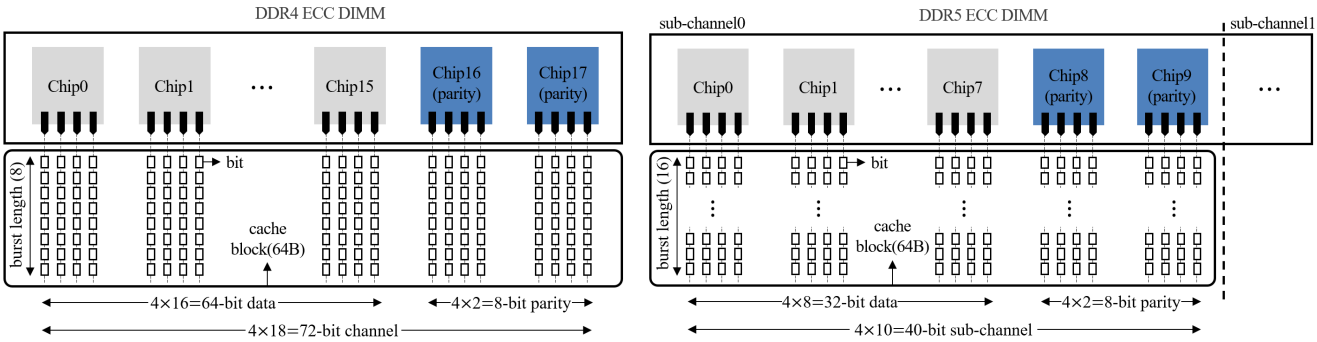
A. Notations and Definitions

Let an m-bit symbol be an element over the Galois field $GF(2^m)$. A code of length N and dimension K is denoted as an (N, K) code. If the code is binary, N and K represent the number of bits; if it is nonbinary (e.g., RS codes), they denote the number of symbols.

B. DRAM Organization

Since the development of DRAM in 1968 [20], it has become an indispensable component in computer systems due to its low cost and high density. A single xN DRAM chip

This work was supported in part by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2024-00398449, Network Research Center: Advanced Channel Coding and Channel Estimation Technologies for Wireless Communication Evolution), 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.)



(a) DDR4 ECC DIMM. (b) DDR5 ECC DIMM.

Fig. 1. DRAM ECC DIMM architectures.

features an N-bit input/output interface through N data pins (DQs), and multiple such chips are assembled on a Dual In-line Memory Module (DIMM) to provide a wider memory interface. For example, a DDR4 non-ECC DIMM typically consists of either sixteen x4 chips or eight x8 chips to form a 64-DQ interface. This enables data transfers in 64-bit chunks, which are repeated to compose a cache block—typically 64 bytes in size—for communication between the processor and memory. A total of eight 64-bit transfers are performed to construct a 64-byte cache block, and this repetition is referred to as the burst length.

To provide ECC redundancy, DDR4 ECC DIMMs extend the standard 64-bit data interface to 72 bits by incorporating an additional x8 DRAM chip [21]. With this 12.5% redundancy, RL-ECC schemes such as SEC-DED and chipkill-level correction can be applied.

DDR5 DIMMs introduced several architectural changes, the most notable of which is the increase in burst length from 8 to 16. As a result, only 32 DQs are needed to construct a 64-byte cache block, and each DIMM is divided into two independent 32 DQ-wide sub-channels. In ECC-enabled DIMMs, each sub-channel adds 8 extra DQs for redundancy, forming a 40 DQ-wide interface, which can be implemented using ten x4 chips or five x8 chips [22]. We illustrate these ECC DIMM organizations in Fig. 1.

Although the redundancy ratio increases to 25%, each sub-channel still uses 8 redundant DQs—equivalent to DDR4—allowing existing ECC schemes to be reused. Furthermore, since fewer data chips are used per sub-channel, the overall probability of error occurrence, particularly those exceeding chipkill-correction capabilities, is reduced. Motivated by this DDR5 ECC DIMM configuration, this paper proposes a new chipkill-correction scheme for x4 DDR5 architecture.

C. Prior Works

1) Interleaved SEC-DED: One of the early techniques to provide chipkill-level protection was to apply SEC-DED codes in an interleaved manner [13]. This approach employed four (72,64) SEC-DED codewords, distributing the 4-bit data from a x4 DRAM chip across four different codewords. As a result, the scheme enabled single-chip error correction and double-chip error detection.

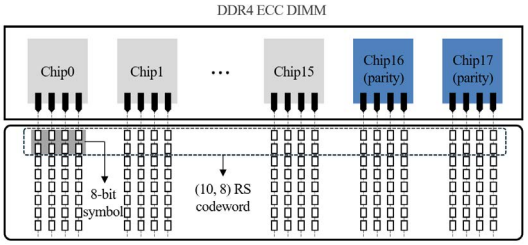
However, it required accessing a total of 256 bits to decode a single logical data unit, which significantly increased memory access overhead. This large access granularity degraded system performance and efficiency, and consequently, this method is no longer used in modern DRAM systems.

2) 4-bit symbol BCH codes: Another approach to ensuring chipkill-level correction is to use symbol-based ECC. Earlier AMD DRAM systems supported two ECC modes: a normal ECC mode using a (72,64) Hamming code for SEC-DED, and a chipkill ECC mode [23]. The chipkill ECC mode employs a (36,32) nonbinary BCH code over 4-bit symbols, where each symbol corresponds to 4 bits of data from a x4 DRAM chip. With single symbol error correction capability, this scheme can tolerate a full-chip failure.

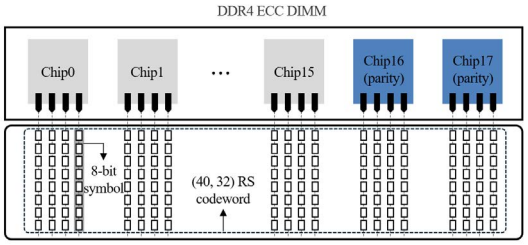
Although RS codes using 4-bit symbols can correct up to two symbol errors with four parity symbols, nonbinary BCH codes were adopted. This choice appears to be due to the limitation of RS codeword length: an m-bit symbol RS code has a fixed length of $N = 2^m - 1$, which is too short to fit the configuration of DDR4 ECC DIMMs.

3) 8-bit symbol RS codes: Subsequently, AMD adopted an 8-bit symbol RS code scheme [15]. In this approach, as shown in Fig. 2(a), each symbol is composed of 8 bits from the same chip, obtained by reading 4 bits twice from a single x4 chip. The 18 symbols from the 18 chips of a DDR4 ECC DIMM form an (18,16) RS code, enabling chipkill-level protection through single symbol error correction. By employing 8-bit symbols, the codeword length can be extended to 255, which can then be shortened to design a codeword length suitable for the DDR4 ECC DIMM configuration.

Furthermore, AMD introduced a mechanism to enhance the detection capability for errors exceeding a single chip failure by maintaining ECC history. Each 64B cache block is composed of four (18,16) RS codewords. The system checks whether a correction occurred for each codeword and records the location of the corrected symbol. If any corrected symbol location differs among the codewords, the corresponding cache block is considered to contain an uncorrectable error due to



(a) 8-bit symbol RS codes.



(b) Bamboo ECC.

Fig. 2. Configurations of existing DRAM ECC schemes.

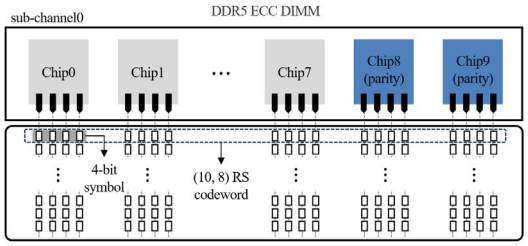


Fig. 3. Configuration of proposed scheme.

miscorrection. This mechanism can also be applied to our proposed scheme, and further details are discussed in Section III.

4) Bamboo ECC: Another DRAM ECC technique employing RS codes is Bamboo ECC [18]. Unlike the previous chip-boundary-aligned symbol organization, Bamboo ECC groups per-pin 8-bit data into a single symbol, as illustrated in Fig. 2(b), and applies 8-bit symbol RS codes to provide chipkill-level protection. In both DDR4 and DDR5 environments, Bamboo ECC can correct all possible faults in x4 chips using a (72,64) and a (40,32) RS code, respectively, each with four-symbol error correction capability. Furthermore, it can also correct up to four random pin faults. By employing long codewords, Bamboo ECC enhances error detection capability and additionally proposes a variant that further improves detection capability by sacrificing a portion of the random pin error correction capability.

III. Proposed Scheme

In this section, we present the code construction of the proposed chipkill scheme for x4 DDR5 ECC DIMM.

A. ECC Construction

Similar to prior works excluding Bamboo ECC, we use symbols aligned with chip boundaries. As shown in Fig. 3, 4-bit data from a single chip are grouped into one symbol, and errors are corrected using a (10,8) RS code. While this RS symbol configuration was impractical for DDR4 DIMMs due to the codeword length limitation described in Section II, the narrower sub-channel bandwidth in DDR5 DIMMs makes it feasible. A 64B cache block consists of 16 codewords, and single symbol error correction for each codeword enables correction of any single chip failure. If any of the codewords in a cache block contains a detectable but uncorrectable error (DUE), the entire cache block is considered to have encountered a DUE.

This simplified codeword construction offers several advantages. Unlike an 8-bit symbol that must handle 256 elements in $GF(2^8)$, a 4-bit symbol only deals with 16 elements in $GF(2^4)$, thereby reducing the complexity of computation circuits and the size of lookup tables (LUTs). Moreover, decoding a single-symbol error-correcting RS code does not require complex procedures such as the Berlekamp–Massey algorithm, Chien search, or the Forney algorithm [24]. Consequently, the proposed scheme achieves a simpler encoder/decoder architecture, reduced circuit size, and lower power consumption.

B. ECC History Mechanism

While a 4-bit symbol RS code offers several advantages, it suffers from reduced error detection capability compared to using 8-bit symbols. The mother codes for 4-bit and 8-bit symbol RS codes have length 15 and 255, respectively; however, when both are shortened to a length of 10, the minimum distance remains the same. Nevertheless, in the 8-bit symbol RS code, the number of valid codewords is significantly reduced, which leads to a much lower probability of miscorrection. This difference is also evident in the error correction outcome ratios presented in Section IV.

To enhance error detection capability, we consider an additional mechanism similar to that used in [17], which leverages ECC history. A 64B cache block with a burst length of 16 is divided into two 32B data blocks, each with a burst length of 8. For each data block, we record whether a correction occurred and the location of the corrected symbol for all eight constituent codewords. In the case of a single chip failure, all errors will be corrected and the correction locations will be identical across all codewords. However, when errors exceed a single chip failure, some codewords may either fail to correct the errors or result in miscorrection.

For example, as illustrated in Fig. 4, when a single chip failure coincides with a single bit error, the codeword containing the single bit error will have two symbol errors, making it uncorrectable. In the worst case, this may result in a miscorrection, delivering erroneous data to the processor. To prevent this, we utilize the recorded correction locations of the eight codewords. If all corrected codewords report identical error locations, the failure is classified as a single-chip error and the decoded results are accepted. However, if any corrected error occurs at a different location, it is considered a case

TABLE I. Decoding results under different error scenario.

Error Scenario	Decoding Result	8-bit RS	Bamboo	4-bit RS	4-bit RS + ECC History
1 Chip Error	CE(%)	100	100	100	100
1 DQS Error + 1 bit Error	CE(%)	75.0372	93.3315	87.4942	0
	DUE(%)	24.2225	6.6685	5.8253	93.3195
	SDC(%)	0.7403	0	6.6805	6.6805
1 Chip Error + 1 bit Error	CE(%)	0.3918	1.5398	6.2437	0
	DUE(%)	96.4792	98.4587	43.7247	100
	SDC(%)	3.129	0.0015	50.0316	0
2 Chip Error	CE(%)	0	0	0	0
	DUE(%)	96.8623	99.998	98.5609	100
	SDC(%)	3.1377	0.002	1.4391	0

where errors beyond a single chip failure have caused a miscorrection. Although this history mechanism may classify certain correctable errors as uncorrectable—thereby lowering the correction success rate—it can significantly reduce the rate of undetected errors.

IV. Simulation Results

To evaluate the proposed scheme, we compare its performance against 8-bit symbol RS codes and Bamboo ECC. In a DDR5 ECC DIMM environment, a 32B data block is encoded according to each scheme, after which errors are injected and decoding is performed. Four error scenarios are considered: (i) one chip error, (ii) one DQS error plus one bit error, (iii) one chip error plus one bit error, and (iv) two chip errors. For each scenario, the affected bits are assumed to flip independently with probability 1/2, and each scenario is repeated 10^6 times.

The decoding results are categorized into three outcomes: CE (Correctable Error), DUE (Detectable but Uncorrectable Error), and SDC (Silent Data Corruption). The distribution of these outcomes is reported in Table I. Among these, SDC represents cases where errors occur but remain undetected; such cases are particularly critical because corrupted data may be silently propagated within the system. This can lead to delayed error discovery and high recovery costs, making the reduction of SDC probability as important as maximizing the number of correctable errors.

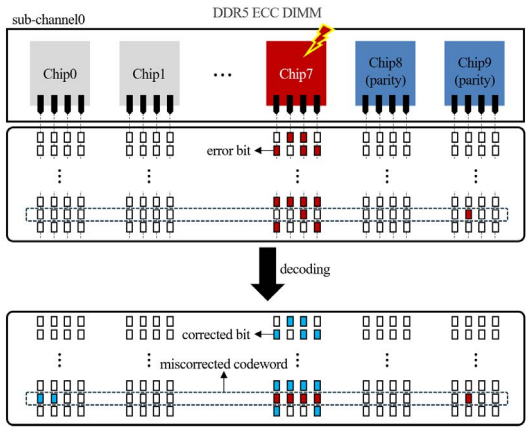


Fig. 4. Miscorrection caused by multi-chip errors.

For the 4-bit RS codes without ECC history, the performance was generally weaker than Bamboo ECC but stronger than 8-bit symbol RS codes across most scenarios. However, in the 1 chip error + 1 bit error scenario, the limited detection capability resulted in an excessively high SDC rate exceeding 50%. To address this detection weakness, we incorporated the ECC history mechanism. While this enhancement did not enable correction of errors beyond a single chip failure, it significantly reduced the SDC rate to negligible levels.

V. Conclusion

This paper presents an RL-ECC scheme for DDR5 ECC DIMMs. We introduced a DRAM ECC architecture that leverages 4-bit symbol RS codes to provide x4 chipkill-level protection and compared its performance against two prior schemes. The proposed design provides simplicity, but has the limitation of weaker error detection. Our simulation demonstrats that adding an ECC history mechanism can greatly reduce SDCs. As future work, we plan to evaluate and compare the area, latency, and power consumption of the 4-bit symbol RS code encoder/decoder to further assess its practicality.

References

[1] S. E. Schuster, “Multiple word/bit line redundancy for semiconductor memories,” *IEEE J. Solid-State Circuits*, vol. 13, no. 5, pp. 698–703, Oct. 1978.

[2] 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,” *IEEE Trans. on Electron Devices*, vol. 26, no. 6, pp. 853-860, June 1979.

[3] R. Nair, “Evolution of Memory Architecture,” *Proc. IEEE*, vol. 103, no. 8, pp. 1331–1345, Aug. 2015.

[4] O. Mutlu, “Memory scaling: A systems architecture perspective,” in *Proc. IEEE Int. Memory Workshop (IMW)*, May 2013, pp. 21–25.

[5] S. Cha, S. O, H. Shin, S. Hwang, K. Park, and S. J. Jang, et al., “Defect analysis and cost-effective resilience architecture for future DRAM devices,” in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Austin, TX, USA, Feb. 2017, pp. 61–72.

[6] H. Ju, D.-H. Kong, K. Lee, M.-K. Lee, S. Cho and S.-H. Kim, ”How to use redundancy for memory reliability: Replace or code?,” Electronics, vol. 14, no. 9, p. 1812, 2025.

[7] G. Jung, H. J. Na, S.-H. Kim and J. Kim, ”Dual-Axis ECC: Vertical and Horizontal Error Correction for Storage and Transfer Errors,” *Proc. IEEE Int. Conf. Computer Design (ICCD)*, Milan, Italy, 2024, pp. 409–417.

[8] 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), Osaka, Japan, 2025, pp. 1–4.

[9] S. Park and J. Kim, ”Evaluating the impact of in-band ECC on GPU performance,” 2024 21st International SoC Design Conference (ISOCC), Sapporo, Japan, 2024, pp. 320–321.

[10] G. Kim, T. Ha, J. Kim, C. Kim, and S.-H. Kim, ”Partial-Chip Extensions of Single-Chip Erasure Decoding for Flexible Use of Redundancy,” Proc. of the 2025 16th International Conference on Information and Communication Technology Convergence (ICTC), to appear.

[11] K. Furutani, K. Arimoto, H. Miyamoto, T. Kobayashi, K. Yasuda, and K. Mashiko, “A built-in Hamming code ECC circuit for DRAMs,” *IEEE J. Solid-State Circuits*, vol. 24, no. 1, pp. 50–56, Feb. 1989.

[12] J. Shin and J. Kim, ”ROSE: Reliability-Optimized OD-ECC and S-ECC Enhancements for HBM3,” in *Proc. 2025 Int. Conf. on Electronics, Information, and Communication (ICEIC)*, Osaka, Japan, 2025, pp. 1-4.

[13] T. J. Dell, “A white paper on the benefits of Chipkill-correct ECC for PC server main memory,” IBM Microelectronics Div., White Paper 11.1-23, pp. 5–7, 1997.

[14] H. M. Abdullah, U. U. Fayyaz, T. Mahmood and S. Hong, "Enhanced positional SECDED: Achieving maximal double-error correction in racetrack memories," IEEE Transactions on Magnetics, vol. 60, no. 3, pp. 1–10, Mar. 2024.
[15] M. V. Beigi, Y. Cao, S. Gurumurthi, C. Recchia, A. Walton, and V. Sridharan, "A systematic study of DDR4 DRAM faults in the field," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Montreal, QC, Canada, Feb. 2023, pp. 991–1002.
[16] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jan. 1960.
[17] Advanced Micro Devices (AMD), Inc., *BIOS and Kernel Developer's Guide (BKDG) for AMD Family 15h Models 00h–0Fh Processors*, Jan. 2013.
[18] J. Kim, M. Sullivan, and M. Erez, "Bamboo ECC: Strong, safe, and flexible codes for reliable computer memory," in *Proc. IEEE 21st Int. Symp. High Perform. Comput. Archit. (HPCA)*, Burlingame, CA, USA, Feb. 2015, pp. 101–112.
[19] R. Yeleswarapu and A. K. Somani, "Addressing multiple bit/symbol errors in DRAM subsystem," *PeerJ Comput. Sci.*, vol. 7, Art. no. e528, 2021.
[20] R. H. Dennard, "Field-effect transistor memory," U.S. Patent 3,387,286 A, Jun. 4, 1968.
[21] JEDEC Solid State Technology Association, *DDR4 SDRAM Standard*, JESD79-4, Sep. 2012.
[22] JEDEC Solid State Technology Association, *DDR5 SDRAM Standard*, JESD79-5, Jul. 2020.
[23] Advanced Micro Devices (AMD), Inc., *BIOS and Kernel Developer's Guide for AMD NPT Family 0Fh Processors*, Jul. 2007.
[24] S. Pontarelli, P. Reviriego, M. Ottavi, and J. A. Maestro, "Low delay single symbol error correction codes based on Reed–Solomon codes," *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1497–1501, May 1, 2015.