Towards Robust Memory Systems: Enhancements to DRAM On-Die ECC for Emerging Integrity Issues

Himanshi Gupta, Krishil Gandhi

Abstract—Ensuring memory integrity and reliability in DRAM requires stronger error correction to combat scaling issues introduced with each new generation of DRAM process nodes. This research focuses on enhancing on-die ECC efficiency by leveraging strong error-correcting codes for robust error mitigation. By designing a lightweight decoder that can correct multibit errors in DRAM, we aim to strengthen DRAM resilience while minimizing performance and power overhead. This approach seeks to provide a scalable and future-proof solution to protect memory against evolving reliability issues due to DRAM scaling.

1 Introduction

As DRAM technology continues to scale to smaller nodes and higher densities, it increasingly suffers from reliability challenges that compromise data integrity and system stability. Faults in DRAM can arise from a variety of sources—transient faults caused by cosmic radiation, as well as permanent faults stemming from failures in internal components such as the memory cells, sense amplifiers, wordline drivers, I/O pins, and even entire chips. Traditional on-die Error-Correcting Codes (ECC), such as Single-Error Correction, Double-Error Detection (SECDED), are effective against isolated single-bit faults but offer limited protection against more complex failure modes, such as those triggered by circuit failure induced multi-bit errors. In modern computing systems, especially those using LPDDR or HBM memory, where replacing faulty modules is difficult or impractical, enhancing the Reliability, Availability, and Serviceability (RAS) of DRAM is crucial. This necessitates stronger error correction schemes that can tolerate and recover from multiple faults within a cacheline, thereby extending DRAM lifetime and improving system resilience in applications ranging from data centers to safety-critical environments like automotive systems.

2 Background

Error Correction Codes (ECC) are fundamental to addressing data reliability challenges in memory systems. By adding redundancy in the form of parity bits to the original data, ECC enables detection and correction of bit errors introduced during storage or transmission. A common representation of an ECC scheme is the (n, k) code, where n is the total number of bits in the codeword and k is the number of data bits. The code rate r = k/n indicates the storage efficiency of the scheme. Over time, various ECC schemes—such as Hamming codes, repetition codes, BCH codes, and Reed-Solomon codes—have been developed to support different levels of error resilience, from single-bit to multi-bit correction.

With the growing complexity and density of DRAM, especially in modern DDR, LPDDR, and HBM technologies, error correction has become increasingly critical. In ECC-enabled DDR modules (typically used in servers), the standard SECDED scheme, a variant of the Hamming code, offers single-bit correction and double-bit error detection. Traditionally, this decoding and correction are handled by the memory controller (MC). However, the emergence of DDR5 introduced on-die ECC—embedding SECDED directly on the DRAM chip—to tackle the rising incidence of cell-level faults and scaling-induced errors.

High-bandwidth memory (HBM), due to its 3D stacking and dense architecture, is particularly prone to multi-bit and burst errors. To mitigate these, more robust ECC schemes like Reed-Solomon codes are employed. These symbol-based codes can correct entire bytes or symbols (typically 8–16 bits), offering protection against clustered faults that simpler bit-level codes like SECDED cannot handle.

While these advancements have improved DRAM resilience, existing solutions present significant limitations. As shown in the surveyed solutions, memory-controller-based and hybrid ECC techniques (e.g., Chipkill, SafeGuard, XED, COMET) [2], [6], [1] rely heavily on off-chip correction, introducing latency, storage overhead, and increased system complexity. On the other hand, on-die ECC solutions—although beneficial for reducing correction latency—are typically limited to correcting only one bit per word. They struggle with clustered or random multi-bit errors, suffer from hardware overhead due to redundant logic, and are vulnerable to silent data corruption due to limited detection capabilities. Given these constraints, stronger on-die ECC solutions are essential to ensure data integrity in future DRAM technologies.

3 Our Insights

Designing effective error correction for modern DRAM systems requires more than just stronger codes—it requires aligning the correction mechanism with the nature of the faults, memory architecture, and real-world error behavior.

First, the type and granularity of faults significantly influence the effectiveness of ECC. Burst errors, such as those caused by coupling or particle strikes, demand strategies different from random single-bit errors. Likewise, permanent faults behave differently from intermittent or transient faults, which require tailored approaches. To be effective, ECC must operate at the appropriate level —bit, word, or symbol—depending on where the errors tend to occur.

Second, error correction should be guided by observed error patterns in the field. Real-world DRAM faults are not uniformly distributed; they tend to cluster or follow specific activation behaviors. By tailoring ECC schemes to these patterns, rather than assuming uniformly random faults, we can improve both efficiency and protection.

Third, the memory type impacts the ECC design space. Different DRAM technologies like DDR5, LPDDR5, and HBM come with varying performance, power, and area constraints. For example, LPDDR5 prioritizes energy efficiency, limiting the complexity of ECC, while the dense and stacked HBM layout increases the risk of burst faults, making it a better candidate for multi-bit correction codes such as Reed-Solomon or BCH.

Additionally, simply applying stronger codes like BCH or Reed-Solomon is not sufficient. The effectiveness of ECC also depends heavily on how codewords are mapped in memory and on the statistical likelihood of faults. For example, prior work has shown that remapping codewords to column planes without changing the ECC logic can reduce vulnerability to failure . Other studies propose scrambling chip-to-DRAM address mappings to introduce randomness, which, when combined with on-die ECC (e.g., SECDED) and higher-level protection like Chipkill, mitigates RowHammer-induced multi-bit errors [4].

Therefore, as we evaluate the BCH codes for DDR5, our focus extends beyond the correction strength. We aim to provide a realistic picture of the storage, latency, and reliability tradeoffs with architectural constraints and real-world fault behavior, to arrive at a practical and scalable on-die ECC solution.

4 Simulators for Evaluating ECC Schemes

This study aims to demonstrate that the proposed ECC scheme can effectively correct multi-bit errors in both current and future memory systems while minimizing overhead. To validate this, we rely on both analytical and empirical methods, using simulators capable of modeling various DRAM fault types and ECC mechanisms. Key features of the simulator include empirical FIT rates for DRAM components, support for a broad range of ECC solutions, and flexibility in simulating diverse fault types, with adjustable fault rates to account for scaling

4.1 FaultSim Evaluation

FaultSim is a fast, configurable memory reliability simulator for 2D and 3D-stacked memory systems. It uses Monte Carlo simulations based on real-world failure statistics and an event-based fault injection framework to evaluate standard ECC schemes like SECDED and ChipKill. However, its evaluation is limited to DDR5 memory only, and it does not support other memory configurations such as LPDDR5 or HBM. Moreover, ECC word layout is fixed, and the simulator lacks features to assess on-die ECC schemes or scaling-induced faults, making it less suitable for exploring emerging reliability techniques.

4.2 DRAM Fault Sim Evaluation

DRAM Fault SIM [3] advances DRAM reliability simulation by going beyond traditional fault injection techniques that only assess whether a fault could lead to failure. Instead, it refines reliability estimation through a two-level Monte Carlo simulation: first, injecting faults; then, simulating the resulting error patterns. These patterns are subsequently processed through error-correcting code (ECC) mechanisms to provide a realistic estimate of system-level reliability, including Silent Data Corruption (SDC) rates. This methodology is particularly valuable for evaluating the impact of scaling-induced faults.

At its core, DRAM Fault SIM uses an empirically derived, component-level DRAM fault model. This model is parameterized and designed for extensibility to accommodate both existing and future memory technologies. To develop this model, the simulator leverages a large-scale DRAM error log dataset released by Alibaba, encompassing over 75.1 million correctable errors recorded across 30,496 servers and more than three million DIMMs over eight months. The dataset includes detailed metadata—timestamps, addresses (row, column, bank, rank), memory IDs, and server IDs—as well as issue tickets for server failures, with around 12% of servers reporting memory errors.

Using this dataset, DRAM Fault SIM maps observed errors to root causes, building a comprehensive fault taxonomy. Faults are categorized into types such as cell faults, bitline sense amplifier faults, row/column decoder faults, sub-wordline and driver faults, column multiplexer faults, and higher-level faults involving banks and ranks.

Unlike prior simulators that categorize faults only at a logical level, DRAM Fault SIM improves accuracy by accounting for physical component-level correlations. This prevents unrealistic fault overlaps, such as two row-level faults being incorrectly treated as independent, resulting in more precise reliability estimates.

The simulator also integrates the scaled-DRAM inherent fault model proposed by Gong et al., which captures variation in weak cell susceptibility due to technology scaling. It models fault occurrence based on the ratio of weak cells and a constant activation probability per cell.

Finally, DRAM Fault SIM supports reliability evaluation across various memory technologies and ECC schemes, including emerging ones like LPDDR5 and HBM3. This allows researchers to assess how effectively different ECC implementations mitigate reliability concerns under realistic fault conditions.

5 Evaluating different ECC schemes

We will be looking at three potential on-die ECC schemes for DRAM, which include SECDED (1-bit error correction and 2-bit error detection), BCH (multi-bit error detection and correction), and Reed-Solomon codes (single-symbol correction and double-symbol detection) as shown in Fig 1. We have looked at the tradeoffs of these codes in terms of reliability, storage overhead, latency, and area in the following section.

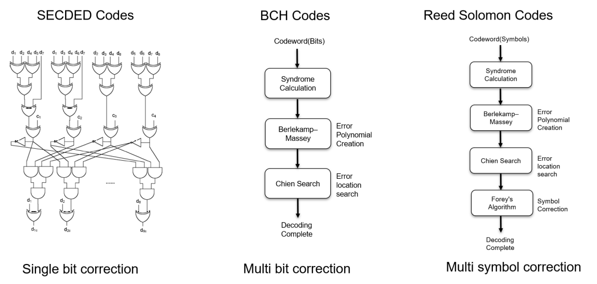
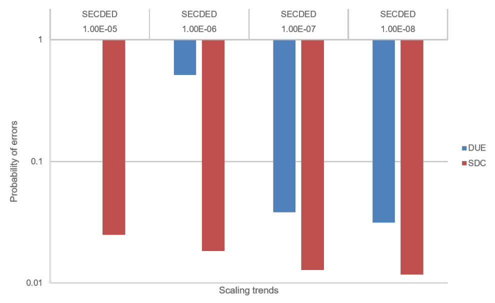


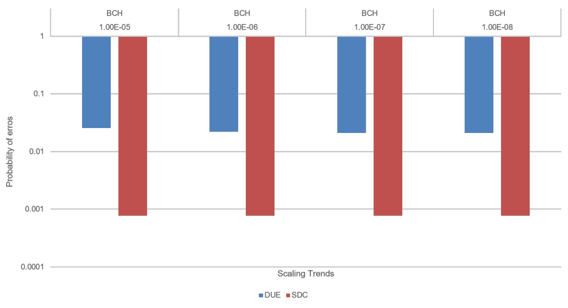
Fig. 1: Algorithm of various ECC schemes

5.1 Reliability analysis

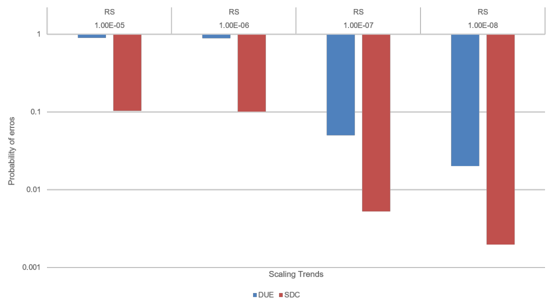
In this section, we use DRAM Fault Sim [3] to understand the error correction capability of the different modalities of error patterns present in the Alibaba dataset.



(a) SECDED Codes



(b) BCH Codes



(c) Reed Solomon codes

Fig. 2: Error probability of a DUE or an SDC in 5 years in DDR5 chips for the three error correction codes

5.1.1 *Experiment setup*

ECC Type: SEC-DED, BCH, Reed Solomon
Iterations: 1 million
Mode: S (System Evaluation Mode)
Stochastic Parameters:

- 10^{-9}: Permanent Rate — Fraction of DRAM cells with permanent single-bit faults
- 10^{-5}: Permanent Activation Probability — Chance that a permanent fault causes an error
- 10^{-5}: Intermittent Rate — Fraction of DRAM cells with intermittent (rare) faults
- 10^{-7}: Intermittent Activation Probability — Chance that an intermittent fault causes a VRT error

We vary the fraction of DRAM cells with intermittent faults to model the effect of an increasing number of weak cells due to technology scaling.

DRAM Configuration:

- **Type:** DDR5
- **Structure:** 4 fault domains/group, 4 ranks/domain, 4 devices/rank, 8 pins/device, 16 burst length/pin

5.1.2 *Reliability results*

The comparison between SECDED, BCH, and Reed–Solomon (RS) error correction schemes across varying bit error rates highlights the limitations of SECDED and the robustness of more advanced codes like BCH and RS. SECDED, with its single-bit correction capability, exhibits a constant and high silent data corruption (SDC) probability of around 10^{-3} across all scaling trends, indicating poor resilience to multi-bit errors. Although its detected unrecoverable error (DUE) rate decreases from approximately 10^{-2} to 10^{-4} as the bit error rate improves, it remains insufficient for modern systems where multi-bit faults are prevalent.

In contrast, BCH codes exhibit superior resilience due to their ability to correct multiple random bit errors—typically two or more—within a data word, rather than being limited to specific error patterns. This capability enables BCH to maintain a significantly lower and more stable SDC rate of about 10^{-6}, offering stronger protection against undetected errors. Its DUE rate ranges from 10^{-3} to 10^{-5}, representing a reasonable trade-off between detection and correction.

On the other hand, Reed–Solomon (RS) codes, such as RS8(34,32), are optimized for burst error correction due to their symbol-based structure, where each symbol spans multiple bits (e.g., 8 bits). While RS achieves a slightly better DUE rate—dropping as low as 10^{-6}—its SDC remains higher than that of BCH at around 10^{-5}, as it is less effective at correcting randomly scattered bit errors.

Given the critical importance of minimizing silent data corruption in reliable systems, BCH emerges as the most suitable choice due to its consistent suppression of SDC and its inherent strength in correcting random, rather than burst, bit errors.

5.2 Hardware overhead analysis

It is important to understand the hardware trade-offs associated with implementing different ECC (Error Correction Code) schemes on-die, as DRAM real estate is highly cost-sensitive. Ideally, ECC decoders should introduce minimal hardware overhead. Additionally, since ECC logic lies on the critical path of DRAM read and write operations, it must not introduce significant latency in the normal execution flow.

To make a fair comparison, we evaluate different ECC schemes such as SEC-DED, BCH, and Reed-Solomon (RS) using a similar codeword size—approximately 256 bits of information data. Between SEC-DED and BCH, the increase in parity overhead for comparable data group sizes is around 1.8%.

To estimate the number of gates required for SEC-DED, we follow a deduction-based approach. While BCH and

Metrics	ECC Code		
	SECDED [7]	**BCH [5]**	**Reed Solomon [8]**
Error correction capability	1 bit	2 bits (random)	1 symbol (burst)
Additional parity bit storage	4.2% (11 bits for 256 bits data)	6% (16 bits for 255 bits data)	11% (32 bits for 272 bits of data)
Latency	0.825 ns for 45 nm (80 inv delay)	2.5 ns for 90 nm	1.3 ns for 28 nm
Area	$\sim$1500 gates	$\sim$4400 gates	$\sim$7980 gates

TABLE 1: Comparison of SECDED, BCH, and Reed Solomon error correction codes

RS decoders involve significantly more complexity, the gate count and latency metrics used for them are derived from synthesizable RTL implementations. However, a similar gate-level breakdown as used for SEC-DED can be applied to analyze these codes in detail.

SECDED Decoder Gate Count Estimation
Given: 256 information bits and 266 received bits (including 9 parity bits and 1 overall parity bit).

Gate Estimation: For syndrome computation, each of the 9 syndrome bits is computed by XOR-ing approximately half of the 265 received bits, requiring about 131 2-input XOR gates per bit. This results in $9 \times 131 = 1179$ XOR gates. The overall parity check involves XOR-ing all 265 bits, requiring 264 XOR gates. For error correction, a single XOR gate is used to flip the erroneous bit. Control logic for error classification and correction is estimated at 30–50 gates; we assume 50 gates.

Total Estimated Gate Count: $1179 + 264 + 1 + 50 = \boxed{1494}$ gates.

6 Future Work

To improve DRAM reliability without incurring high performance or area penalties, we propose a multi-tier ECC scheme that combines lightweight and selective strong error correction mechanisms. In this approach, each die is equipped with low-overhead Single Error Correction (SEC) to efficiently correct frequent single-bit errors. In parallel, a hard-decision LDPC decoder monitors the ECC codewords to detect the presence of multibit errors. When a multibit error is detected, the die issues a catchword to the memory controller, signaling both the presence of the error. Rather than invoking full-chip correction, soft LDPC decoding is done only for the affected bank in DRAM. Meanwhile, other banks remain accessible, allowing continued data access and minimizing latency penalties. This multi-tier approach balances correction strength and system overhead, offering scalable protection against increasing fault rates while preserving system throughput and energy efficiency.

6.1 LDPC standalone module

Low Density Parity Codes (LDPC), known for approaching the Shannon limit, offer a principled approach to achieving robust and correctable error patterns. While prior work (e.g., Cai et al.) has demonstrated LDPC's effectiveness in SSDs using soft decoding, its application to DRAM is less explored due to the latency and storage overhead associated with large codewords. Our idea investigates LDPC configurations optimized for DRAM, targeting a balance between correction capability and performance.

We evaluate a standalone LDPC decoder based on the iterative belief propagation algorithm. The decoder operates on soft inputs in the form of log-likelihood ratios (LLRs), derived from signals transmitted over an Additive White Gaussian Noise (AWGN) channel. Variable and check nodes iteratively exchange LLRs to refine bit estimates, leveraging both the parity-check structure and probabilistic information. After a fixed number of iterations, hard decisions are made based on the sign of the final LLRs. Using a (7,4) Hamming code as a test case, our setup demonstrates the decoder's ability to correct channel-induced errors with improved confidence through soft decoding.

7 Conclusion

The need for smarter, low-overhead error correction is crucial, where adaptive ECC schemes can utilize fault granularity and spatial spread to reduce hardware costs while maintaining efficiency. We found that BCH codes have the potential to be used as an on-die ECC solution as they can correct multi-bit errors while not suffering from issues present in Reed Solomon codes, such as a complex decoder and inability to correct random errors in the codeword. Transparency from server vendors is vital, as sharing detailed memory error data allows system architects to design more effective protection mechanisms. Without awareness of real-world faults, even the best protection strategies are ineffective. Looking forward, multi-tier ECC presents a promising solution, where layered and adaptive fault-tolerance strategies can address both known failure modes and emerging fault patterns.

References

[1] I. Alam and P. Gupta, "Comet: On-die and in-controller collaborative memory ecc technique for safer and stronger correction of dram errors," in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2022, pp. 124–136.

[2] A. Fakhrzadehgan, Y. N. Patt, P. J. Nair, and M. K. Qureshi, "Safeguard: Reducing the security risk from row-hammer via low-cost integrity protection," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 373–386.

[3] J. Jung and M. Erez, "Predicting future-system reliability with a component-level dram fault model," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 944–956. [Online]. Available: https://doi.org/10.1145/3613424.3614294

[4] M. J. Kim, M. Wi, J. Park, S. Ko, J. Choi, H. Nam, N. S. Kim, J. H. Ahn, and E. Lee, "How to kill the second bird with one ecc: The pursuit of row hammer resilient dram," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 986–1001. [Online]. Available: https://doi.org/10.1145/3613424.3623777

[5] Y.-M. Lin, H.-C. Chang, and C.-Y. Lee, "Improved high code-rate soft bch decoder architectures with one extra error compensation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 11, pp. 2160–2164, 2013.

[6] P. J. Nair, V. Sridharan, and M. K. Qureshi, "Xed: Exposing on-die error detection information for strong memory reliability," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 341–353.

[7] P. Reviriego, S. Pontarelli, J. A. Maestro, and M. Ottavi, "A method to construct low delay single error correction codes for protecting data bits only," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 3, pp. 479–483, 2013.

[8] C. Shin and J. Park, "Dbb-ecc: Random double bit and burst error correction code for hbm3," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2025.