

Driving Into the Memory Wall

The Role of Memory for Advanced Driver Assistance Systems and Autonomous Driving

POSITION PAPER

Matthias Jung

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

Sally A. McKee

Electrical and Computer Engineering,
Clemson University
Clemson, South Carolina
samckee@clemson.edu

Chirag Sudarshan

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

Christoph Dropmann

Fraunhofer Institute for Experimental
Software Engineering (IESE)
Kaiserslautern, Germany
christoph.dropmann@iese.fraunhofer.de

Christian Weis

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

Norbert Wehn

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

ABSTRACT

Autonomous driving is disrupting conventional automotive development. Underlying reasons include control unit consolidation, the use of components originally developed for the consumer market, and the large amount of data that must be processed. For instance, Audi's zFAS or NVIDIA's Xavier platform integrate GPUs, custom accelerators, and CPUs within a single domain controller to perform sensor fusion, processing, and decision making. The communication between these heterogeneous components and the algorithms for *Advanced Driver Assistance Systems* and *Autonomous Driving* require low latency and huge memory bandwidth, bringing the *Memory Wall* from high-performance computing in data centers directly to our cars. In this paper we discuss these and other requirements in using DRAM for near-term autonomous driving architectures.

CCS CONCEPTS

• **Computer systems organization** → **Embedded hardware**; • **Hardware** → **Dynamic memory**; **Safety critical systems**;

KEYWORDS

DRAM, ADAS, Autonomous Driving

ACM Reference Format:

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 POSITION PAPER. In *The International Symposium on Memory Systems (MEMSYS)*, October 1–4, 2018, Old Town Alexandria, VA, USA. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3240302.3240322>

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, October 1–4, 2018, Old Town Alexandria, VA, USA

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-6475-1/18/10...\$15.00

<https://doi.org/10.1145/3240302.3240322>

1 INTRODUCTION

The automotive industry is undergoing major changes with respect to *Electric-Electronic* (E/E) architectures for enabling autonomy. First, the current approach of using a hundred distributed *Electronic Control Units* (ECUs) connected by a network does not scale and is no longer cost-effective [7]. There is thus a trend transitioning from distributed ECUs to a dozen networked, consolidated domain controllers [58]. This transition dramatically changes current approaches to automotive design. Second, there is a convergence of mainstream and mission-critical markets [17]. Due to cost and performance, processing elements like GPUs that were originally developed for the consumer sector are now being used for safety-critical applications. Third, more and more data coming from numerous perception sensors must be processed in real time in order to guarantee the functionality of *Advanced Driver Assistance Systems* (ADAS) and even fully *Autonomous Driving* (AD). According to Intel, the averagely driven car will process 4 TB of data per hour [54].

The Society of Automotive Engineers (SAE International) standardization body defines different levels of autonomy for on-road vehicles. These range from issuing warnings only (Level 0) to hands-on (Level 1), hands-off (Level 2), eyes-off (Level 3), mind-off (Level 4), and steering-wheel-optional (Level 5). Level 4 allows user control under specific circumstances (e.g., congested traffic), whereas Level 5 vehicles are fully autonomous [62]. Therefore, the levels 1-3 represent ADAS, where the levels 4-5 represent AD. A major task for future AD platforms with autonomy Level 4 and Level 5 is Neural Network (NN) inference.

As a consequence of this ADAS/AD revolution, to meet computational requirements future automotive systems are forced to contain heterogeneous computing platforms and memories similar to those in embedded consumer devices. For instance, Audi recently presented its A8 car featuring TTTech's "zentrales Fahrerassistenzsteuergerät" (zFAS) advanced autonomy Level 3 driver assistance platform. zFAS is Audi's entry point for autonomy, and it demonstrates the major changes outlined above. In contrast to classical distributed-ECU automotive solutions, Audi's zFAS system is designed to be a consolidated platform on a single PCB, as shown in

Figure 1. The reason for this consolidation is that the system’s main task is sensor fusion. In many cases, computation on raw sensor data is more accurate than processing meta data (like object lists) computed from preprocessing ECUs [38].

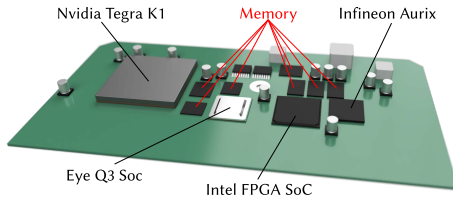


Figure 1: Simplified PCB View of zFAS

In particular, zFAS features [38]:

- An NVIDIA K1 GPU for 360° image processing;
- An Intel Mobile Eye *System on Chip* (SoC) for detecting traffic signs, pedestrians, collisions, lights, and lanes;
- An Intel Altera Cyclone FPGA for enabling communication between the other components, object and map fusion, parking piloting, and data pre-processing;
- An Infineon Aurix CPU for assistance systems like the Level 3 traffic jam pilot; and
- Several types of memory for communication, buffering, and computation.

A look inside today’s HPC data centers shows that ASICs instead of GPUs are employed for efficient NN processing (training and inference). This suggests that ASICs and heterogeneous SoCs with high memory bandwidth will be used in future automotive platforms. NVIDIA recently presented their Xavier platform, which confirms this outlook. Compared to the zFAS platform, which uses an FPGA for communication among all devices, Xavier’s functionality is integrated into a single SoC with more than nine billion transistors in a 12nm FinFET technology. This is currently the most complex SoC available for the automotive market [66].

The Xavier SoC will support up to 30 trillion operations per second while consuming only 30 watts of power, making it a promising compute platform for autonomy Level 4 and Level 5. It features [66]:

- An eight-core ARM 64 bit CPU with functional safety features like dual execution, parity and ECC;
- A 512-core Volta GPU;
- A deep learning accelerator;
- computer vision accelerators;
- An 8K HDR video processor; and
- Eight x32 LPDDR4 DRAMs at 2.1 GHz, providing a theoretical bandwidth of 137 GB/s.

However, in all the discussions on automotive electronics, the aspect of memory has not yet been sufficiently addressed [15]. The ever increasing number of large datasets that heterogeneous compute platforms like zFAS and Xavier must process in real-time need to be buffered in large, fast, and high-endurance consumer memories like DRAMs, which historically have only been part of the car’s entertainment systems — in other words, ADAS and AD will move DRAM from entertainment components into safety-critical components [24]. For example, the amount of input data, temporary

data, weights, and activation parameters of NN algorithms is very large compared to usual on-chip memories (e.g., caches or scratch pad), therefore, DRAM access is largely the bottleneck for real-time inference of NNs [72]. The DRAM standardization committee JEDEC and its members also responded to this trend by organizing an automotive electronics forum in September 2017 [74], where car makers (OEMs), suppliers (Tier 1s), SoC and memory vendors came together to discuss the new role of DRAM in the automotive context.

Using DRAM for non-entertainment purposes poses challenges ranging from providing sufficient bandwidth to latency, reliability, thermal issues, and security/safety. Some of these challenges are new to automobile designers as well as memory engineers. We discuss these challenges in the following sections of this paper.

2 BACKGROUND AND RELATED WORK

All major automotive manufacturers are developing some kind of AD solution. For instance, Audi will introduce a Level 4 highway pilot in 2020-2021 [45]. Together with partners Lyft, Domino’s Pizza, and Postmates, Ford aims to produce a vehicle that will be Level 4 or above by 2021 [27]. Deutsche Post DHL will upgrade its fleet of up to 3,400 autonomous streetscooter vehicles in 2018 [18]. General Motors plans to launch a Level 5 self-driving car with no steering wheel in 2019 [10]. Nissan intends to deploy commercially viable semi-autonomous vehicles by 2020 [16]. And Renault has already deployed its SYMBIOZ concept car with Level 4 autonomy [75].

ADAS and AD systems rely on multiple input sources ranging from LiDAR, radar, image processing, computer vision, and high-speed networking within the vehicle, to name a few. Together with electric and hybrid vehicles, ADAS and AD represents one of the fastest-growing segments in automotive electronics [59]. A 2017 Cadence blog post points out that ADAS/AD requires NNs which require higher bandwidth, safety, security, and reliability compared to traditional automotive semiconductors [51].

Other researchers have highlighted non-memory constraints and challenges inherent in building these mission-critical, autonomous driving systems. For instance, Lin et al. identify six classes of constraints including performance, predictability, storage, thermal, power, and others [42]. They find that in order to respond to real-time traffic conditions, ADAS architectures must complete their end-to-end processing in less than 100 ms with a frame rate higher than 10 frames per second. They focus on the computational bottlenecks of object detection, object tracking, and localization, studying implications and trade-offs across devices when accelerating these activities with GPUs, FPGAs, and ASICs. Okuda et al. survey trends in the algorithms used to perform these tasks [55].

3 DRAM CHALLENGES FOR THE AUTOMOTIVE SECTOR

The selection of a DRAM subsystem usually has three main dimensions: bandwidth, latency, and capacity. Bandwidth is the amount of data that can be transferred between DRAM and a computational unit within one second. The maximum DRAM bandwidth is limited to the number of data pins times the interface frequency. Latency is the access time that it takes to complete an access. In fact, latency helps bandwidth, but not vice versa [56]. For instance, lower DRAM

latency results in more accesses per second, and therefore higher bandwidth, whereas increasing the number of data pins increases the bandwidth without decreasing latency. In the context of AD, NN inference requires high bandwidth, but real-time and safety critical tasks require predictable latency. Section 3.1 presents different DRAM technologies in light of ADAS bandwidth challenges. Section 3.2 discusses challenges with respect to latency and the non-deterministic timing behavior of DRAM controllers. Section 3.3 shows the DRAM capacity requirements for ADAS and autonomous solutions. Sections 3.4, 3.5 and 3.6 discuss additional dimensions such as power, reliability, thermal problems, and safety/security, all of which play an important role.

3.1 Bandwidth

As noted, ASICs are used in today’s data centers for neural network training and inference. A prominent example of a custom NN ASIC is Google’s *Tensor Processing Unit* (TPU). Google showed that four out of six studied neural networks are bandwidth-limited by the DRAM memory [30]. Fast NN execution must not only be supported by the computational units, but the memory subsystem must be designed to avoid hitting the *Memory Wall* [77].

While the automotive industry is focused largely on sensor perception, processing, and car-to-car communication, the problem of the memory wall for AI is currently not addressed and probably underestimated. Current automotive applications require a DRAM bandwidth of less than 60 GB/s [15], which can be provided easily with standard DDR and LPDDR solutions. However, advanced AD applications for autonomy Level 4 and Level 5 will require between 400-1024 GB/s memory bandwidth [15, 72], which is hard to realize with the current DRAM technologies. To put the problem in perspective, we survey current memory architectures. Figure 2 shows different DRAM-based memory subsystems, and Figure 3 and Figure 4 show their properties with respect to interface frequency, maximum theoretical bandwidth, and energy consumption per transferred bit¹.

The maximum bandwidth of conventional DIMM-based DDR solutions is limited by the I/O count and interface speed. This limitation arises due to the package, power considerations, and costs on both the memory and processing sides.

To avoid pin limitations, designers and vendors are using *Buffer on Board* (BoB) organizations [9] in which an additional logic component is interposed between the CPU and DRAM to control the memory and to communicate with the CPU over a narrow, high-speed, serial interface. This technique is mainly used in applications where several terabytes of DRAM are required. The required storage capacity in the automotive domain is much smaller than in the high-performance systems BoB targets, and thus this organization is inappropriate (cf. Section 3.3).

Package on Package (PoP) organizations reduce the distance between the DRAM and the MPSoC (from centimeters to millimeters), providing higher bandwidth, lower latency, better power efficiency, and smaller form factors, all of which are especially important for smartphones and tablets. Low power DDR DRAMs (e.g., LPDDR4),

can be used either as a device on a PCB or mounted directly as PoP. The latter organization permits only one device to be connected, requiring DRAM commands to be serialized due to the resulting low pin count. The Xavier platform instead uses eight LPDDR4 devices on the PCB to deliver a theoretical bandwidth of 137 GB/s.

To address the huge memory demand of highly parallel GPUs, graphic DDR DRAMs (e.g., GDDR5X or GDDR6) use techniques like *Quad Data Rate* (QDR) to deliver high bandwidth compared to conventional DDR DRAM. While LPDDR4 devices are designed and optimized for ultra-low power consumption with aggressive power gating and higher-threshold transistors, GDDR5X/6 devices focus on delivering the highest achievable bandwidth. Both use an architecture with distributed banks (heavy sub-banking) due to the wider data I/O widths of x16/x32 and the larger data prefetch of up to 16 bit per data I/O. However, GDDR5X/6 devices improve the column-to-column cycle time (t_{CCD}) by reducing data path delays from primary sense amplifiers to the global sense amplifiers. Furthermore, GDDR5X/6 chips use an on-die *Phase Lock Loop* (PLL) to achieve very high I/O performance in QDR mode. In contrast, LPDDR4 devices have no on-die PLL or *Delay Lock Loop* (DLL). Combining 16 GDDR6 devices in QDR mode yields a theoretical bandwidth of 1 TB/s, as shown in Figure 3.

Another way of achieving high bandwidth is 3D stacking; examples include WIDE I/O [48], Micron’s *Hybrid Memory Cube* (HMC) [47] and Samsung’s *High Bandwidth Memory* (HBM) [49]. These memories reduce the distance between CPU and external RAM from centimeters to micrometers by means of *through-silicon via* (TSV) technology. The available bandwidth increases due to more pins provided by the TSVs, but, more importantly, this technology provides a major boost in energy efficiency compared to standard off-chip (G)DDR devices [57, 76]. For example, four parallel HBM2 devices on a 2.5D silicon interposer can provide up to 1 TB/s [72]. However, 3D memories suffer from thermal issues, which we discuss in Section 3.5.

Tezzeron’s DiRAM4 is a 3D variant that moves the sense amplifiers, peripheral circuitry, and control logic to a separate layer from the storage array. This disintegration enables increased memory density due to higher cell efficiency within each layer. DiRAM uses fine-grain tungsten super contacts in place of thick copper TSVs², reducing the size and increasing the number of parallel vertical channels to dramatically increase bandwidth. The energy and bandwidth numbers presented in Figure 4 and Figure 3 make DiRAM a perfect choice for HPC. However, other 2.5D and 3D memory solutions are better suited to the automotive mass market than this expensive, niche product.

It is becoming clear that automotive applications must concentrate on DRAM solutions like GDDR and HBM in order to cope with their high bandwidth requirements. However, the bandwidth numbers presented above are theoretical maxima: the sustainable memory bandwidth is much less, and it strongly depends on how the data are stored in the memories, i.e., the memory access pattern [31]. Therefore, it’s not only important to choose a memory that provides high bandwidth, it is also important to design a DRAM controller that can bring the sustainable bandwidth nearer to the theoretical maximum.

¹Note that the latency, actual sustainable bandwidth, and the total energy consumption of a DRAM strongly depend on the application being executed. Reaching the maximum theoretical bandwidths in Figure 3 is practically impossible on general-purpose systems.

²One copper TSV is equal to 100 super contacts.

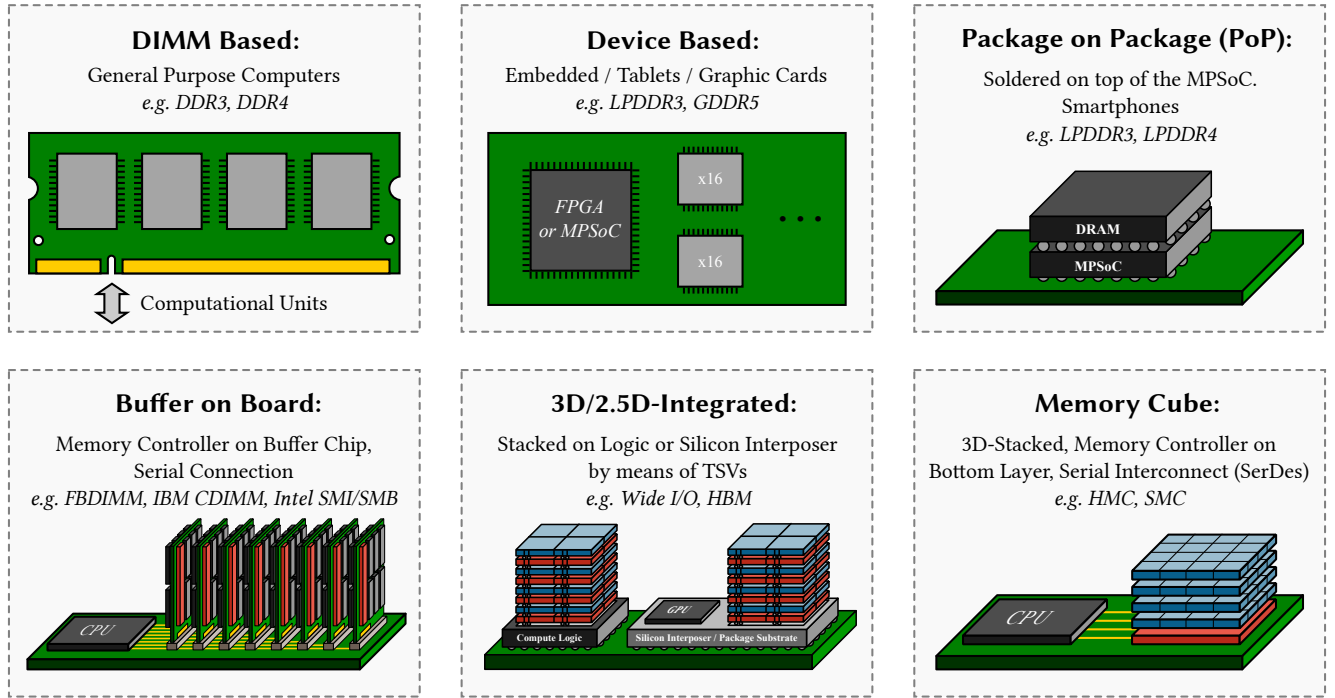


Figure 2: DRAM based Memory Subsystems

General purpose DRAM controllers use online scheduling techniques in order to improve the sustainable bandwidth, e.g., by reducing the number of row misses or read/write transitions. In order to reduce the number of read/write transitions, DRAM controllers buffer read and write commands in two distinct queues. An arbiter switches between read and write mode to diminish the t_{WTR} penalty, the minimum time interval between the end of a WR burst and a RD command.

However, in the ADAS and AD application space, many applications (e.g., signal, image, or NN processing) have regular, fixed, and deterministic memory access patterns. On the computer architecture side, this inherent application-specific knowledge has been heavily exploited in DSPs, GPUs, and ASICs. However, on the memory side there is limited research that exploits application knowledge for customized *Application-Specific Memory Controllers* (ASMC) [31].

General purpose memory controllers that use online scheduling algorithms and quality of service methods have only a *local* view on the application due to the limited depth of the scheduler’s buffer. Jung et al. [31] propose a memory controller based on an optimized DRAM address mapping for applications with deterministic memory access patterns. This ASMC can outperform online schedulers because it was designed with a *global* application view. This offline ASMC maximizes the number of row buffer hits and exploits the bank-level parallelism of the DRAM device in order to keep up the sustainable bandwidth near the maximum.

The scheduling and offline techniques described above (if applicable) are key for an efficient memory subsystem design for future ADAS and autonomous vehicle solutions. However, it is not easy to

nail down the optimal memory strategy only from the bandwidth perspective — there are other important considerations, which we highlight in the following sections.

3.2 Non-Deterministic Latency

From an application point of view, the DRAM subsystem has non-deterministic timing behavior [21] due to its complex protocol (i.e., the latency of a DRAM request depends on previous issued commands) and the runtime optimizations of the memory controller; This makes it difficult to provide predictable performance and thus to guarantee real-time task predictability [1]. Figure 5 shows a histogram of the CHStone ADPCM benchmark [26] simulated on the DRAMSys framework [34]³. Although the average latency is concentrated around 40 ns, the memory latency can easily vary by an order of magnitude.

As with the bandwidth issues discussed in Section 3.1, the memory controller plays an integral role in this non-deterministic timing behavior. In the following we list contributors to the DRAM latency.

- **Row Misses:** The latency of a bank access varies depending on the state of its row buffer. If a memory access targets the same row as the row currently cached in the buffer (a row hit), it results in lower latency and lower energy memory accesses. On the other hand, if a memory access targets a different row from that currently in the buffer (a row miss), it results in higher latency and energy consumption.
- **Close vs. Open Page Policy:** *Commercial off-the-shelf* (COTS) DRAM controllers usually support two major modes, an *Open Page Policy* (OPP) and *Closed Page Policy* (CPP). The

³The simulated DRAM is a DDR3 with a RBC address mapping and disabled scheduler.

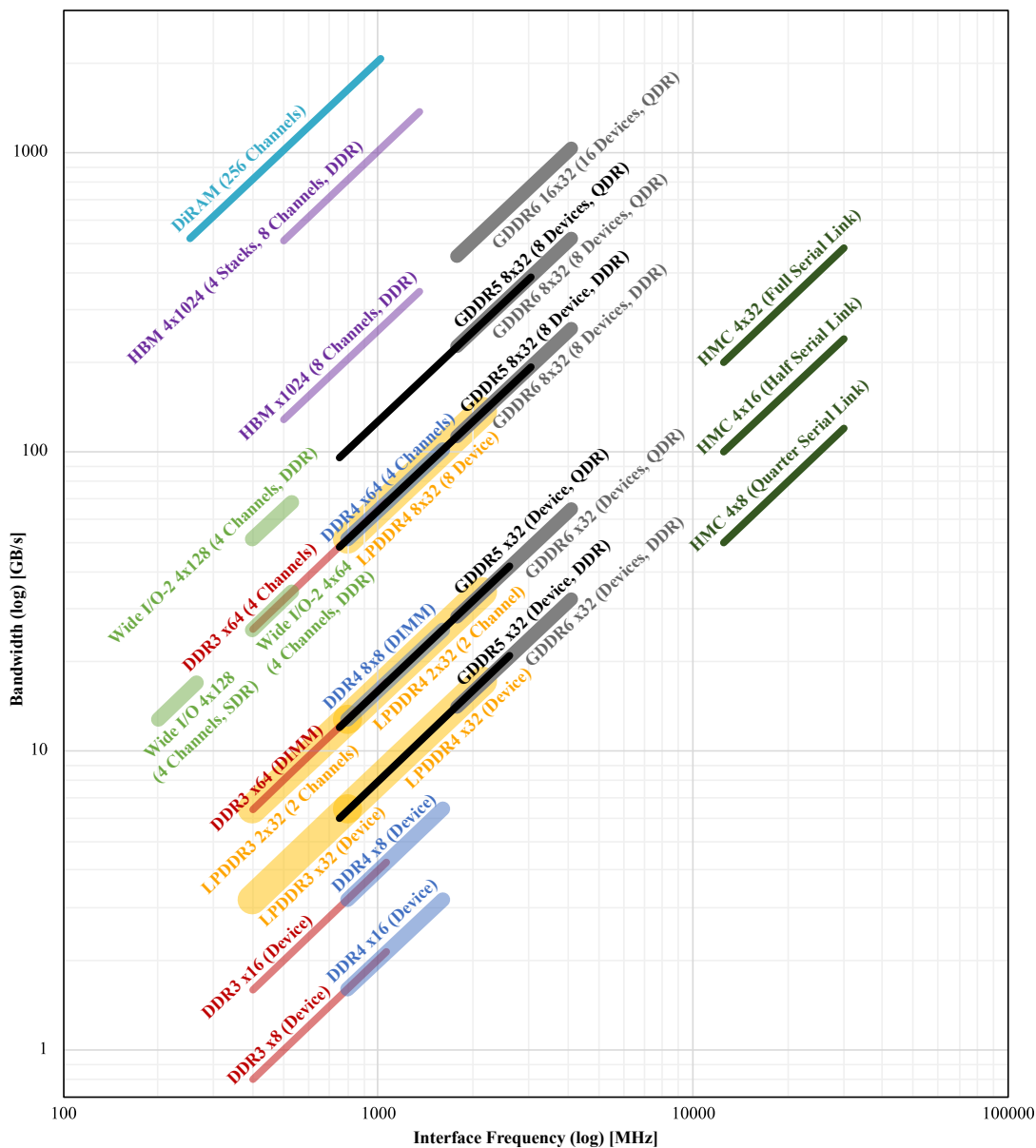


Figure 3: Interface Frequency and Maximum Bandwidth of Different DRAMs (Inspired by [23])

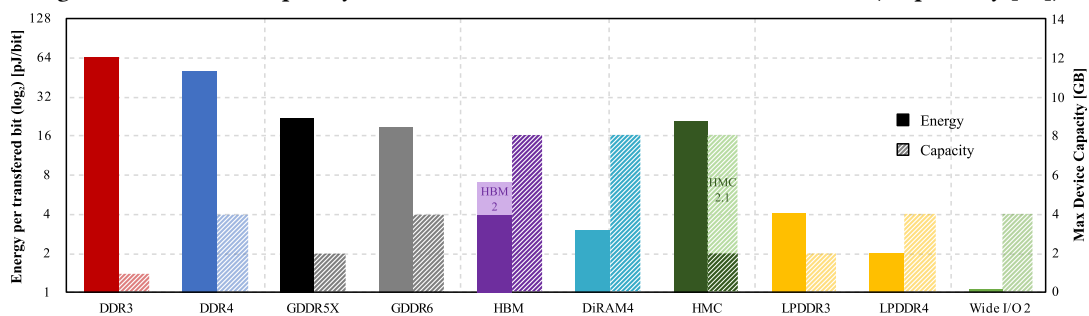


Figure 4: Properties of Today's DRAMs (Sources: Micron, Hynix, Nvidia, Xilinx, JEDEC, Tezzaron)

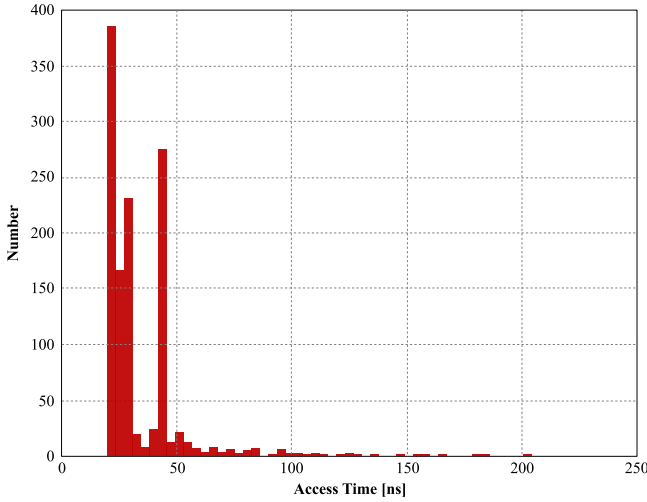


Figure 5: Latency for an Application Running on DDR3

OPP keeps the current row active after a read or write, whereas the CPP precharges the row automatically after each access. The latter makes the latency for each access more predictable, but it also decreases performance for access patterns with high row-hit potential.

- **Refresh:** DRAMs must be refreshed regularly due to their charge-based bit storage property (capacitor). The memory controller issues this refresh operation periodically (e.g., every 64 ms). Normal accesses to the DRAM are blocked for the duration of the refresh operation t_{RFC} (350 ns for DDR4), degrading performance with respect to both bandwidth and latency and increasing energy consumption⁴. If a memory access arrives at the same time that a refresh happens it will experience unpredictable latency. Recent work by Jung et al. [32, 33, 35] proposes safe refresh disabling, e.g., for frame buffers where the data lifetime is less than the refresh period.
- **Scheduling:** COTS memory controllers are designed for good average case performance, and therefore they employ runtime scheduling of requests (c.f. Section 3.1) for online optimization. For example, with schedulers that attempt to maximize row hits it is possible that a request that misses the row could starve, which again results in a hardly predictable latency.
- **Arbitration:** A major challenge arises when several computational units are issuing read and write requests to the memory controller. The different applications running on these units will place their requests in different input buffers, and arbitration must be performed. This leads to interference that can cause high unpredictability.
- **Command/Address and Data bus Contention:** All banks in a DRAM share the same command/address and data buses, which can limit overall performance. If the data bus utilization is 100% the maximum bandwidth is reached. On the

⁴In fact, the degradation grows linearly with the capacity, which means it grows exponentially with each density generation. Liu et al. [43] and Bhati et al. [6] predicted that 40% to 50% of the power consumption of future DRAM devices will be caused by refresh commands, and the maximum DRAM bandwidth will be significantly reduced.

other hand if the command bus utilization is 100%, it often has a negative impact on the available bandwidth and the latency, since WR and RD commands must be issued in later cycles.

- **Current Limiting and Power Supply Network:** In order to limit peak currents there exists a rolling time-frame in which a maximum of four banks can be activated, called *Four Activate Window* (t_{FAW}). There is also a minimum time interval between two ACT commands to different banks, (t_{RRD}). Also these constraints can influence bandwidth, latency, and predictability in specific scenarios.
- **Further Effects:** Bank-Groups in DDR4 or GDDR or rank-to-rank switching constraints in DDR memories, which limit the predictability.

Due to the above mentioned complexity of DRAM and its controller, the automotive community has thus far largely avoided using DRAM for safety-critical and real-time applications. For example, Infineon’s Aurix CPU, which is widely used for safety-critical applications, does not provide a DRAM controller. As discussed before, due to the requirement for larger memories and the convergence of mainstream and mission-critical markets, the automotive community finds itself forced to address DRAM memory.

In past years there were many investigations with respect to DRAM controllers for real-time and mixed-criticality applications [2, 3, 11–13, 20, 22, 41, 73, 79]. Most of these approaches concentrate operating the DRAM with statically precomputed command patterns which guarantee a predictable behavior. However, this predictability often comes with a degradation of sustainable bandwidth. The authors of [19] present DRAM selections and configurations for real-time systems. A detailed book, which summarizes real-time challenges related to DRAM has been presented by Goossens et al. [21]. The previously mentioned techniques and investigations have to be carried out to COTS DRAM controllers in order to make way for DRAMs in real-time applications in the automotive ADAS and autonomy context.

3.3 Capacity

The memory capacity is defined as the amount of data that can be stored in it. Prior to the self driving era, most of the safety critical real-time constrained ECUs (e.g. power train ECU) were only equipped with low capacity SRAMs. However, the applications running on automotive ADAS/AD platforms like zFAS or Xavier demand a high memory capacity which are beyond the limits of on-chip SRAMs. Therefore DRAMs must be applied because they offer a much higher capacity. As of today, most of the ADAS applications (Level 1 to Level 2) require not more than 4GB of DRAM capacity [53]. Such memory capacity requirements can be easily addressed by any DRAM shown in Figure 4, either by using it as a single device or by a combination of several devices. Very recently Micron estimated that by 2020 with full autonomous driving (Level 4 to Level 5) the DRAM memory capacity will increase to 18 – 24 GB [53], which is still manageable by most of the available DRAM variants. For example, by using HBM or GDDR6 for a maximum bandwidth of 1 TB/s a capacity of 32 GB or 64 GB is achieved implicitly. Hence, the capacity of DRAM is not at all the bottleneck for such applications.

3.4 Power Consumption

Power plays an important role for autonomous systems. For example today's AD prototypes use around 2,500 W and it is estimated that commercial AD systems e.g. based on NVIDIA's Xavier will use around 500 W in the future [71]. For both, combustion-engine and electric cars, this power consumption will reduce the driving range significantly.

Therefore, this section explores the possible DRAM candidates for AD from the power/energy perspective. According to Figure 4, the preliminary choice in this regard are LPDDR4 and Wide I/O2 due to their very low energy numbers. However, When aiming at the specified high memory bandwidth of 1 TB/s these devices are not feasible. For example, to achieve the aforementioned bandwidth with LPDDR4 64 devices (x32) are required. Although the average power would be only ~ 17 W at a peak frequency of 2000 MHz, the high number of resulting I/O pins (2048) is unfeasible. Hence, the only alternative candidates for such an AD system (Level 4 and Level 5) with a required bandwidth of 1 TB/s are HBM2 and GDDR5X/6, as shown in Figure 3⁵. According to Figure 3 and Figure 4 the average power consumed⁶ by the HBM (4 stack x1024) and GDDR6 (16 devices, QDR, x32) devices are ~ 60 W and ~ 150 W, respectively. These numbers show that DRAM will be a significant contributor to future AD systems. Such a high power consumption also fosters a high thermal dissipation. Therefore, the next section presents the exploration from the temperature and reliability perspective.

3.5 Temperature vs. Reliability

DRAMs are very sensitive to high temperature, which increases the leakage at the cells. In order to avoid data corruption by retention errors, the refresh frequency needs to be increased. The general rule of thumb is to double the refresh rate for every 10°C increase after 85 °C [35]. Usually, the operating temperature range for automotive electronics is specified between - 40°C up to 125°C [46]. For example, many automotive manufactures put their ADAS perception modules behind the windshield where they profit from the cleaning through the wipers. However, the ADAS system may be exposed to direct sunlight resulting in potentially very high temperatures [24]. Some DRAMs like DDR4, LPDDR4 or GDDR6 have been already qualified for automotive usage by the DRAM manufactures providing high temperatures, e.g. 105°C or 125°C [24, 50]. However, also for these DRAMs the refresh period must be decreased from 64 ms to 4 – 8 ms for 125°C, which leads to a serious collapse of the sustainable bandwidth [61].

Additionally, DRAM manufacturers also employ several conservative and non-optimized circuit techniques to make the DRAM cells and control logic operating at this extreme conditions. However, these solutions are associated with following side effects like higher power consumption, larger area, lower operating frequency, reduced bandwidth and higher cost per device. This situation is even worse for today's 3D stacked DRAM systems (e.g., Wide I/O, HBM, HMC, etc.), which aggravate the thermal crisis: i.e. these DRAMs are even more sensitive to temperature changes because of the stacked thin dies [61]. Additionally, when aiming for highest

bandwidths with HBM or HMC, these devices will consume, as mentioned before, a significant amount of power on a small area compared to their commodity counterparts. Thus, the self-heating of 3D-DRAMs is even more accelerated.

Due to shrinking technology nodes, reliability is a major concern in DRAMs. Thus, many techniques exist to improve the reliability e.g. by using *Error Correcting Codes* (ECC). State-of-the-art ECC DDR DIMMs, for instance, consist of 8 DRAM devices and a 9th device, used to store the ECC redundancy. But also for the automotive device oriented DRAM candidates like LPDDR4, GDDR6 and HBM, different architectural strategies for ECC data storage are currently under discussion [24].

The code that performs the error correction and detection for DDR DRAMs is usually a Hamming code, that is a *Single Error Correcting / Double Error Detecting* (SEC/DED) code [25]. Oftentimes, a variation of the Hamming code is used, the so called Hsiao code [28], that aims at minimizing the implementation complexity of the SEC/DED code. Recent publications like e.g. [8, 37, 44, 65] show that also other codes or coding schemes should be considered by combining coding and device knowledge. For example, two vendors recently introduced on-die ECC for LPDDR4 [39, 40] to avoid retention errors. They lower the refresh rates by a factor of 4× in order to reduce power consumption. This shows the importance of ECC for retention errors in DRAMs.

3.6 Safety and Security

Safety is defined as absence of catastrophic consequences on users and environment, whereas security is defined as the concurrent existence of availability for authorized users only, confidentiality, and integrity [5].

The automotive ISO26262 standard [29] defines so called *Automotive Safety Integrity Levels* (ASIL), which define techniques and safety mechanisms that engineers should apply to achieve a given safety goal. Additionally the standard defines also acceptable failure rates. The ASIL levels range from no ASIL (not safety critical), QM (a defined quality management is required), ASIL A to ASIL D (highest integrity). In short, the ASIL represents the tolerable probability that a corresponding failure contributes to a safety goal violation. By using DRAMs in a safety critical ADAS system, errors of the DRAM subsystem have the potential to violate safety goals like "*In autonomous driving mode the ADAS must prevent steering in unintended directions with ASIL D*". Hence, a DRAM subsystem that realizes safety critical functionality has to fulfill strict hardware safety requirements.

In this section, we will briefly list potential memory safety requirements based on the ISO26262 and discuss *random*, *systematic* and *malicious* DRAM failures that could violate a safety goal. The ISO26262 Part 5, Annex D mentions volatile and non-volatile memory with corresponding failures and safety mechanisms, however in a very generic way and not specific to DRAM:

- **Failures:** transient and permanent (stuck-at) failure, linked failures, memory or memory interconnect access delays.
- **Safety Mechanisms:** pattern tests, march based self-tests, parity bits, ECC, memory block replication with comparison, checksums e.g. CRC.

⁵Since DiRAM is a niche product it is not considered in this exploration

⁶operated at respective peak frequency

In case, the corresponding safety requirement has the integrity ASIL A, inductive analysis as failure mode and effect analysis, design inspections, hierarchical design and documentation have to be performed. Higher ASILs (B, C, D) require in addition deductive analysis, e.g. fault tree analysis and the evaluation of the effectiveness of safety mechanisms handling random hardware failure via calculating a hardware failure rate metric. For instance, in this qualitative analysis the probability of the sum of all not covered faults must be derived from the following target values: 100 FIT⁷ for ASIL B and C and 10 FIT for ASIL D [29]. In the following, we will discuss *random*, *systematic* and *malicious* DRAM failures that could violate a safety goal.

3.6.1 Random DRAM Failures. Apart from the temperature based retention errors discussed in Section 3.5 DRAM is also prone to transient soft errors i.e. effects of cosmic particle strikes [24, 67]. Moreover, due to the high frequency of DRAM interfaces there exists the possibility of transient transmission errors on the DRAM bus. Furthermore, there can be hard errors related to stuck at failures or aging which could lead for instance to a defect column decoder. There exists only a limited amount of studies on DRAM error rates in the field since manufacturers and data centers are very careful to share this sensitive information [52, 63, 70]. Sridharan et al. report 20 – 66 FIT for a single DRAM device [68, 69]. This highlights again the need for appropriate safety mechanisms in order to decrease the FIT rates. For example the DRAM controller contains an additional logic, which tests the interface periodically in order to detect errors or a strong ECC that is able to correct errors online in order to guarantee functional safety.

3.6.2 Systematic DRAM Failures. From ISO26262 point of view, systematic failures are related in a deterministic way to a cause that can only be eliminated via a change of a relevant factor, e.g. the design, documentation, operational procedures or the manufacturing process. Underestimating the memory wall, the non-deterministic latency or designing a faulty partitioning concept are potential systematic DRAM failures. A DRAM partitioning concept is needed in case of an integrated system that hosts ADAS software components with different integrity levels that share the DRAM, called mixed criticality [60]. The ISO26262 requests so called *freedom from interferences* to avoid that a failure of one software component propagates to another component with a higher ASIL via platform’s shared resources (e.g. DRAM). An example scenario for interference can be the starving of accesses for a safety critical application A because a non-critical application B has a high row buffer hit rate, as mentioned in Section 3.2. The effect for application A will be an increased memory access latency and therefore the execution time of the application’s process increases. The safety requirement is that this latency will be taken into account, e.g. via simulation or worst case estimation. Existing off-the-shelf DRAM controllers, however, do not support mixed criticality and therefore allow interference between applications [14].

3.6.3 Malicious Failures. Apart from random and systematic failure causes, malicious cause can lead to a safety goal violation, too. The ISO26262 states that a system that is to be safe must also be secure. However, because of transitions to open environments

for *Car2Car* and *Car2X* communication the vulnerability of DRAM must be considered. As DRAM process technology scales down, the electrical interference between the memory cells increases, which leads to disturbance errors. Recently, the *Row Hammer* problem [36] and its exploit [64] have caused a lot of attention in research and newspapers. By repeatedly opening and closing a DRAM row, called *Hammering*, bits in adjacent rows can flip. This effect can be exploited to write on memory locations with prohibited access rights to e.g. gain kernel privileges or escape a sandbox or hypervisor. The Row Hammer security attack [36] is a potential malicious behavior that has to be avoided. Controller techniques like *Target Row Refresh*, where rows will be refreshed when their activation count exceed a threshold or techniques on the device level like [4, 78] must be employed.

In summary, as mentioned before, the communication between computational units within a domain controller will be realized with shared memory. The challenge is to deal with random faults, establish a clear segregation between mixed critical applications running on the same platform and employ countermeasures against malicious behavior.

4 CONCLUSION

ADAS and AD force the usage of consumer components like GPUs or DRAMs in the context of safety critical systems, to keep up with the computational requirements. In this work we highlight the boundary and environmental conditions for an ADAS and AD systems from a DRAM perspective with respect to bandwidth, latency, reliability, temperature, and safety/security. From the pure functional requirements like bandwidth, latency and capacity we select LPDDR4, GDDR5X/6 and HBM as most promising DRAM candidates for today’s ADAS systems, shown in Table 1. If we target autonomy Level 4 and 5 only GDDR5X/6 and HBM fulfill the functional requirements.

However, for the automotive industry, non-functional requirements like reliability, temperature and safety/security are also of high importance. So far, only LPDDR4 and GDDR5X/6 are certified for the high automotive temperature range. The complex stacking process and package integration of a 3D technology like HBM increases the failure probability, thereby not guaranteeing the required reliability for automotive grades. The high bandwidth, technological maturity, product availability and mass production of GDDR5X/6, in combination with thermal stability and reliability compared to 3D stack based DRAMs makes it one of the best choices in the near future for Level 4 and Level 5 applications. However, if the roadmaps for autonomy solutions and reliability of 3D integration merge, HBM will be also a viable solution due to its energy efficiency.

ACKNOWLEDGMENTS

This work was partially funded by the DFG grant no. WE2442/10-1, supported by the the *Fraunhofer High Performance Center for Simulation- and Software-based Innovation* and supported by the OPRECOMP project⁸ (Union’s Horizon 2020 research and innovation programme, under grant agreement No 732631). Furthermore, we thank the anonymous reviewers for their valuable suggestions.

⁷Failure in Time (FIT) is one failure in 10⁹ hours.

⁸<http://oprecomp.eu>

Table 1: Different AD (Level 4 and Level 5) DRAM Candidates

DRAM	Peak Bandwidth	Low Energy	Reliability	Automotive Grade (Temp.)
LPDDR4	o	++	+	yes
GDDR5X/6	+	o	+	yes
HBM	++	+	o	no
Legend	marginal (o), medium (+), high (++)			

REFERENCES

- [1] Ankit Agrawal and Gerhard Fohler. 2017. DRAM-Related Challenges in Task Scheduling with Timing Predictability on COTS Multi-cores for Safety-critical Systems. In *Proceedings of the International Symposium on Memory Systems (MEMSYS '17)*. ACM, New York, NY, USA, 265–267. <https://doi.org/10.1145/3132402.3132417>
- [2] B. Akesson, K. Goossens, and M. Ringhofer. 2007. Predator: A predictable SDRAM memory controller. In *2007 5th IEEE/ACM/IFIP International Conference on Hardware/Software CoDesign and System Synthesis (CODES+ISSS)*. 251–256. <https://doi.org/10.1145/1289816.1289877>
- [3] B. Akesson, W. Hayes Jr., and K. Goossens. 2011. Automatic Generation of Efficient Predictable Memory Patterns. In *2011 IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications*, Vol. 1. 177–184. <https://doi.org/10.1109/RTCSA.2011.33>
- [4] A. Amaya, H. Gomez, and E. Roa. 2017. Mitigating Row Hammer attacks based on dummy cells in DRAM. In *2017 IEEE International Conference on Consumer Electronics (ICCE)*. 442–443. <https://doi.org/10.1109/ICCE.2017.7889389>
- [5] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. 2004. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Trans. Dependable Secur. Comput.* 1, 1 (Jan. 2004), 11–33. <https://doi.org/10.1109/TDSC.2004.2>
- [6] Ishwar Bhati, Mu-Tien Chang, Z. Chishti, Shih-Lien Lu, and B. Jacob. 2015. DRAM Refresh Mechanisms, Trade-Offs, and Penalties. *IEEE Trans. Comput. PP*, 99 (2015), 1–1. <https://doi.org/10.1109/TC.2015.2417540>
- [7] Thomas Bloor. 2016. The Automotive Shift to Software-Defined, Consolidated Controller Architectures. <http://qxauto.blogspot.de/2016/10/automotive-shifting-software-defined.html>. (October 2016).
- [8] H. M. Chen, S. Jeloka, A. Arunkumar, D. Blaauw, C. J. Wu, T. Mudge, and C. Chakrabarti. 2016. Using Low Cost Erasure and Error Correction Schemes to Improve Reliability of Commodity DRAM Systems. *IEEE Trans. Comput.* 65, 12 (Dec 2016), 3766–3779. <https://doi.org/10.1109/TC.2016.2550455>
- [9] E. Cooper-Balis, P. Rosenfeld, and B. Jacob. 2012. Buffer-on-board memory systems. In *39th Annual International Symposium on Computer Architecture*. 392–403. <https://doi.org/10.1109/ISCA.2012.6237034>
- [10] Alex Davies. 2018. GM Will Launch Robocars Without Steering Wheels Next Year. <https://www.wired.com/story/gm-cruise-self-driving-car-launch-2019/>. (January 2018).
- [11] L. Ecco and R. Ernst. 2017. Architecting high-speed command schedulers for open-real-time SDRAM controllers. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2017. 626–629. <https://doi.org/10.23919/DATE.2017.7927063>
- [12] L. Ecco and R. Ernst. 2017. Tackling the Bus Turnaround Overhead in Real-Time SDRAM Controllers. *IEEE Trans. Comput.* 66, 11 (Nov 2017), 1961–1974. <https://doi.org/10.1109/TC.2017.2714672>
- [13] L. Ecco, S. Saidi, A. Kostrzewa, and R. Ernst. 2015. Real-time DRAM throughput guarantees for latency sensitive mixed QoS MPSoCs. In *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*. 1–10. <https://doi.org/10.1109/SIES.2015.7185038>
- [14] Leonardo Ecco, Sebastian Tobuschat, Selma Saidi, and Rolf Ernst. 2014. A mixed critical memory controller using bank privatization and fixed priority scheduling. In *RTCSA*. IEEE Computer Society, 1–10.
- [15] Sven Evers. 2017. Cinco-Play: Memory IS that Critical to Autonomous Driving. <https://www.micron.com/about/blogs/2017/october/cinco-play-memory-is-that-critical-to-autonomous-driving>. (November 2017).
- [16] Jon Fingas. 2017. Nissan's Rogue is its first US car with semi-autonomous driving. <https://www.engadget.com/2017/10/19/nissans-rogue-is-its-first-us-car-with-semi-autonomous-driving/>. (October 2017).
- [17] S. Girbal, M. Moreto, A. Grasset, J. Abella, E. Quiñones, F.J. Cazorla, and S. Yehia. 2013. On the Convergence of Mainstream and Mission-Critical Markets. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–10. <https://doi.org/10.1145/2463209.2488962>
- [18] Staff Global Trade. 2017. Deutsche Post DHL Selects NVIDIA for Autonomous Trucks. <http://www.globaltrademag.com/global-logistics/deutsche-post-dhl-selects-nvidia-autonomous-trucks>. (October 2017).
- [19] Manil Dev Gomony, Christian Weis, Benny Akesson, Norbert Wehn, and Kees Goossens. 2012. DRAM Selection and Configuration for Real-Time Mobile Systems. In *Design and Test in Europe*. 51–56.
- [20] S. Goossens, B. Akesson, and K. Goossens. 2013. Conservative open-page policy for mixed time-criticality memory controllers. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*. 525–530. <https://doi.org/10.7873/DATE.2013.118>
- [21] S. Goossens, K. Chandrasekar, B. Akesson, and K. Goossens. 2016. *Memory Controllers for Mixed-Time-Criticality Systems: Architectures, Methodologies and Trade-offs*. Springer International Publishing. https://books.google.de/books?id=19_7CwAAQBAJ
- [22] S. Goossens, K. Chandrasekar, B. Akesson, and K. Goossens. 2016. Power/Performance Trade-Offs in Real-Time SDRAM Command Scheduling. *IEEE Trans. Comput.* 65, 6 (June 2016), 1882–1895. <https://doi.org/10.1109/TC.2015.2458859>
- [23] Hiroshige Goto. 2015. Performance Memory Bandwidth Roadmap. <https://pc.watch.impress.co.jp/video/pcw/docs/1054/618/p5.pdf>. (2015).
- [24] Marc Greenberg. 2017. Understanding Automotive DDR DRAM. <https://www.synopsys.com/designware-ip/technical-bulletin/automotive-ddr-dram.html>. (2017).
- [25] Richard Wesley Hamming. 1950. Error Detecting and Error Correcting Codes. *Bell System Technical Journal* 26, 2 (1950), 147–160.
- [26] Yuko Hara, Hiroyuki Tomiyama, Shinya Honda, and Hiroaki Takada. 2009. Proposal and Quantitative Analysis of the CHStone Benchmark Program Suite for Practical C-based High-level Synthesis. *Journal of Information Processing* 17 (2009), 242–254. <https://doi.org/10.2197/ipsjip.17.242>
- [27] Andrew J. Hawkins. 2018. Ford wants to be the self-driving OS for the future of transportation. <https://www.theverge.com/2018/1/9/16868814/ford-self-driving-autonomous-vehicle-ces-2018>. (January 2018).
- [28] 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.
- [29] ISO. 2011. Road vehicles – Functional safety. (2011).
- [30] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Salek, et al. 2017. In-Datcenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17)*. ACM, New York, NY, USA, 1–12. <https://doi.org/10.1145/3079856.3080246>
- [31] Matthias Jung, Irene Heinrich, Marco Natale, Deepak M. Mathew, Christian Weis, Sven Krumke, and Norbert Wehn. 2016. ConGen: An Application Specific DRAM Memory Controller Generator. In *International Symposium on Memory Systems (MEMSYS 2016)*.
- [32] Matthias Jung, Deepak M. Mathew, Christian Weis, and Norbert Wehn. 2016. Approximate Computing with Partially Unreliable Dynamic Random Access Memory: Approximate DRAM. In *IEEE/ACM Design Automation Conference (DAC)*.
- [33] Matthias Jung, Deepak M. Mathew, Christian Weis, and Norbert Wehn. 2016. Efficient Reliability Management in SoCs - An Approximate DRAM Perspective. In *21st Asia and South Pacific Design Automation Conference (ASP-DAC)*.
- [34] Matthias Jung, Christian Weis, and Norbert Wehn. 2015. DRAMSys: A flexible DRAM Subsystem Design Space Exploration Framework. *IPSP Transactions on System LSI Design Methodology (T-SLDM)* (August 2015).
- [35] Matthias Jung, Éder Zulian, Deepak Mathew, Matthias Herrmann, Christian Brugger, Christian Weis, and Norbert Wehn. 2015. Omitting Refresh - A Case Study for Commodity and Wide I/O DRAMs. In *1st International Symposium on Memory Systems (MEMSYS 2015)*. Washington, DC, USA.
- [36] Yoongu Kim, R. Daly, J. H. Kim, C. Fallin, Ji Hye Lee, Donghyuk Lee, C. Wilkerson, K. Lai, and O. Mutlu. 2014. Flipping Bits in Memory without Accessing Them: An Experimental Study of DRAM Disturbance Errors. In *ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. 361–372. <https://doi.org/10.1109/ISCA.2014.6853210>
- [37] Kira Kraft, Matthias Jung, Chiarg Sudarshan, Deepak M. Mathew, Christian Weis, and Norbert Wehn. 2018. Improving the Error Behavior of DRAM by Exploiting its Z-Channel Property. In *IEEE Conference Design, Automation and Test in Europe (DATE)*.

- [38] Ingo Kuss. 2016. Audi zFAS — Enorme Datenmengen bewältigen. <http://www.elektroniknet.de/elektronik-automotive/assistenzsysteme/enorme-datenmengen-bewaeltigen-131797.html>. (July 2016).
- [39] 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>
- [40] C. K. Lee, Y. J. Eom, J. H. Park, J. Lee, H. R. Kim, K. Kim, Y. Choi, H. J. Chang, J. Kim, J. M. Bang, S. Shin, H. Park, S. Park, Y. R. Choi, H. Lee, K. H. Jeon, J. Y. Lee, H. J. Ahn, K. H. Kim, J. S. Kim, S. Chang, H. R. Hwang, D. Kim, Y. H. Yoon, S. H. Hyun, J. Y. Park, Y. G. Song, Y. S. Park, H. J. Kwon, S. J. Bae, T. Y. Oh, I. D. Song, Y. C. Bae, J. H. Choi, K. I. Park, S. J. Jang, and G. Y. Jin. 2017. 23.2 A 5Gb/s/pin 8Gb LPDDR4X SDRAM with power-isolated LVSTL and split-die architecture with 2-die ZQ calibration scheme. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 390–391. <https://doi.org/10.1109/ISSCC.2017.7870425>
- [41] Y. Li, B. Akesson, and K. Goossens. 2014. Dynamic Command Scheduling for Real-Time Memory Controllers. In *2014 26th Euromicro Conference on Real-Time Systems*. 3–14. <https://doi.org/10.1109/ECRTS.2014.18>
- [42] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E. Haque, Lingjia Tang, and Jason Mars. 2018. The Architectural Implications of Autonomous Driving: Constraints and Acceleration. In *23rd International Symposium on Architectural Support for Programming Languages and Operating Systems*.
- [43] Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu. 2012. RAIDR: Retention-Aware Intelligent DRAM Refresh. In *Proceedings of the 39th Annual International Symposium on Computer Architecture (ISCA '12)*. IEEE Computer Society, Washington, DC, USA, 1–12. <http://dl.acm.org/citation.cfm?id=2337159.2337161>
- [44] Alirad Malek, Evangelos Vasilakis, Vassilis Papaefstathiou, Pedro Trancoso, and Ioannis Sourdis. 2017. Odd-ECC: On-demand DRAM Error Correcting Codes. In *Proceedings of the International Symposium on Memory Systems (MEMSYS '17)*. ACM, New York, NY, USA, 96–111. <https://doi.org/10.1145/3132402.3132443>
- [45] Audi. 2018. Audi piloted driving. <http://media.audiusa.com/models/piloted-driving>. (2018).
- [46] Automotive Electronics Council. 2007. FAILURE MECHANISM BASED STRESS TEST QUALIFICATION FOR INTEGRATED CIRCUITS (AEC-Q100). (May 2007).
- [47] Hybrid Memory Cube Consortium. 2013. Hybrid Memory Cube Specification. (2013). http://www.hybridmemorycube.org/files/SiteDownloads/HMC_Specification%201_0.pdf
- [48] Jedic Solid State Technology Association. 2011. Wide I/O Single Data Rate (JESD 229). (Dec. 2011).
- [49] JEDEC Solid State Technology Association. 2015. High Bandwidth Memory (HBM) DRAM. *JEDEC Standard JESD235A* (2015).
- [50] Micron Technology, Inc. 2016. Automotive DDR4 SDRAM (MT40A1G8). (2016).
- [51] Paul McLellan. 2017. CDNDrive: Cadence Automotive IP Solutions. https://community.cadence.com/cadence_blogs_8/b/breakfast-bytes/posts/cndrive-cadence-automotive-solutions. (July 2017).
- [52] Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu. 2015. Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.
- [53] Raj Narasimhan. 2017. Fulfilling Quality Requirements for Memory in Automotive Applications. <http://www.arena-international.com/Journals/2017/04/04/y/l/g/1-Raj-Narasimhan-Micron.pdf>. (April 2017).
- [54] Patrick Nelson. 2016. Just one autonomous car will use 4,000 GB of data/day. <https://www.networkworld.com/article/3147892/internet/one-autonomous-car-will-use-4000-gb-of-dataday.html>. (December 2016).
- [55] Ryosuke Okuda, Yuki Kajiwar, and Kazuaki Terashima. 2014. A Survey of Technical Trend of ADAS and Autonomous Driving. In *International Symposium on VLSI-Technology, Systems, and Applications*.
- [56] David A. Patterson. 2004. Latency Lags Bandwidth. *Commun. ACM* 47, 10 (Oct. 2004), 71–75. <https://doi.org/10.1145/1022594.1022596>
- [57] J. T. Pawlowski. 2011. Hybrid Memory Cube. HotChips 23. (Aug. 2011).
- [58] Dominik Reinhardt and Markus Kucera. 2013. Domain Controlled Architecture - A New Approach for Large Scale Software Integrated Automotive Systems. In *Proceedings of the 3rd International Conference on Pervasive Embedded Computing and Communication Systems - Volume 1: PECCS*. INSTICC, SciTePress, 221–226. <https://doi.org/10.5220/0004340702210226>
- [59] Ian Riches. 2014. Automotive Ethernet Market Growth Outlook. standards.ieee.org/events/automotive/2014/00_Automotive_Ethernet_Market_Growth_Outlook. (Oct. 2014).
- [60] John Rushby. 1999. Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance. (1999).
- [61] MohammadSadeh Sadri, Matthias Jung, Christian Weis, Norbert Wehn, and Luca Benini. 2014. Energy Optimization in 3D MPSoCs with Wide-I/O DRAM Using Temperature Variation Aware Bank-Wise Refresh. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2014. 1–4. <https://doi.org/10.7873/DATE2014.294>
- [62] SAE International. 2014. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Technical Report. https://www.sae.org/standards/content/j3016_201609
- [63] Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. 2009. DRAM Errors in the Wild: A Large-Scale Field Study. In *SIGMETRICS*.
- [64] Mark Seaborn and Thomas Dullien. 2015. Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges. <http://googleprojectzero.blogspot.de/2015/03/exploiting-dram-rowhammer-bug-to-gain.html>. (March 2015).
- [65] Seyed Mohammad Seyedzadeh, Donald Kline, Jr, Alex K. Jones, and Rami Melhem. 2017. Mitigating Bitline Crosstalk Noise in DRAM Memories. In *Proceedings of the International Symposium on Memory Systems (MEMSYS '17)*. ACM, New York, NY, USA, 205–216. <https://doi.org/10.1145/3132402.3132410>
- [66] Danny Shapiro. 2018. NVIDIA DRIVE Xavier, World's Most Powerful SoC, Brings Dramatic New AI Capabilities. <https://blogs.nvidia.com/blog/2018/01/07/drive-xavier-processor/>. (January 2018). <https://blogs.nvidia.com/blog/2018/01/07/drive-xavier-processor/>
- [67] C. Slayman. 2011. Soft error trends and mitigation techniques in memory devices. In *2011 Proceedings - Annual Reliability and Maintainability Symposium*. 1–5. <https://doi.org/10.1109/RAMS.2011.5754515>
- [68] 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>
- [69] 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>
- [70] Ioan Stefanovici, Andy Hwang, and Bianca Schroeder. 2015. DRAM's Damning Defects - and How They Cripple Computers. *IEEE Spectrum* (November 2015).
- [71] Jack Stewart. 2018. Self-Driving Cars use Crazy Amounts of Power, and it's Becoming a Problem. <https://www.wired.com/story/self-driving-cars-power-consumption-nvidia-chip/>. (2018).
- [72] Kar Yee Tang. 2018. The New Deep Learning Memory Architectures You Should Know About. *Whitepaper eSilicon* (April 2018).
- [73] G. Thomas, K. Chandrasekar, B. Akesson, B. Juurlink, and K. Goossens. 2012. A Predictor-Based Power-Saving Policy for DRAM Memories. In *2012 15th Euromicro Conference on Digital System Design*. 882–889. <https://doi.org/10.1109/DSD.2012.11>
- [74] Federico Tiziani. 2017. Memory and Storage for L5 Autonomy from Automotive JEDEC Forum. <https://www.micron.com/about/blogs/2017/november/memory-and-storage-for-l5-autonomy-from-automotive-jedec-forum>. (November 2017).
- [75] Peter Valdes-Dapena. 2017. Renault, Nissan and Mitsubishi team up on self-driving and electric cars. <http://money.cnn.com/2017/09/15/technology/renault-nissan-mitsubishi-alliance-electric-self-driving-cars/index.html>. (September 2017).
- [76] C. Weis, I. Loi, L. Benini, and N. Wehn. 2012. An energy efficient DRAM subsystem for 3D integrated SoCs. In *proc. DATE 2012*.
- [77] Wm. A. Wulf and Sally A. McKee. 1995. Hitting the Memory Wall: Implications of the Obvious. *SIGARCH Comput. Archit. News* (March 1995), 5. <https://doi.org/10.1145/216585.216588>
- [78] C. M. Yang, C. K. Wei, Y. J. Chang, T. C. Wu, H. P. Chen, and C. S. Lai. 2016. Suppression of Row Hammer Effect by Doping Profile Modification in Saddle-Fin Array Devices for Sub-30-nm DRAM Technology. *IEEE Transactions on Device and Materials Reliability* 16, 4 (Dec 2016), 685–687. <https://doi.org/10.1109/TDMR.2016.2607174>
- [79] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha. 2013. MemGuard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms. In *2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 55–64. <https://doi.org/10.1109/RTAS.2013.6531079>