

This is a summary of the original paper, entitled “Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture” which appears in HPCA 2013 [37].

Tiered-Latency DRAM (TL-DRAM)

Donghyuk Lee Yoongu Kim Vivek Seshadri Jamie Liu Lavanya Subramanian Onur Mutlu
Carnegie Mellon University

Abstract

This paper summarizes the idea of Tiered-Latency DRAM, which was published in HPCA 2013 [37]. The key goal of TL-DRAM is to provide low DRAM latency at low cost, a critical problem in modern memory systems [55]. To this end, TL-DRAM introduces heterogeneity into the design of a DRAM subarray by segmenting the bitlines, thereby creating a low-latency, low-energy, low-capacity portion in the subarray (called the near segment), which is close to the sense amplifiers, and a high-latency, high-energy, high-capacity portion, which is farther away from the sense amplifiers. Thus, DRAM becomes heterogeneous with a small portion having lower latency and a large portion having higher latency. Various techniques can be employed to take advantage of the low-latency near segment and this new heterogeneous DRAM substrate, including hardware-based caching and software based caching and memory allocation of frequently used data in the near segment. Evaluations with simple such techniques show significant performance and energy-efficiency benefits [37].

1 Summary

1.1 The Problem: High DRAM Latency

Primarily due to its low cost-per-bit, DRAM has long been the choice substrate for architecting main memory subsystems. In fact, DRAM’s cost-per-bit has been decreasing at a rapid rate as DRAM process technology scales to integrate ever more DRAM cells into the same die area. As a result, each successive generation of DRAM has enabled increasingly large-capacity main memory subsystems at low cost.

In stark contrast to the continued scaling of cost-per-bit, the latency of DRAM has remained almost constant. During the same 11-year interval in which DRAM’s cost-per-bit decreased by a factor of 16, DRAM latency (as measured by the t_{RCD} and t_{RC} timing constraints) decreased by only 30.5% and 26.3% [5, 25], as shown in Figure 1 of our paper [37]. From the perspective of the processor, an access to DRAM takes hundreds of cycles — time during which the processor may be stalled, waiting for DRAM. Such wasted time leads to large performance degradations.

1.2 Key Observations and Our Goal

Bitline: Dominant Source of Latency. In DRAM, each bit is represented as electrical charge in a capacitor-based cell. The small size of this capacitor necessitates the use of an auxiliary structure, called a sense-amplifier, to detect the small amount of charge held by the cell and amplify it to a full digital logic value. But, a sense-amplifier is approximately one hundred times larger than a cell [61]. To amortize their large size, each

sense-amplifier is connected to many DRAM cells through a wire called a bitline.

Every bitline has an associated parasitic capacitance whose value is proportional to the length of the bitline. Unfortunately, such parasitic capacitance slows down DRAM operation for two reasons. First, it increases the latency of the sense-amplifiers. When the parasitic capacitance is large, a cell cannot quickly create a voltage perturbation on the bitline that could be easily detected by the sense-amplifier. Second, it increases the latency of charging and precharging the bitlines. Although the cell and the bitline must be restored to their quiescent voltages during and after an access to a cell, such a procedure takes much longer when the parasitic capacitance is large. Due to the above reasons and a detailed latency break-down (refer to our HPCA-19 paper [37]), we conclude that long bitlines are the dominant source of DRAM latency [22, 70, 51, 52].

Latency vs. Cost Trade-Off. The bitline length is a key design parameter that exposes the important trade-off between latency and die-size (cost). Short bitlines (few cells per bitline) constitute a small electrical load (parasitic capacitance), which leads to low latency. However, they require more sense-amplifiers for a given DRAM capacity (Figure 1a), which leads to a large die-size. In contrast, long bitlines have high latency and a small die-size (Figure 1b). As a result, neither of these two approaches can optimize for both latency and cost-per-bit.

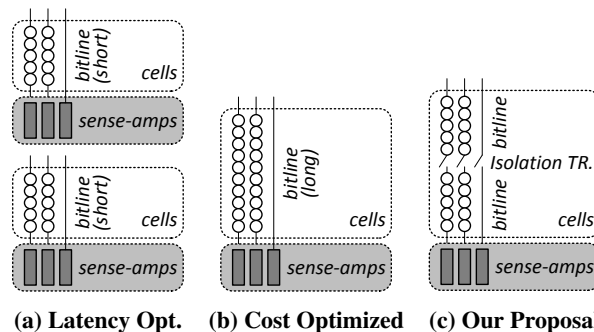


Figure 1. DRAM: Latency vs. Cost Optimized, Our Proposal

Figure 2 shows the trade-off between DRAM latency and die-size by plotting the latency (t_{RCD} and t_{RC}) and the die-size for different values of cells-per-bitline. Existing DRAM architectures are either optimized for die-size (commodity DDR3 [64, 50]) and are thus low cost but high latency, or optimized for latency (RLDRAM [49], FCRAM [65]) and are thus low latency but high cost.

The goal of our paper [37] is to design a new DRAM architecture to approximate the best of both worlds (i.e., low latency and low cost), based on the key observation that that long bitlines are the dominant source of DRAM latency.

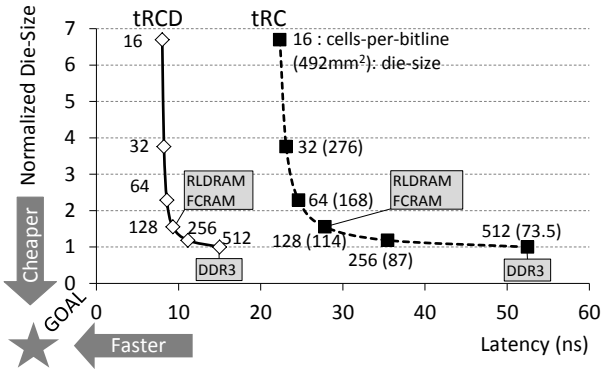


Figure 2. Bitline Length: Latency vs. Die-Size

1.3 Tiered-Latency DRAM

To achieve the latency advantage of short bitlines and the cost advantage of long bitlines, we propose the *Tiered-Latency DRAM* (TL-DRAM) architecture, which is shown in Figure 1c and 3a. The key idea of TL-DRAM is to divide the long bitline into two shorter segments using an *isolation transistor*: the *near segment* (connected directly to the sense-amplifier) and the *far segment* (connected through the isolation transistor).

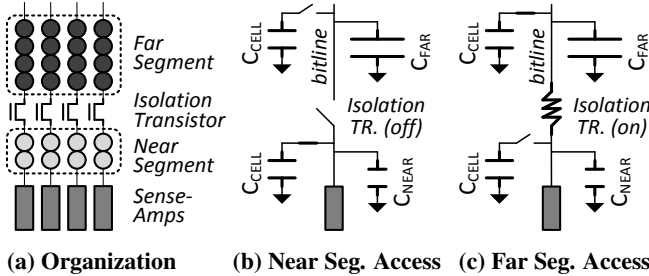


Figure 3. TL-DRAM: Near vs. Far Segments

The primary role of the isolation transistor is to electrically decouple the two segments from each other. This changes the effective bitline length (and also the effective bitline capacitance) as seen by the cell and sense-amplifier. Correspondingly, the latency to access a cell is also changed — albeit differently depending on whether the cell is in the near or the far segment.

When accessing a cell in the near segment, the isolation transistor is turned off, disconnecting the far segment (Figure 3b). Since the cell and the sense-amplifier see only the reduced bitline capacitance of the shortened near segment, they can drive the bitline voltage more easily. As a result, the bitline voltage is restored more quickly, so that the latency (t_{RC}) for the near segment is significantly reduced. On the other hand, when accessing a cell in the far segment, the isolation transistor is turned on to connect the entire length of the bitline to the sense-amplifier. In this case, the isolation transistor acts like a resistor inserted between the two segments (Figure 3c) and limits how quickly charge flows to the far segment. Because the far segment capacitance is charged more slowly, it takes longer for the far segment voltage to be restored, so that the latency (t_{RC}) is increased for cells in the far segment.

Latency, Power, and Die-Area. Table 1 summarizes the latency, power, and die-area characteristics of TL-DRAM to other DRAMs, estimated using circuit-level SPICE simulation [56] and power/area models from Rambus [61]. Compared to commodity DRAM (long bitlines) which incurs high latency

(t_{RC}) for all cells, TL-DRAM offers significantly reduced latency (t_{RC}) for cells in the near segment, while increasing the latency for cells in the far segment due to the additional resistance of the isolation transistor. In DRAM, a large fraction of the power is consumed by the bitlines. Since the near segment in TL-DRAM has a lower capacitance, it also consumes less power. On the other hand, accessing the far segment requires toggling the isolation transistors, leading to increased power consumption. Mainly due to additional isolation transistors, TL-DRAM increases die-area by 3%. Our paper includes detailed circuit-level analyses of TL-DRAM (Section 4 of [37]).

	Short Bitline (Fig 1a)	Long Bitline (Fig 1b)	Segmented Bitline (Fig 1c)	
	Unsegmented	Unsegmented	Near	Far
Length (Cells)	32	512	32	480
Latency (t_{RC})	Low (23.1ns)	High (52.5ns)	Low (23.1ns)	Higher (65.8ns)
Normalized Power Consump.	Low (0.51)	High (1.00)	Low (0.51)	Higher (1.49)
Normalized Die-Size (Cost)	High (3.76)	Lower (1.00)	Low (1.03)	

Table 1. Latency, Power, and Die-Area Comparison

1.4 Leveraging TL-DRAM

TL-DRAM enables the design of many new memory management policies that exploit the asymmetric latency characteristics of the near and the far segments. Our HPCA-19 paper (in Section 5) describes four ways of taking advantage of TL-DRAM. Here, we describe two approaches in particular.

In the first approach, the memory controller uses the near segment as a *hardware-managed cache* for the far segment. In our HPCA-19 paper [37], we discuss three policies for managing the near segment cache. (The three policies differ in deciding when a row in the far segment is cached into the near segment and when it is evicted.) In addition, we propose a new data transfer mechanism (*Inter-Segment Data Transfer*) that efficiently migrates data between the segments by taking advantage of the fact that the bitline is a bus connected to the cells in both segments. By using this technique, the data from the source row can be transferred to the destination row over the bitlines at very low latency (additional 4ns over t_{RC}). Furthermore, this Inter-Segment Data Transfer happens exclusively within DRAM bank without utilizing the DRAM channel, allowing concurrent accesses to other banks.

In the second approach, the near segment capacity is exposed to the OS, enabling the OS to use the full DRAM capacity. We propose two concrete mechanisms, one where the memory controller uses an additional layer of indirection to map frequently accessed pages to the near segment, and another where the OS uses static/dynamic profiling to directly map frequently accessed pages to the near segment. In both approaches, the accesses to pages that are mapped to the near segment are served faster and with lower power than in conventional DRAM, resulting in improved system performance and energy efficiency.

1.5 Results: Performance and Power

Our HPCA-19 paper [37] provides extensive detail about both of the above approaches. But, due to space constraints, we present the evaluation results of only the first approach, in which the near segment is used as hardware-managed cache managed under our best policy (*Benefit-Based Caching*) to show the advantage of our TL-DRAM substrate.

Performance & Power Analysis. Figure 4 shows the average performance improvement and power-efficiency of our proposed mechanism over the baseline with conventional DRAM, on 1-, 2- and 4-core systems. As described in Section 1.3, access latency and power consumption are significantly lower for near segment accesses, but higher for far segment accesses, compared to accesses in a conventional DRAM. We observe that a large fraction (over 90% on average) of requests hit in the rows cached in the near segment, thereby accessing the near segment with low latency and low power consumption. As a result, TL-DRAM achieves significant performance improvement by 12.8%/12.3%/11.0% and power savings by 23.6%/26.4%/28.6% in 1-/2-/4-core systems, respectively.

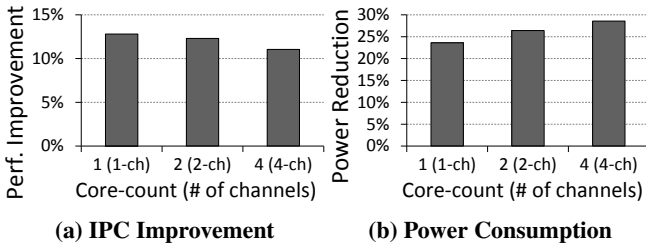


Figure 4. IPC Improvement & Power Consumption

Sensitivity to Near Segment Capacity. The number of rows in the near segment presents a trade-off, since increasing the near segment’s size increases its capacity but also increases its access latency. Figure 5 shows the performance improvement of our proposed mechanisms over the baseline as we vary the near segment size. Initially, performance improves as the number of rows in the near segment since more data can be cached. However, increasing the number of rows in the near segment beyond 32 reduces the performance benefit due to the increased capacitance.

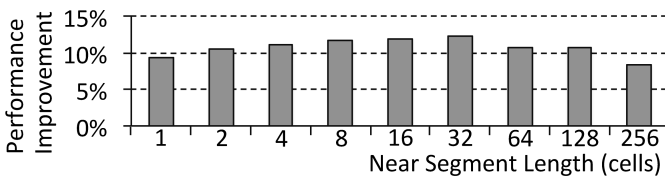


Figure 5. Effect of Varying Near Segment Capacity

Other Results. In our HPCA-19 paper, we provide a detailed analysis of how timing parameters and power consumption vary when varying the near segment length, in Section 4 and 6.3, respectively. We also provide a comprehensive evaluation of the mechanisms we build on top of the TL-DRAM substrate for single- and multi-core systems in Section 8.

All of our results are gathered using an in-house version of Ramulator [31], an open-source DRAM simulator [30], which is integrated into an in-house processor simulator.

2 Significance

2.1 Novelty

To our knowledge, our HPCA-19 paper is the first to enable latency heterogeneity in DRAM without significantly increasing cost-per-bit and to propose hardware/software mechanisms that leverage this latency heterogeneity to improve system performance. We make the following major contributions.

A Cost-Efficient Low-Latency DRAM. Based on the key observation that long internal wires (bitlines) are the dominant source of DRAM latency, we propose a new DRAM architecture called Tiered-Latency DRAM (TL-DRAM). To our knowledge this is the first work to enable low-latency DRAM without significantly increasing the cost-per-bit. By adding a single isolation transistor to each bitline, we carve out a region within a DRAM chip, called the near segment, that is fast and energy-efficient. This comes at a modest overhead of 3% increase in DRAM die-area. While there are two prior approaches to reduce DRAM latency (using short bitlines [49, 65], adding an SRAM cache in DRAM [20, 18, 16, 84]), both of these approaches significantly increase die-area due to additional sense-amplifiers or additional area for SRAM cache, as we evaluate in our paper [37]. Compared to these prior approaches, TL-DRAM is a much more cost-effective architecture for achieving low latency.

There are many works that reduce *overall memory access latency* by modifying DRAM, the DRAM-controller interface, and DRAM controllers. These works enable more parallelism and bandwidth [29, 10, 66, 40], reduce refresh counts [42, 43, 26, 79, 60], accelerate bulk operations [66, 68, 69, 11], accelerate computation in the logic layer of 3D-stacked DRAM [2, 1, 83, 17], enable better communication between CPU and other devices through DRAM [39], leverage process variation and temperature dependency in DRAM [38], leverage DRAM access patterns [19], reduce write-related latencies by better designing DRAM and DRAM control policies [13, 36, 67], and reduce overall queuing latencies in DRAM by better scheduling memory requests [53, 54, 27, 28, 75, 73, 21, 78]. Our proposal is orthogonal to all of these approaches and can be applied in conjunction with them to achieve higher latency and energy benefits.

Inter-Segment Data Transfer. By implementing latency heterogeneity within a DRAM subarray, TL-DRAM enables efficient data transfer between the fast and slow segments by utilizing the bitlines as a wide bus. This mechanism takes advantage of the fact that both the source and destination cells share the same bitlines. Furthermore, this inter-segment migration happens only within a DRAM bank and does not utilize the DRAM channel, thereby allowing concurrent accesses to other banks over the channel. This inter-segment data transfer enables fast and efficient movement of data within DRAM, which in turn enables efficient ways of taking advantage of latency heterogeneity.

Son et al. proposes a low latency DRAM architecture [71] that has fast (long bitline) and slow (short bitline) subarrays in DRAM. This approach provides largest benefit when allocating latency critical data to the low latency regions (the low latency subarrays). Therefore, overall memory system performance is sensitive to the page placement policy. However,

our inter-segment data transfer enables efficient relocation of pages, leading to dynamic page placement based on the latency criticality of each page.

2.2 Potential Long-Term Impact

Tolerating High DRAM Latency by Enabling New Layers in the Memory Hierarchy. Today, there is a large latency cliff between the on-chip last level cache and off-chip DRAM, leading to a large performance fall-off when applications start missing in the last level cache. By introducing an additional fast layer (the near segment) within the DRAM itself, TL-DRAM smoothens this latency cliff.

Note that many recent works added a DRAM cache or created heterogeneous main memories [33, 35, 59, 47, 81, 62, 57, 48, 44, 12, 63, 41, 14] to smooth the latency cliff between the last level cache and a longer-latency non-volatile main memory, e.g., Phase Change Memory [33, 35, 59], or to take advantage of the advantages of multiple different types of memories to optimize for multiple metrics. Our approach is similar at the high-level (i.e., to reduce the latency cliff at low cost by taking advantage of heterogeneity) yet we introduce the new low-latency layer within DRAM itself instead of adding a completely separate device.

Applicability to Future Memory Devices. We show the benefits of TL-DRAM's asymmetric latencies. Considering that most memory devices adopt a similar cell organization (i.e., a 2-dimensional cell array and row/column bus connections), our approach of reducing the electrical load of connecting to a bus (bitline) to achieve low access latency can be applicable to other memory devices.

Furthermore, the idea of performing inter-segment data transfer can also potentially be applied to other memory devices, regardless of the memory technology. For example, we believe it is promising to examine similar approaches for emerging memory technologies like Phase Change Memory [33, 59, 58, 46, 82, 34] or STT-MRAM [32, 80], as well as the NAND flash memory technology [45, 8, 9, 7, 6].

New Research Opportunities. The TL-DRAM substrate creates new opportunities by enabling mechanisms that can leverage the latency heterogeneity offered by the substrate. We briefly describe three directions, but we believe many new possibilities abound.

- *New ways of leveraging TL-DRAM.* TL-DRAM is a substrate that can be utilized for many applications. Although we describe two major ways of leveraging TL-DRAM in our HPCA-19 paper, we believe there are several more ways to leverage the TL-DRAM substrate both in hardware and software. For instance, new mechanisms could be devised to detect data that is latency critical (e.g., data that causes many threads to become serialized [15, 77, 23, 76, 24] or data that belongs to threads that are more latency-sensitive [27, 28, 72, 78, 3, 4, 73, 75, 74]) or could become latency critical in the near future and allocate/prefetch such data into the near segment.
- *Opening up new design spaces with multiple tiers.* TL-DRAM can be easily extended to have multiple latency tiers by adding more isolation transistors to the bitlines, providing more latency asymmetry. (Our HPCA-19 paper provides an analysis of the latency of a TL-DRAM design with three

tiers, showing the spread in latency for three tiers.) This enables new mechanisms both in hardware and software that can allocate data appropriately to different tiers based on their access characteristics such as locality, criticality, etc.

- *Inspiring new ways of architecting latency heterogeneity within DRAM.* To our knowledge, TL-DRAM is the first to enable latency heterogeneity within DRAM by significantly modifying the existing DRAM architecture. We believe that this could inspire research on other possible ways of architecting latency heterogeneity within DRAM or other memory devices.

References

- [1] J. Ahn et al. A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing. In *ISCA*, 2015.
- [2] J. Ahn et al. PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture. In *ISCA*, 2015.
- [3] R. Ausavarungrun et al. Staged memory scheduling: achieving high performance and scalability in heterogeneous systems. In *ISCA*, 2012.
- [4] R. Ausavarungrun et al. Exploiting Inter-Warp Heterogeneity to Improve GPGPU Performance. In *PACT*, 2015.
- [5] S. Borkar and A. A. Chien. The future of microprocessors. In *CACM*, 2011.
- [6] Y. Cai et al. Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation. In *ICCD*, 2013.
- [7] Y. Cai et al. Neighbor-cell Assisted Error Correction for MLC NAND Flash Memories. In *SIGMETRICS*, 2014.
- [8] Y. Cai et al. Data retention in MLC NAND flash memory: Characterization, optimization, and recovery. In *HPCA*, 2015.
- [9] Y. Cai et al. Read Disturb Errors in MLC NAND Flash Memory: Characterization, Mitigation, and Recovery. In *DSN*, 2015.
- [10] K. K. Chang et al. Improving DRAM performance by parallelizing refreshes with accesses. In *HPCA*, 2014.
- [11] K. K. Chang et al. Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM. In *HPCA*, 2016.
- [12] N. Chatterjee et al. Leveraging Heterogeneity in DRAM Main Memories to Accelerate Critical Word Access. In *MICRO*, 2012.
- [13] N. Chatterjee et al. Staged Reads: Mitigating the Impact of DRAM Writes on DRAM Reads. In *HPCA*, 2012.
- [14] G. Dhiman et al. PDRAM: A hybrid PRAM and DRAM main memory system. In *DAC*, 2009.
- [15] E. Ebrahimi et al. Parallel Application Memory Scheduling. In *MICRO*, 2011.
- [16] Enhanced Memory Systems. Enhanced SDRAM SM2604, 2002.
- [17] Q. Guo et al. 3D-Stacked Memory-Side Acceleration: Accelerator and System Design. In *WoNDP*, 2013.
- [18] C. A. Hart. CDRAM in a unified memory architecture. In *Compon Spring '94, Digest of Papers*, 1994.
- [19] H. Hassan et al. ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality. In *HPCA*, 2016.
- [20] H. Hidaka et al. The Cache DRAM Architecture: A DRAM with an On-Chip Cache Memory. In *IEEE Micro*, 1990.
- [21] E. Ipek et al. Self optimizing memory controllers: A reinforcement learning approach. In *ISCA*, 2008.
- [22] JEDEC. DDR3 SDRAM STANDARD. <http://www.jedec.org/standards-documents/docs/jesd-79-3d>, 2010.
- [23] J. A. Joao et al. Bottleneck identification and scheduling in multithreaded applications. In *ASPLOS*, 2012.
- [24] J. A. Joao et al. Utility-Based Acceleration of Multithreaded Applications on Asymmetric CMPs. In *ISCA*, 2013.
- [25] T. S. Jung. Memory technology and solutions roadmap. http://www.sec.co.kr/images/corp/ir/irevent/techforum_01.pdf, 2005.
- [26] S. Khan et al. The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study. In *SIGMETRICS*, 2014.
- [27] Y. Kim et al. ATLAS: A scalable and high-performance scheduling algorithm for multiple memory controllers. In *HPCA*, 2010.
- [28] Y. Kim et al. Thread cluster memory scheduling: Exploiting differences in memory access behavior. In *MICRO*, 2010.
- [29] Y. Kim et al. A case for exploiting subarray-level parallelism (SALP) in DRAM. In *ISCA*, 2012.
- [30] Y. Kim et al. Ramulator source code. <https://github.com/CMU-SAFARI/ramulator>, 2015.
- [31] Y. Kim, W. Yang, and O. Mutlu. Ramulator: A Fast and Extensible DRAM Simulator. In *IEEE CAL*, 2015.
- [32] E. Kultursay et al. Evaluating STT-RAM as an energy-efficient main memory alternative. In *ISPASS*, 2013.
- [33] B. C. Lee et al. Architecting Phase Change Memory As a Scalable DRAM Alternative. In *ISCA*, 2009.

- [34] B. C. Lee et al. Phase Change Memory Architecture and the Quest for Scalability. In *CACM*, 2010.
- [35] B. C. Lee et al. Phase-Change Technology and the Future of Main Memory. In *IEEE Micro*, 2010.
- [36] C. J. Lee et al. DRAM-Aware Last-Level Cache Writeback: Reducing Write-Caused Interference in Memory Systems. In *UT Tech Report TR-HPS-2010-002*, 2010.
- [37] D. Lee et al. Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture. In *HPCA*, 2013.
- [38] D. Lee et al. Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case. In *HPCA*, 2015.
- [39] D. Lee et al. Decoupled Direct Memory Access: Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM. In *PACT*, 2015.
- [40] D. Lee et al. Simultaneous Multi-Layer Access: Improving 3D-Stacked Memory Bandwidth at Low Cost. In *ACM TACO*, 2016.
- [41] Y. Li et al. Managing Hybrid Main Memories with a Page-Utility Driven Performance Model. In *CoRR abs/1507.03303*, 2015.
- [42] J. Liu et al. RAIDR: Retention-Aware Intelligent DRAM Refresh. In *ISCA*, 2012.
- [43] J. Liu et al. An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms. In *ISCA*, 2013.
- [44] Y. Luo et al. Characterizing Application Memory Error Vulnerability to Optimize Data Center Cost via Heterogeneous-Reliability Memory. In *DSN*, 2014.
- [45] Y. Luo et al. WARM: Improving NAND flash memory lifetime with write-hotness aware retention management. In *MSST*, 2015.
- [46] J. Meza et al. A case for small row buffers in non-volatile main memories. In *ICCD*, 2012.
- [47] J. Meza et al. Enabling Efficient and Scalable Hybrid Memories Using Fine-Granularity DRAM Cache Management. In *IEEE CAL*, 2012.
- [48] J. Meza et al. A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory. In *WEED*, 2013.
- [49] Micron. RLD RAM 2 and 3 Specifications. <http://www.micron.com/products/dram/rldram-memory>.
- [50] Y. Moon et al. 1.2V 1.6Gb/s 56nm 6F2 4Gb DDR3 SDRAM with hybrid-I/O sense amplifier and segmented sub-array architecture. *ISSCC*, 2009.
- [51] O. Mutlu. Memory Scaling: A Systems Architecture Perspective. In *IMW*, 2013.
- [52] O. Mutlu. Main Memory Scaling: Challenges and Solution Directions. In *More than Moore Technologies for Next Generation Computer Design*. Springer, 2015.
- [53] O. Mutlu and T. Moscibroda. Stall-time fair memory access scheduling for chip multiprocessors. In *MICRO*, 2007.
- [54] O. Mutlu and T. Moscibroda. Parallelism-aware batch scheduling: Enhancing both performance and fairness of shared DRAM systems. In *ISCA*, 2008.
- [55] O. Mutlu and L. Subramanian. Research Problems and Opportunities in Memory Systems. In *SUPERFRI*, 2015.
- [56] S. Narasimha et al. High performance 45-nm SOI technology with enhanced strain, porous low-k BEOL, and immersion lithography. In *IEDM*, 2006.
- [57] S. Phadke and S. Narayanasamy. MLP aware heterogeneous memory system. In *DATE*, 2011.
- [58] M. K. Qureshi et al. Enhancing Lifetime and Security of PCM-based Main Memory with Start-gap Wear Leveling. In *MICRO*, 2009.
- [59] M. K. Qureshi et al. Scalable High Performance Main Memory System Using Phase-change Memory Technology. In *ISCA*, 2009.
- [60] M. K. Qureshi et al. AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems. In *DSN*, 2015.
- [61] Rambus. DRAM Power Model. <http://www.rambus.com/energy>, 2010.
- [62] L. E. Ramos et al. Page placement in hybrid memory systems. In *ICS*, 2011.
- [63] J. Ren et al. ThyNVM: Enabling Software-Transparent Crash Consistency in Persistent Memory Systems. In *MICRO*, 2015.
- [64] Samsung. DRAM Data Sheet. <http://www.samsung.com/global/business/semiconductor/product>.
- [65] Y. Sato et al. Fast Cycle RAM (FCRAM); a 20-ns random row access, pipe-lined operating DRAM. In *Symposium on VLSI Circuits*, 1998.
- [66] V. Seshadri et al. RowClone: Fast and Energy-efficient in-DRAM Bulk Data Copy and Initialization. In *MICRO*, 2013.
- [67] V. Seshadri et al. The Dirty-Block Index. In *ISCA*, 2014.
- [68] V. Seshadri et al. Fast Bulk Bitwise AND and OR in DRAM. In *IEEE CAL*, 2015.
- [69] V. Seshadri et al. Gather-Scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses. In *MICRO*, 2015.
- [70] S. M. Sharroush et al. Dynamic random-access memories without sense amplifiers. In *Elektrotechnik & Informationstechnik*, 2012.
- [71] Y. H. Son et al. Reducing Memory Access Latency with Asymmetric DRAM Bank Organizations. In *ISCA*, 2013.
- [72] L. Subramanian et al. MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems. In *HPCA*, 2013.
- [73] L. Subramanian et al. The Blacklisting Memory Scheduler: Achieving high performance and fairness at low cost. In *ICCD*, 2014.
- [74] L. Subramanian et al. The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory. In *MICRO*, 2015.
- [75] L. Subramanian et al. The Blacklisting Memory Scheduler: Balancing Performance, Fairness and Complexity. In *TPDS*, 2016.
- [76] M. A. Suleman et al. Accelerating critical section execution with asymmetric multi-core architectures. In *ASPLOS*, 2009.
- [77] M. A. Suleman et al. Data Marshaling for Multi-core Architectures. In *ISCA*, 2010.
- [78] H. Usui et al. DASH: Deadline-Aware High-Performance Memory Scheduler for Heterogeneous Systems with Hardware Accelerators. In *ACM TACO*, 2016.
- [79] R. Venkatesan et al. Retention-aware placement in DRAM (RAPID): software methods for quasi-non-volatile DRAM. In *HPCA*, 2006.
- [80] J. Wang et al. Enabling High-performance LPDDR-compatible MRAM. In *ISLPED*, 2014.
- [81] H. Yoon et al. Row Buffer Locality Aware Caching Policies for Hybrid Memories. In *ICCD*, 2012.
- [82] H. Yoon et al. Efficient Data Mapping and Buffering Techniques for Multilevel Cell Phase-Change Memories. In *ACM TACO*, 2014.
- [83] D. Zhang et al. TOP-PIM: Throughput-oriented Programmable Processing in Memory. In *HPCA*, 2014.
- [84] Z. Zhang et al. Cached DRAM for ILP processor memory access latency reduction. *IEEE Micro*, July 2001.