Chapter 3
The Dynamic Random Access Memory Challenge in Embedded Computing Systems

Matthias Jung, Christian Weis, and Norbert Wehn

3.1 Introduction

Dynamic random access memories (DRAMs) are key components in all computing systems that require large working memory. Due to the strong increase in data volume in many embedded applications, such as machine learning, image processing, autonomous systems, etc., DRAMs largely impact the overall system performance and power consumption. In many of these applications, the overall system performance is often limited by the memory bandwidth or latency and not by the computation itself. Due to the dynamic storage scheme of DRAMs and shrinking technology nodes, reliability is also a major concern in current and future DRAMs.

Therefore, new challenges arise, which we will discuss in this chapter. The most important metrics, which are typically considered for DRAM subsystems (especially in the *high-performance computing* (HPC) domain), are *bandwidth*, *latency*, and *capacity*. However, in the context of embedded systems it requires to consider further metrics, such as *power*, *temperature*, *reliability*, *safety*, and *security*. In the following we will highlight these challenges and refer to some of our recent contributions, which tackle these challenges.

M. Jung
Fraunhofer IESE, Kaiserslautern, Germany
e-mail: Matthias.Jung@iese.fraunhofer.de

C. Weis · N. Wehn (✉)
TU Kaiserslautern, Kaiserslautern, Germany
e-mail: weis@eit.uni-kl.de; wehn@eit.uni-klde

J.-J. Chen (ed.), *A Journey of Embedded and Cyber-Physical Systems*,
https://doi.org/10.1007/978-3-030-47487-4_3

3.2 Bandwidth and Latency

Bandwidth is the amount of data that can be transferred between DRAM and a computational unit within 1 s. 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 [33]. 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. A fast execution of applications on embedded systems must not only be supported by the computational units, but the memory subsystem must be designed to avoid hitting the *memory wall* [43]. For example, embedded applications for autonomous driving will require between 400 and 1024 GB/s of memory bandwidth [16], which is hard to realize with the current DRAM technologies. To put the problem in perspective, we survey current memory architectures.

Figure 3.1 shows different DRAM-based memory subsystems, and Figs. 3.2 and 3.3 show their properties with respect to interface frequency, maximum theoretical bandwidth, and energy consumption per transferred bit.[1] 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.

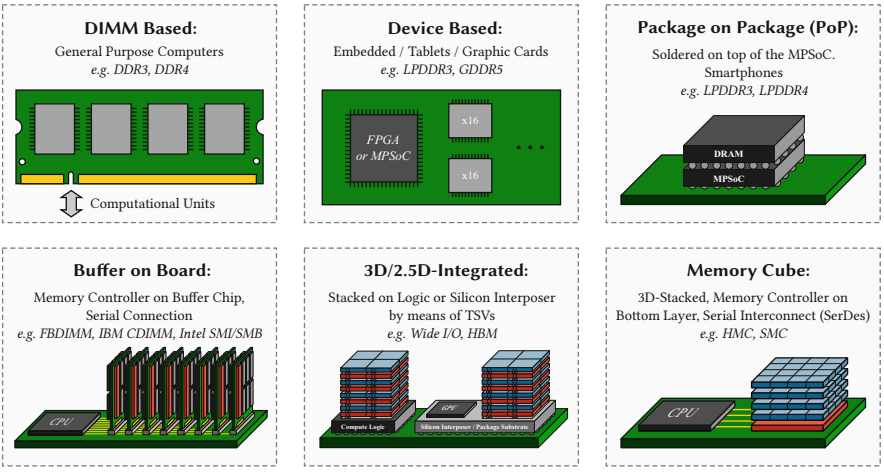

Fig. 3.1 DRAM-based memory subsystems

[1] Note 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 Fig. 3.2 is practically impossible on general-purpose systems.

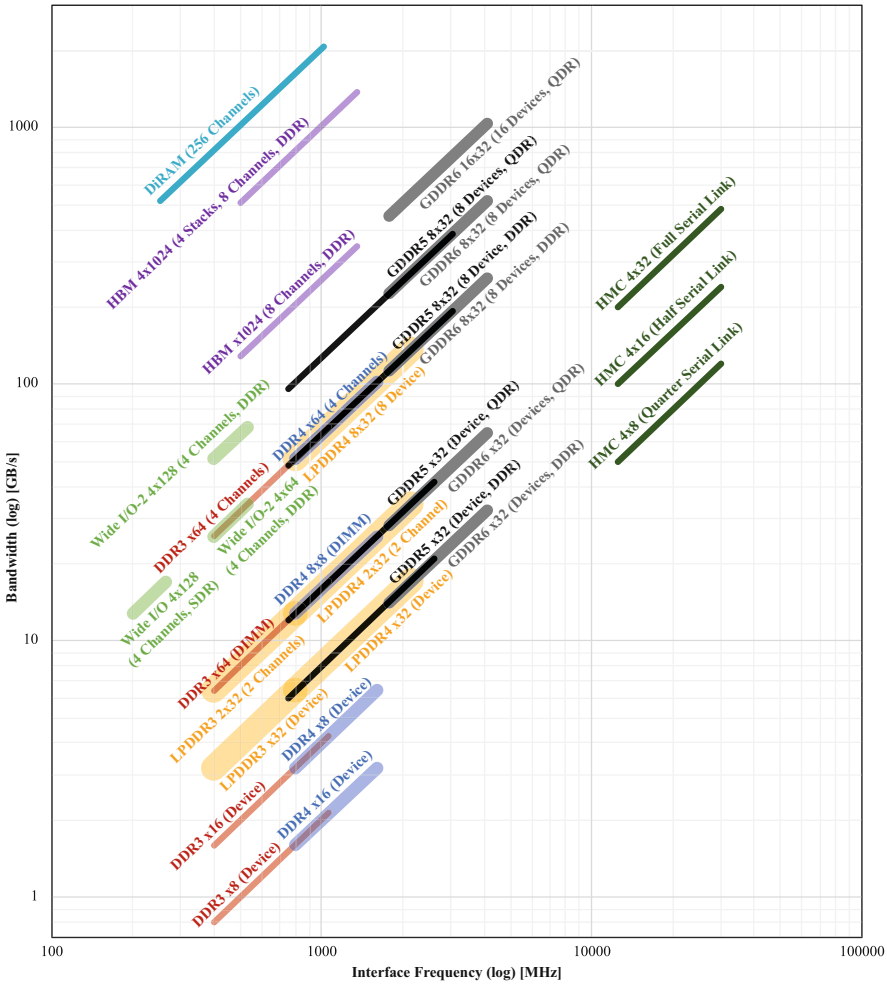

Fig. 3.2 Interface frequency and maximum bandwidth of different DRAM types

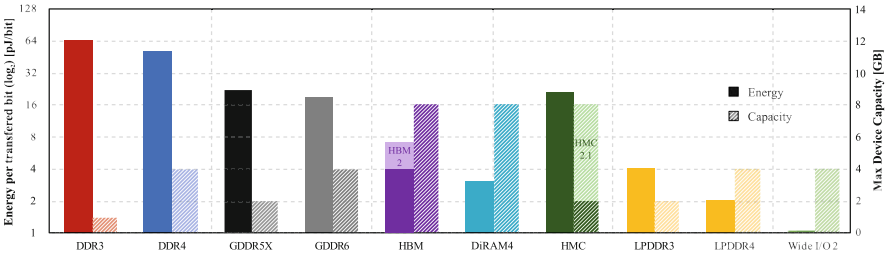

Fig. 3.3 Properties of today's DRAMs (Sources: Micron, Hynix, Nvidia, Xilinx, JEDEC)

To avoid pin limitations, designers and vendors are using *Buffer on Board* (BoB) organizations [7], 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 server applications where several terabytes of DRAM are required. The required storage capacity in embedded systems is much smaller than in the high-performance systems BoB targets, and thus this organization is inappropriate. All the other following DRAM devices can achieve easily several GB capacity, which is enough for most of the embedded applications.

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. For example, if eight LPDDR4 devices are used on a PCB, they 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 $\times 16/\times 32$ 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 Fig. 3.2.

Another way of achieving high bandwidth is 3D stacking: examples include WIDE I/O, Micron's *Hybrid Memory Cube* (HMC), and Samsung's *High Bandwidth Memory* (HBM). 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.

The combination of high bandwidth communication and the lower power consumption of 3D integrated memory is an ideal fit for embedded systems. For example, four parallel HBM2 devices on a 2.5D silicon interposer can provide up to 1 TB/s [16]. However, 3D memories suffer from thermal issues, which we discuss in Sect. 3.4.

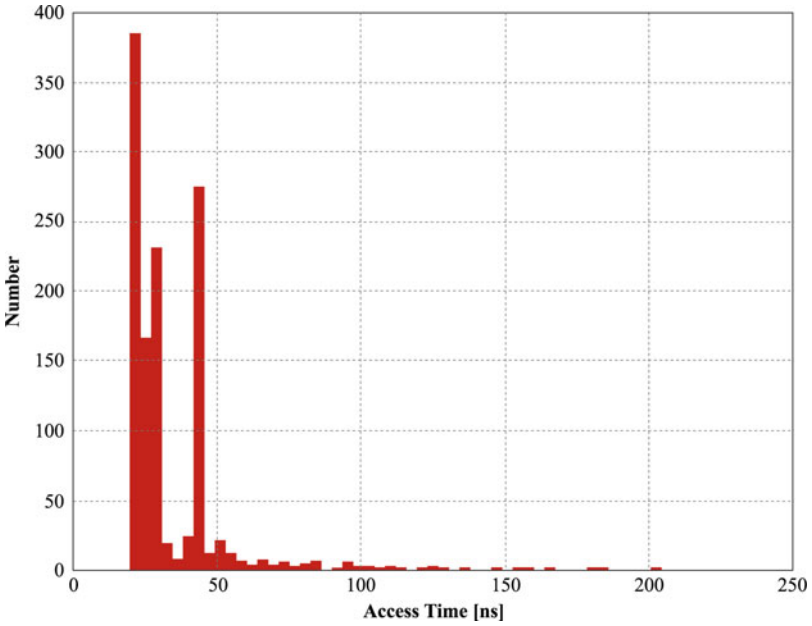

Fig. 3.4 Latency for an application running on DDR3 DRAM

From an application point of view, the DRAM subsystem has non-deterministic timing behavior [8] due to its complex protocol (i.e., the latency of a DRAM request depends on previous issued commands) and the runtime optimization of the memory controller; this makes it difficult to provide predictable performance and thus to guarantee real-time task predictability [1]. Figure 3.4 shows a histogram of the CHStone ADPCM benchmark [11] simulated on the DRAMSys framework [18].[2] 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 above, the memory controller plays an integral role in this non-deterministic timing behavior. The memory controller has to manage, on one side, accesses to the DRAM memory from the compute fabric and, on the other side, the complex interface protocol of the DRAMs. In the following we discuss the main contributions to the DRAM latency that origin from the complex internal memory architecture and the memory controller.

- **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

[2]The simulated DRAM is a DDR3 with a RBC address mapping and disabled scheduler.

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 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 architecture. The memory controller has to issue this refresh operation periodically (e.g., every 64 ms). Normal accesses to the DRAM have to be 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 consumption.[3] If a memory access arrives at the same time that a refresh happens it will experience unpredictable latency.
- **Scheduling:** COTS memory controllers are optimized for average case performance and therefore employ runtime scheduling of requests (c.f. Sect. 3.2) 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 compute 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 other hand if the command bus utilization is 100%, WR and RD commands must be issued in later cycles that negatively impacts the bandwidth and the latency.
- **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 also impact the predictability.

[3] In fact, the degradation grows linearly with the capacity, which means it grows exponentially with each density generation. Liu et al. [27] and Bhati et al. [3] predicted that 40–50% of the power consumption of future DRAM devices will be caused by refresh commands, and the maximum DRAM bandwidth will be significantly reduced.

Due to this unpredictable timing behavior, processors for embedded applications with real-time and strict latency constraints have thus far largely avoided using DRAM. For example, Infineon's Aurix CPU, which is widely used for safety-critical applications, does not provide a DRAM controller.

In past years there were many investigations with respect to DRAM controllers for real-time and mixed-criticality applications in embedded systems. A detailed book which summarizes those approaches has been presented by Goossens et al. [8]. Most of these approaches concentrate operating the DRAM with statically pre-computed command patterns which guarantee a predictable behavior. However, this predictability often comes with a degradation of sustainable bandwidth. Moreover, the bandwidth numbers presented in Fig. 3.2 are theoretical maxima: the sustainable memory bandwidth is much less, and it strongly depends on how the data is stored in the memories, i.e., the memory access pattern [12]. Therefore, it is 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 closer to the theoretical maximum.

As already mentioned, general-purpose DRAM controllers use online scheduling techniques 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 embedded systems, many applications (e.g., signal, image, or neural network processing) have regular, fixed, and deterministic memory access patterns. On the compute side, inherent application-specific knowledge has been heavily exploited for efficient compute architectures. However, on the memory side there is limited research that exploits application knowledge to improve the memory access behavior. In [12] we presented an *application-specific memory controller* (ASMC). Key of this controller is an optimized mapping of the logical addresses to physical DRAM addresses such that the row misses in the access pattern stream are minimized. The corresponding mathematical optimization problem is an *integer linear programming* problem. The solution of this problem maximizes the number of row buffer hits and exploits the bank-level parallelism of the DRAM device in order to reduce the latency and therefore to keep up the sustainable bandwidth near to the maximum. Therefore, such an ASMC can outperform online schedulers because it was designed with a *global* application view. Furthermore, for real-time embedded systems with this method we can easily determine WCET bounds, since no non-deterministic online scheduling is involved.

The efficiency of this approach is demonstrated on an industrial embedded image processing application that consists of image rotation and FFT. Due to real-time requirements this application requires a minimum bandwidth of 9.57 GB/s. Figure 3.5 shows the bandwidth and energy for the standard address mappings of a standard memory controller with standard row-bank-column (RBC) mapping and bank-row-column (BRC) mapping, a manual optimization of the mapping of an

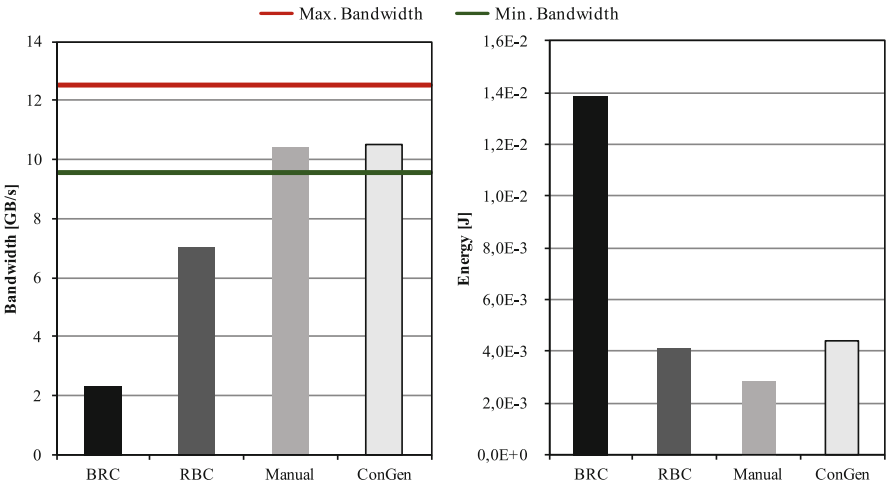

Fig. 3.5 Industrial image processing application

experienced engineer and the ASMC approach. The ASMC approach has a runtime of ∼50 min, whereas the manual approach requires ∼1 week for an engineer to fully understand the application and analyze the behavior. Furthermore, by using the generated address mapping, all the online scheduling capabilities of the memory controller could be removed, which reduced the required area of the memory controller by 35%.

As mentioned already in Sect. 3.2 the refresh has a large impact on DRAM's bandwidth and latency. The overhead of refreshes can be reduced by only refreshing the memory cells inside the DRAM that hold data that are still alive. A large body of research exists developing schemes that manually refresh the DRAM row-by-row, characterizing each row's ability to retain data and eliminating unnecessary *Refresh* operations on rows that can be refreshed less often. These schemes have been shown to be extremely efficient. Since eliminating refresh improves both energy and performance of the memory system, these schemes offer the potential for significant gains in DRAM-system efficiency. However, these schemes are incompatible with the modern *auto-refresh* mechanism that is widely used: *auto-refresh* operates on multiple rows at once and not on a row-by-row basis. In addition, *auto-refresh* cannot skip any row, whether that row needs to be refreshed or not. Thus, the manual schemes use explicit row-level *Activate* (`ACT`) and *Precharge* (`PRE`) commands to refresh row-by-row, called *row granular refresh* (RGR). However, it was shown in [4] that techniques based on RGR could never be as effective as the DRAM's internal auto-refresh.

In [28] we presented a technique called optimized RGR which allows a row-by-row refresh with the same efficiency as the auto-refresh. Here, we investigated the timings that are relevant to *Activate* and *Precharge* commands and showed that these DRAM timing parameters can be reduced for performing the *Refresh*

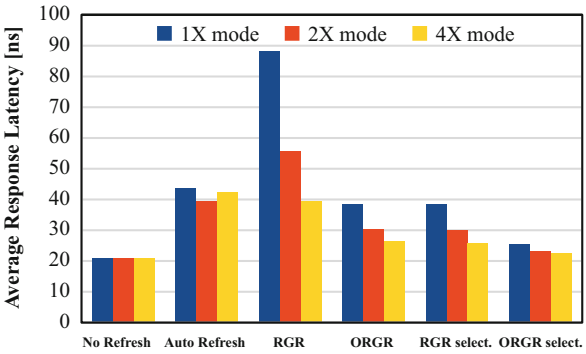

Fig. 3.6 Average response latency using different refresh techniques and modes according to JEDEC: 1X—all rows are refreshed per *Refresh* command, 2X—half of the rows are refreshed, 4X—a quarter of the rows are refreshed

operation row-by-row. We could demonstrate a reduction of latency and increase of bandwidth compared to standard *auto-refresh*, as shown in Fig. 3.6. The results can be even improved if only alive data is refreshed (ORGR select). Additionally, ORGR improved the energy efficiency compared to RGR.

It is becoming clear that embedded applications must concentrate on DRAM solutions like GDDR and HBM in combination with ASMCs and sophisticated refresh mechanisms in order to cope with their high bandwidth and low latency requirements.

3.3 Power Consumption

Power is one of the major challenges in today's embedded system development. According to Fig. 3.3, the preliminary choice for low power designs is LPDDR4 and Wide I/O2 due to their very low energy consumption. However, when aiming at high memory bandwidth, e.g., 1 TB/s these devices are not optimal. For example, to achieve the aforementioned bandwidth with LPDDR4, 64 devices ($\times 32$) are required. Although the average power would be only $\sim$17 W at a peak frequency of 2000 MHz, the high number of resulting I/O pins (2048) becomes unfeasible. Hence, the only alternative candidates for high bandwidth are HBM2 and GDDR5X/6. According to Figs. 3.2 and 3.3 the average power consumed[4] by the HBM (4 stack $\times 1024$) and GDDR6 (16 devices, QDR, $\times 32$) devices are $\sim$60 W and $\sim$150 W, respectively. These numbers show that DRAM will be a significant power contributor to embedded systems which require a high memory bandwidth. Therefore, it is mandatory to efficiently use DRAM's power-down modes in order to reduce power consumption.

In state-of-the-art memory controllers the entry to a power-down mode is scheduled when there was no activity in a period of time called timeout. DRAMs

[4]Operated at respective peak frequency.

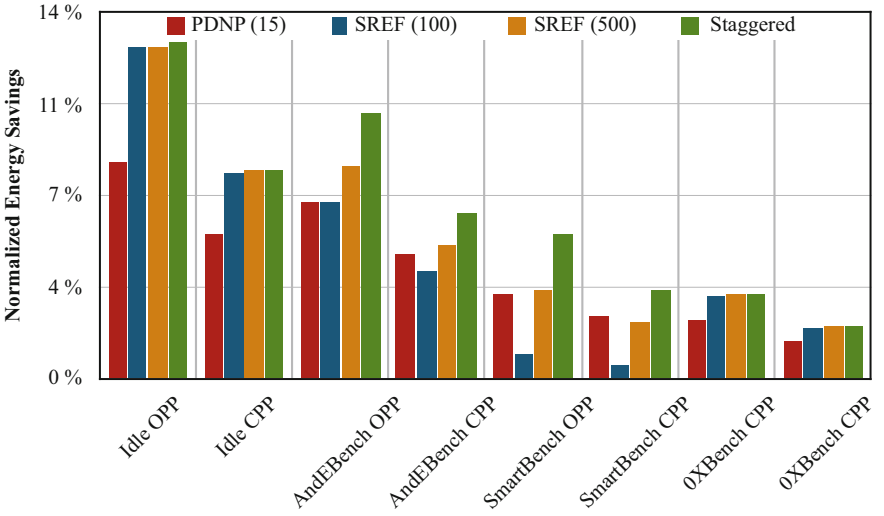

Fig. 3.7 Comparison of energy savings normalized to power-down disabled

offer three power-down modes, called *active power-down* (PDNA), *precharge power-down* (PDNP), and *self-refresh* (SREF). In [35] we showed that a highly opportunistic SREF entry results in an increased power consumption, since the SREF will always execute an internal refresh in the beginning. Therefore, the timeout for a SREF entry should be at least 500 clock cycles for a Wide I/O DRAM.

In [19] we presented an optimized power-down policy, called *staggered power-down*, which considers all three available DRAM power-down modes to achieve the maximum saving in energy and the minimum in slow-down on the execution of the applications. The basic idea is to change to the next more efficient power-down state on a refresh event. With this method, unnecessary SREF entries will be avoided and the hardware timeout counters, as used in state-of-the-art controllers, are not required anymore. As shown in Fig. 3.7 for Wide I/O DRAMs an energy reduction up to 10% in high activity periods and up to 13% in idle phases is feasible.

A high power consumption also fosters a high thermal dissipation that largely impacts the reliability of a DRAM. This challenge is discussed in more detail in the next paragraph.

3.4 Temperature vs. Reliability

DRAMs are very sensitive to high temperature, which increases the leakage in the memory cells. Figure 3.8 shows the different leakage paths in a DRAM cell:

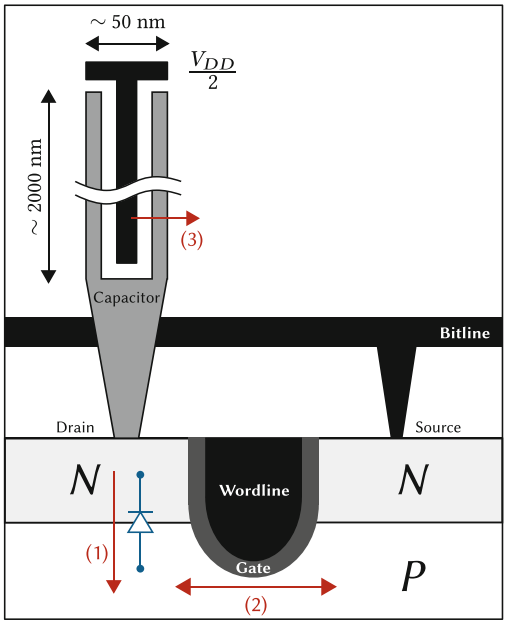

Fig. 3.8 Leakage paths in modern buried wordline DRAM architecture [23, 34]

- **Drain Leakage (1)**, which includes the P-N junction leakage as well as *gate induced drain leakage* (GIDL). GIDL is mainly caused by *trap assisted tunneling* (TAT), and it is influenced by the number and distribution of traps in the band-gap region as well as the electric field. Since the negative wordline voltage and the positive charge stored in the cell capacitor (when a `1` is stored in the cell) increase the electric field in the band-gap region (gate-drain overlap region), GIDL is the major source of leakage for a stored `1` in the DRAM cell [31].
- **Sub-threshold Leakage (2)**, which is the drain-source leakage of the cell transistor when it is in the OFF state. This current depends on various factors such as negative wordline voltage, bulk voltage, etc. When the bitlines are in precharged state ($V_{DD}/2$) this can slightly charge the cell capacitor and therefore cause the degradation of a `0` stored in the cell. It can also degrade a `1` stored in the cell by discharging to the bitline, but the leakage will be very small due to the increased threshold voltage of the access transistor when a `1` is stored (body-bias effect).
- **Cell Capacitor Leakage (3)**, which is the leakage through the cell capacitor dielectric. With the technology scaling, also the capacitor area is decreasing. Therefore, to maintain the cell capacitance at the previous value, dielectric thickness has to be reduced, which increases the leakage. The use of new *metal insulator metal* (MIM) structure with high-k dielectric materials has helped to reduce this leakage. Capacitor leakage influences both stored `0`'s and stored `1`'s.

In order to avoid data corruption by retention errors due to leakage, the refresh frequency needs to be increased. The general rule of thumb is to double the refresh

rate for every 10 °C increase over 85 °C [20]. For example, 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 [16].

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. 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. Besides the leakage currents, crosstalk on bitlines and wordlines can also disturb the data stored in the cells or disturb their sensing. Due to the aforementioned effects and shrinking technology nodes, reliability is a major concern in DRAMs. Many techniques exist to improve the reliability, e.g., using *error correcting codes* (ECC) and/or spatial redundancy.

Approximate and *probabilistic computing* evolved as design paradigms that exploit the error resilience of applications to increase their performance and decrease the power consumption [10]. This paradigm can be extended to DRAMs resulting in *approximate DRAMs* (ADRAM) that enable a trade-off between energy efficiency, performance, and reliability. The inherent error resilience of applications allows sacrificing data storage robustness and stability by lowering the refresh rate or disabling refresh in DRAMs completely, as shown in Fig. 3.9. However, to apply ADRAM the statistical DRAM behavior with respect to retention time, process variation, and temperature has to be characterized.

Several studies for the usage of ADRAM are presented in [13–15, 20]. One scenario is safe refresh disabling, i.e., if the data lifetime is smaller than the refresh period, the refresh can be completely switched off without impact on the system

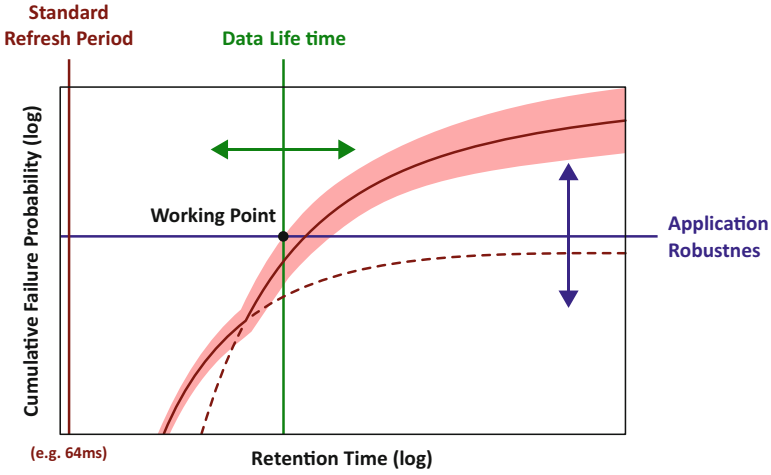

Fig. 3.9 Approximate DRAM design space

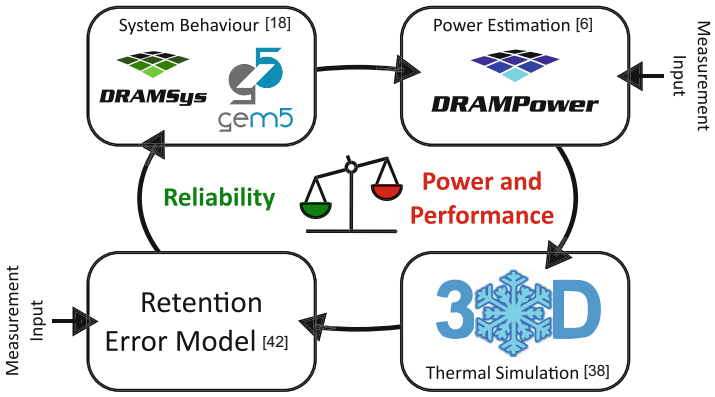

Fig. 3.10 Simulation framework for approximate DRAM explorations

behavior. To perform an accurate characterization we measured state-of-the-art DRAM devices, such as DDR3, DDR4, and Wide I/O. These measurements were the base for a simulation platform for ADRAM investigations.

Exploring ADRAM in a system context is challenging, since a trade-off between accuracy and simulation performance must be considered. Our framework relies on SystemC *Transaction Level Models* (TLM) for fast and accurate simulation. Figure 3.10 shows the closed loop simulation. This simulation loop consists of (1) *DRAM and gem5 Core Models* [5, 18], (2) a DRAM power model [6, 29], which uses either parameters from datasheets, or real measurements [13, 15], (3) a *thermal model* based on 3D-ICE [38], and (4) a *DRAM retention error model* [42].

As mentioned before, ECC is an efficient technique to improve DRAM's reliability, e.g., retention errors or errors induced by crosstalk. State-of-the-art ECC DDR DIMMs, for instance, consist of 8 DRAM devices and a further device for storing the ECC redundancy. Moreover, vendors recently introduced on-die ECC for LPDDR4 [24, 26] to correct retention errors. With ECC the refresh rate can be lowered by 4×, which largely reduces the power consumption. Finding an efficient ECC is a non-trivial task. Traditional ECC techniques for DRAMs assume a symmetric behavior of the retention errors, i.e., the error probability for a stored 0 and 1 is identical. In [22] and [23] we presented a more accurate error model for the retention behavior that exploits the internal cell structure (the so-called true- or anti-cells) of a DRAM. This model is asymmetric and we could show that the channel capacity according to Shannon's capacity definition (the memory cell is considered as a noisy channel) of a single memory cell is larger than in the traditional commonly used symmetrical model. Hence, a more efficient coding must exist. In [23] we presented a new and low-overhead coding scheme that improves the reliability with respect to retention errors.

3.5 Safety and Security

Since DRAMs are more and more used in safety-critical applications like automotive, safety and security are major concerns for DRAMs that were originally mainly developed for consumer applications. Apart from the temperature based retention errors discussed in Sect. 3.4, DRAMs are also prone to transient soft errors, i.e., effects of cosmic particle strikes [9]. Moreover, due to the high frequency of DRAM interfaces transient transmission errors on the DRAM bus can occur. Furthermore, there can be hard errors related to stuck at failures or aging, which could result in 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 [30, 36, 41]. Sridharan et al. report 20–66 FIT for a single DRAM device [39, 40]. This highlights the need for appropriate safety mechanisms in order to decrease the FIT rates. For example, the memory controller contains an additional logic that 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.

Apart from random failures, malicious causes can lead to a safety goal violation, too. Because of transitions to open environments for IoT or *Car2X* communication the vulnerability of DRAMs for embedded systems 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 [21, 32] and its exploits [25, 37] 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. In [25] the author showed that secret data can be read with a combination of row-hammer and data dependencies [23]. The row-hammer security attack [21] is a potential malicious behavior that has to be avoided. Controller triggered techniques like *target row refresh* where rows will be refreshed when their activation count exceeds a threshold or techniques on the device level like [2, 44] can alleviate this problem. In [17] a methodology for reverse engineering DRAMs by reconstructing the physical location of memory cells without opening the device package and microscoping the device was presented. This method consists of a retention error analysis while a temperature gradient is applied to the DRAM device. With this insight into the internal DRAM structure row-hammer countermeasure techniques can be improved.

3.6 Conclusion

Emerging applications executed on embedded computing systems require ever increasing main memory sizes. Thus, DRAMs are indispensable to be integrated in such systems. However, the use of DRAMs implies many new challenges.

In this chapter, we highlighted some of the major challenges for the integration of DRAM subsystems into embedded computing systems. These challenges are namely: *bandwidth*, *latency*, *power*, *temperature*, *reliability*, *safety*, and *security*. Furthermore, we showed several solutions from our recent research activities in order to tackle and overcome these challenges.

References

1. A. Agrawal, G. Fohler, 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, 2017), pp. 265–267. https://doi.org/10.1145/3132402.3132417
2. A. Amaya, H. Gomez, E. Roa, Mitigating Row Hammer attacks based on dummy cells in DRAM, in *2017 IEEE International Conference on Consumer Electronics (ICCE)* (2017), pp. 442–443. https://doi.org/10.1109/ICCE.2017.7889389
3. I. Bhati, M.T. Chang, Z. Chishti, S.L. Lu, B. Jacob, DRAM refresh mechanisms, trade-offs, and penalties. IEEE Trans. Comput. **PP**(99), 1 (2015). https://doi.org/10.1109/TC.2015.2417540
4. I. Bhati, M.T. Chang, Z. Chishti, S.L. Lu, B. Jacob, DRAM refresh mechanisms, penalties, and trade-offs. IEEE Trans. Comput. **65**(1), 108–121 (2016). https://doi.org/10.1109/TC.2015.2417540
5. N. Binkert, B. Beckmann, G. Black, S.K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D.R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M.D. Hill, D.A. Wood, The gem5 simulator. SIGARCH Comput. Archit. News **39**(2), 1–7 (2011). https://doi.org/10.1145/2024716.2024718
6. K. Chandrasekar, C. Weis, Y. Li, B. Akesson, O. Naji, M. Jung, N. Wehn, K. Goossens, DRAMPower: open-source DRAM power and energy estimation tool (2014). http://www.drampower.info
7. E. Cooper-Balis, P. Rosenfeld, B. Jacob, Buffer-on-board memory systems, in *2012 39th Annual International Symposium on Computer Architecture (ISCA)* (2012), pp. 392–403. https://doi.org/10.1109/ISCA.2012.6237034
8. S. Goossens, K. Chandrasekar, B. Akesson, K. Goossens, Memory controllers for mixed-time-criticality systems: architectures, methodologies and trade-offs, in *Embedded Systems* (Springer, Berlin, 2016). https://books.google.de/books?id=l9_7CwAAQBAJ
9. M. Greenberg, Understanding automotive DDR DRAM (2017). https://www.synopsys.com/designware-ip/technical-bulletin/automotive-ddr-dram.html
10. J. Han, M. Orshansky, Approximate computing: an emerging paradigm for energy-efficient design, in *2013 18th IEEE European Test Symposium (ETS)* (2013), pp. 1–6. https://doi.org/10.1109/ETS.2013.6569370
11. Y. Hara, H. Tomiyama, S. Honda, H. Takada, Proposal and quantitative analysis of the CHStone benchmark program suite for practical c-based high-level synthesis. J. Inf. Process. **17**, 242–254 (2009). https://doi.org/10.2197/ipsjjip.17.242
12. M. Jung, I. Heinrich, M. Natale, D.M. Mathew, C. Weis, S. Krumke, N. Wehn, ConGen: an application specific dram memory controller generator, in *Proceedings of the Second International Symposium on Memory Systems, MEMSYS '16* (ACM, New York, 2016), pp. 257–267. https://doi.org/10.1145/2989081.2989131
13. M. Jung, D. Mathew, C. Rheinländer, C. Weis, N. Wehn, A platform to analyze DDR3 DRAM's power and retention time. IEEE Design Test **34**(4), 52–59 (2017). https://doi.org/10.1109/MDAT.2017.2705144

14. M. Jung, D. Mathew, C. Weis, N. Wehn, Approximate computing with partially unreliable dynamic random access memory—approximate DRAM, in *Proceedings of the 53rd Annual Design Automation Conference, DAC '16* (ACM, New York, 2016), pp. 100:1–100:4. https://doi.org/10.1145/2897937.2905002
15. M. Jung, D.M. Mathew, C. Weis, N. Wehn, Efficient reliability management in SoCs—an approximate DRAM perspective, in *21st Asia and South Pacific Design Automation Conference (ASP-DAC)* (2016)
16. M. Jung, S.A. McKee, C. Sudarshan, C. Dropmann, C. Weis, N. Wehn, 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, MEMSYS '18* (ACM, New York, 2018), pp. 377–386. https://doi.org/10.1145/3240302.3240322
17. M. Jung, C. Rheinländer, C. Weis, N. Wehn, Reverse engineering of DRAMs: Row Hammer with Crosshair, in *International Symposium on Memory Systems (MEMSYS 2016)* (2016)
18. M. Jung, C. Weis, N. Wehn, DRAMSys: a flexible DRAM subsystem design space exploration framework. IPSJ Trans. Syst. LSI Design Methodol. **8**, 63–74 (2015). https://doi.org/10.2197/ipsjtsldm.8.63
19. M. Jung, C. Weis, N. Wehn, M. Sadri, L. Benini, Optimized active and power-down mode refresh control in 3D-DRAMs, in *Proceedings of the 2014 22nd International Conference on Very Large Scale Integration (VLSI-SoC)* (2014), pp. 1–6. https://doi.org/10.1109/VLSI-SoC.2014.7004159
20. M. Jung, E. Zulian, D. Mathew, M. Herrmann, C. Brugger, C. Weis, N. Wehn, Omitting refresh—a case study for commodity and wide I/O DRAMs, in *Proceedings of the 1st International Symposium on Memory Systems (MEMSYS 2015)* (Washington, 2015)
21. Y. Kim, R. Daly, J.H. Kim, C. Fallin, J.H. Lee, D. Lee, C. Wilkerson, K. Lai, O. Mutlu, Flipping bits in memory without accessing them: an experimental study of DRAM disturbance errors, in *ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)* (2014), pp. 361–372. https://doi.org/10.1109/ISCA.2014.6853210
22. K. Kraft, M. Jung, C. Sudarshan, D.M. Mathew, C. Weis, N. Wehn, Improving the error behavior of DRAM by exploiting its Z-channel property, in *IEEE Conference Design, Automation and Test in Europe (DATE)* (2018)
23. K. Kraft, D.M. Mathew, C. Sudarshan, M. Jung, C. Weis, N. Wehn, F. Longnos, Efficient coding scheme for DDR4 memory subsystems, in *ACM International Symposium on Memory Systems (MEMSYS 2018)* (2018)
24. 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. Park, Y.S. Park, H.J. Kwon, S.J. Bae, J.H. Choi, K.I. Park, S.J. Jang, G.Y. Jin, 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)* (2017), pp. 394–395. https://doi.org/10.1109/ISSCC.2017.7870427
25. A. Kwong, D. Genkin, D. Gruss, Y. Yarom, RAMBleed: reading bits in memory without accessing them, in *Proceedings of the 41st Annual IEEE Symposium on Security and Privacy* (2020)
26. 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, G.Y. Jin, 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)* (2017), pp. 390–391. https://doi.org/10.1109/ISSCC.2017.7870425
27. J. Liu, B. Jaiyen, R. Veras, O. Mutlu, RAIDR: retention-aware intelligent DRAM refresh, in *Proceedings of the 39th Annual International Symposium on Computer Architecture, ISCA '12* (IEEE Computer Society, Washington, 2012), pp. 1–12. http://dl.acm.org/citation.cfm?id=2337159.2337161

28. D.M. Mathew, d.F. Zulian, M. Jung, K. Kraft, C. Weis, B. Jacob, N. Wehn, Using run-time reverse-engineering to optimize DRAM refresh, in *International Symposium on Memory Systems (MEMSYS17)* (2017)
29. D.M. Mathew, E.F. Zulian, S. Kannoth, M. Jung, C. Weis, N. Wehn, A Bank-Wise DRAM power model for system simulations, in *Proceedings of the 9th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools, RAPIDO '17* (ACM, New York, 2017), pp. 5:1–5:7. https://doi.org/10.1145/3023973.3023978
30. J. Meza, Q. Wu, S. Kumar, O. Mutlu, 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)* (2015)
31. N.J. Min Hee Cho, An innovative indicator to evaluate DRAM cell transistor leakage current distribution. J. Electron Devices Soc. **6**, 494–499 (2017)
32. O. Mutlu, The Row–Hammer problem and other issues we may face as memory becomes denser, in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pp. 1116–1121 (2017). https://doi.org/10.23919/DATE.2017.7927156
33. D.A. Patterson, Latency lags bandwith. Commun. ACM **47**(10), 71–75 (2004). https://doi.org/10.1145/1022594.1022596
34. T. Schloesser, $6F^2$ buried wordline DRAM cell for 40 nm and beyond, in *IEEE International Electron Devices Meeting* (San Francisco, 2008)
35. D. Schmidt, N. Wehn, DRAM power management and energy consumption: a critical assessment, in *Proceedings of the 22nd Annual Symposium on Integrated Circuits and System Design* (Natal, 2009)
36. B. Schroeder, E. Pinheiro, W.D. Weber, DRAM errors in the wild: a large-scale field study. ACM SIGMETRICS Perform. Eval. Rev. **37**(1), 193–204 (2009)
37. M. Seaborn, T. Dullien, Exploiting the DRAM Row–Hammer bug to gain kernel privileges (2015). http://googleprojectzero.blogspot.de/2015/03/exploiting-dram-rowhammer-bug-to-gain.html
38. A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, D. Atienza, 3D-ICE: fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling, in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2010)
39. V. Sridharan, N. DeBardeleben, S. Blanchard, K.B. Ferreira, J. Stearley, J. Shalf, S. Gurumurthi, Memory errors in modern systems: the good, the bad, and the ugly. SIGARCH Comput. Archit. News **43**(1), 297–310 (2015). https://doi.org/10.1145/2786763.2694348
40. V. Sridharan, D. Liberty, A study of DRAM failures in the field, in *2012 International Conference on High Performance Computing, Networking, Storage and Analysis (SC)* (2012), pp. 1–11. https://doi.org/10.1109/SC.2012.13
41. I. Stefanovici, A. Hwang, B. Schroeder, DRAM's Damning defects—and how they cripple computers. IEEE Spectrum (2015)
42. C. Weis, M. Jung, P. Ehses, C. Santos, P. Vivet, S. Goossens, M. Koedam, N. Wehn, Retention time measurements and modelling of bit error rates of WIDE I/O DRAM in MPSoCs, in *Proceedings of the IEEE Conference on Design, Automation and Test in Europe (DATE)* (European Design and Automation Association, 2015)
43. W.A. Wulf, S.A. McKee, Hitting the memory wall: implications of the obvious. SIGARCH Comput. Archit. News (1995). https://doi.org/10.1145/216585.216588
44. C.M. Yang, C.K. Wei, Y.J. Chang, T.C. Wu, H.P. Chen, C.S. Lai, Suppression of Row Hammer effect by doping profile modification in Saddle-Fin array devices for sub-30-nm DRAM technology. IEEE Trans. Device Mater. Reliab. **16**(4), 685–687 (2016). https://doi.org/10.1109/TDMR.2016.2607174