Building and Optimizing MRAM-Based Commodity Memories

JUE WANG, Pennsylvania State University

XIANGYU DONG, Qualcomm Technologies, Inc.

YUAN XIE, University of California, Santa Barbara

Emerging non-volatile memory technologies such as MRAM are promising design solutions for energy-efficient memory architecture, especially for mobile systems. However, building commodity MRAM by reusing DRAM designs is not straightforward. The existing memory interfaces are incompatible with MRAM small page size, and they fail to leverage MRAM unique properties, causing unnecessary performance and energy overhead. In this article, we propose four techniques to enable and optimize an LPDDRx-compatible MRAM solution: *ComboAS* to solve the pin incompatibility; *DynLat* to avoid unnecessary access latencies; and *EarlyPA* and *BufW* to further improve performance by exploiting the MRAM unique features of non-destructive read and independent write path. Combining all these techniques together, we boost the MRAM performance by 17% and provide a DRAM-compatible MRAM solution consuming 21% less energy.

Categories and Subject Descriptors: B.7.1 [Integrated Circuits]: Types and Design Styles—*Memory technologies*

General Terms: Design, Experimentation, Performance

Additional Key Words and Phrases: Nonvolatile memory, MRAM, spin-transfer torque, main memory, LPDDR2, LPDDR3, performance, energy

ACM Reference Format:

Jue Wang, Xiangyu Dong, and Yuan Xie. 2014. Building and optimizing MRAM-based commodity memories. *ACM Trans. Architect. Code Optim.* 11, 4, Article 36 (December 2014), 22 pages.

DOI: <http://dx.doi.org/10.1145/2667105>

1. INTRODUCTION

Reducing power consumption is one of the key design goals for mobile devices. Among all the mobile device components, main memories consume a significant portion of power. For example, the DRAM in a smartphone can consume up to 34.5% of the total power [Duan et al. 2011]. How to improve the memory subsystem power efficiency is a key issue in designing future mobile devices.

To address the challenge, many techniques were proposed to reduce DRAM power consumption [Lebeck et al. 2000; Zhou et al. 2004; Udipi et al. 2010; Zheng et al. 2008;

This work is supported in part by NSF 1218867, 1213052, 1409798, and Department of Energy under Award Number DE-SC0005026. This submission is extended from “Enabling High-Performance LPDDRx-Compatible MRAM” published on ISLPED’14. The additional material provided in the submission includes: (1) a new technique, Buffered Write, with its detailed implementation; (2) the simulation results and analysis for the Buffered Write technique; (3) sensitivity study on the number of channels, ranks, and banks; (4) sensitivity study on the number of cores; and (5) sensitivity study on the number of write buffer entries.

Authors’ addresses: J. Wang, 111N IST Building, University Park, PA 16802; email: jzw175@cse.psu.edu; X. Dong, 5775 Morehouse Drive San Diego, CA 92121; email: xydong@acm.org; Y. Xie, Electrical and Computer Engineering Department, Santa Barbara, CA 93106; email: yuanxie@ece.ucsb.edu.

Permission to make digital or hard copies of part or all 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 show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2014 ACM 1544-3566/2014/12-ART36 \$15.00

DOI: <http://dx.doi.org/10.1145/2667105>

Ahn et al. 2009]. In addition, the low-power, double-data rate, synchronous DRAM (LPDDR) was first introduced in 2009, and LPDDR2 or LPDDR3 DRAM is now widely used in tablets and smartphones (e.g., Apple iPad, Samsung Galaxy [Gowri 2014]). Compared to DDRx, LPDDR_x¹ reduces the memory power by adopting lower core voltage, deep power down mode, wider data path, and on-die DLL removal. The power-saving feature of LPDDR_x even inspires server architects [Malladi et al. 2012; Yoon et al. 2012] to replace DDR3 DRAM with LPDDR2 DRAM to reduce the memory power for the data center by 3–5x [Malladi et al. 2012]. However, DRAM is still a volatile technology that needs refresh. Previous work [Liu et al. 2012] shows that DRAM refresh could contribute as high as 20% of total DRAM power; the DRAM refresh issue will soon become the system performance bottleneck in the near future [Nair et al. 2013; Mukundan et al. 2013]. Therefore, it is necessary to explore alternative memory technologies.

Spin-transfer torque magnetic RAM (STT-MRAM or MRAM²) is an attractive emerging nonvolatile memory (NVM) technology, and its power-saving opportunities have been heavily exploited [Meza et al. 2012a; Kultursay et al. 2013; Meza et al. 2012b]. However, enjoying the MRAM power-saving benefit is not free. There is a consensus that MRAM cannot compete with DRAM in terms of performance, and more importantly, the MRAM chip internal structure is not compatible with today's DRAM interface. This compatibility is not only highly required for a successful MRAM early adoption but also critical to enable a tiered memory system using refresh-free MRAM and high-performance DRAM [Hellmold 2013; Qureshi et al. 2009; Ramos et al. 2011]. Everspin's effort [Rizzo et al. 2013] to implement a DDR3-compatible MRAM is an example to meet this request.

The industry has realized the internal structure difference between NVM and DRAM, and the LPDDR2 standard once included an NVM-based solution called LPDDR2-NVM [JEDEC Solid State Technology Association 2010], in which the activation command is divided into Pre-Active and Activate to transfer the long row address. However, it needs a dedicated controller for 3-phase access and a new software interface with much higher design complexity. Due to the design complexity, this LPDDR2-NVM spec was removed from LPDDR3 standard. How to design LPDDR3-compatible NVM architecture remains a challenge.

In this article, we study MRAM features that are different from DRAM. In particular, three unique MRAM properties, *small page size*, *nondestructive read*, and *independent write path*, are investigated, and we propose four techniques to enable the design of LPDDR3-compatible high-performance and low-power MRAM architecture:

- We keep the LPDDR_x interface but propose a new *Combinational Row/Column Address Strobe* (ComboAS) to handle the unbalanced row and column address bits caused by the MRAM's *small page size*;
- We add *Dynamic Latency* (DynLat) to alleviate the performance degradation introduced by ComboAS;
- We devise *Early Precharge/Activation* (EarlyPA) to improve the MRAM performance by utilizing the MRAM *nondestructive read* property;
- We improve the MRAM write performance by using a *Buffered Write* (BufW) technique, which buffers multiple writes and leverages the MRAM's unique property of *separate write path*.

Combining these, we come up with a DRAM-swappable MRAM solution with a significant performance improvement (17% on average and up to 42%). In other words,

¹We use LPDDR_x to refer LPDDR2 and LPDDR3 in this article.

²We use MRAM to refer to STT-MRAM in the rest of this article.

Fig. 1. The memory organization for LPDDRx chips.

our techniques enable an LPDDRx-compatible MRAM with DRAM-competitive performance and freely exploit the MRAM low-power features.

2. BACKGROUND

2.1. Architecture of Traditional LPDDRx Devices

LPDDRx is the dominant memory interface for modern mobile devices. JEDEC released the first LPDDR specification in 2009. Today, almost all the mobile DRAM chips use LPDDR2 or LPDDR3 interface [JEDEC Solid State Technology Association 2012].

Figure 1 is an exemplary LPDDRx memory subsystem. LPDDRx devices have wider I/O (e.g., x16 or x32) than DDRx ones (e.g., x4 or x8). In our example, four LPDDRx devices form a 2-rank memory subsystem with a 64-bit data bus. LPDDRx uses a multiplexed Command/Address (CA) bus to reduce the pin count. The 10-bit CA bus contains command, address, and bank information.

Each LPDDRx device has 8 banks internally, and each bank can independently process a different memory request. The internal LPDDR3 data path uses an 8n prefetch architecture (LPDDR2-S2 is 2n prefetch, and LPDDR2-S4 is 4n prefetch). The LPDDRx interface transfers 2 data bits per DQ pin during every clock period.

Same as DDRx, LPDDRx accesses begin with an activation command (ACT), which includes an row access strobe (RAS) signal, a bank address, and a row address. Memory controllers send ACT commands to memory devices; memory devices then activate the corresponding bank and the row. The data from the activated row are latched in the sense amplifier (S/A) after a tRCD delay (row address to column address delay). Then, memory controllers can continue to issue column read or write commands with a column access strobe (CAS) signal and the starting column address for the burst access.

The S/A acts as a temporary data storage and drives the amplified data until the array is pre-charged again. Therefore, the S/A is essentially a row buffer that caches the entire row data (which can be 1KB to 4KB in modern DRAMs, and it is called “a page”). Each memory bank has its own S/A.

2.2. MRAM Technology

LPDDRx only optimizes the memory interface for higher power efficiency. However, DRAM technology itself is volatile and needs periodic refresh. MRAM [Chung et al. 2010; Kim et al. 2011; Tsuchida et al. 2010; Rizzo et al. 2013] is an emerging nonvolatile memory technology and does not consume any standby power. MRAM has been widely considered as a potential DRAM replacement.

Figure 2 illustrates the basic concept of MRAM. Instead of using electrical charges, MRAM uses magnetic tunnel junctions (MTJs) to store its binary data. Each MTJ consists of two ferromagnetic layers: a pinned layer with a fixed magnetization direction and a free layer with a switchable direction. The relative direction of these two layers

Fig. 2. The conceptual view of an MRAM cell.

Table I. Comparison of DRAM and MRAM LPDDR3 Devices

Parameters	LPDDR3 DRAM		LPDDR3 MRAM	
Clock	533MHz		533MHz	
Page size	4KByte		256 Byte	
Bank bit	BA2-BA0		BA2-BA0	
Row bit	R13-R0		R17-R0	
Column bit	C9-C0		C5-C0	
tREF	3,900ns		N/A	
tRCD	10 cycle		13 cycle	
tRL	8 cycle		6 cycle	
tWL	4 cycle		4 cycle	
tRP	10 cycle		7 cycle	
tRC	32 cycle		18 cycle	
tRTP	4 cycle		2 cycle	
tRRD	6 cycle		6 cycle	
tCCD	4 cycle		4 cycle	
tWTR	4 cycle		4 cycle	
tWR	8 cycle		14 cycle	
tFAW	27 cycle		27 cycle	
tRFC	70 cycle		N/A	
Voltage	VDD1 (1.8V)	VDD2 (1.2V)	VDD1 (1.8V)	VDD2 (1.2V)
IDD0	7.8mA	28.3mA	4.8mA	41.2mA
IDD2N	1.8mA	3.7mA	1.8mA	3.7mA
IDD2P	1.8mA	1.7mA	1.8mA	1.7mA
IDD3N	3.5mA	6.9mA	3.5mA	6.9mA
IDD3P	3.5mA	6.9mA	3.5mA	6.9mA
IDD4R	3.5mA	140.4mA	0mA	170.9mA
IDD4W	3.5mA	145.4mA	0mA	267.2mA
IDD5	23.8mA	78.2mA	1.8mA	1.7mA
IDD6	1.0mA	3.8mA	0mA	0mA

determines the data stored in the MTJ. Previous work [Chung et al. 2010] has shown that the unit cell dimension of MRAM below 30nm can be smaller than $8 F^2$, which is comparable to DRAM's $6 F^2$ size.

2.3. Architecture of MRAM LPDDR_x Devices

We can use the same memory organization in Figure 1 for MRAM devices. We simulate two 4Gb LPDDR3 modules with DRAM and MRAM, respectively, on a 28-nm DRAM process node. Table I lists the parameters. We use our modified version of CACTI [Thoziyoor et al. 2008] and NVSim [Dong et al. 2012] to generate this estimation. The major parameter differences between DRAM and MRAM include:

- Small page size*: MRAM uses current sensing, which is generally more complex than DRAM voltage sensing and significantly bigger. To get the similar area, MRAM reduces number of S/A and thus has a smaller page size. Our circuit simulation shows that the page size of a 4Gb LPDDR3 MRAM device is only 256B, 16X smaller than its DRAM counterpart. MRAM industry companies also have consensus on this. For example, a 2012 EverSpin patent [Alam et al. 2012] discloses that their MRAM page is only 512-bit large, 32X smaller than a DRAM page.
- Nonvolatility*: MRAM is nonvolatile and needs no refresh. Hence, both the t_{REF} and t_{RFC} of MRAM are zeros. MRAM auto-refresh current ($IDD5^3$) and self-refresh current ($IDD6$) are zeros as well.
- Nondestructive read*: MRAM has smaller t_{RTP} and can issue precharge command sooner because MRAM reads are nondestructive and do not need write-back.
- Fast page close*: MRAM has faster precharge speed and smaller t_{RP} because DRAM precharge needs to balance the bitlines (BL and $\overline{BL}$) to $VDD/2$, but MRAM precharge can skip this step.
- Slow page open*: MRAM MTJ has small on/off resistance ratio (e.g., 200%), and it is hard to sense the data. Therefore, the MRAM row activation speed is slower, and t_{RCD} of MRAM is larger.
- Slow write*: MRAM has longer write latency and higher write energy. Thus, MRAM has larger t_{WR} and $IDD4W$.

3. TECHNIQUE 1: COMBINATIONAL ROW/COLUMN ADDRESS STROBE (COMBOAS)

3.1. Motivation: Balance Row/Column Address Transfers

LPDDR2 and LPDDR3 uses multiplexed CA bus. Each command occupies the CA bus for one cycle and is clocked at both positive and negative edges. CA bus is 10-bit width, hence each command contains 20 bits in total. The activation command (ACT) uses 2 bits for command decoding, 3 bits of the bank address, and the remaining 15 bits are for the row address. Therefore, this scheme can address up to 32K row. It works well for existing DRAM devices but not for our targeted MRAM devices.

The major problem is caused by the MRAM small page size. As explained in Section 2.3, as MRAM S/A is much more complex and larger, the number of S/A per bank is correspondingly reduced. Compared to DRAM devices usually equipping 1KB–4KB pages, MRAM page size is much smaller. For example, EverSpin's MRAM page size is 512 bits [Slaughter et al. 2012]. This is also a common drawback for other nonvolatile memories that use current sensing (e.g., Micron's 1Gb PCM only has page size of 512 bits [Micron 2012]). In this work, our simulated 4Gb MRAM device page size is 256B, 16 times smaller than its DRAM counterpart (see Table I). Since the total memory capacity and the bank count are the same, MRAM needs 4 additional row address bits than DRAM does. While it is not a problem for low-density MRAM devices (e.g., EverSpin 64Mb MRAM [Rizzo et al. 2013] with a 64B page can still be DDR3 interface-compatible), the existing LPDDR x interface is not compatible with gigascale MRAM devices.

A naive solution to this problem is to add two more CA bus pins, but we do not consider it an option. First, the row and column address bits are highly unbalanced, and the extra two pins are only useful in ACT commands. It is a contradiction to the basic multiplexing concept behind the CA bus design. Second, adding two pins implies that the MRAM LPDDR x bus is not compatible to the existing DRAM bus. The bad consequences include but are not limited to: (1) industry-wise pin ball redesigns and PHY interface redesigns; (2) for the host that tends to mix DRAM and MRAM,

³In auto-refresh mode, MRAM peripheral circuitry still consumes power so that $IDD5$ is essentially $IDD2P$.

Fig. 3. The timing diagrams of unlimited-pin and ComboAS.

two different memory interfaces are required, thereby increasing both area and cost. Neither is good for the MRAM early adoption. Instead, our goal is to make a DRAM-swappable MRAM solution with a fully LPDDR_x-compatible interface.

3.2. ComboAS Operation

We propose *Combinational Row/Column Address Strobe (ComboAS)* to balance the MRAM long row address and short column address caused by its smaller page size. The basic concept of *ComboAS* is to offload the overflowed row address from RAS commands (i.e., ACT) to CAS commands (i.e., READ or WRITE). Hence, RAS commands only carry parts of the row address, and we transfer the remaining row address together with the column address in CAS commands.

Considering we split the row address into RAS and CAS commands, we need both of them before a new row activation. Consequently, instead of waiting for t_{RCD} , we should issue a CAS command immediately after every RAS command. Figure 3 compares the timing diagram of ComboAS against the incompatible solution of adding 2 more CA pins.

The ComboAS timing is similar to posted-CAS [La 2002] and posted-RAS [Udipi et al. 2010] as they all issue RAS and CAS commands back-to-back, but they are essentially different. The actual row activation in posted-CAS [La 2002] starts before CAS is received, and its CAS command does not carry any row information. The posted-RAS scheme proposed by Udipi et al. [2010] is the most similar work to our ComboAS technique. However, posted-RAS only works for close-page policy where there is only one CAS command after opening one row, and it does not optimize the timing parameters to mitigate the performance overhead.

To make ComboAS work for both close- and open-page policies, we adjust memory timing parameters as follows:

- Minimize t_{RCD}* : Because we now wait for CAS to start a row activation, and there is no circuit dependency between RAS and CAS, we can reduce t_{RCD} to 1 clock cycle.
- Adjust t_{RL} , t_{WL} , and t_{RTP}* : In ComboAS, the arrival of CAS only means the start of a row activation. It delays the actual column access (read or write) by the physical row activation time. Therefore, we need to increment both the column read and write latencies (t_{RL} and t_{WL}) by a row activation delay. Similarly, it is necessary to adjust t_{RTP} (read to precharge delay) in the same way.

Figure 3 compares the ComboAS external and internal command buses. The actual row activation in the memory is delayed by 1 cycle to wait for the remaining row address bits carried by CAS, and the read/write accesses are delayed by t_{RCD}_0 (the original row activation delay). Table II lists the detailed adjustments.

3.3. ComboAS Implementation

To implement ComboAS, the key issue is to let the memory chip wait until the first CAS command arrives before activating the accessing row. Our modifications include: a register to store the partial row address in RAS; a set of registers to hold the column

Table II. Timing Parameters for Unlimited-Pin and ComboAS

	Unlimited-pin system	ComboAS
tRCD	tRCD ₀	1 tCK
tCCD	tCCD ₀	tCCD ₀
tRL	tRL ₀	tRCD ₀ + tRL ₀
tWL	tWL ₀	tRCD ₀ + tWL ₀
tRTP	tRTP ₀	tRCD ₀ + tRTP ₀

Fig. 4. The architecture of ComboAS. The zoom-in figure shows the detailed circuit design of the RA_EN generator.

addresses in the early-arrived CAS commands; and a signal generator to latch the remaining row address from the first-arrived CAS command.

Address Registers. Normally, we only require one address register because the row address and column address are completely separate. However, as ComboAS divides row addresses into both RAS and CAS, we need two registers, as shown in Figure 4: *P_RowAddrReg* stores the partial row address in RAS commands; *Comb_AddrRegs* hold the combination of the remaining row address and the column address in CAS commands.

In Figure 4, RAC and CAC are the signals to represent the arrivals of RAS and CAS, respectively. They can be decoded from the LPDDR_x command truth table. Since RAS and CAS share the same CA bus, we add a multiplexer to choose from these two registers, with CAC as the select signal.

Note that we designed *Comb_AddrRegs* to be capable of holding more than one entry because multiple CAS commands can arrive to the memory chip before the requested row is opened as the latency to open a row (tRCD₀) is larger than the minimum delay between two column commands (tCCD₀). We decided the number of column address registers by $\lceil tRCD_0/tCCD_0 \rceil$ (i.e., 3 in this work). Similar to the traditional posted-CAS DRAM, ComboAS uses countdown circuits to delay the external CAS command by tRCD₀.

RA_EN Generator. Normal memory devices use the RAC signal to trigger row activations. However, in ComboAS, only the first-arrived CAC signal shall trigger this step; all the later CAC signals should be filtered from the row activation control. We designed a *RA_EN generator* for this purpose, as shown in Figure 4. In a *RA_EN generator*, the signal A is 0 when RAC and CAC are both 0. When RAS arrives and RAC becomes 1, A changes to 1; when the first CAS command comes and CAC becomes 1, A toggles to 0 and remains unchanged when later CAS commands arrive. Thus, we ensure only the first CAS triggers the row activation.

Memory Controller. The modification to the memory controller is negligible in ComboAS. Only a small latch (4-bit in this work) after the PHY interface is needed to temporarily hold the extra row address bits from RAS command and later deposit

Fig. 6. The DynLat implementation.

—Accesses **R2** and **R3** are not back to back. In ComboAS, the t_{RL} of **R3** remains $t_{RL_0} + t_{RCD_0}$, which causes bubbles on both the internal command bus and the data bus (the bubble between DATA2 and DATA3). In DynLat, the bubble on the data bus is eliminated by setting the t_{RL} for **R3** to be $\max(t_{RL} - \text{bubbleLength}, t_{RL_0})$. By forcing t_{RL} larger than t_{RL_0} , we ensure that the command meets the memory chip internal hardware constraint; by subtracting bubbleLength , we guarantee that the bubble is removed.

To implement DynLat, we track *bubble* and update t_{RL} , t_{WL} , and t_{RTP} after each access.

Accumulated Bubble Length (ABL). We use ABL to store the total bubble length during transferring access commands for each memory rank.⁵ We reset ABL to 0 upon every ACT command and accumulate the bubble length upon every READ or WRITE command according to Equation (1):

$$ABL' = ABL + (\text{curCycle} - \text{lastCmdCycle}) - \text{minReqDelay}. \quad (1)$$

In general, it is the summation of the old ABL value and the newly detected bubble length, and it keeps increasing during a page open cycle. In practice, we can set an upper limit to the ABL value to reduce the hardware counter overhead.

Updating Parameters. Based on ABL, we then calculate the new t_{RL} , t_{WL} , and t_{RTP} ⁶ according to Equation (2),

$$\begin{aligned} t_{RL} &= \max(t_{RL_0} + t_{RCD_0} - ABL, t_{RL_0}) \\ t_{WL} &= \max(t_{RL_0} + t_{RCD_0} - ABL, t_{WL_0}) \\ t_{RTP} &= \max(t_{RTP_0} + t_{RCD_0} - ABL, t_{RTP_0}), \end{aligned} \quad (2)$$

where t_{RL_0} , t_{WL_0} , t_{RTP_0} , and t_{RCD_0} are the original timing parameters defined by the memory device.

4.3. DynLat Implementation

Figure 6 shows a memory architecture with DynLat scheme adopted. A DynLat control logic is added to both memory device and memory controller.

Memory Device. Since DynLat introduces variable read and write latencies, the memory device shall track the latest t_{RL} and t_{WL} , so that it can return the data for read or latch the data for write at the correct cycle. For this purpose, we add a new component called *TimeCtrl* to each memory device as shown in Figure 6. TimeCtrl tracks the ABL value and updates the timing parameters to the device internal signal delaying circuitry according to Equation (2). If a memory rank contains multiple memory devices, their TimeCtrl logics behave in a lockstep mode.

⁵DynLat is designed to track the rank-level timings since all banks share the same interface and tCCD constrains the access interval in one rank.

⁶Write-to-precharge latency equals to $t_{WL} + BL/2 + t_{WR}$, and it is adjusted together with t_{WL} .

Fig. 7. The IPC and page-hit ratio comparison between a DRAM system and an MRAM system.

Memory Controller. The same TimeCtrl logic is duplicated in the memory controller so that the optimized command intervals can be correctly generated from the controller. The number of duplications is the same as the memory subsystem rank count. For example, in Figure 6, we duplicate two TimeCtrls in the memory controller for a 2-rank configuration.

5. TECHNIQUE 3: EARLY PRECHARGE/ACTIVATION (EARLYPA)

5.1. Motivation: Leveraging Nondestructive Read

DynLat can remove the unnecessary latency and alleviate the performance drop brought by ComboAS, but it cannot mitigate the performance drop caused by reduced page-hit ratio, which is another side effect of the MRAM small page size. Although a smaller page size is preferred to avoid overactivation and reduce the energy waste [Udipi et al. 2010], it is only meaningful to close-page memory systems for which page locality is not utilized. While today's servers and data centers mostly use close-page policy due to their low data locality, mobile devices still commonly use open-page policy, and their performance is highly sensitive to memory page size.

Figure 7 compares the performance of a DRAM system with 4KB page size and an MRAM LPDDR3 system with 256B pages (see Section 7 for the detailed simulation methodology). The performance difference greatly depends on the page-hit ratio change. The biggest performance loss occurs when page hit ratio is significantly reduced.⁷ On average, the MRAM page-hit ratio is decreased by 66% due to its 16X smaller page size. As a result, replacing LPDDR3 DRAM with MRAM degrades the performance by 10% on average and up to 24%.

Although the fundamental MRAM sensing scheme results in such a disadvantage, we shall notice that the same sensing scheme gives MRAM a unique advantage: unlike DRAM reads that destroy the data stored in the DRAM cell, MRAM reads are nondestructive. Based on this unique property, we devise our third optimization technique: *Early Precharge/Activation (EarlyPA)*.

5.2. EarlyPA Operation

Because DRAM reads are destructive and need data restoration, the DRAM S/A always connects to the bitline during page open, and it also serves as a “row buffer.” On the contrary, MRAM reads are nondestructive. The row buffer part of the S/A is not necessary to connect to the input bitline after the data are correctly sensed out. The only reason to reconnect row buffers and bitlines is to write new data. Previous works also considered decoupling S/As and row buffers [Lee et al. 2009], but they did not

⁷Note that for the workloads whose page hit ratio is less sensitive to the page size, MRAM outperforms DRAM (e.g., *e.nat*) because MRAM has faster precharge speed and needs no refresh.

Fig. 8. The timing diagrams of DynLat and EarlyPA.

utilize this characteristic to optimize the operation timing. Our EarlyPA technique is to precharge bitlines right after data sensing so that the next ACT command can be issued earlier.

To decouple the “data latching” function out of a normal S/A, we first extract the last stage amplifier (usually a pair of cross-coupled inverters) from the S/A, and evolve it into a full SRAM cell. After this change, we still call the remaining part of the “data sensing” circuit the S/A, and the SRAM cells then become the row buffer.

The decoupling enables the EarlyPA operations. A read-only example is illustrated in Figure 8:

- Time slot 1:** Upon the first ACT arrival, the S/A starts data sensing, and a self-precharge counter starts counting down from $tRCD_0$.
- Time slot 2:** The counter triggers a bitline self-precharge (an internal PRE command⁸) after $tRCD_0$. The S/A finishes data sensing, and the row buffer holds a copy of the data.
- Time slot 3:** When the second ACT arrives, bitlines and S/As are ready for another row activation (row 1). At the same time, all the column read accesses to row0 keep proceeding from the row buffer to I/Os.

The decoupled row buffer allows bitlines to be early-precharged during the buffer column accesses, and we can improve the read performance by issuing PRE and ACT commands for the next row in advance. However, if there is a write access, we need another PRE after the dirty data write-back. Therefore, when memory write occurs, the minimum required delay to issue the next PRE operation (*write-to-precharge delay*) is the same to the conventional scheme (i.e., $tWL + BL/2 + tWR$). Our proposed EarlyPA technique handles write accesses as follows:

- If a write comes before the self-precharge is internally issued, we postpone the self-precharge so that we can leverage the unfinished row activation cycle. To do that, we update the self-precharge counter and reset it to the write-to-precharge delay (i.e., $tWL + BL/2 + tWR$).
- If a write comes after the self-precharge is internally issued, it means that we already disconnect the corresponding row in the memory array. In this case, we need to

⁸This self-precharge is different from the auto-precharge operation used in close-page policy. In EarlyPA, the self-precharge only precharges the bitlines but the row buffer data remain intact.

turn on the corresponding wordline again for writing the data, which brings some latency overhead (i.e., 3 cycles in this work). In addition, we have to reset the self-precharge counter to the *wordline-turn-on delay* plus the *write-to-precharge delay* (e.g., $3+t_{WL}+BL/2+t_{WR}$), and the previous self-precharge operation is wasted.

Although frequent write accesses still undermine the EarlyPA performance, it does not cause any timing violation. That is because the memory access order remains unchanged, and the next ACT command is never issued until all the WRITE commands to that row are drained.

5.3. EarlyPA Implementation

Similar to the previously proposed ComboAS scheme, the EarlyPA implementation can be transparent to the memory controller and only requires some timing parameter manipulations. The controlling policy for column access commands (READ/WRITE) remains the same. As shown in Figure 8, we modify two precharge-related timing parameters:

- t_{RAS} (activation-to-precharge delay) of EarlyPA is set as $t_{RCD_0} + t_{RP_0}$.
- t_{RP} (precharging time) value is set as 1 so that the next ACT command can be issued immediately when the self-precharge is finished.

Memory Device. Devices ignore all the PRE commands from the memory controller as EarlyPA automatically precharges the bitlines in advance. Instead, a self-precharge counter is added to each memory device control logic. The counter is set to t_{RCD_0} after every ACT command and reset to $t_{WL}+BL/2+t_{WR}$ or $3+t_{WL}+BL/2+t_{WR}$ after every WRITE command depending on whether the counter reaches zero or not at the WRITE command arrival. Furthermore, the memory device skips precharge-related timing rule (e.g., t_{RTP} checking) except the t_{RAS} checking as S/As and row buffers are decoupled in the EarlyPA mode.

Memory Controller. Symmetrically, the memory controller manipulates the timing parameters in the same way as memory devices do. An additional modification to the memory controller changes the write-to-precharge latency control: after issuing a WRITE command, the minimum required delay for the next PRE command is $t_{WL}+BL/2+t_{WR}+t_{RP_0}$ instead of $t_{WL}+BL/2+t_{WR}$.

6. TECHNIQUE 4: BUFFERED WRITES (BUFV)

6.1. Motivation: Independent Write Path

EarlyPA utilizes the MRAM nondestructive read to issue the PRE command in advance, but we also discuss that a WRITE command might undermine the EarlyPA scheme in terms of performance. Fortunately, we can leverage our decoupled row buffers and bitlines to make another optimization.

Traditional DRAM access protocols handle writes during the row activation cycle. It is a good choice for DRAMs because DRAM reads are destructive and need data restoration. Therefore, it is beneficial to leverage the row buffer for write operations so that we can overlap the write latency with the data restoration process. However, MRAM reads are nondestructive, and we no longer need to buffer the write data in the row buffer. Instead, since we now have row buffers disconnected from bitlines, we can set up a read-independent data write path and buffer the write data in a separate place. This observation leads us to a *buffered write scheme (BufW)*.

6.2. BufW Operation

The basic concept of BufW is to store the incoming WRITE commands in a small buffer placed in the memory bank, use a dedicated write path, and try to issue the internal

Fig. 9. The timing diagrams comparison between EarlyPA and BufW.

write operations only when we detect a bus idle period or the buffer becomes full. The detailed BufW description is as follows:

- If a write comes and the write buffer is not full, we allocate a new write buffer entry and store the data together with its address into this entry. Because the buffer is essentially a small SRAM array, we assume the buffer allocation can be finished within one memory clock cycle. Also, this write does not affect the EarlyPA operation.
- If a write comes and the write buffer is full, we fall back to basic EarlyPA and complete the write through normal write data path (using row buffer as the data latch).
- When the number of idle cycles for a memory bank exceeds a threshold,⁹ we switch the memory rank into a write buffer draining phase, in which an FSM moves the data from the write buffer to the memory array. During each drain, we use the address information in write buffer to turn on the corresponding wordline but without data sensing. Only the specified column is written while the other columns are masked. The procedure is repeated until the write buffer becomes empty. After that, the bitlines are precharged. Or, if a new command arrives from the memory controller during a write, the write is cancelled, and the coming command is served first. In other words, the write buffer is read-preemptive [Sun et al. 2009].
- For each read access, a write buffer lookup is needed since the write buffer might hold the latest copy of the data. Since the write buffer is usually small (e.g., 10-entry), this lookup process can be operated in parallel with the normal column access to the row buffer, hiding the lookup latency. In case of write buffer hit, we add extra read latency to pretend that the data is returned by the memory array.

Figure 9 shows the difference between EarlyPA and BufW. When WRITE arrives, in BufW, DATA0 is held in the write buffer until an idle period is detected on the memory bus. BufW outperforms EarlyPA because it can issue the next PRE and ACT earlier when a write occurs. Note that the memory device enters the write buffer draining phase automatically (e.g., an idle period is detected) and exits it implicitly (e.g., the write buffer becomes empty or a bus activity is detected). Therefore, unlike entering/exiting DRAM self-refresh mode, we do not need explicit LPDDR_x commands for the entry and exit of the write buffer draining phase.

6.3. BufW Implementation

Memory Device. Figure 10 shows write buffer implementation. First, an SRAM-based, first-in, first-out (FIFO)-organized buffer with n entries is added. Each entry includes row address, column address, the data field (32 bytes in this work), and a bit indicating if it is occupied. We evaluate the hardware overhead of these additional components using NVSim [Dong et al. 2012] under a 32nm technology node. The result shows

⁹This functionality and the counter are similar to the circuit that controls the DRAM power-down mode.

Fig. 10. BufW implementation.

that the area overhead is about 0.004mm^2 for each memory bank (smaller than 0.1% compared to a 4Gb DRAM LPDDR3 chip fabricated using 32nm technology with die area of 82mm^2 [Bae et al. 2012]). The energy overhead of one access is about 3.6pJ and the latency is about 0.5ns. Second, 3 multiplexers are added on the memory array interface to choose different sources for row address, column address, and input data. Third, a small controller is added to monitor bank states and drain the write buffer when a long idle state is detected. As a summary, the hardware overhead is negligible.

Memory Controller. The write buffer draining process is transparent to the memory controller until the buffer is full and a new write command comes. In this case, we need to switch the memory controller back to the basic EarlyPA mode. To synchronize the write buffer overflow event between the memory controller and the memory device, we add a *virtual write buffer* to the memory controller for each memory rank. This buffer has the same number of entries as the ones on the memory device side, but each entry only has one bit to indicate if the corresponding entry in the memory device is occupied or not. We update this virtual write buffer using the same algorithm as the one in the memory device. Therefore, the memory controller is able to know the status of the actual write buffer on the memory devices and switch to the basic EarlyPA mode when necessary.

6.4. BufW Discussion

First, as shown in Section 7, the BufW technique is most beneficial to the workloads with heavy write traffic. Therefore, unlike the previous three techniques (ComboAS, DynLat, and EarlyPA) that we consider the essential techniques for the commodity MRAM success, the BufW technique might serve as an optional technique that is only added for a system that is known to handle heavy memory write traffic.

Second, the BufW technique is very different from the previous “cached DRAM” effort [Hidaka et al. 1990; Zhang et al. 2001]. Cached DRAM adds a large amount of SRAMs on a DRAM chip to buffer multiple DRAM rows. The hardware overhead on the DRAM device side is very large because their buffer entry is in the unit of page size. For example, an 8KB SRAM cache is added to a 4MB DRAM [Hidaka et al. 1990], and it can cause at least a 5% die size increase (assuming a 24:1 SRAM/DRAM cell-area ratio). On the contrary, our BufW buffers data in the unit of a memory burst length, and it causes less than 0.1% die size overhead. In addition, fabricating SRAM on a DRAM process degrades the performance. Our BufW technique is immune to this degradation because the buffer is filled on a noncritical path. However, cached DRAM suffers from this problem as cache is latency-sensitive.

Table III. Simulation Settings

Core	1-8 cores, 2GHz, out-of-order ARM cores
SRAM I-L1/D-L1 caches	private, 32KB/32KB, 8-way, LRU, 64B cache line, write-back, write allocate
SRAM L2 cache	shared, 1MB, 16-way, LRU, 64B cache line, write-back, write allocate
Memory controller	open page policy, 32-entry read/write request queues per controller, FR-FCFS scheduler
Main memory	1/2/4 channel, 1/2/4 ranks-per-channel, 4/8 banks-per-rank. Timing is configured as Table I.

Fig. 11. Normalized IPC of main memory system with each technique: ComboAS as the baseline, unlimited-pin, DynLat, EarlyPA, BufW, and DRAM systems.

7. EXPERIMENTS

7.1. Simulation Methodology

We model a 2GHz out-of-order ARMv7 microprocessor using our modified version of gem5 [Binkert et al. 2011]. DRAMSim2 [Rosenfeld et al. 2011] is integrated and modified to model the main memory system. Open-page policy with FR-FCFS [Rixner et al. 2000] scheduling is accurately modeled.

We used the timing and power parameters in Table I to simulate our DRAM and MRAM devices. Both parameters of DRAM and MRAM are projected to work on a 533MHz LPDDR3 bus. Unless specified, our default system configuration comprises a single-core processor with a main memory system with 1 channel, 2 ranks, and 8 banks. We also give detailed sensitivity studies to vary the number of cores, channels, ranks, and banks in Section 7.4. We use open-page policy with row-interleaving, which is widely used to maximize the memory-level parallelism. More details for the simulation setting are provided in Table III.

We select 20 memory-intensive benchmarks from SPEC 2006 [SPEC CPU 2006], EEMBC 2.0 [EEMBC 2014], and HPEC [HPEC 2006]. We form the multi-core workloads by randomly choosing from all the workloads. We fast-forward each simulation to the predefined breakpoint at the code region of interest, warm-up 10 million instructions, and simulate for at least 1 billion instructions. To measure the system performance, we use instruction per cycle (IPC) as the metric.

7.2. Performance Speedup of Individual Benchmarks

Figure 11 shows the performance speedup of the MRAM system with each proposed technique. The DRAM system performance is also provided for comparison. We use ComboAS as the baseline, which has the worst performance. The second bar is the performance of an impractical implementation where two more pins are added (referred to as unlimited-pin in the chart). Compared to unlimited-pin, Figure 11 shows that the performance of ComboAS is degraded by 5% on average. However, after adopting DynLat to reduce the unnecessary latency overhead caused by ComboAS, the system performance is bounced back by 3% on average (up to 14%). Thus, the performance of the ComboAS system with DynLat is comparable to the unlimited-pin system in most cases.

Fig. 12. Normalized energy consumption of the DRAM and MRAM systems adopted different techniques.

In addition, by leveraging the MRAM nondestructive read with EarlyPA, we improve the performance by 14% (up to 36%). Furthermore, by adding BufW scheme, we provide another performance boost, and the overall performance improvement reaches 17% on average (up to 42%). After adopting all the proposed techniques, the overall performance of the projected MRAM system is competitive to the DRAM counterpart (about 98%).

Furthermore, as shown in Figure 11, compared to the unlimited-pin system, our proposed techniques also improve the MRAM system performance by 10%. It means that even without maintaining an LPDDR-compatible interface, our techniques can still improve MRAM main memory system performance.

The performance improvement of each workload is different for two reasons. First, memory-intensive workloads benefit more from our proposed techniques because more efficient memory accesses provide larger system performance improvement. Second, write-intensive workloads benefit more from BufW, but the benefit is reduced if the write ratio is too high and the write buffer is always full. We also give the sensitivity study on the number of write buffer entries in Section 7.4.

7.3. Energy Consumption Analysis

Sleeping Mode. The battery life is critical to every mobile device. To reduce standby power, modern devices (e.g., smartphones) turn off as many components (e.g., CPU, GPU, GPS) as possible during sleeping mode. However, DRAM cannot be turned off because it is volatile. Commonly, DRAM is switched from auto-refresh mode to self-refresh mode before the memory controller goes offline. Although the self-refresh mode can generally reduce the DRAM refresh power by 50% to 80% (depending on the ambient temperature), it is still a dominant power contributor to standby power. For instance, a smartphone usually consumes 25mW to 30mW during standby (e.g., iPhone 4S), but its 512MB DRAM still consumes 6mW even using self-refresh. Replacing DRAM with MRAM can eliminate memory standby power (MRAM IDD6 is 0) and easily improve the mobile device battery life.

Operating Mode. While the performance of our optimized MRAM system is similar to the conventional DRAM system, the real deal breaker is the energy consumption saving. Figure 12 shows the comparison of energy consumption between the DRAM and MRAM systems, in which each value is divided to refresh energy, burst energy, activation/precharge energy and background peripheral circuit energy.¹⁰ The energy overhead of each proposed technique is also included.

¹⁰In this work, we do not model the self-refresh mode for DRAM systems and the power-collapse mode for MRAM systems. The realistic background energy may be smaller than our simulated numbers.

Fig. 13. Normalized IPC of each technique when the number of channels, ranks, and banks are changed.

Fig. 14. Normalized IPC of each technique when the number of cores are changed.

Compared to DRAM, the MRAM-based system does not consume any refresh energy because of its nonvolatility, and it is the major source of MRAM energy saving. We need to mention that, as shown in Table I, the read/write energy of MRAM is larger than DRAM because MRAM has smaller sense margin and the memory cell is difficult to write. Thus, the MRAM burst energy is usually larger than the DRAM one. The energy consumed by peripheral circuits is similar between DRAM and MRAM because we do not apply any circuit optimization to it in this work. But we should note that the peripheral energy of MRAM can be further reduced if MRAM is allowed to go into power-collapse mode frequently during the idle state.

Figure 12 shows that the ComboAS MRAM system reduces total energy consumption by more than 17% compared to DRAM system. After adopting proposed DynLat, EarlyPA and BufW techniques, the energy consumption of MRAM system can be further reduced by 4.5%, on average, since the performance is increased and the total execution time is reduced. Considering the comparable performance and smaller energy consumption, MRAM is an attractive candidate to build the main memory system.

7.4. Sensitivity Study

The Number of Channels, Ranks, and Banks. To evaluate our techniques under different memory configurations, we change the number of memory channels, ranks, and banks but keep the total memory capacity the same. Figure 13 is the normalized IPC under each configuration showing that the proposed techniques improve the performance by 16% to 20% under different configurations. Each channel has its own CA and data bus, so the overall memory bandwidth is increased with adding more channels. Thus, when the number of higher-level parallelism (channels) is increased, the performance difference between naive MRAM and DRAM system is decreased. In this case, the performance of the MRAM system (e.g., 4c1r4b) after adopting all proposed techniques may be better than the DRAM system. But adding higher-level parallelism is expensive, especially for mobile systems. Therefore, using our proposed techniques is a more effective method to boost the performance.

The Number of Cores. Figure 14 shows the performance improvement of our proposed techniques under multicore system configuration. After adopting all the proposed techniques, the performance of 2core/4core/8core system is improved by 14%/10%/5%, on average. When the number of cores is increased, the page-hit ratio of the main memory system is decreased as the number of processes simultaneously accessing memory goes up, which is also proved in some previous works [Udipi et al. 2010; Meza et al. 2012a]. The good news is that the performance hit caused by MRAM smaller page size is also less severe in these cases, and our techniques bring extra performance gain.

Fig. 15. The IPC improvement of BufW over EarlyPA and the percentage of repeated PRE commands when the write buffer size increases.

The Number Write Buffer Entries. For BufW, the number of write buffer entries is a predetermined parameter that should be studied. The possibility of write buffer overflow decreases as the number of entries increases, and the system performance is improved. In case of write buffer overflow, our solution is to utilize the conventional write path, which is through the row buffer. We need another PRE command after that to close the row, and the previous PRE command issued by EarlyPA in advance becomes useless. A larger write buffer can reduce the chances of repeated PRE commands. However, adding more write buffer entries generates area overhead, necessitating a trade-off between performance and cost.

Figure 15 shows the IPC improvement and percentage of repeated precharge when the number of write entries is increased from 2 to 50. The result shows that IPC is increased rapidly when the number of write buffer entries increases from 2 to 10, and the trend becomes flat after 10. Therefore, we select 10 as our write buffer size.

8. RELATED WORK

8.1. DRAM Optimizations

Increase the Energy Efficiency. Lebeck et al. [2000] explored the interaction of page placement with static and dynamic hardware policies. DMA-aware memory energy management is developed [Pandey et al. 2006]. Techniques [Zhou et al. 2004] are proposed to dynamically track miss ratio curve of applications at run time. Other techniques are proposed to reshape the memory traffic and coalesce short idle periods [Huang et al. 2005] and design adaptive memory controller policy [Hur and Lin 2008]. The other major volume of work aimed at redesigning DRAM organization architecture, Zheng et al. [2008], proposed to break a conventional DRAM rank into multiple smaller mini-ranks to reduce the number of devices involved in a single memory access. Ahn et al. [2009] proposed the Multicore DIMM, in which chips are grouped into multiple virtual memory devices to control them individually. Udipi et al. [2010] proposed SBA and SSA to segment global wordlines and control them separately. They mitigate the overfetching problem and save power, but increase the access latency.

Improve the Performance of the DRAM System. Sudan et al. [2010] proposed a new scheme to improve the row buffer utilization using the co-location of chunks. The technique of staged reads [Chatterjee et al. 2012] is proposed to reduce the bus-turnaround penalty. Kim et al. [2012] proposed SALP to exploit the subarray-level parallelism in the DRAM system, which is compatible with our work and can be used together in the MRAM system. Cached DRAM is also proposed [Hidaka et al. 1990; Zhang et al. 2001] to store recently accessed data to improve performance. The design of row buffer cache is proposed in previous works [Loh 2008] to store the accessed rows in the cache and reduce the resulting access latency if there is a hit. It is different from the proposed EarlyPA in which the row buffer stores the current row in order to activate the next

row access early. Some other works focus on designing a memory address mapping scheme [Zhang et al. 2000] and memory scheduling algorithms [Ebrahimi et al. 2011; Kim et al. 2010; Muralidhara et al. 2011; Mutlu et al. 2008]. In addition, new 3D DRAM architecture is proposed by Loh [2008], and systems with photonic interconnects are also studied in some work [Beamer et al. 2010; Vantrease et al. 2008].

8.2. NVM Optimizations

Nonvolatile memories include MRAM and phase-change memory (PCM), which have been considered as scalable DRAM alternatives in some previous works. Some works have put the focus on how to reduce energy overheads and improve the performance of PCM, such as data comparison write [Yang et al. 2007; Zhou et al. 2009], data-inverting schemes [Joo et al. 2010; Cho and Lee 2009], selective-XOR operations [Xu et al. 2009]. Lee et al. [2009] proposed the technique to separate sense amplifiers and row buffers in PCM-based main memory, but unlike EarlyPA, their technique did not issue precharge commands in advance to hide the latency. Meza et al. [2012a] also noticed the NVM smaller page size problem. However, they treated it as an advantage in enabling energy-efficient fine-grained row activation for server applications where the row-hit ratio is inevitably low. The other technique they proposed [Meza et al. 2012b] issues the precharge as soon as sensing completes, and it sends the row address in PRE commands. This technique can be used only in close-page policy. For open-page policy, the write operation needs additional precharge, which is what we addressed in EarlyPA. Kultursay et al. [2013] evaluated MRAM as a main memory alternative and proposed a write scheme to bypass the row buffer write bypass, but they did not buffer multiple writes and did not use the separate write path to optimize the row buffer precharge operations for reads. Last, but not least, Smullen et al. [2011] and Sun et al. [2011] traded off MRAM nonvolatility for improved write speed and energy.

9. CONCLUSION

The shift from PCs to mobile devices necessitates low-power memory solutions; NVM technologies such as MRAM are promising candidates. Compared to DRAM, MRAM has many unique features such as *small page size*, *nondestructive read*, and *independent write path*. The smaller page size brings challenges in designing commodity MRAM that can be deployed on the same LPDDR_x interface as DRAM, and can cause performance degradation to mobile systems in which page-hit ratio is important. In this work, we propose four techniques: ComboAS and DynLat to solve the DRAM-compatibility issue; EarlyPA and BufW to further improve the performance by exploiting the MRAM unique features, and they mitigate the performance loss caused by lower page-hit ratio. Combined, our solution enables a commodity MRAM on LPDDR3 interface with a much optimized performance (17% on average and up to 42%). It makes LPDDR3 MRAM a competitive performer but saves 21% of the energy compared to LPDDR3 DRAM. The proposed architecture is a step forward in energy-efficient memory design.

REFERENCES

- J. H. Ahn, J. Leverich, R. S. Schreiber, and N. P. Jouppi. 2009. Multicore DIMM: An energy efficient memory module with independently controlled DRAMs. *Computer Architecture Letters* 8, 1 (2009), 5–8.
- S. M. Alam, T. Andre, and D. Gogl. 2012. Memory controller and method for interleaving DRAM and MRAM accesses. Patent No. 2012/0155160 A1, Filed Dec 16, 2011, Issued Jun 21, 2012.
- Y.-C. Bae, J.-Y. Park, S. J. Rhee, S. B. Ko, Y. Jeong, K.-S. Noh, Y. Son, J. Youn, Y. Chu, H. Cho, M. Kim, D. Yim, H.-C. Kim, S.-H. Jung, H.-I. Choi, S. Yim, J.-B. Lee, J.-S. Choi, and K. On. 2012. A 1.2V 30nm 1.6Gb/s/pin 4Gb LPDDR3 SDRAM with input skew calibration and enhanced control scheme. In *Proceedings of the International Solid-State Circuits Conference*. 44–46.

- S. Beamer, C. Sun, Y.-J. Kwon, A. Joshi, C. Batten, V. Stojanovic, and K. Asanovic. 2010. Re-architecting DRAM memory systems with monolithically integrated silicon photonics. In *Proceedings of the International Symposium on Computer Architecture*. 129–140.
- 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, and D. A. Wood. 2011. The gem5 simulator. *SIGARCH Computer Architecture News* 39, 2, 1–7.
- N. Chatterjee, N. Muralimanohar, R. Balasubramanian, A. Davis, and N. P. Jouppi. 2012. Staged reads: Mitigating the impact of DRAM writes on DRAM reads. In *Proceedings of the International Symposium on High Performance Computer Architecture*. 1–12.
- S. Cho and H. Lee. 2009. Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance. In *Proceedings of the International Symposium on Microarchitecture*, 347–357.
- S. Chung, K.-M. Rho, S.-D. Kim, H.-J. Suh, D.-J. Kim, H. J. Kim, S. H. Lee, J.-H. Park, H.-M. Hwang, S.-M. Hwang, J. Y. Lee, Y.-B. An, J.-U. Yi, Y.-H. Seo, D.-H. Jung, M.-S. Lee, S.-H. Cho, J.-N. Kim, G.-J. Park, G. Jin, A. Driskill-Smith, V. Nikitin, A. Ong, X. Tang, Y. Kim, J.-S. Rho, S.-K. Park, S.-W. Chung, J.-G. Jeong, and S. J. Hong. 2010. Fully integrated 54nm STT-RAM with the smallest bit cell dimension for high density memory application. In *Proceedings of the International Electron Devices Meeting*. 12.7.1–12.7.4.
- X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. 2012. NVSim: A circuit-level performance, energy, and area model for emerging non-volatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 7, 944–1007.
- R. Duan, M. Bi, and C. Gniady. 2011. Exploring memory energy optimizations in smartphones. In *Proceedings of the International Green Computing Conference and Workshops*. 1–8.
- E. Ebrahimi, R. Miftakhutdinov, C. Fallin, C. J. Lee, J. A. Joao, O. Mutlu, and Y. N. Patt. 2011. Parallel application memory scheduling. In *Proceedings of the International Symposium on Microarchitecture*. 362–373.
- EEMBC. 2014. EEMBC benchmark. Retrieved September 10, 2014 from <http://www.eembc.org/>.
- V. Gowri. 2014. Samsung Galaxy Tab. *AnandTech Review*.
- S. Hellmold. 2013. Delivering nanosecond-class persistent memory. In *Flash Memory Summit*.
- H. Hidaka, Y. Matsuda, M. Asakura, and K. Fujishima. 1990. The cache DRAM architecture: A DRAM with an on-chip cache memory. *IEEE Micro* 10, 2, 14–25.
- HPEC. 2006. HPEC benchmark. Retrieved September 10, 2014 from <http://www.omgwiki.org/hpec/files/hpec-challenge/>.
- H. Huang, K. G. Shin, Lefurgy C., and T. Keller. 2005. Improving energy efficiency by making DRAM less randomly accessed. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 393–398.
- I. Hur and C. Lin. 2008. A comprehensive approach to DRAM power management. In *Proceedings of the International Symposium on High Performance Computer Architecture*. 305–316.
- JEDEC Solid State Technology Association. 2010. Standard JESD209-2E LPDDR2.
- JEDEC Solid State Technology Association. 2012. Standard JESD209-3 LPDDR3.
- Y. Joo, D. Niu, X. Dong, G. Sun, N. Chang, and Y. Xie. 2010. Energy- and endurance-aware design of phase change memory caches. In *Proceedings of the Design, Automation and Test in Europe Conference*, 136–141.
- W. Kim, J. H. Jeong, Y. Kim, W. C. Lim, J.-H. Kim, J. H. Park, H. J. Shin, Y. S. Park, K. S. Kim, S. H. Park, Y. J. Lee, K. W. Kim, H. J. Kwon, H. L. Park, H. S. Ahn, S. C. Oh, J. E. Lee, S. O. Park, S. Choi, H.-K. Kang, and C. Chung. 2011. Extended scalability of perpendicular STT-MRAM towards sub-20nm MTJ node. In *Proceedings of the International Electron Devices Meeting*. 24.1.1–24.1.4.
- Y. Kim, D. Han, O. Mutlu, and M. Harchol-Balter. 2010. ATLAS: A scalable and high-performance scheduling algorithm for multiple memory controllers. In *Proceedings of the International Symposium on High Performance Computer Architecture*. 1–12.
- Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu. 2012. A case for exploiting subarray-level parallelism (SALP) in DRAM. In *Proceedings of the International Symposium on Computer Architecture*, 368–379.
- E. Kultursay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu. 2013. Evaluating STT-RAM as an energy-efficient main memory alternative. In *Proceedings of the International Symposium on Performance Analysis of Systems and Software*, 256–267.
- O.-G. La. 2002. SDRAM having posted CAS function of JEDEC standard. Patent No. 6,483,769, Filed May 2, 2001, Issued Nov 19, 2002.

- A. R. Lebeck, X. Fan, H. Zeng, and C. Ellis. 2000. Power aware page allocation. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*. 105–116.
- B. C. Lee, E. Ipek, O. Mutlu, and D. Burger. 2009. Architecting phase change memory as a scalable DRAM alternative. In *Proceedings of the International Symposium on Computer Architecture*. 2–13.
- J. Liu, B. Jaiyen, R. Veras, and O. Mutlu. 2012. RAIDR: Retention-aware intelligent DRAM refresh. In *Proceedings of the International Symposium on Computer Architecture*, 1–12.
- G. H. Loh. 2008. 3D-stacked memory architectures for multi-core processors. In *Proceedings of the International Symposium on Computer Architecture*, 453–464.
- K. T. Malladi, B. C. Lee, F. A. Nothaft, C. Kozyrakis, K. Periyathambi, and M. Horowitz. 2012. Towards energy-proportional datacenter memory with mobile DRAM. In *Proceedings of the International Symposium on Computer Architecture*. 37–48.
- J. Meza, J. Li, and O. Mutlu. 2012a. A case for small row buffers in non-volatile main memories. In *Proceedings of the International Conference on Computer Design*. 484–485.
- J. Meza, J. Li, and O. Mutlu. 2012b. *Evaluating Row Buffer Locality in Future Non-Volatile Main Memories*. Technical Report CMU-2012-002. Carnegie Mellon University, Pittsburgh, PA.
- Micron. 2012. 121-Ball LPDDR2-PCM and LPDDR2 MCP.
- J. Mukundan, H. Hunter, K.-H. Kim, J. Stuecheli, and J. F. Martínez. 2013. Understanding and mitigating refresh overheads in high-density DDR4 DRAM systems. In *Proceedings of the International Symposium on Computer Architecture*. 48–59.
- S. P. Muralidhara, L. Subramanian, O. Mutlu, M. Kandemir, and T. Moscibroda. 2011. Reducing memory interference in multicore systems via application-aware memory channel partitioning. In *Proceedings of the International Symposium on Microarchitecture*. 374–385.
- O. Mutlu and T. Moscibroda. 2008. Parallelism-aware batch scheduling: Enhancing both performance and fairness of shared DRAM systems. In *Proceedings of the International Symposium on Computer Architecture*. 63–74.
- P. Nair, C.-C. Chou, and M. K. Qureshi. 2013. A case for refresh pausing in DRAM memory systems. In *Proceedings of the International Symposium on High Performance Computer Architecture*. 627–638.
- V. Pandey, W. Jiang, Y. Zhou, and R. Bianchini. 2006. DMA-aware memory energy management. In *Proceedings of the International Symposium on High Performance Computer Architecture*. 133–144.
- M. K. Qureshi, V. Srinivasan, and J. A. Rivers. 2009. Scalable high performance main memory system using phase-change memory technology. In *Proceedings of the International Symposium on Computer Architecture*. 24–33.
- L. E. Ramos, E. Gorbato, and R. Bianchini. 2011. Page placement in hybrid memory systems. In *Proceedings of the International Conference on Supercomputing*. 85–95.
- S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens. 2000. Memory access scheduling. In *Proceedings of the International Symposium on Computer Architecture*. 128–138.
- N. D. Rizzo, D. Houssameddine, J. Janesky, R. Whig, F. B. Mancoff, M. L. Schneider, M. DeHerrera, J. J. Sun, K. Nagel, S. Deshpande, H.-J. Chia, S. M. Alam, T. Andre, S. Aggarwal, and J. M. Slaughter. 2013. A fully functional 64 Mb DDR3 ST-MRAM built on 90 nm CMOS technology. *IEEE Transactions on Magnetics* 49, 7, 4441–4446.
- P. Rosenfeld, E. Cooper-Balis, and B. Jacob. 2011. DRAMSim2: A cycle accurate memory system simulator. *Computer Architecture Letters* 10, 1, 16–19.
- J. M. Slaughter, N. D. Rizzo, J. Janesky, R. Whig, F. B. Mancoff, D. Houssameddine, J. J. Sun, S. Aggarwal, K. Nagel, S. Deshpande, S. M. Alam, T. Andre, and P. LoPresti. 2012. High density ST-MRAM technology. In *Proceedings of the International Electron Devices Meeting*. 29.3.1–29.3.4.
- C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan. 2011. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *Proceedings of the International Symposium on High Performance Computer Architecture*. 50–61.
- SPEC CPU. 2006. SPEC CPU2006. Retrieved September 10, 2014 from <http://www.spec.org/cpu2006/>.
- K. Sudan, N. Chatterjee, D. Nellans, M. Awasthi, R. Balasubramanian, and A. Davis. 2010. Micro-pages: Increasing DRAM efficiency with locality-aware data placement. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*. 219–230.
- G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen. 2009. A novel 3D stacked MRAM cache architecture for CMPs. In *Proceedings of the International Symposium on High Performance Computer Architecture*. 239–249.
- Z. Sun, X. Bi, H. Li, W.-F. Wong, Z.-L. Ong, X. Zhu, and W. Wu. 2011. Multi retention level STT-RAM cache designs with a dynamic refresh scheme. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*. 329–338.

- S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman, and N. P. Jouppi. 2008. A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies. In *Proceedings of the International Symposium on Computer Architecture*. 51–62.
- K. Tsuchida, T. Inaba, K. Fujita, Y. Ueda, T. Shimizu, Y. Asao, T. Kajiyama, M. Iwayama, K. Sugiura, S. Ikegawa, T. Kishi, T. Kai, M. Amano, N. Shimomura, H. Yoda, and Y. Watanabe. 2010. A 64Mb MRAM with clamped-reference and adequate-reference schemes. In *Proceedings of the International Solid-State Circuits Conference*. 258–259.
- A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. P. Jouppi. 2010. Rethinking DRAM design and organization for energy-constrained multi-cores. In *Proceedings of the International Symposium on Computer Architecture*. 175–186.
- D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. G. Beausoleil, and J. H. Ahn. 2008. Corona: System implications of emerging nanophotonic technology. In *Proceedings of the International Symposium on Computer Architecture*. 153–164.
- W. Xu, J. Liu, and T. Zhang. 2009. Data manipulation techniques to reduce phase change memory write energy. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 237–242.
- B.-D. Yang, J.-E. Lee, J.-S. Kim, J. Cho, S.-Y. Lee, and B.-G. Yu. 2007. A low power phase-change random access memory using a data-comparison write scheme. In *Proceedings of the International Symposium on Circuits and Systems*. 3014–3017.
- D. H. Yoon, J. Chang, N. Muralimanohar, and P. Ranganathan. 2012. BOOM: Enabling mobile memory based low-power server DIMMs. In *Proceedings of the International Symposium on Computer Architecture*. 25–36.
- Z. Zhang, Z. Zhu, and X. Zhang. 2000. A permutation-based page interleaving scheme to reduce row-buffer conflicts and exploit data locality. In *Proceedings of the International Symposium on Microarchitecture*. 32–41.
- Z. Zhang, Z. Zhu, and X. Zhang. 2001. Cached DRAM for ILP processor memory access latency reduction. *IEEE Micro* 21, 4, 22–32.
- H. Zheng, J. Lin, Z. Zhang, E. Gorbato, H. David, and Z. Zhu. 2008. Mini-rank: Adaptive DRAM architecture for improving memory power efficiency. In *Proceedings of the International Symposium on Microarchitecture*. 210–221.
- P. Zhou, V. Pandey, J. Sundaresan, A. Raghuraman, Y. Zhou, and S. Kumar. 2004. Dynamic tracking of page miss ratio curve for memory management. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*. 177–188.
- P. Zhou, B. Zhao, J. Yang, and Y. Zhang. 2009. A durable and energy efficient main memory using phase change memory technology. In *Proceedings of the International Symposium on Computer Architecture*. 14–23.

Received June 2014; revised August 2014; accepted August 2014