Improving Memory Reliability by Bounding DRAM Faults

DDR5 improved reliability features

KJERSTEN CRISS
Intel

KULJIT BAINS
Intel

RAJAT AGARWAL
Intel

TANJ BENNETT
Microsoft

TERRY GRUNZKE
Microsoft

JANGRYUL KEITH KIM
SK Hynix

HOEJU CHUNG
SK Hynix

MUNSEON JUNG
SK Hynix

ABSTRACT

DRAM reliability is an increasingly difficult to address as error correction code (ECC) overhead increases and process nodes shrink, driving up DRAM cell errors. As single bit errors increase, failures of larger hardware structures within the memory are also an ever-present threat to data integrity.This paper presents a set of methods aimed at reducing the required ECC overhead for correcting many DRAM failures types or fault types. This enables more reliable ECC protection of DIMMs that have traditionally been lower reliability, for example x8 DIMMs that have only one device allocated to ECC check-bits. Reducing the ECC overhead required to correct failures is done by limiting or bounding the area of a cache line that is impacted by those failure types. Individually these methods are all reliability features that a DDR5 memory vendor may choose to implement in their design; collectively this paper refers to these features as DDR5 – Bounded Fault or DDR5-BF.This paper will give an overview of the features needed to bound failures. Specifically, failure pattern requirements for common fault types and on-die ECC modifications to contain miscorrections in the event of multi-bit errors.

CCS CONCEPTS

• **Computer systems organization**; • **Dependable and fault-tolerant systems and networks**; • **Reliability**;

KEYWORDS

DDR5-BF, DRAM, DDR5, On-Die ECC, DRAM Fault, Fault Bounding, DRAM Failure

ACM Reference Format:
KJERSTEN CRISS, KULJIT BAINS, RAJAT AGARWAL, TANJ BENNETT, TERRY GRUNZKE, JANGRYUL KEITH KIM, HOEJU CHUNG, and MUNSEON JUNG. 2020. Improving Memory Reliability by Bounding DRAM Faults: DDR5 improved reliability features. In *The International Symposium on Memory Systems (MEMSYS 2020), September 28–October 01, 2020, Washington, DC, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3422575.3422803

MEMSYS 2020, September 28–October 01, 2020, Washington, DC, USA

ACM ISBN 978-1-4503-8899-3/20/09…$15.00
https://doi.org/10.1145/3422575.3422803

1 INTRODUCTION

System ECC protected DIMMs in servers often use full device error correction codes to recover from errors and memory failures. Server ECC is often uses symbolic codes. In these types of ECC schemes a DRAM chip failure causing data corruption in a cache line, requires at least twice as many ECC check-bits to recover the data [1]. For example, an error correction code that can recover from a single device or chip failure requires two ECC chips worth of check-bits. In DDR5, the 10x4 DIMM which has 8 data chips and two ECC chips is used in servers for full device correction ECC schemes. The ECC capacity overhead of the DDR5 10x4 is 25%, 2x the ECC capacity overhead of a DDR4 18x4 device.

There has been a trend in the cloud computing industry to move from easily protected DDR5 10x4 DIMMs to wider devices. There are many reasons for Cloud Service Providers to favor wider devices: cost, preferred capacity, performance, lower power, etc., but these benefits come at the cost of full device ECC protection. Unlike 10x4 DIMMs, which have two devices to store ECC check-bits, DDR5 5x8 DIMMs have one device to store ECC check-bits. A failure in a x8 device could corrupt data from any or all of the 8 data lines or DQs. A 5x8 DIMM has one ECC device, which does not provide enough ECC check-bits to correct an error that spreads over an entire x8 device, two x8 ECC device would be required to correct such an error. If an error from a failure was limited to a subset of the DQs from a x8 device, say 4 or 2 DQs, then the one ECC chip provides enough check-bits to recover the corrupted data. Systems with 5x8 DIMMs can use ECC schemes that correct half device errors instead of full device ECC schemes. A half device ECC scheme may correct any error that occurs in the data from the left 4 DQs of a device or any error that occurs in the right 4 DQs of a device, but would not correct an error that spreads over all the DQs from a device. Correction of half the device can be considered in DIMMs where only one ECC chip is used. In DDR5 this includes 9x4 DIMMs, which maintains the 12.5% system ECC overhead of DDR4 (but not the full device correction), or 5x8 DIMMs.

This paper discusses a set of features developed to limit some common DRAM failures, so that they are correctable with half device correction codes. The set of features encompasses DDR5 Bounded Fault or DDR5-BF design. The term DDR5-BF is being used

here to distinguish between DDR5 with these features added into the design and DDR5 without these features. DDR5-BF is a DDR5 DIMM with this collection of features, not a separate technology, memory product, or industry term.

2 INTERNAL FAULT BOUNDING

The aim of fault bounding is to limit the number of failure patterns seen by the memory controller such that correction of major faults can be reliably performed in DIMMs with one ECC device (e.g. DDR5 5x8 or DDR5 9x4). A failure pattern is the pattern of bits, as seen by the memory controller, that are impacted by a failure during one read access (e.g. a DQ or pin failure pattern in DDR5 is 1 DQ x 16 bursts). The following are components to DDR5-BF design:

- Agree on failure pattern guidelines for common failure types with memory vendors.
- Align failure patterns of components (e.g. failure pattern for internal bounded fault and pin failure are the same in 9x4).
- Controlling on-die ECC miscorrections

In a non-bounded fault DDR5 device, a single failure type can have numerous failure patterns among DRAM vendors and process generations. The proliferation of failure patterns or the size of failure patterns limits the effectiveness of any system ECC that cannot correct all the data from a device. DDR5-BF features limit failure patterns for common internal DRAM failures. The limited failure patterns can then be correct with half device correction codes. Half device correction may be performed on 9x4 DIMMs, which has a 12.5% system ECC overhead (compared to DDR5 10x4 DIMMs 25%), and 5x8 DIMMs. Half device correction partitions the data from each device in half and then the ECC can correct errors limited to a single partition. For example, the data from the first four DQs and the last four DQs of a x8 device would be an example of a half device partitioning. If an error was contained on the first four DQs it would be correctable, but if an error had bit flips on the first DQ and the last DQ it would not be correctable.

Figure 1 and Figure 2 show possible styles of fault boundaries for x8 and x4 devices, respectively. The x8 devices are separated into data on the first four DQs (nibble A) and last four DQs (nibble B). The fault boundary can be defined as any two DQs on a nibble. Each nibble is partitioned into two fault boundaries. Even though a 5x8 DIMM could use up to half device correction, the fault boundaries here are limited to a quarter of the device or 32 bits. The failure boundaries are limited to 32 bits because miscorrection of multi-bit errors by the on-die ECC (a single error correction scheme in each DRAM chip) are easier to contain for 32-bit failure boundaries than for 64-bit failure boundaries. The fault boundary for 9x4 DIMMs is one DQ. In 10x4 DIMMs the same faults could be bounded to either one or two DQs. A failure pattern for bounded fault will be some sub-set of the bits that form a fault boundary.

2.1 Faults in 9x4 and 5x8 DDR5-BF DIMMs

DRAM architecture does not allow all faults to be bounded. For error correction in DDR5-BF the main distinguishers between faults is whether the fault impacts all data from the device during a read access (unbounded fault) or only a portion of the data from the device during a read access (bounded fault). For that reason, the

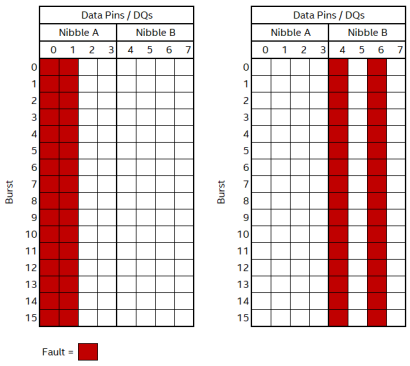

Figure 1: Two possible styles of fault boundary for DDR5-BF x8 devices. Major sub-CL faults will impact, at most, data on two DQs in either nibble A or nibble B. Major sub-CL faults should not impact both nibble A and nibble B.

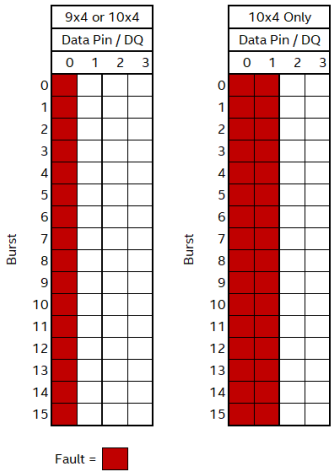

Figure 2: Fault boundary for DDR5-BF x4 devices. In all 9x4 devices major sub-CL faults will impact data on one DQ. Major sub-CL faults in 10x4 devices can impact data on either one or two DQs.

fault types are divided into sub cache line faults (sub-CL fault) and full cache line faults (full-CL fault) and a selection of common faults in DRAM are presented.

2.1.1 Sub-CL Faults. These are the faults that will only impact a portion of the data from a device during one read access. They include sub-row failures, in which only a portion of the row fails, column failures, or data pin failures. Components that make up the row may fail without causing the entire row to fail. Many cache lines in the same page or row will be impacted by these failures, but only a portion of the data from the device will be corrupted in each cache line. Column faults impact multiple cache lines that share a column address. They may impact a single bit in a cache line or several bits in a cache line. Data pin failures may impact

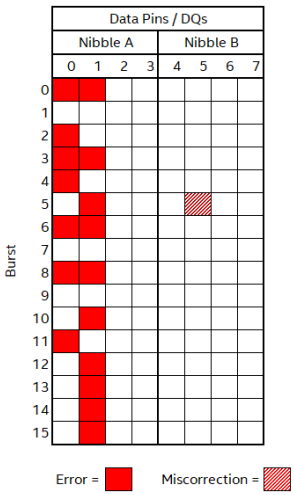

Figure 3: Example of an unconstrained on-die ECC miscorrection. A sub-CL faults has caused a multi-bit error in the data from DQ0 and DQ1. The on-die ECC erroneously attempts to correct a single bit nibble B. The miscorrection causes the original error to be uncorrectable.

any read from the device. Failure patterns in DDR5-BF devices will align with those of internal faults.

2.1.2 Full-CL Faults. A device, bank, or entire row may fail. These are faults in components that affect every bit of data in a cache line from the device. To recover data from the faults with ECC, full device correction is needed. Some of these faults may benefit from screening processes or more robust component designs.

3 BOUNDING ON-DIE ECC MISCORRECTIONS

Shrinking DRAM cells have increased the single bit error rate and number of weak bits, bits that do not maintain charge reliably between refreshes [2]. DRAM vendors have introduced on-die ECC to deal with increase in single bit errors [2]. On-Die ECC is an error correction scheme within a DRAM chip. On-die ECC corrects errors that occur during storage in the DRAM.

The on-die ECC used in DDR5 is a single error correction (SEC) code [3]. When the on-die ECC decoder is presented with a multi-bit error, it may mistake the error for a single bit error and attempt to make a correction. This is a miscorrection and causes an additional bit flip. Figure 3 shows this in the context of DDR5-BF; a bounded fault has caused an error on DQ0 and DQ1, but the on-die ECC has miscorrected a bit on DQ5, creating an unbounded error pattern. An additional unconstrained error from an on-die ECC miscorrection with a multi-bit error (e.g. an error due to a bounded fault) can cause a potentially uncorrectable error pattern in systems using half device correction codes.

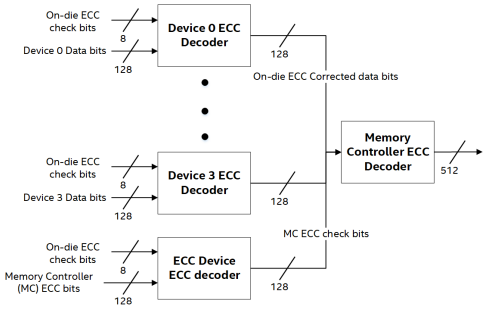

Figure 4: Arrangement of on-die ECC decoders and memory controller ECC decoder for a DDR5 5x8 DIMM. Each device independently performs a single bit error correction on 128 bits of data before sending it to the memory controller. In the memory controller, the system ECC corrects the aggregated data.

3.1 On-Die ECC Brief

The DDR5 on-die ECC performs correction on chunks of 128 bits of data with 8 check-bits. The 128 data bits are joined with 8 check-bits to form a 136-bit ECC codeword. The memory controller only receives the corrected 128 bits of data and has no visibility into errors the on-die ECC detects or corrects. A DDR5 device will not signal the controller if it detects or corrects an error. Figure 4 shows the arrangement of the memory controller ECC decoder and the on-die ECC decoders for a DDR5 5x8 DIMM. An ECC decoder checks and corrects the data. The on-die ECC will check data from each device individually before sending the data to the memory controller. The memory controller performs ECC correction and detection on the aggregate data. Likewise, during encoding the on-die ECC generates check-bits for the data going to each device independently of the data going to the other device. The memory controller will generate check bits from the entire cache line. In the case of DDR5 x8, the 128-bit on-die ECC region coincides with the amount of data the memory controller receives from a device in one read or write access. However, in DDR5 x4 a single read or write access is only 64 bits or half an on-die ECC region. As Figure 5 shows this requires a read-modify-write to be performed to update the check-bits during write [3].

Because the DDR5 on-die ECC uses a SEC code, any error with 2 or more bit flips in the 136 bit codeword may cause the on-die ECC to miscorrect. In the case of DDR5-BF it is problematic to have bit flips in multiple bounded failure regions. The best case for any bit flips from on-die ECC miscorrection is into the bounded failure region that already contains the original multi-bit error, because this will be correctable by a memory controller ECC designed for DDR5-BF DIMMs.

3.2 SEC Boundary Codes

One solution to prevent on-die ECC miscorrections outside of failure boundaries defined in DDR5-BF is to use SEC codes that will miscorrect into whichever bounded region contains the multi-bit error causing the miscorrection (or into the check-bits, which are not sent to the memory controller).

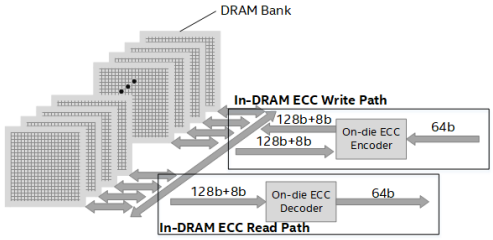

Figure 5: Read and write paths for a x4 DDR5 device. During a read access, 128 data bits and 8 check-bits are read from a DRAM bank. The data is corrected and sent out. During writes a RMW is performed. 128 data bits and 8 check-bits are read from a DRAM bank and the data is corrected, the new data is merged and the new ECC bits calculated. The corrected data may or may not be written back to the DRAM array.

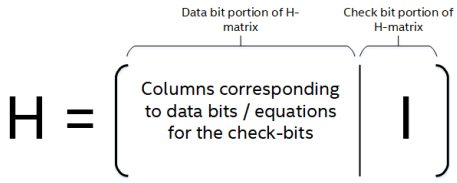

Figure 6: Diagram of a parity check matrix or H-matrix in systematic form. The right-hand side of the matrix is the identity matrix, each column corresponds to an ECC check-bits. The left-hand side of the matrix consists of a set of unique columns, one for each data bit, and determines the equations for the check-bits.

Codes that miscorrect into the failure boundary containing the multi-bit error or the check-bits can be created by using the mathematical structures of the columns of the H-matrix. The H-matrix or parity check matrix can be thought of as one way to define an ECC code. It describes how the check-bits are calculated from the data bits. Each column corresponds to a data bit or one of the ECC check-bits and each row gives the equation for one of the check-bits. Figure 6 gives pictorial representation of an H-matrix.

The H-matrix is primarily used for decoding during correction. The multiplication of the H-matrix with the received codeword produces a vector called the error syndrome, which identifies which bit needs to be corrected. If the error syndrome matches a column in the H-matrix, then the corresponding bit is corrected. If the error syndrome does not appear as a column in the H-matrix, then the decoder recognizes the error as uncorrectable. The received codeword is just the data read from the memory with the stored check-bit appended to it.

All possible columns of the H-matrix and the all zero column under bit-wise XOR (bits are XORed together to calculate the check-bits) form a mathematical group. A group is a set of elements, in this case possible columns of the H-matrix and the all zero column, along with an operation, bit-wise XOR, that meet a few basic requirements. By examining the possible interactions between members

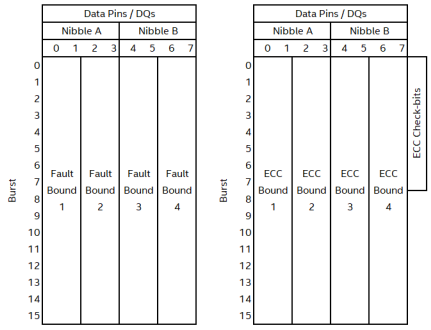

Figure 7: Fault boundaries compared to on-die ECC boundaries in a x8 device. An ECC boundary overlaps a fault boundary, preventing on-die ECC miscorrections outside of a boundary in the case of a fault.

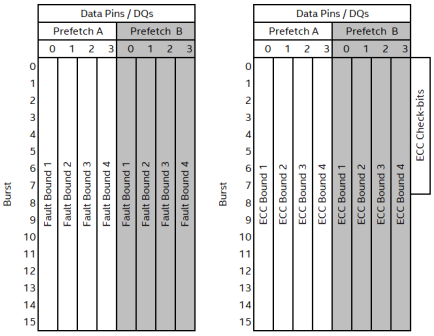

Figure 8: Fault boundaries compared to on-die ECC boundaries in a x4 device. An ECC boundary overlaps a fault boundary, preventing on-die ECC miscorrections outside of a boundary in the case of a fault. Notice that each boundary is divided in two with one half in prefetch A and one half in prefetch B.

of this group, that is the error syndromes that would result if bits corresponding to those columns were flipped, one can observe the structure of the group. From examination of the group structure, one can obtain the subsets of the group that will only miscorrect into a boundary (or the on-die ECC check-bits) if there is a multi-bit error confined to that boundary. Figure 7 and Figure 8 show a comparison of the arrangement of on-die ECC code boundaries and the fault boundaries of a DDR5-BF device for x8 and x4 respectively.

There exist multiple constructions of such codes and multiple styles of fault boundary, so Figure 7 and Figure 8 are not universal. Regardless of how the specific on-die ECC scheme is constructed, errors due to the sub-CL fault types addressed by a vendors DDR5-BF design should not violate the DDR5-BF failure boundaries before or after on-die ECC correction.

3.3 SEC Boundary Codes

Unlike errors confined to a failure boundary, multi-bit errors in the on-die ECC check-bits can miscorrect anywhere in the on-die ECC codeword. Failures impacting the on-die ECC check-bits will cause unconstrained miscorrected regardless of the SEC code design, because every column in the H-matrix must be a linear combination of the columns corresponding to the check-bits. The fact that any column in the H-matrix can be produced by XORing the columns of the check-bits means any error syndrome can be produced by an error in the check-bits. If only the check-bits are impacted by a failure, this will result in at most a single bit miscorrection per an on-die ECC correction cycle. However, if a piece of hardware has failed that drives both the ECC check-bits and a portion of the data, then the data may contain an error confined to a boundary and the on-die ECC may miscorrect a bit outside of that boundary.

The most effective strategy to prevent failures involving the on-die ECC check-bits would be to physically isolate the ECC check-bits so they do not share hardware that may cause sub-CL faults. However, this comes with a die area penalty and design complexity. A DRAM designer may also choose to make such hardware elements more robust or, if the chance of failure for these hardware elements is already very low, leave the hardware as is.

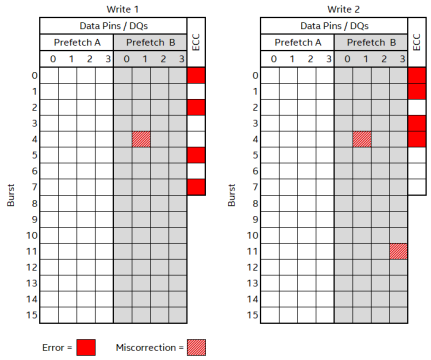

Figure 9: Accumulation of miscorrected errors in after on-die ECC check-bit failure. The on-die ECC may write back miscorrections in prefetch B, when the memory controller writes to prefetch A, unless write back of on-die ECC corrections during writes is disables. Accumulation of miscorrection during writes only occurs in x4, but similar accumulation can occur in x4 or x8 devices during on-die ECS unless scrubbing is disabled.

3.4 Write and Scrub Concerns

Failures impacting only the on-die ECC check-bits may also cause problems during writes and on-die ECC scrubbing. Particularly in the case of x4 devices, where only half the data in an on-die ECC region is written/read per cache line. For reads this requires the device to read all 136 bits in the on-die ECC region to perform correction. For writes a read-modify-write (RMW) needs to be done to maintain the integrity of the data and the on-die ECC check-bits. The RMW must go through the following steps:

- All 128 data bits and 8 check-bits are read from the array and sent to the ECC logic.
- ECC calculations are performed (may or may not include correction).
- New data is merged with existing data.
- ECC logic updates the check-bits.
- New codeword is written to the array.

Problems may arise from the RMW if the decoder is not judicious in the correction of bits. Suppose the memory controller is performing a write operation to the cache line shown in Figure 9. It is overwriting the data in prefetch A. Prefetch B is the portion of data not being overwritten. The ECC check-bits have failed and are reading out random values each time they are accessed. During the initial write to the cache line, the on-die ECC decoder believes there is an error in DQ1 burst 4. If the on-die ECC writes back this miscorrection to prefetch B during the RMW cycle, then it has created a single bit error in the data. If prefetch A is written to a second time before prefetch B is accessed, then the error pattern in the ECC check-bits may have changed and the decoder may miscorrect again. The second miscorrection could produce a random double bit error in prefetch B. In such a way single bit errors may accumulate in prefetch B during multiple reads to A.

Errors will not accumulate if on-die ECC does not write back corrected data during writes (the DDR5 specification already prescribes that corrections are not written back during read accesses). Some bookkeeping needs to be done to maintain valid check-bits; In the case that there are single bit errors, the on-die ECC should be able to correct them. To maintain usable check-bits the on-die ECC must not use the uncorrected data to update the check-bits during writes.

During on-die ECC error check and scrub (ECS) cycles the same type of error accumulation may occur. In this case, the on-die ECC may miscorrect once when scrubbing and then again during a read or, if the data has gone through multiple scrub cycles, many times during scrubbing and possibly again during read. Either way, there is the potential for unbounded multi-bit errors in both x4 and x8 devices. A simple solution is to turn off the scrubbing functionality of the on-die ECS and use it only as a single bit error counter. The system ECC will then need to take over scrubbing operations to ensure single bit errors are corrected. It can do this by going through the array and reading, correcting any multi-bit errors, and writing back each cache line. The system scrub will also track any multi-bit errors in the memory.

There may be systems in which write back of corrections and scrubbing by on-die ECC is preferred. Both write back correction and on-die ECC scrubbing could be controlled using mode register or MR bits.

4 CONCLUSIONS

The features presented have all been proposed to increase the reliability of DDR5 DIMMs that cannot support full device data correction such as 9x4 and 5x8. By consolidating failure patterns seen by the memory controller ECC, the memory controller ECC can target the most likely patterns. This allows more sub-CL faults to be corrected. In order to consolidate these failure patterns, there must be

agreement with memory vendors on how failures impact the data as seen by the memory controller. There also must be consideration in the design and use of on-die ECC to ensure that miscorrection of multi-bit errors do not breach the prescribed boundaries.

ACKNOWLEDGMENTS

We thank multiple participants from SK Hynix, Samsung, and Micron who helped come up with the solutions that are implemented in DRAMs. We are also thankful to our joint development partners at Microsoft, Google, Amazon, and HPE that reviewed our work.

REFERENCES

[1] Martyn Riley and Iain Richardson. 1998. Reed-Solomon Codes. Retrieved September 15 th, 2020 from https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html
[2] Uksong Kank *et al.* 2014. Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling. The Memory Forum. https://www.cs.utah.edu/thememoryforum/kang.pdf
[3] JEDEC DDR5 Specification Rev 1.0 (79-5). https://www.jedec.org