Error Detecting and Correcting Codes for DRAM Functional Safety

Steffen Buch
Embedded Business Unit
Micron Technology, Inc.
Munich Germany
sbuch@micron.com

KEYWORDS

DRAM, LPDDR, DDR, RAS, ECC, EDC, CRC, SECDED, SEC, functional safety, ISO26262, failure modes, diagnostic coverage, Hamming

ABSTRACT

With the evolution of Advanced Driver Assistance Systems (ADAS) and Autonomous Driving (AD) in vehicles, processing requirements in Electronic Control Units (ECUs) increase significantly. Traditional microcontrollers using embedded flash and/or embedded SRAM to store data and execute program code are not performant enough. These automotive microcontrollers have for a long time been optimized for best quality and highest safety standards. New architectures with high performance multi-core System-on-Chip (SoC) designs and significant external DRAM need to follow the same path to assure functional safety.

In this paper, we first provide an overview over failure modes of DRAM devices. After this we'll analyze the behavior of common Error Correcting (ECC) and Error Detecting (EDC) Codes with the respect to their coverage of DRAM failure modes. We will show that simple Single-Bit Correcting Double-Bit Detecting (SEC-DED) ECC codes do not provide sufficient Diagnostic Coverage (DC) for multi-bit (MBE) and addressing errors that occur in DRAM devices with significantly high FIT (Failure-In-Time) rates. The properties of popular ECC and EDC codes for their MBE DC will be derived analytically and in addition verified through simulations. Coding schemes with sufficient DC to build functionally safe DRAM sub-systems will be proposed.

We will finally provide a generic equation to calculate MBE DC for any linear block code including CRC, Hamming, Bose-Chaudhuri-Hocquenghem (BCH) and Reed-Solomon (RS) codes.

Introduction

Functional Safety (FuSa) is defined by ISO 26262 [1] as the "*absence of unreasonable risk … due to hazards … caused by malfunctioning behaviour … of E/E systems.*" E/E stands for electrical and electronic. As a part of the efforts that need to be taken to certify a component under ISO 26262, a detailed quantitative analysis of failure modes and FIT distributions for so-called random HW faults needs to be conducted [2]. Two approaches - Failure Modes, Effects, and Diagnostics Analysis (FMEDA) and quantitative Fault Tree Analysis (FTA) [3] – can be used.

The starting point of the FMEDA is detailed a block-by-block failure mode analysis. Figure 1 shows a block diagram of a typical LPDDR DRAM component. Next, FIT (Failure-In-Time) rates are assigned to each block based on its relative gate count or die size and to the identified block-level failure modes. More details about this process can be found in [2]. Finally, to simplify system analysis, we aggregate all individual block-level failure mode FIT rates up to a more manageable set of top-level failure modes (TLFM). The TLFM represents the failure behavior that the host sees in case a low-level failure occurs.

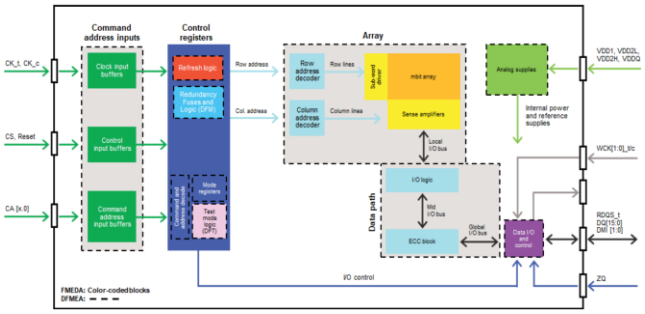

Figure 1: Block diagram of a typical LPDDR DRAM

DRAM Top-Level Failure Modes (TLFM)

Table 1 shows the TLFM identified for LPDDR4 and LPDDR5 components [4]. There are eight distinct TLFM that can be grouped into three main failure mode types:

- single-bit errors – SBE (TLFM-01)
- multi-bit errors – MBE (TLFM-02, -03, -04, -07, -08)
- addressing errors (TLFM-05, -06)

ID	Description	Comment	Critical FIT	Failure Type
TLFM-01	Corrupt data: Single-bit error (SBE)	Single corrupted bit in one or multiple words	~70%	SBE
TLFM-02	Corrupt data: Double-bit error (DBE)	Two corrupted bits in one or multiple words		MBE
TLFM-03	Corrupt data: Multiple-bit error (MBE)	Multiple corrupted bits or random data vector		MBE
TLFM-04	Corrupt data: Continuous MBE (CMBE)	Repeated MBE for many/every read access		MBE
TLFM-05	Wrong data	Example: Data from a wrong address	~30%	Address
TLFM-06	Lost / old data	Example: No data written, old data at this address		Address
TLFM-07	No data driven during read operation	Termination pulls data to VSS — all-0 received		MBE
TLFM-08	DQ bus disturbance	Leading to MBE on the shared DQ bus		MBE

Table 1: LPDDR top-level failure modes

Note that all failure modes are single-point fault failures which means, for example, that a single fault event causes multiple bits in a read burst to be wrong (MBE failure type).

For LPDDR DRAM, a significant percentage of the safety critical FIT budget is allocated to single-bit errors that can easily be covered by a standard ECC on DRAM or on the host side, but the remaining percentage of the FIT are more difficult to detect because they are multi-bit and addressing errors. We found that for a 64Gb LPDDR5x DRAM, roughly 70% of the FIT is in single-bit errors (TLFM-01) while the other 30% are in multi-bit and addressing failure mode types (TLFM-02 … -08). A recent large scale field study on commodity servers found a distribution of even 50%:50% for single- vs. multi-bit faults in DDR4 components [5].

Because of the significant allocation to MBE and addressing failure types, we found that a standard JEDEC LPDDR DRAM with commonly used host inline ECC schemes (e.g., 64+8 single error correction, double error detection (SEC-DED)) is not able achieve hardware Key Performance Indicators (KPIs) required for functional safety [2][7]. This finding has recently also been confirmed independently by a team of researchers from TU Kaiserslautern, Fraunhofer Institute and Mercedes-Benz using a different analysis methodology — fault tree analysis (FTA) instead of FMEDA [3]. The issue is the mediocre detection capability of traditional ECC schemes for multi-bit and addressing errors. We'll analyze the characteristics of common error detecting and error correcting codes with respect to their MBE detection capability in the next chapters.

Error Correcting and Error Detecting Codes

An error correcting code (ECC) adds redundant information – check bits or symbols – on sender side. The resulting codeword is transmitted over a noise channel where errors occur. The receiver can then correct a certain number of errors based on the redundancy added. Commonly used ECC codes for memory protection are SEC (Single Error Correct) or SEC-DED (Single Error Correct – Double Error Detect) Hamming codes. These codes can correct a single-bit error per codeword. Other alternatives are Bose-Chaudhuri-Hocquenghem (BCH) and Reed-Solomon (RS) codes. BCH codes can correct multiple single-bit errors and RS codes one or multiple symbols (e.g. bytes).

Error detecting codes (EDC) also add redundant bits or symbols on sender side. Again, the codeword is transmitted over a noisy channel that, with a certain probability, creates an error in the codeword. The receiver now can detect errors without correcting them. An example for and EDC is the Cyclic Redundancy Check code (CRC) [6]. However, any Hamming code can also be operated in EDC mode as the decoding procedure consists of 3 steps: 1. detection of an error in the codeword, 2. determination of the location of the erroneous bit, and 3. flipping of the respective bit location. A Hamming code is operated in EDC mode if the decoding procedure is stopped after the 1st step.

Multi-Bit Error Diagnostic Coverage

Diagnostic Coverage (DC) is a KPI used in ISO 26262 to determine the fault detection capability of a safety mechanism. It is specified as the ration of detected dangerous faults to all possible dangerous faults [1].

$$DC = \frac{\text{detected failures}}{\text{all failures}} = 1 - \frac{\text{undetected failures}}{\text{all failures}}$$

Let d be the number of data and e the number of check bits in an error detecting or error correcting code. The code has then 2^d valid code words and 2^{d+e} possible words.

In an EDC, all arbitrary multi-bit errors that fall on a valid codeword cannot be detected. The error and the valid codeword could not be distinguished. The EDC, hence, has 2^d undetectable failure variants and 2^{d+e} overall possible failure variants. The MBE DC can be calculated as

$$DC(MBE)_{EDC} = 1 - \frac{2^d}{2^{d+e}} = 1 - 2^{-e}$$

It is interesting to note that the MBE DC of an error detecting code does not depend on the number of data bits but solely on the number of check bits of the code. This is only valid within the capacity limits of the code that in turn depend on the number of check bits.

A single-bit correcting ECC (SEC or SEC-DED Hamming code) also has 2^{d+e} overall possible MBE and all MBE that fall on the 2^d valid codewords cannot be detected. In addition, all $(d+e)2^d$ possible multi-bit error combinations that appear as a single-bit error cannot be distinguished from real SBE and are, hence, undetectable. The MBE DC of a single-bit correcting ECC can be calculated as

$$DC(MBE)_{ECC} = 1 - \frac{2^d + (d+e)2^d}{2^{d+e}} = 1 - (\mathrm{d} + \mathrm{e} + 1)2^{-e}$$

For the same number of check bits e, $DC(MBE)_{EDC}$ is larger than $DC(MBE)_{ECC}$. Figures 2, 3, and 4 show this trade-off in a graphical way. The 9x9 chess board represents the code space (81 possible words). The green boxes are the valid code words (9 valid codewords). We assume that errors can propagate only vertically and horizontally. As can be seen, the minimal Hamming distance is 3 so that a single-bit correcting code can be implemented in this code space. Minimum 3 steps are required to get from one green field to the next one.

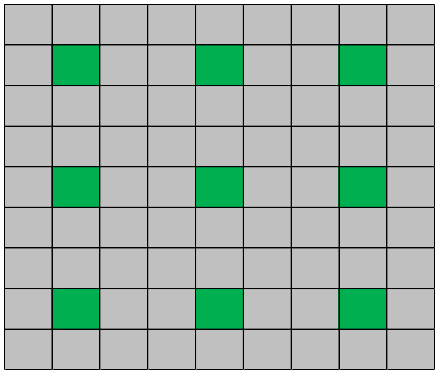

Figure 2: Hypothetical EDC

The MBE DC of the hypothetical EDC in Figure 2 can be calculated as DC(MBE) = 1 – 9/81 = 89%.

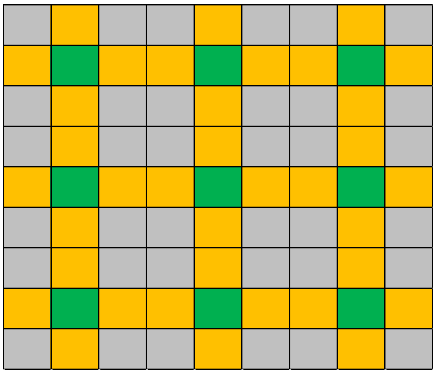

Figure 3: Hypothetical SEC-ECC

The yellow boxes in Figure 3, being one horizontal or vertical step away from a green box, represent single-bit errors. There are 36 possible singe-bit errors. MBE DC of the hypothetical SEC-ECC can be calculated as DC(MBE) = 1 – (9+36)/81 = 44%.

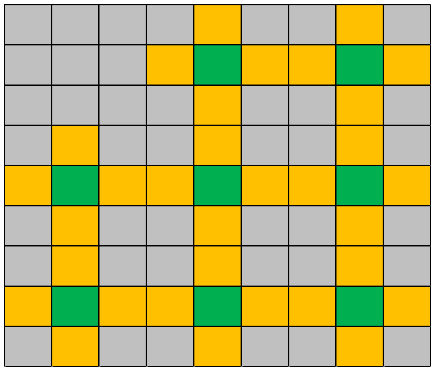

Figure 4: Hypothetical SEC-ECC with an unused codeword

In Figure 4, one of the possible codewords remains unused – one green box less. This method can be used to implement error correcting codes that combine the single-bit correction capability with high multi-bit detection as we will show later. This is commonly called overprovisioned ECC. The MBE DC of this code increases to DC(MBE) = 1 – (8+32)/81 = 51%.

ECC and EDC Fault Simulation

To validate the above analytical calculations, fault simulations of several CRC and Hamming codes in ECC and EDC mode have been conducted in Matlab. For all codes investigated, an all-0 data word yielded an all-0 codeword. Words with combinations of 1 could, hence be used to simulate errors in the all-0 codeword. All possible combinations with *i* bit errors for the binary codeword were generated and ran through the decode operation. Detected and undetected bit errors were counted. Simulations started with *i=1* (SBE) and *i* was increased until the number of possible combinations was too high to be executed in reasonable time spans.

A Hamming code can be described by the number total bits *n*, data bits *k*, and the minimum Hamming distance *h*. A common representation is *Hamming(n, k, h)*.

# bit errors	# error combinations	DC(ECC) SEC-DED	DC(EDC) TED
1	39	100% (corrected)	100%
2	741	100%	100%
3	9 139	41.2%	100%
4	82 251	98.4%	98.4%
5	575 757	38.9%	100%
6	3 262 623	98.4%	98.4%
7	15 380 937	39.1%	100%
8	61 523 748	98.4%	99.4%
9	211 915 132	39.1%	100%

Table 2: Hamming(39,32,4) code fault simulation

Table 2 shows a Hamming code with 32 data bits and 7 check bits. With its minimum Hamming distance of 4, it can correct 100% of all single-bit errors and detect 100% of double-bit errors when operated in ECC mode (SEC-DED ECC). In EDC mode, it can detect 100% of all bit error combinations up to 3 bits (TED EDC). In addition to its single-bit correction and double-bit detection capabilities, the Hamming code can also detect a percentage of MBE. The simulations show that even bit errors are detected with a very high probability, while the DC for odd bit errors is rather low.

Even bit errors can be detected because the additional parity bit used in *h*=4 Hamming codes to implement the DED function has the capability to detect all even bit errors - not just 2. Since valid codewords have the same parity as even bit errors, the overall DC is be bit lower than 100%.

Certain odd bit errors can also be detected in this code since Hamming(39,32,4) is a so-called truncated Hamming code truncated from the full Hamming(64,57,4). Some MBE are decoded to unused (truncated) codewords and can, hence, be identified as uncorrectable MBE. It should be noted that additional logic must be implemented in the ECC decode

hardware to leverage the odd MBE detection capability. A simple textbook implementation might not have this.

The analytical number for $DC(MBE)_{ECC}$ from the equations derived in this paper is 68.75%. To compare with the simulations, we need to average over odd and even MBE. We cannot use the weighted average to compare with the analytical number since the number of bit error combination differs significantly between *i* and *i+1*. Moreover, the coverage values simulated for individual *i*-bit errors can deviate from the ideal average as they depend on the actual code word selection in a truncated Hamming code. The higher *i*, the lower is the deviation from the ideal average. Averaging over *i=8* (even MBE) and *i=9* (odd MBE) gives the expected 68.75% from the analytical calculation.

In EDC mode, all odd bit errors are detected with 100% probability. Since also here valid codewords have the same parity as even bit errors, even MBE falling on valid codewords cannot be detected. This is why the DC for even MBE is a bit less than 100%. The expected analytical $DC(MBE)_{EDC}$ is 99.6%. Averaging over *i=8* (even MBE) and *i=9* (odd MBE) gives 99.7%.

# bit errors	# error combinations	DC(ECC) SEC-DED	DC(EDC) TED
1	72	100% (corrected)	100%
2	2 556	100%	100%
3	59 640	24.0%	100%
4	1 028 790	98.9%	98.9%
5	13 991 544	32.5%	100%
6	156 238 908	99.1%	99.1%
7	1 473 109 704	37.7%	100%
8	11 969 016 345	99.1%	99.1%
9	85 113 005 120	40.8%	100%

Table 3: Hamming(72,64,4) code fault simulation

Table 3 shows a Hamming code with 64 data bits and 8 check bits. Such codes are often used as an inline ECC in memory host controllers to protect against single-bit flips in DRAMs. In addition to its single-bit correction and double-bit detection capabilities, this Hamming code can also detect 71.48% of all arbitrary MBE in ECC mode. It can detect 100% of all bit errors up to *i=3* and detect 99.6% of all higher order MBE in EDC mode. Averaging over *i=8* and *i=9* in the simulation results gives 69.95% in ECC mode and 99.55% in EDC mode.

# bit errors	# error combinations	DC(ECC) SEC-DED	DC(EDC) TED
1	272	100% (corrected)	100%
2	36 856	100%	100%
3	3 317 040	99.592%	100%
4	223 070 940	99.991%	99.991%
5	11 956 602 384	99.037%	100%
6	532 068 806 088		
7			
8			

Table 4: Hamming(272,256,4) code fault simulation

Table 4 shows the simulation results for an overprovisioned Hamming code with 256 data and 16 check bits. The minimal number of check bits required for a Hamming code with *h=4* to protect 256 data bits would be 10. This code uses 6 additional check bits so that it is "overprovisioned". The overprovisioning leads to many unused codewords and can, hence, be used to increase $DC(MBE)_{ECC}$ while maintaining the single-bit correction capability as it has been shown in Figure 4.

The Hamming code has been derived from the IBM-CRC-16 polynomial ($x^{16} + x^{15} + x^2 + 1$) which can protect up to 32751 bits with a minimal Hamming distance *h=4* [6]. It should be noted that any CRC code with a given minimal Hamming distance *h* can be converted into a Hamming code with the same *h*.

The analytical values are $DC(MBE)_{ECC}=99.583\%$ and $DC(MBE)_{EDC}=99.998\%$. The simulated average for *i=4* and *i=5* is 99.514% and 99.996% for ECC and EDC mode respectively.

Extension to BCH and RS codes

With growing memory density and bandwidth requirements, improved multi-bit or symbol correcting codes are likely to be required. HBM3, for example, uses Reed-Solomon (RS) codes for improved robustness [8].

The generic equation for the detection capability (in functional safety terminology Diagnostic Coverage) of uncorrectable errors in any linear block code is given below:

$$DC_{uncorrectable} = 1 - \frac{(2^{mk} - 1) * \sum_{i=0}^{t} \binom{n}{i} (2^m - 1)^i}{2^{mn} - \sum_{i=0}^{t} \binom{n}{i} (2^m - 1)^i}$$

With n being the codeword size, k the data word size, m the symbol size, and t the number of correctable bit or symbol errors. It should be noted that this equation is now truly generic. It can be used for CRC codes (m=1, t=0), Hamming codes (m=1, t=1), BCH codes (m=1, t>1), and RS codes (m>1, t≥1).

Table 5 shows the fault simulation of all possible bit errors for a RS(7,3,3,5) code. The code has *n=7* symbols in a code word of which there are *k=3* data symbols. The symbol size is *m=3* bits. The code has a minimum Hamming distance of 5 so that it can correct up to *t=2* symbol errors. The code consists of overall *m*n=21* bits. All error combinations with 1 or 2 bit errors can be corrected as they cannot span more than 2 symbols. For bit error combinations with 3 to 6 bit errors, some combinations can still be corrected if they fall into maximum 2 symbols. Bit error combinations with more than 6 bit errors always span more than 2 symbols and can, hence, not be corrected anymore. The simulation results in an average DC over all uncorrectable error combinations of 73.6951% which is identical to the result obtained with the generic equation presented above.

# bit errors	# error combinations	# correctable errors	# detectable errors	DC(ECC)
1	21	21	0	100.%
2	210	210	0	100%
3	1 330	385	819	90.5%
4	5 985	315	4 515	80.7%
5	20 349	126	15 456	76.6%
6	54 264	21	40 418	74.5%
7	116 280	0	85 428	73.5%
8	203 490	0	149 394	73.4%
9	293 930	0	216 363	73.6%
10	352 716	0	259 959	73.7%
11	352 716	0	259 959	73.7%
12	293 930	0	216 363	73.6%
13	203 490	0	149 394	73.4%
14	116 280	0	85 428	73.5%
15	54 264	0	40 418	74.5%
16	20 349	0	15 456	76.0%
17	5 985	0	4 515	75.4%
18	1 330	0	819	61.6%
19	210	0	0	0%
20	21	0	0	0%
21	1		0	0%

Table 5: RS(7,3,3,5) code fault simulation

Conclusions

The main top-level failure modes (TLFM) of DRAM components show significant FIT in multi-bit (MBE) and addressing errors. MBE are the 2nd highest FIT contributor after SBE in DRAM. MBE can occur in the memory array, the periphery, and the package – with package and periphery being the biggest contributors. Protection mechanisms on DRAM and/or host side, hence, require a very high MBE detection capability DC(MBE). Simple ECC schemes are often not sufficient to achieve the required DC.

Analytical equations to calculate DC(MBE) for error detecting (CRC, Hamming code in detection mode) and single-bit error correcting codes (Hamming code in correction mode) have been derived. Table 5 provides an overview of DC(MBE) for commonly used coding schemes in ECC as well as EDC mode.

Coding Algorithm	DC(MBE) in ECC mode	DC(MBE) in EDC mode
Hamming(39,32,4)	68.75%	99.22%
Hamming(72,64,4)	71.48%	99.61%
Hamming(266,256,4)	73.93%	99.90%
Hamming(267,256,4)	86.91%	99.95%
Hamming(272,256,4)	99.58%	99.99%
CRC8	n/a	99.61%
CRC16	n/a	99.99%
CRC32	n/a	~100%
Hamming(136,128,3)	46.48%	99.61%
Hamming(137,128,4)	73.05%	99.80%
BCH(286,256,7)	99.64%	~100%
RS(18,16,16,3)	99.97%	~100%

Table 5: DC(MBE) of commonly used ECC/EDC schemes

We extended the DC calculation to multi-bit (BCH) and symbol (RS) correcting codes and derived a truly generic equation to calculate the detection capability for uncorrectable errors of any linear block code, be it CRC, Hamming, BCH, or RS.

In addition, fault simulations have been conducted to analyze the behavior in more detail and to validate the analytical equations. Monte Carlo simulations can be added as part of future work for higher order bit errors where exhaustive simulations of all bit error combinations are not feasible anymore.

While for functional safety, error detection (aka diagnostic coverage) suffices, system availability (aka error correction or fault prevention) is still very important in automotive embedded systems. There are two approaches to effectively combine (single-bit) error correction and high MBE detection for DRAM subsystems:

1. **EDC after ECC**
   Executing a host end-to-end EDC scheme (Hamming code in detection mode or CRC) combined with link and array ECC mechanisms on interface and DRAM side.
2. **Overprovisioned ECC**
   Applying a host ECC with more check bits than necessary to achieve the pure SEC-DED functionality.

More detailed investigations are required to assess pros and cons of both options in different DRAM subsystems.

ACKNOWLEDGMENTS

The author would like to thank his colleagues **Aaron Boehm**, **Mellissa Uribe**, and **Scott Schaefer** who were instrumental in completing the above analysis. The author is grateful to **Alexander Griessing** of Exida for his continuous training and guidance in all aspects of functional safety and ISO 26262. Finally, the author wants to thank **Prof. Dr.-Ing. Matthias Jung** for the excellent technical discussions around DRAM RAS capabilities and his encouragement to write this paper.

REFERENCES

[1] International Organization for Standardization (ISO). 2018. *Road vehicles – Functional safety, ISO 26262*
[2] S. Buch. 2022. *DRAM in Safety Critical Automotive Systems – A Micron White Paper.* Micron Technology, Inc. https://media-www.micron.com/-/media/client/global/documents/products/white-paper/lpddr5_safety_critical_auto_systems_white_paper.pdf
[3] L. Steiner et al. 2021. *An LPDDR4 Safety Model for Automotive Applications.* MEMSYS 2021
[4] S. Buch. 2020. *Questions to Ask Your Memory Supplier … About Functional Safety for DRAM.* Electronica. https://www.youtube.com/watch?v=mzcbtXdWDcg
[5] M. V. Beigi et al. 2023. *A Systematic Study of DDR4 DRAM Faults in the Field.* 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)
[6] P. Koopman. 2018. *Best CRC Polynomials.* Carnegie Mellon University. http://users.ece.cmu.edu/~koopman/crc/
[7] A. Boehm. 2021. *DRAM – More Important Than You Think for Achieving Automotive Functional Safety.* Design News. https://www.designnews.com/electronics/dram-more-important-you-think-achieving-automotive-functional-safety
[8] Gurumurthi et al. 2021. *HBM3 RAS: Enhancing Resilience at Scale.* IEEE Computer Architecture Letters 2021