An LPDDR4 Safety Model for Automotive Applications

Lukas Steiner
Microelectronic Systems Design Research Group, TU Kaiserslautern
Kaiserslautern, Germany
lsteiner@eit.uni-kl.de

Kira Kraft
Microelectronic Systems Design Research Group, TU Kaiserslautern
Kaiserslautern, Germany
kraft@eit.uni-kl.de

Denis Uecker
Fraunhofer Institute for Experimental Software Engineering (IESE)
Kaiserslautern, Germany
denis.uecker@iese.fraunhofer.de

Matthias Jung
Fraunhofer Institute for Experimental Software Engineering (IESE)
Kaiserslautern, Germany
matthias.jung@iese.fraunhofer.de

Michael Huonker
Design High-Computing Platforms, Mercedes-Benz
Sindelfingen, Germany
michael.huonker@daimler.com

Norbert Wehn
Microelectronic Systems Design Research Group, TU Kaiserslautern
Kaiserslautern, Germany
wehn@eit.uni-kl.de

ABSTRACT

The increasing demand for DRAM in modern vehicles creates new challenges for automobile manufacturers. To allow DRAM subsystems to be used in safety-critical tasks like autonomous driving, a special Automotive Safety Integrity Level (ASIL) grading according to the ISO 26262 is required. While the classification process is already well-established for processors and on-chip memories with dedicated automotive hardware introduced to the market, no similar research has been conducted for DRAM yet. As a consequence, the process proves to be difficult for car manufacturers at this point. Therefore, a methodology that captures all the DRAM subsystem complexity in a comprehensive but yet understandable way is required. In this paper we use Component Fault Trees to create a clearly-structured safety model of an exemplary LPDDR4 memory subsystem. Based on the proposed model, we also evaluate the ASIL that it can reach. For the automotive industry, our work can serve as a foundation for future classification processes, therefore taking one more step towards full autonomy.

CCS CONCEPTS

• **Hardware** → **Safety critical systems**; **Dynamic memory**; • **Computing methodologies** → **Modeling methodologies**.

KEYWORDS

DRAM, LPDDR4, Safety, ECC, ASIL, Automotive, Fault Tree Analysis, Component Fault Trees

ACM Reference Format:
Lukas Steiner, Kira Kraft, Denis Uecker, Matthias Jung, Michael Huonker, and Norbert Wehn. 2021. An LPDDR4 Safety Model for Automotive Applications. In *The International Symposium on Memory Systems (MEMSYS 2021), September 27–30, 2021, Washington DC, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488423.3488441

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MEMSYS 2021, September 27–30, 2021, Washington DC, DC, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8570-1/21/09…$15.00
https://doi.org/10.1145/3488423.3488441

1 INTRODUCTION

Currently, the automotive industry is in the middle of a great revolution. The shift to electric cars, the rapid increase of *Advanced Driver-Assistance Systems* (ADAS) and the advent of *Autonomous Driving* (AD) presents great opportunities for future transportation infrastructure, but also creates lots of challenges for manufacturers. Vehicles have to be designed in compliance with safety metrics like the *Automotive Safety Integrity Level* (ASIL) to guarantee a flawless operation at all times and in all circumstances, since the responsibility is fully transferred from humans to machines. In order to classify the overall ASIL of a system, the ASIL of all subsystems have to be classified first. For microcontrollers and *System-on-Chips* (SoCs) consisting of processing cores and on-chip memories (SRAM, Flash) this procedure is state-of-the-art. A lot of research has been conducted in this area [6, 9, 14, 17, 21, 22] and dedicated automotive products like the Infineon AURIX microcontroller family [2], the ARM Cortex-A76AE [30] or the Synopsys DesignWare ARC EM22FS Safety Processor [16] already enable operation at the highest level ASIL D. One type of subsystems with particular importance in modern vehicles however are *Dynamic Random-Access Memories* (DRAMs), since the need for memory with high capacity and bandwidth as well as low latency has grown dramatically over the past few years due to the introduction of ADAS and AD [15]. Unfortunately, these devices originate from the consumer and server domain, so they are not specialized to reach a certain ASIL. Their strict optimization for cost per bit leads to a complex internal hardware architecture and, moreover, they are highly sensitive to errors: in addition to classical hard and soft errors, leakage effects can lead to retention errors, because information is stored in the form of extremely small amounts of charge. Technology scaling exacerbates this problem more and more. Therefore, memory vendors try to compensate the high error sensitivity by protecting devices with *Error Correction Codes* (ECC) [18, 19, 27]. In older DRAM standards the corresponding ECC engines were placed externally on the memory controller side and parity bits had to be stored in an extra chip or in line with data by reducing the effective storage capacity. LPDDR4, on the other hand, is the first DRAM standard that provides vendors the opportunity to integrate ECC directly on the chip [24], reducing both latency and power consumption. Driven by these facts and the overall low power consumption compared to other available standards [10, 15], LPDDR4 is the first choice for most automobile

manufacturers and suppliers in their search for the most suitable solution for their current products (see, e.g., Tesla Full Self-Driving Computer [35], NVIDIA Drive Xavier [25]). However, while ECC decreases failure rates and can enable a higher ASIL grading, at the same time it complicates the classification process even more because additional hardware is inserted. The automotive standard ISO 26262 [13] defines a procedure to rate the ASIL of silicon hardware and also gives some examples for diagnosis and coverage with respect to memories. Unfortunately, this procedure relies on spreadsheets and cannot capture the full complexity of today's DRAM device and subsystem architectures. Therefore, in this paper we present for the first time a new approach for the safety modeling of ECC-protected DRAM subsystems using a *Fault Tree Analysis* (FTA) methodology, which is based on *Component Fault Trees* (CFT). These fault trees allow a clear model structuring and fully capture the device complexity at once. Based on the model, we evaluate the ASIL that can be reached with state-of-the-art LPDDR4 technology. This will help automobile manufacturers to design, classify and certify their current products. Finally, we give an outline on coverage mechanisms that DRAM vendors should include in the emerging LPDDR5 standard to make their products more suitable for automotive applications by achieving a higher ASIL.

In summary, we make the following contributions:

- We present, to the best of our knowledge, for the first time an DRAM safety modeling methodology based on CFTs, which is compliant with the ISO 26262.
- In a case study, we show how this methodology can be applied to an exemplary LPDDR4 DRAM subsystem. Additionally, we estimate the ASIL rating for this system.

The paper is structured as follows: Section 2 discusses related work and introduces the resources that our assumptions in subsequent sections are based on. Section 3 gives a short overview of the LPDDR4 DRAM standard. The ECC architecture is presented and explained in Section 4. Our CFT apprach is presented in Section 5. A case study with an exemplary LPDDR4 subsystem is shown in Section 6. The paper is then concluded with the Sections 7 and 8.

2 RELATED WORK

Lately, a lot of research has been carried out in the field of hardware development and classification with regard to ISO 26262. The authors of [14] list individual steps of the automotive hardware development procedure. In [21] an ASIL-oriented hardware design framework based on FTA is presented, while in [9] a methodology for automated exploring of generic automotive architecture solutions is shown. Similarly, the paper [6] demonstrates the ISO 26262 hardware assessment process based on an exemplary safety microprocessor, and the authors of [17] introduce a real microcontroller that already satisfies ASIL D. Unfortunately, these works are either theoretical or only focus on smaller microprocessor solutions with on-chip memory.

A larger SoC that reaches ASIL D and is specialized to achieve high CNN performance was presented by the authors of [22] just recently. Their platform uses dual-channel LPDDR4 memory to satisfy the requirements for high bandwidth and capacity. The requirements for ASIL D are achieved by the use of DRAM ECC and dual-core lock step. However, neither the ECC architecture nor any failure rates are included.

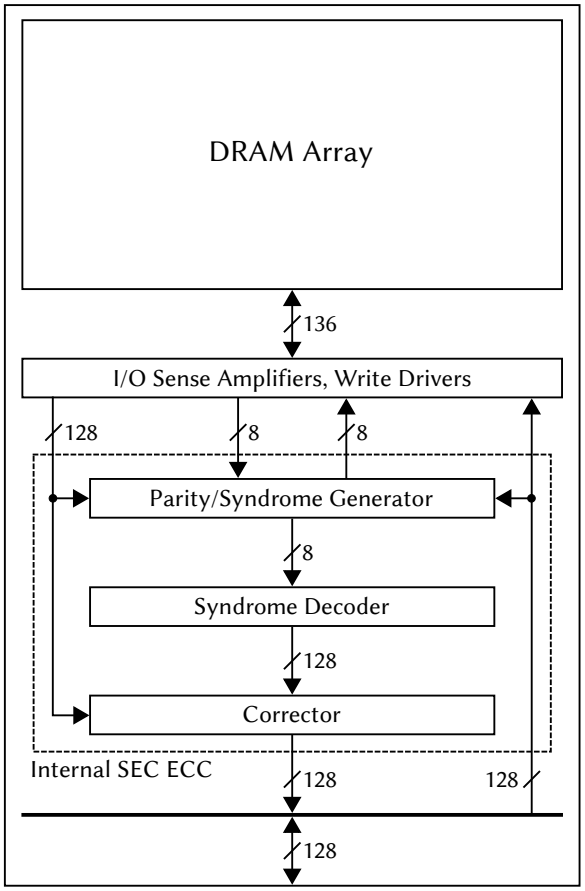

Figure 1: LPDDR4 Bank Architecture with Internal SEC ECC Engine [18, 19, 27]

The variety of problems that the use of consumer DRAM devices in modern automotive applications brings along has already been outlined in [15] in great detail. One section also targets the safety issues with regard to ISO 26262.

Field studies on DRAM failures and error sources have been conducted extensively over the last years and can be found numerously in literature [3, 11, 26, 29, 31–33]. They cover different application fields, DRAM standards, DRAM configurations, error-correction schemes, DRAM failure modes, etc., creating a solid base of error probabilities for our experiments in Section 6.

In addition to an experimental root cause analysis, the authors of [20] also perform a fault tree analysis for DDR4 DIMMs. Their analysis focuses on mechanical failures and electrical failures, however, no microarchitectural insights are considered. Besides, neither external nor internal ECC engines are modelled.

In order to find out the internal ECC architecture that is used in current LPDDR4 devices, we refer to various independent resources. Both Samsung and SK Hynix, two of the three large DRAM manufacturers, propose LPDDR4 devices with on-chip ECC using a (136, 128) shortened Hamming code [18, 19, 27]. In contrast to the

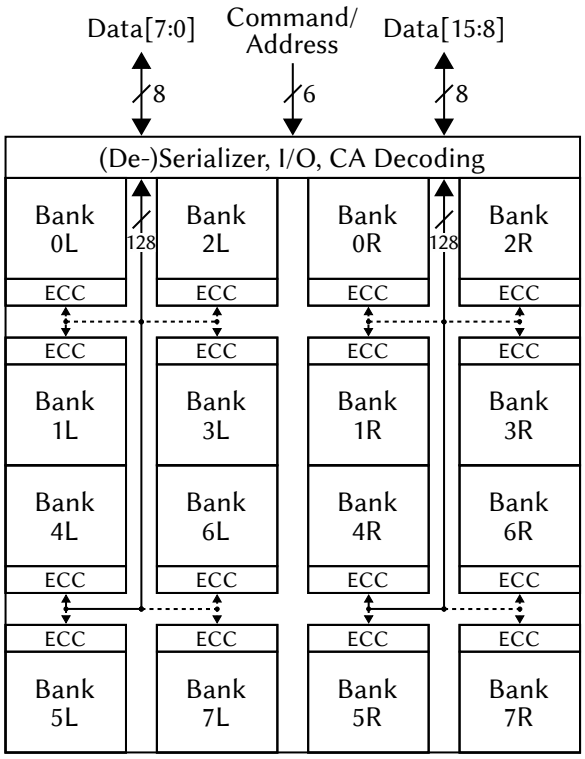

Figure 2: LPDDR4 Channel Architecture [18, 19, 27]

purpose in our research, they exploit the error correction capability to reduce power by reducing the supply voltage or increasing the refresh interval. The authors of [5] have investigated different on-chip ECC solutions with regard to area and latency overhead, which led them to the same preferred architecture. By reverse-engineering a large set of different LPDDR4 devices, the paper [28] also reveals this architecture. For that reason, a (136, 128) shortened Hamming code is assumed for the rest of this work.

3 LPDDR4 BACKGROUND

Low Power Double Data Rate 4 (LPDDR4) [24] is the fourth generation of low power *Dynamic Random-Access Memory* (DRAM) specified by the *JEDEC Solid State Technology Association* (short JEDEC) in 2014. Compared to its predecessor LPDDR3 [23], the maximum data rate is doubled from 2133 MT/s to 4266 MT/s. Instead of a single 32-bits-wide channel, LPDDR4 comes with either a single or two 16-bits-wide channels per die. Besides the standard dies, there are special byte-mode dies with only 8 bits per channel. Two of these dies can be combined into a 16-bit standard configuration with twice the density. Each memory channel consists of 8 banks. In order to match the low internal access frequency, which is caused by the architecture optimized for cost per bit, and the increased interface data rate, LPDDR4 uses a 16n prefetch and a burst length of 16, resulting in 32 bytes of data transferred per access on one channel. Alternatively, the burst length can be set to 32 to transfer 64 bytes of data with a single burst. As a novelty, LPDDR4 is the first DRAM standard that allows vendors to implement efficient *Error Correction Codes* (ECC) based on larger blocks of data, i.e., a full burst and not a single byte, in the device itself. This requires the DRAM to treat a write command with a data mask differently from a normal one, because ECC is then calculated on a combination of already stored data and newly transmitted data. Internally, the command triggers a so-called *Read-Modify-Write* (RMW) sequence. It first reads out a complete burst of data to the ECC engine and corrects errors, then partially overwrites old data with new one, and finally calculates new parity bits and writes the complete burst back to the cells like a normal write. This RMW sequence introduces a higher latency for masked writes, which is considered in the standard separately. In contrast, previous DRAM generations could only be equipped with an external ECC engine located in the DRAM controller, because all writes were associated with the same latency. External RMW for data masking in combination with ECC was then realized by sending a separate read command before the actual write. Compared to this approach, the on-chip solution has three major benefits:

- The power consumption is reduced because both the read data for RMW and the ECC bits are not transferred over the interface but only inside the device.
- No additional device for parity storage is required and the effective storage capacity is not affected.
- The achievable performance is higher because internal RMW introduces shorter latencies than external RMW (shorter transfer paths, no data bus turnarounds).

4 ECC ARCHITECTURE

The following sections describe both the on-chip ECC architecture that is used inside the LPDDR4 devices as well as an additional exemplary external ECC architecture. Placing a second, external ECC around devices is one way of further decreasing the overall failure rates to reach a higher ASIL for the subsystem.

4.1 On-Chip ECC Architecture

Although the JEDEC standard defines a special masked write command with increased latency to allow vendors the implementation of on-chip ECC in LPDDR4 devices, neither the specific code nor the internal hardware implementation are specified in the standard. However, vendors are constrained by the standardized delays, the area overhead and the minimum access granularity of 256 bits. This usually limits them to *Single Error Correction* (SEC) codes, since correcting more than one error in a codeword results in a significantly higher computational complexity and exceeds the maximum area and latency [5].[1] Assuming the smallest access granularity of 256 bits, a (265, 256) shortened SEC Hamming code could be applied. But as we have seen in Section 2, the vendors only use a (136, 128) code. To be more precise, they split each DRAM bank up into two physical banks [18, 19, 27], leading to 16 and not only 8 banks per channel as specified in the standard. One bank is storing the lower half of each data beat while the other one is storing the upper half, and each half of a complete burst is protected with a (136, 128) shortened SEC Hamming code. This architecture enables a cheap

[1] Only detecting but not correcting errors in LPDDR4 brings no advantage because the information about occurrences cannot be propagated to the controller and the remaining system.

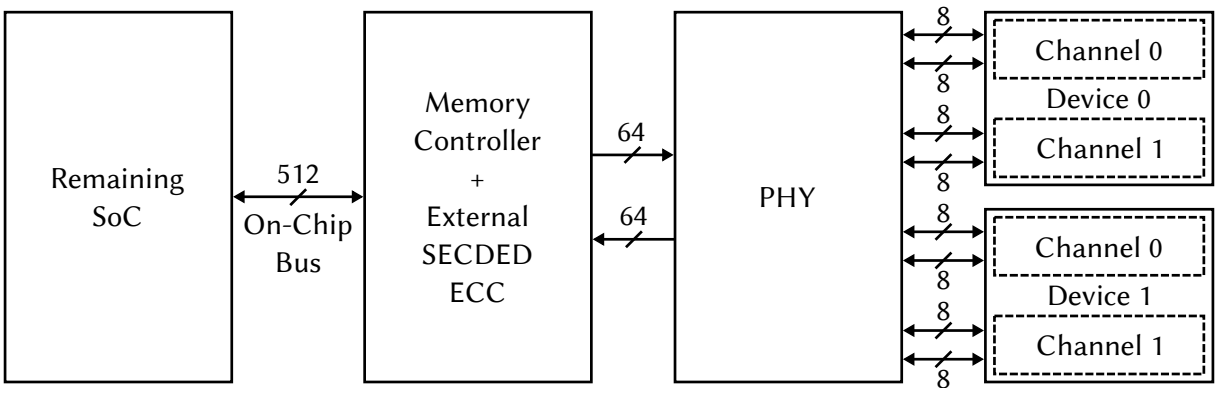

Figure 3: Complete Memory Subsystem with External SEC-DED ECC

implementation of byte mode with minimal hardware overhead by using the same devices and simply accessing only one physical bank per read or write access. In contrast, a byte-mode device with only eight physical banks and a (265, 256) code would require an internal RMW operation also for normal write accesses because only half of the data would be updated each time. This would worsen the power efficiency significantly. One additional advantage the two-bank architecture brings along with is that also 50 % of all possible double bit errors of the merged 256-bit codeword can be corrected. At the same time, storing in total 16 parity bits of two (136, 128) codes might not even require more chip area than 9 parity bits of a single (265, 256) code due to fixed layout rules. To hide the additional latency of RMW for masked writes, each physical bank uses its own local ECC engine, making the ECC encoding in one bank and decoding in another bank happen simultaneously and allowing gapless masked writes in a bank-interleaved fashion [27]. For illustration, the simplified architectures of a single bank and a complete LPDDR4 channel are shown in Figures 1 and 2.

4.2 External ECC Architecture

In addition to the on-chip ECC, an external ECC can be wrapped around one or several LPDDR4 devices to further decrease failure rates. At this point board manufacturers have more flexibility on selecting a suitable ECC architecture by combining multiple devices to one wider channel. As stated in Section 3, the parity bits for external ECC either have to be stored on an extra device or in the same device by reducing the effective storage capacity. Without any loss of generality, we choose the latter to demonstrate our methodology. Similar to the approach of Micron introduced in a presentation about the use of DRAM in automotive applications [4, 34], the 128 bits of data fetched from one internal bank are protected with an external (72, 64) shortened *Single Error Correction, Double Error Detection* (SEC-DED) Hamming code. While for the on-chip ECC of LPDDR4 devices the detection of double-bit errors brings no advantage (see Section 4.1), spending one additional parity bit to detect double-bit errors in the external ECC engine is a worthwhile investment in automotive platforms. This information can be communicated to the remaining system in order to enter a safe state when proper operation is not guaranteed any more. Since the parity bits are stored in line with the actual data, only 64 of 128 bits can effectively be used, meaning that the device capacity is reduced by half. A more efficient external ECC storage could be achieved by spreading one codeword across several internal banks, devices or burst accesses. However, it would result in a much more complex error propagation and is omitted in this work for the sake of simplicity. To match the 64 bytes usual last level cache line size of modern processors, two dual-channel devices are combined and form a single 64-bits-wide channel. With each burst access, eight internal banks are then targeted simultaneously. The remaining system is coupled to the memory controller via a 512-bits-wide on-chip bus to transfer the data of one DRAM access in a single beat. This exemplary subsystem architecture is shown in Figure 3.

5 LPDDR4 FAULT TREE MODEL

For safety in automotive applications, error detection is more important than error correction in order to bring the car into a safe state. Therefore, intelligent coverage mechanisms are required to meet the safety goals. To ensure that a safety goal is not violated, a careful analysis must be employed. Fault tree analysis is a well-known method to perform deductive safety analysis. On the basis of Boolean logic (OR, AND, and NOT gates) a series of lower-level events with certain probabilities is combined. However, for safety analyses at the hardware level, inductive methods are better suited to analyze the faults from their origin in the hardware elements, the failure propagation through the system, and the effects at the system boundary. Therefore, we use the more advanced variant of *Component Fault Trees* (CFT) as proposed by Adler et al. [1] in this work. A particular advantage they bring along with is the structuring of the fault trees based on the system components, in our case hardware elements. This way, the failure modes that can occur at the respective interfaces can be described for each component. By modularizing the fault trees into small subproblems, it allows for both deductive and inductive approaches. The failure propagation through the system is mapped onto a composite fault tree, which combines the individual CFTs of the components. At the system boundary, qualitative and quantitative analyses can finally be performed with tool support.

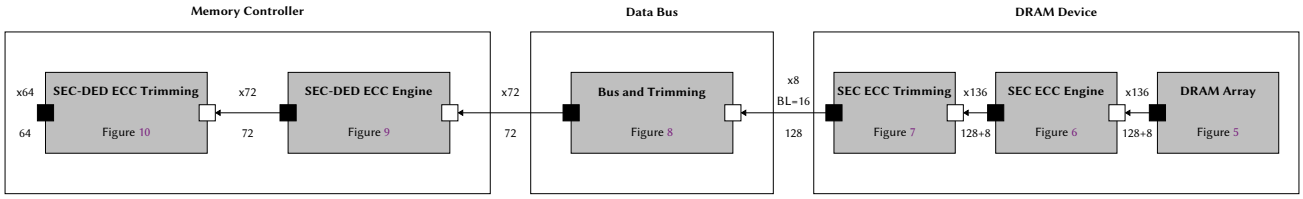

Figure 4: Error Propagation Chain used for Fault Tree Analysis (Half Channel)

The overall model consists of three major components including 6 CFTs (see Figures 4-10), which are explained in the following subsections in more detail. For the sake of simplicity, we only model one half of a 16-bits-wide channel, i.e., accesses to one internal bank. The full system is then composed of eight identical independent subsystems. It is important to note that our presented fault tree is a template for an exemplary LPDDR4 system, which, of course, has to be adopted to or extended for the specific use case (e.g., DRAM subsystem configuration or ECC implementation).

5.1 DRAM Array

Most of the errors in our model originate in the DRAM array itself. According to [4, 34], we model four main errors that may occur in the DRAM array: *Single-Bit Errors* (SBE), *Double-Bit Errors*, *Multi-Bit Errors* (MBE) and *Wrong Data* (WD). The latter includes for example an error in a row decoder, which will deliver wrong data, but most likely with correct ECC information. This error propagates directly to the DRAM subsystem boundary. The exact distribution of these errors is described in Section 6 by means of different scenarios. As shown in Figure 4, these errors are propagating upwards in the system to the next component, which is the internal DRAM SEC ECC.

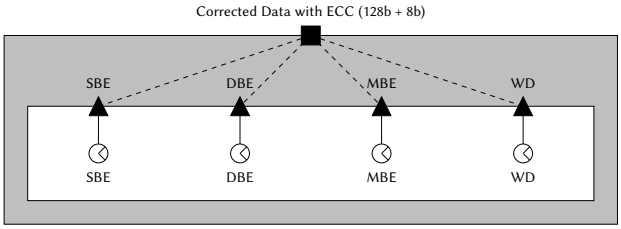

Figure 5: Component Fault Tree of the DRAM Array

5.2 SEC ECC Engine

The CFT of the SEC ECC engine is depicted in Figure 6. It describes the propagation of bit errors happening inside the DRAM when a (136, 128) Hamming SEC ECC is used (see Section 4.1). Accordingly, the input consists of 136 bits in total, whereof 128 bits correspond to the actual data protected by 8 parity bits. The input error events considered in this CFT are SBEs, DBEs and MBEs.

All SBEs will be corrected by the SEC ECC. Thus, the incoming SBE can only stay a SBE if there is a defect in the SEC engine, which we assume to happen with 0.1 FITs. In case of an incoming DBE, two cases have to be differentiated. First, if there is a defect in the SEC engine, the DBE will stay a DBE. Second, in case there is no defect in the SEC engine, the SEC will either detect that there is an uncorrectable error or attempt to correct the data, resulting in the introduction of a third error. The probability for introducing a third error largely depends on the specific code that is used, in particular on the number of minimal weight-3 codewords of the code. For the (136, 128) shortened SEC Hamming code from [7] there are 512 minimal codewords. Each of those minimal codewords results in 3 different double-bit-error patterns that will be miscorrected by the decoder. Thus, the probability for a miscorrection calculates to

$$P(\text{“Third Error”}) = \frac{3 \cdot 512}{\binom{136}{2}} \approx 17\%.$$

The complementary probability $1 - P(\text{“Third Error”}) \approx 83\%$ accounts for the probability that the SEC engine detects a double-bit error and does not perform an attempt of correction. In theory, the information about a detected uncorrectable error can be propagated to the system boundary (marked as $(*)$ in Figure 6) and, e.g., in an automotive application, the car could go into a safe state. But as this is not done for LPDDR4, we omit the differentiation between detected an undetected DBEs at this point and combine them with the other DBEs. However, for future DRAM generations like LPDDR5 it would be beneficial to propagate this information out of the memory such that the control unit can react on the DBE detection. In case of an incoming MBE, the SEC engine will not be able to correct any bit errors. Thus, a MBE is always propagated as a MBE.

5.3 SEC ECC Trimming

Since the DRAM's SEC ECC is only used internally, the redundancy is not further propagated out of the DRAM and is therefore discarded. This so-called *trimming* has severe effects on the error distribution. For example, if a TBE exists, but two of the erroneous bits are located in the redundancy part, these bits are discarded, which results in a SBE that is propagating towards the memory controller (i.e., 2/3 trimmed). Figure 7 shows all possibilities of how different errors transform to other errors as a CFT. The probabilities for these transformations can be calculated as follows,

$$P(x/y) = \frac{\binom{n-y}{r-x} \cdot \binom{y}{x}}{\binom{n}{r}}, \tag{1}$$

where n denotes the total number of bits, r the number of parity bits (or bits that will be trimmed), y the total number of errors before trimming, and x the number of errors after trimming.

For MBEs, we always assume the worst case, in which MBEs will always stay MBEs during all trimming events.

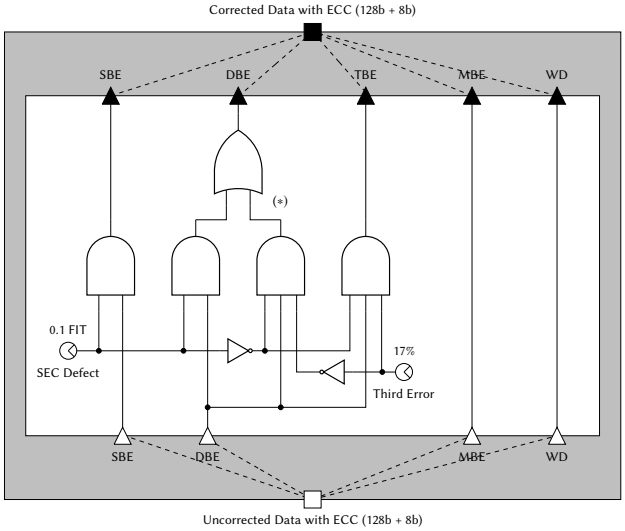

Figure 6: Component Fault Tree of the SEC ECC Engine

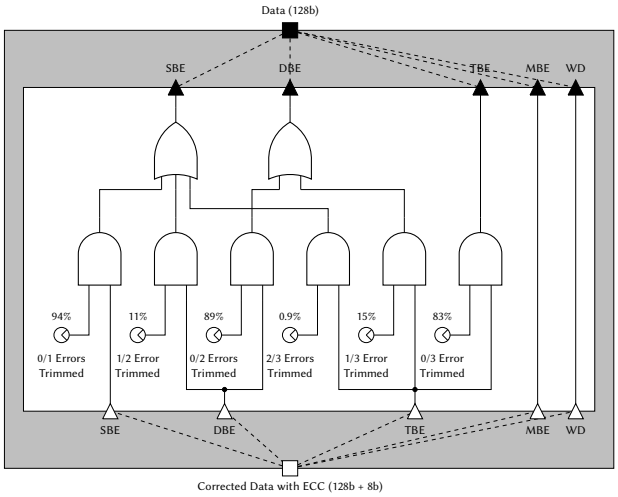

Figure 7: Component Fault Tree of the SEC ECC Trimming

5.4 Bus and Trimming

As explained in Section 4.2, only 72 of the 128 bits of one internal bank fetch are used for data and parity storage of the external SEC-DED ECC. The remaining 56 bits are discarded and trimmed away. Figure 8 shows the effects of this trimming and the error behavior of the DRAM's data bus. The probabilities for the trimming are computed similarly as shown in Section 5.3 (see Equation 1). Furthermore, this component also models errors of the DRAM bus as basic events, e.g., errors like DQ bus disturbances or no data driven (AZ) (i.e., the termination pulls all pins to V_{SS} such that bits are all zero), which will lead to MBEs.

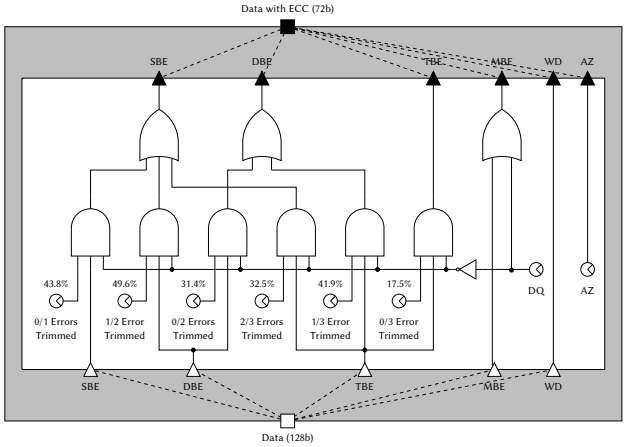

Figure 8: Component Fault Tree of the Bus and Trimming

5.5 SEC-DEC ECC Engine

The CFT of the SEC-DED ECC engine is depicted in Figure 9. The incoming data consists of 72 bits in total, whereof 64 bits correspond to the actual data protected by 8 parity bits. The input error events correspond to the output error events of the bus and trimming CFT (see Section 5.4 and Figure 8).

Similar to the SEC ECC engine (see Section 5.2 and Figure 6), we assume the SEC-DED ECC engine to have 0.1 FITs. As in the SEC ECC CFT, all types of errors are propagated as is in case there is a defect in the SEC-DED ECC engine.

In case that the SEC-DED ECC is working properly, it will correct all SBEs and detect all DBEs. In contrast to the on-chip SEC ECC, we now assume that detected errors are propagated to the system boundaries and cause the application to go into a safe state. Thus, detected errors do not count into the final error statistics.

Similar to the DBEs in the SEC ECC, there is a probability that TBEs can be detected in the SEC-DED ECC. As before, it depends on the number of minimal codewords of the code used. A popular SEC-DED code and alternative to the classical Hamming code is the so-called Hsiao code [12]. It is constructed in a certain way to simplify the hardware implementation complexity and is therefore widely used in memory applications. The Hsiao code yields a TBE miscorrection probability of around 56%, which we assume in our CFT model. Other codes aiming to minimize TBE miscorrections can reduce this probability to around 48% [8], but may not yield the same hardware complexity reduction as the Hsiao code.

In case of a MBE, we assume that around 50% of all MBEs will be detected by the SEC-DEC ECC. This assumption is based on the capability of the SEC-DED to detect an even number of errors.

5.6 SEC-DED ECC Trimming

Similar to the internal SEC of the DRAM, the redundancy information (8 bit) of the external SEC-DED will be discarded from the 72-bit word, leading to a payload data of 64 bits. This effect is considered by the CFT shown in Figure 10 and the probabilities are again computed according to Equation 1.

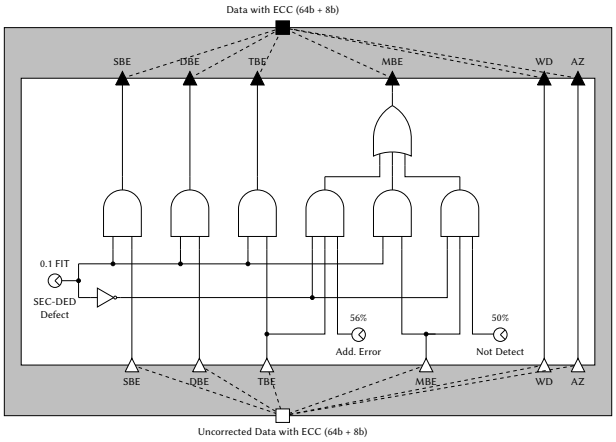

Figure 9: Component Fault Tree of the SEC-DED ECC Engine

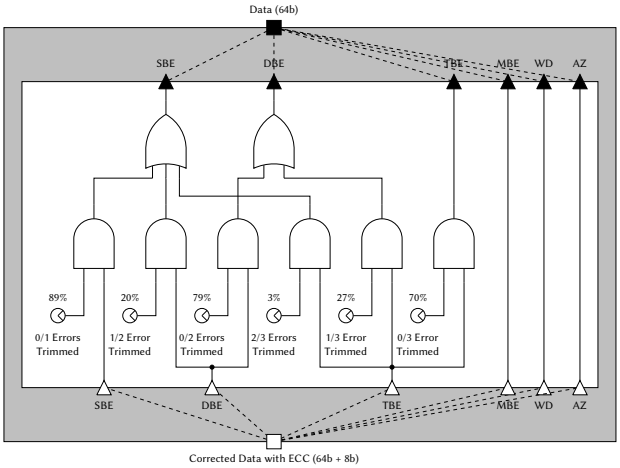

Figure 10: Component Fault Tree of the SEC-DED ECC Trimming

6 EXPERIMENTAL RESULTS

As shown in Section 5, our model has basic events, for instance SBE, DBE or MBE. Unfortunately, these error rates are not disclosed by DRAM vendors. Therefore, we first use the information from publicly-available references and second use different scenarios for a simple Monte Carlo analysis. In the following we will explain the assumptions made for our exemplary case study with the error model. In our experiments we assume a device channel size of 8 Gib. The parity bits for the internal SEC ECC require $\frac{8}{128} \cdot 8\,\text{Gib} = 0.5\,\text{Gib}$ of additional storage. According to [32] a DRAM has 0.066 FIT/Mib, which translates into an overall FIT rate of

$$R = (4\,\text{Gib} + 0.25\,\text{Gib}) \cdot 0.066 \frac{\text{FIT}}{\text{Mib}} \approx 287\,\text{FIT}$$

for one half of a 16-bits-wide DRAM channel in our example. According to [31, 32] SBEs are most prevalent, i.e., they account for 70 % of the total DRAM FIT rate. For the other events no measurement results exist in the literature. Therefore, we assume different scenarios shown in Table 1 for the further analysis, where Scenario 1 assumes that all errors besides SBEs and DQ are equally distributed and Scenarios 2 and 3 vary the distribution between DBE and MBE.

	SBE	DBE	MBE	AZ	DQ	Wrong Data
Scenario 1	70%	7.48%	7.48%	7.48%	0.1%	7.48%
Scenario 2	70%	20%	6%	1.9%	0.1%	2%
Scenario 3	70%	6%	20%	1.9%	0.1%	2%

Table 1: Scenarios for Fault Tree Analysis

ASIL	SPFM	LFM	Residual FIT
A	-	-	< 1000
B	> 90%	> 60%	< 100
C	> 97%	> 80%	< 100
D	> 99%	> 90%	< 10

Table 2: Requirements According to ISO 26262 [13]

For the automated evaluation, we use the tool SafeTBox[2] as well as Enterprise Architect[3]. With these tools, we model the presented architecture and the corresponding CFTs. The SafeTBox tool receives the parameters of our scenarios and calculates the metrics that are required for the ASIL rating. The ISO 26262 [13] specifies the hardware metrics that are used to evaluate the risk posed by hardware elements:

- *Single-Point Fault Metric* (SPFM): This metric reflects the coverage of an item or hardware element with respect to single-point faults either by design or by coverage from safety mechanisms.
- *Latent Fault Metric* (LFM): This metric reflects the coverage of an item or hardware element with respect to latent faults either by design (primarily safe faults), fault coverage via safety mechanisms, or by the driver's recognition of a fault's existence within the fault tolerant time interval of a safety goal.

In addition, target values are specified depending on the ASIL. Table 2 shows the required target values for different ASILs. When calculating the SPFM and LFM, we assume a failure portion of the DRAM of 4 %, and we attribute 96 % of the allowed residual failures to the rest of the hardware of the control unit as shown in [4, 34]. In the case of ASIL B and C, these are 96 FIT for the rest of the hardware, which we consider with initial 1920 FIT (50 % safety-related, 90 % diagnostic coverage) in our calculations. Comparing the results from our scenarios (see Table 3) with the target values, it becomes apparent that not even ASIL B can be reached. While all three scenarios achieve high values in the LFM sufficient for ASIL D, only Scenarios 2 and 3 achieves values in the SPFM in the range of ASIL B. However, the remaining absolute failure rates are too high in all three scenarios and would only satisfy ASIL A. This shows that further safety measures are required to fulfill the requirements of ADAS and AD in the future.

[2] https://www.safetbox.de/publications
[3] https://sparxsystems.com/products/ea/

	SPFM	LFM	Residual FIT
Scenario 1	87.5%	94.3%	529.35
Scenario 2	93.8%	89.9%	262.23
Scenario 3	90.1%	91.3%	418.76

Table 3: Results for Fault Tree Analysis

7 OBSERVATIONS

From these results we can make the following observations:

- The usual ECC mechanisms from commodity DRAMs like internal SEC as well as external SEC-DED do not fulfil ASIL B requirements for the presented LPDDR4 subsystem. If, for example, an ASIL D classification is required for ADAS and AD, new coverage mechanisms or ASIL decomposition must be employed.
- As mentioned in Section 5.2, propagating the detection of a double-bit error to the outside would be helpful to increase the coverage and achieve a higher ASIL. Especially for future LPDDR5 applications this should be considered.
- For safety in automotive applications, error detection is more important than error correction in order to bring the car into a safe state in case of a hazard. From this perspective, the usage of ECC might not be the most suitable choice. If it is more important to detect errors, other coding techniques like *Cyclic Redundancy Check* (CRC) would bring a much better coverage compared to classical Hamming ECC approaches with a similar redundancy and de-/encoding effort. Therefore, DRAM and SoC vendors should consider to implement error detection mechanisms into SoCs and memories, since they are suited much better for safety-critical applications.

8 CONCLUSION

In this paper we showed a CFT-based methodology for the safety analysis of LPDDR4 memory subsystems. Based on the proposed model, we also evaluated the ASIL of an exemplary LPDDR4 subsystem. Since DRAM is an essential component in today's advanced automotive control units, our work can serve as a foundation for future classification processes in the automotive industry, therefore taking one more step towards full autonomous driving. In the future, we will explore other coverage mechanisms with our CFT methodology.

ACKNOWLEDGMENTS

This work was supported within the Fraunhofer and DFG cooperation programme (Grant no. 248750294) and supported by the Fraunhofer High Performance Center for Simulation- and Software-based Innovation. Furthermore, we thank Mercedes-Benz for their support.

REFERENCES

[1] Rasmus Adler, Dominik Domis, Kai Höfig, Sören Kemmann, Thomas Kuhn, Jean-Pascal Schwinn, and Mario Trapp. 2011. Integration of Component Fault Trees into the UML. In *Models in Software Engineering*, Juergen Dingel and Arnor Solberg (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 312–327.

[2] Infineon Technologies AG. 2020. AURIX 32-bit microcontrollers for automotive and industrial applications. https://www.infineon.com/dgdl/Infineon-TriCore_Family_BR-ProductBrochure-v01_00-EN.pdf?fileId=5546d4625d5945ed015dc81f47b436c7, visited 2021-07-15.

[3] Leonardo Bautista-Gomez, Ferad Zyulkyarov, Osman Unsal, and Simon McIntosh-Smith. 2016. Unprotected Computing: A Large-Scale Study of DRAM Raw Error Rate on a Supercomputer. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 645–655. https://doi.org/10.1109/SC.2016.54

[4] Aaron Boehm. 2021. DRAM – More Important Than You Think for Achieving Automotive Functional Safety. https://www.designnews.com/electronics/dram-more-important-you-think-achieving-automotive-functional-safety, visited 2021-07-16.

[5] Sanguhn Cha, O. Seongil, Hyunsung Shin, Sangjoon Hwang, Kwangil Park, Seong Jin Jang, Joo Sun Choi, Gyo Young Jin, Young Hoon Son, Hyunyoon Cho, Jung Ho Ahn, and Nam Sung Kim. 2017. Defect Analysis and Cost-Effective Resilience Architecture for Future DRAM Devices. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 61–72. https://doi.org/10.1109/HPCA.2017.30

[6] Yung-Chang Chang, Li-Ren Huang, Hsing-Chuang Liu, Chih-Jen Yang, and Ching-Te Chiu. 2014. Assessing automotive functional safety microprocessor with ISO 26262 hardware requirements. In *Technical Papers of 2014 International Symposium on VLSI Design, Automation and Test*. 1–4. https://doi.org/10.1109/VLSI-DAT.2014.6834876

[7] Alexander Davydov, Leonid Kaplan, Yury Smerkis, and Grigory Tauglikh. 1981. Optimization of shortened Hamming codes. *Problems of Information Transmission* 17 (12 1981).

[8] Alexander Davydov and Leonid Tombak. 1991. An Alternative To The Hamming Code in The Class of SEC-DED Codes in Semiconductor Memory. *Information Theory, IEEE Transactions on* 37 (06 1991), 897 – 902. https://doi.org/10.1109/18.79958

[9] Alessandro Frigerio, Bart Vermeulen, and Kees Goossens. 2019. Component-Level ASIL Decomposition for Automotive Architectures. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. 62–69. https://doi.org/10.1109/DSN-W.2019.00021

[10] Saugata Ghose, Tianshi Li, Nastaran Hajinazar, Damla Senol Cali, and Onur Mutlu. 2019. Demystifying Complex Workload-DRAM Interactions: An Experimental Study. 93–93.

[11] Jin-Woo Han, Jungsik Kim, Dong-Il Moon, Jeong-Soo Lee, and M. Meyyappan. 2019. Soft Error in Saddle Fin Based DRAM. *IEEE Electron Device Letters* 40, 4 (2019), 494–497. https://doi.org/10.1109/LED.2019.2897685

[12] Mu-Yue Hsiao. 1970. A class of optimal minimum odd-weight-column SEC-DED codes. *IBM Journal of Research and Development* 14, 4 (1970), 395–401.

[13] ISO. 2011. ISO 26262: Road vehicles – Functional safety.

[14] Seo-Hyun Jeon, Jin-Hee Cho, Yangjae Jung, Sachoun Park, and Tae-Man Han. 2011. Automotive hardware development according to ISO 26262. In *13th International Conference on Advanced Communication Technology (ICACT2011)*. 588–592.

[15] Matthias Jung, Sally A. McKee, Chirag Sudarshan, Christoph Dropmann, Christian Weis, and Norbert Wehn. 2018. Driving into the Memory Wall: The Role of Memory for Advanced Driver Assistance Systems and Autonomous Driving. In *Proceedings of the International Symposium on Memory Systems* (Alexandria, Virginia) *(MEMSYS '18)*. ACM, New York, NY, USA, 377–386. https://doi.org/10.1145/3240302.3240322

[16] Kelly James, Synopsys, Inc. 2020. Synopsys Delivers Industry's First Processor IP Certified for Full ISO 26262 ASIL D Compliance. https://news.synopsys.com/2020-09-30-Synopsys-Delivers-Industrys-First-Processor-IP-Certified-for-Full-ISO-26262-ASIL-D-Compliance, visited 2021-07-15.

[17] Hiroyuki Kondo, Sugako Otani, Norimasa Otsuki, Yasufumi Suzuki, Naoto Okumura, Shohei Maeda, Tomonori Yanagita, Takao Koike, Kosuke Yayama, Yasuhisa Shimazaki, Masao Ito, Minoru Uemura, Toshihiro Hattori, and Noriaki Sakamoto. 2020. A 28-nm Automotive Flash Microcontroller With Virtualization-Assisted Processor Supporting ISO26262 ASIL D. *IEEE Journal of Solid-State Circuits* 55, 1 (2020), 133–144. https://doi.org/10.1109/JSSC.2019.2953826

[18] Nohhyup Kwak, Saeng-Hwan Kim, Kyong Ha Lee, Chang-Ki Baek, Mun Seon Jang, Yongsuk Joo, Seung-Hun Lee, Woo Young Lee, Eunryeong Lee, Donghee Han, Jaeyeol Kang, Jung Ho Lim, Jae-Beom Park, Kyung-Tae Kim, Sunki Cho, Sung Woo Han, Jee Yeon Keh, Jun Hyun Chun, Jonghoon Oh, and Seok Hee Lee. 2017. 23.3 A 4.8Gb/s/pin 2Gb LPDDR4 SDRAM with sub-100μA self-refresh current for IoT applications. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 392–393. https://doi.org/10.1109/ISSCC.2017.7870426

[19] H. J. Kwon, E. Seo, C. Y. Lee, Y. H. Seo, G. H. Han, H. R. Kim, J. H. Lee, M. S. Jang, S. G. Do, S. H. Cho, J. K. Park, S. Y. Doo, J. B. Shin, S. H. Jung, H. J. Kim, I. H. Im, B. R. Cho, J. W. Lee, J. Y. Lee, K. H. Yu, H. K. Kim, C. H. Jeon, H. S. Park, S. S. Kim, S. H. Lee, J. W. Park, S. S. Lee, B. T. Lim, J. y. Park, Y. S. Park, H. J. Kwon, S. J. Bae, J. H. Choi, K. I. Park, S. J. Jang, and G. Y. Jin. 2017. 23.4 An extremely low-standby-power 3.733Gb/s/pin 2Gb LPDDR4 SDRAM for wearable devices. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 394–395. https://doi.org/10.1109/ISSCC.2017.7870427

[20] Aanchal Lakhotia, Reuben Chang, Daryl Santos, and Christopher Greene. 2020. Fault Tree Analysis To Understand And Improve Reliability Of Memory Modules Used In Data Center Server Racks. *Procedia Manufacturing* 51 (01 2020), 989–997. https://doi.org/10.1016/j.promfg.2020.10.139

[21] Kuen-Long Lu and Yung-Yuan Chen. 2019. ISO 26262 ASIL-Oriented Hardware Design Framework for Safety-Critical Automotive Systems. In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. 1–6. https://doi.org/10.1109/ICCVE45908.2019.8965235

[22] Katsushige Matsubara, Lieske Hanno, Motoki Kimura, Atsushi Nakamura, Manabu Koike, Kazuaki Terashima, Shun Morikawa, Yoshihiko Hotta, Takahiro Irita, Seiji Mochizuki, Hiroyuki Hamasaki, and Tatsuya Kamei. 2021. 4.2 A 12nm Autonomous-Driving Processor with 60.4TOPS, 13.8TOPS/W CNN Executed by Task-Separated ASIL D Control. In *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, Vol. 64. 56–58. https://doi.org/10.1109/ISSCC42613.2021.9365745

[23] JEDEC Solid State Technology Association. 2015. Low Power Double Data Rate 3 (JESD209-3C).

[24] JEDEC Solid State Technology Association. 2020. Low Power Double Data Rate 4 (JESD209-4C).

[25] Hassan Mujtaba. 2018. NVIDIA Drive Xavier SOC Detailed – A Marvel of Engineering, Biggest and Most Complex SOC Design To Date With 9 Billion Transistors. https://wccftech.com/nvidia-drive-xavier-soc-detailed/, visited 2021-07-15.

[26] Prashant J. Nair, David A. Roberts, and Moinuddin K. Qureshi. 2015. FaultSim: A Fast, Configurable Memory-Reliability Simulator for Conventional and 3D-Stacked Systems. *ACM Trans. Archit. Code Optim.* 12, 4, Article 44 (Dec. 2015), 24 pages. https://doi.org/10.1145/2831234

[27] Tae-Young Oh, Hoeju Chung, Jun-Young Park, Ki-Won Lee, Seunghoon Oh, Su-Yeon Doo, Hyoung-Joo Kim, ChangYong Lee, Hye-Ran Kim, Jong-Ho Lee, Jin-Il Lee, Kyung-Soo Ha, YoungRyeol Choi, Young-Chul Cho, Yong-Cheol Bae, Taeseong Jang, Chulsung Park, Kwangil Park, SeongJin Jang, and Joo Sun Choi. 2015. A 3.2 Gbps/pin 8 Gbit 1.0 V LPDDR4 SDRAM With Integrated ECC Engine for Sub-1 V DRAM Core Operation. *IEEE Journal of Solid-State Circuits* 50, 1 (2015), 178–190. https://doi.org/10.1109/JSSC.2014.2353799

[28] Minesh Patel, Jeremie Kim, Hasan Hassan, and Onur Mutlu. 2019. Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices. 13–25. https://doi.org/10.1109/DSN.2019.00017

[29] Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. 2009. DRAM Errors in the Wild: A Large-Scale Field Study. In *SIGMETRICS*.

[30] Anton Shilov. 2018. Arm Unveils Arm Safety Ready Initiative, Cortex-A76AE Processor. https://www.anandtech.com/show/13398/arm-unveils-arm-safety-ready-initiative-cortexa76ae-processor, visited 2021-07-15.

[31] Vilas Sridharan, Nathan DeBardeleben, Sean Blanchard, Kurt B. Ferreira, Jon Stearley, John Shalf, and Sudhanva Gurumurthi. 2015. Memory Errors in Modern Systems: The Good, The Bad, and The Ugly. *SIGARCH Comput. Archit. News* 43, 1 (March 2015), 297–310. https://doi.org/10.1145/2786763.2694348

[32] V. Sridharan and D. Liberty. 2012. A study of DRAM failures in the field. In *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*. 1–11. https://doi.org/10.1109/SC.2012.13

[33] Vilas Sridharan, Jon Stearley, Nathan DeBardeleben, Sean Blanchard, and Sudhanva Gurumurthi. 2013. Feng Shui of supercomputer memory positional effects in DRAM and SRAM faults. In *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 1–11.

[34] Steffen Buch, Micron Technology. 2020. Questions to Ask Your Memory Supplier… About Functional Safety for DRAM. https://www.youtube.com/watch?v=mzcbtXdWDcg, visited 2021-07-16.

[35] Kyle Wiggers. 2019. Tesla claims its latest self-driving chip is 7 times more powerful than its rivals'. https://venturebeat.com/2019/04/22/tesla-claims-its-latest-self-driving-chip-is-six-times-more-powerful-than-its-rivals/, visited 2021-07-15.