Constructing Large and Fast On-Chip Cache for Mobile Processors with Multilevel Cell STT-MRAM Technology

LEI JIANG, BO ZHAO, JUN YANG, and YOUTAO ZHANG, University of Pittsburgh

Modern mobile processors integrating an increasing number of cores into one single chip demand large-capacity, on-chip, last-level caches (LLCs) in order to achieve scalable performance improvements. However, adopting traditional memory technologies such as SRAM and embedded DRAM (eDRAM) leakage and scalability problems. Spin-transfer torque magnetic RAM (STT-MRAM) is a novel nonvolatile memory technology that has emerged as a promising alternative for constructing on-chip caches in high-end mobile processors. STT-MRAM has many advantages, such as short read latency, zero leakage from the memory cell, and better scalability than eDRAM and SRAM. Multilevel cell (MLC) STT-MRAM further enlarges capacity and reduces per-bit cost by storing more bits in one cell.

However, MLC STT-MRAM has long write latency which limits the effectiveness of MLC STT-MRAM-based LLCs. In this article, we address this limitation with three novel designs: line pairing (LP), line swapping (LS), and dynamic LP/LS enabler (DLE). LP forms fast cache lines by reorganizing MLC soft bits which are faster to write. LS dynamically stores frequently-written data into these fast cache lines. We then propose a dynamic LP/LS enabler (DLE) to enable LP and LS only if they help to improve the overall cache performance. Our experimental results show that the proposed designs improve system performance by 9–15% and reduce energy consumption by 14–21% for various types of mobile processors.

Categories and Subject Descriptors: B.3.2 [**Memory Structures**]: Design Styles

General Terms: Design, Performance

Additional Key Words and Phrases: Spin-transfer torque, magnetic random access memory, multilevel cell

ACM Reference Format:
Lei Jiang, Bo Zhao, Jun Yang, and Youtao Zhang. 2015. Constructing large and fast on-chip cache for mobile processors with multilevel cell STT-MRAM technology. ACM Trans. Des. Autom. Electron. Syst. 20, 4, Article 54 (September 2015), 24 pages.
DOI: http://dx.doi.org/10.1145/2764903

1. INTRODUCTION

Modern mobile processors are integrating an increasing number of cores on one single chip. For example, mobile processors such as the 4-core Atom from Intel [2013, 2014], the 4-core Tegra from nVIDIA [2012, 2013], the 8-core Snapdragon from Qualcomm [2013], and the 8-core MT5692 from MediaTek [2013] have been adopted in tablet [Gralla 2011] and smartphone [Fujitsu 2012; HTC 2014] products.

Multicore mobile processors require large on-chip caches to achieve scalable performance. However, it is challenging to construct large caches that achieve both high performance and low power consumption. Many high-end mobile processors only have 256KB∼4MB [nVIDIA 2012, 2013; Intel 2013, 2014] last-level caches (LLCs). Choosing

This work was supported in part by the National Science Foundation under grant NSF CSR #1012070 and NSF CAREER #0747242.
Authors' addresses: L. Jiang (corresponding author), B. Zhao, and J. Yang, Electrical and Computer Engineering Department, University of Pittsburgh; Y. Zhang, Computer Science Department, University of Pittsburgh; corresponding author's email: lei16.jiang@gmail.com.

DOI: http://dx.doi.org/10.1145/2764903

SRAM for future large on-chip caches is less appealing due to SRAM's low density, serious leakage, and reliability problems at nanoscale (e.g., soft error and NBTI [Kumar et al. 2006]). Embedded DRAM (eDRAM) is not a good option either because (i) as technology scales, eDRAM faces challenging charge placement and data sensing problems [Chang et al. 2013]; and (ii) eDRAM cells consume significant refreshing power in order to maintain data correctness [Alizadeh et al. 2012]. Recent efforts have studied emerging memory technologies, such as phase-change memory (PCM) [Wu et al. 2009] and spin-transfer torque magnetic RAM (STT-MRAM) [Dong et al. 2008], in the cache hierarchy. Among different memory technologies, STT-MRAM has been identified as a promising candidate for on-chip caches. STT-MRAM has comparable read speed to that of SRAM, better scalability than eDRAM, and no leakage from the memory cell [Chang et al. 2013]. Recently proposed multilevel cell (MLC) STT-MRAM [Ishigaki et al. 2010; Chen et al. 2010, 2011; Lou et al. 2008] stores multiple bits in one cell, which reduces per-bit cost and makes STT-MRAM even more attractive for constructing large and low-power on-chip caches.

However, while MLC STT-MRAM improves density, it has almost doubled read and write latencies. This degrades the cache performance and often diminishes the benefits gained from enlarged capacity. In this article, we propose three novel designs, *line pairing* (LP), *line swapping* (LS), and *dynamic LP/LS enabling* (DLE), to address latency limitation and improve the effectiveness of MLC STT-MRAM-based caches. The followings summarize our contributions.

—We observe that a two-bit MLC STT-MRAM cell includes a *hard-bit* and a *soft-bit* [Ishigaki et al. 2010; Lou et al. 2008]. The hard-bit is fast to read but slow to write, while the soft-bit is fast to write but slow to read. We propose LP (line pairing) to reorganize two physical cachelines into a *read slow write fast* (RSWF) soft-bit cacheline and a *read fast write slow* (RFWS) hard-bit cacheline. We further propose LS (line swapping) to prompt frequently-written data into RSWF lines and frequently-read data into RFWS lines. LP and LS improve cache performance by reducing the average cache access latency.
—We identify that LP and LS need to access double the amount of cells per access and thus reduce access parallelism in the MLC STT-MRAM-based L2 cache. To adapt to different system settings, we propose a dynamic LP/LS enabler (DLE), a dynamic detector that monitors the number of concurrent cache accesses in LLC and enables LP and LS only if reducing cache access latency outweighs the loss of access parallelism in the L2 cache.
—We evaluate the proposed schemes using a set of different benchmark programs on both in-order and out-of-order embedded processors. Experimental results show that our designs can improve performance by 9–15% and reduce energy by 14–21% over the naive MLC STT-MRAM-based L2 cache designs.

In the rest of article, Section 2 describes the STT-MRAM background. Section 3 elaborates the design details of LP and LS. Section 4 summarizes the experiment methodology. We report and analyze the simulation results in Section 5. We present more related work in Section 6 and conclude in Section 7.

2. BACKGROUND AND MOTIVATION

In this section, we first introduce the background of mobile processors. We then show the details of STT-MRAM, magnetic tunnel junctions (MTJ), and multilevel cells (MLC). Firally, we illustrate the necessity of large last-level caches (LLC) for mobile processors and discuss the limitations of directly deploying MLC STT-MRAM in the LLC.

Table I. Mobile Processor Trend

	Intel i7 Broadwell	Intel Atom C2000	Intel Atom Z3000	ARM Cortex A7	ARM Cortex A15
# of cores	2	2-8	1-4	1-4	1-4/cluster
Freq (Hz)	3.4G	1.2G-1.8G	2.4G	800M-2G	1G-2.5G
OoO	Yes	No	Yes	No	Yes
L2/core (KB)	256	512	512	≤256	≤256
L3/core (MB)	2	No	No	External	External
Pipe. Stage	14-24	?	16	8	17-25
TDP (W)	14	3	5	2	4
Technology(nm)	14	22	22	28	28

2.1. Mobile Processors Trend

Table I compares the two mainstream mobile processors in the current mobile computing market—Cortex-A from ARM and Atom from Intel. Instead of manufacturing and selling CPU chips, ARM licenses its processor architectures to nVIDIA (Tegra [nVIDIA 2012, 2013]), Qualcomm (Snapdragon [Qualcomm 2013]), MediaTek (MT5692 [MediaTek 2013]), and many other companies. In the table, we compare Intel core-i7 [Intel 2015] with Atom C2000 [Intel 2013], Atom Z3000 [Intel 2014], ARM Cortex A7 [ARM 2012b] and A15 [ARM 2012a].

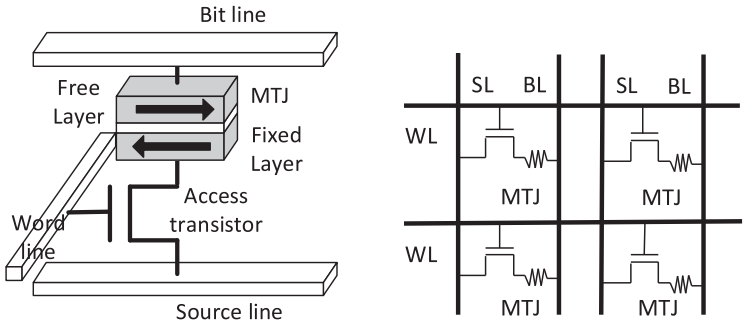
Intel core-i7 is a flagship desktop processor which adopts high operating frequency, out-of-order execution style, a large L3 SRAM cache, and deep pipelines to maximize its performance. It has the largest thermal design power (TDP), which is the average power that a cooling system must dissipate. Mobile processors, that is, the ARM and Atom processors, do not adopt high core frequency. Instead, they adopt ultra low voltage, low operating frequency, small L2 caches, and fewer pipeline stages to prolong battery life. For example, Intel Atom processors only have L2 caches, no L3 caches. The TDP of Intel Atom CPUs is only about 20–30% of that of core-i7. The ARM Cortex series has fewer pipeline stages and achieves even smaller TDPs.

The mobile processors are often designed with different emphases—some for high performance while others may be for high power efficiency. High-performance mobile processors (e.g., Intel Atom Z3000 and ARM Cortex A15) adopt out-of-order execution cores and complex pipelines to enhance performance within a tight power budget. Power-efficient processors (e.g., Intel Atom C2000 and ARM Cortex A7) adopt in-order execution and simple pipeline to maximize power efficiency. The mobile computing industry also takes advantage of hybrid designs that include both high-performance cores and energy-efficient cores in order to achieve benefits from both designs. As an example, the big.LITTLE processing technology of ARM [2011] combines high-performance Cortex A15 cores and high power-efficient Cortex A7 to deliver better peak performance as well as better-sustained performance at lower power cost.

In this article, we propose LP and LS to boost system performance for mobile processors with MLC STT-MRAM-based last-level cache. We further present DLE to dynamically turn on and off LP and LS to improve system performance based on whether applications running on out-of-order mobile cores (Cortex A15 and Atom Z3000) have high memory-level parallelism.

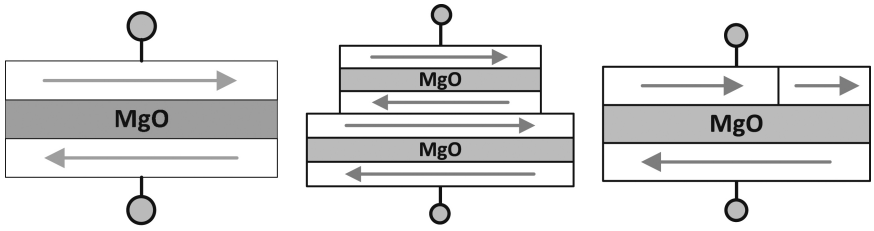
2.2. STT-MRAM Basics

STT-MRAM (spin-transfer torque magnetic RAM) is a new generation of magnetic random access memory (MRAM). As shown in Figure 1(a), a single-level cell (SLC) STT-MRAM uses one MTJ (magnetic tunnel junction) to store binary information [Hosomi et al. 2005]. Figure 2(a) shows one example of single-level cell (SLC) MTJ. An MTJ



(a) 1T1J STT cell. (b) Cell array.

Fig. 1. 1T1J STT-MRAM cell and its cell array structure.



(a) SLC MTJ. (b) Serial MLC MTJ. (c) Parallel MLC MTJ.

Fig. 2. MTJ of SLC and 2-bit MLC.

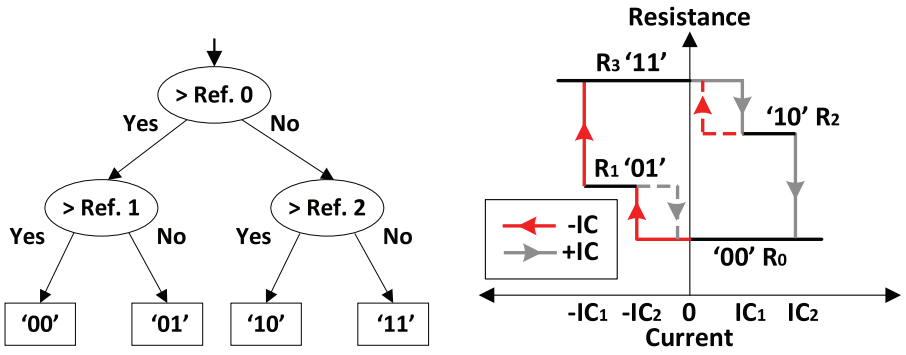
consists of two ferromagnetic layers separated by an oxide barrier layer (MgO). The magnetization direction of one ferromagnetic layer is fixed (reference layer), while the other (free layer) can be changed by injecting an electric current. When the magnetic fields of two layers are parallel, the MTJ resistance is low, representing a logical '0'; when the magnetic fields are anti-parallel, the MTJ resistance is high, indicating a logical '1.'

An STT-MRAM cell typically adopts the '1T1J' structure where one CMOS access transistor is connected to each MTJ (shown in Figure 1(b)) [Zhao et al. 2013]. Raising the voltage of word line (WL) selects the corresponding STT-MRAM cells for operation. STT-MRAM uses spin-polarized current to change the resistance level of the cell. To write bit '1,' the source line (SL) is set to high voltage while the bit line (BL) is connected to ground; to write bit '0,' SL is connected to ground while BL is set high.

STT-MRAM has many advantages. It is nonvolatile such that it has near zero leakage from its memory cell. It has comparable read latency as SRAM. The write is slow but is still much better than that of PCM (phase-change memory). An STT-MRAM cell can be reliably written 10^{12} times and thus does not suffer from the endurance problem that PCM has. STT-MRAM has been proposed as a promising candidate for on-chip caches [Wu et al. 2009].

2.3. STT-MRAM MLC Structure

Advances in device research [Ishigaki et al. 2010; Lou et al. 2008] have enabled fabrication of multilevel cell (MLC) STT-MRAM. There are two types of MLC cells in the literature (Figures 2(b) and 2(c)). A 2-bit *serial* MLC MTJ (Figure 2(b)) possesses two vertically-stacked MTJs that have different MgO and layer thicknesses. Since serial MTJ has a simple structure, it is relatively easy to design and fabricate; for



(a) Two-step read. (b) Two-step write.

Fig. 3. Accessing MLC STT-MRAM cells.

example, Hitachi has announced the successful tape out of serial MLC MTJs [Ishigaki et al. 2010]. However, serial MTJ requires a large critical switching current [Lou et al. 2008], which increases the dynamic power of on-chip caches. Therefore, Seagate has proposed another type of MLC MTJ, *parallel* MLC MTJ (Figure 2(c)), which utilizes a single MgO MTJ whose free layer has two fields [Lou et al. 2008]. Parallel MLC STT-MRAM has been adopted in on-chip caches [Chen et al. 2010, 2011]. In this article, we assume parallel MLC STT-MRAM is used. The work presented here has impelled a similar cache design that adopts serial MLC STT-MRAM [Bi et al. 2013].

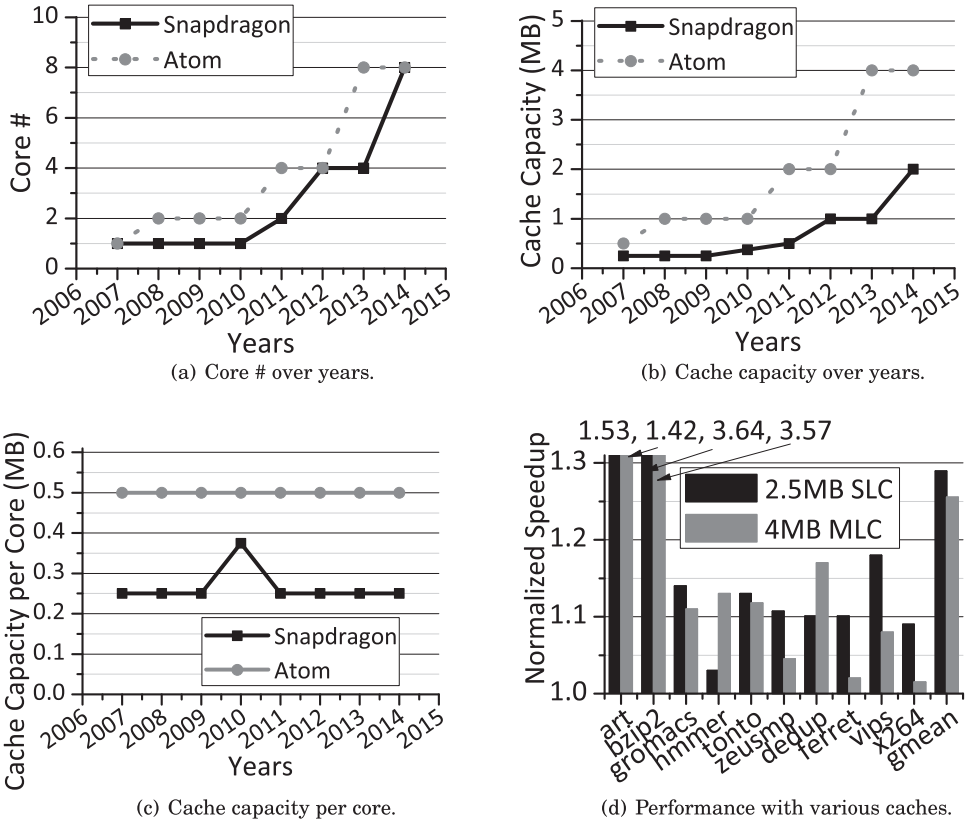
For both types of MLC cells, the two magnetic fields are switched at different spin-polarized currents. Their combinations form multiple resistance levels to represent multibit values. The field that requires large current to switch is referred to as the *hard bit*; the field that requires smaller current to switch is referred to as the *soft bit*. For 2-bit data, the least significant bit (LSB) is the soft bit, while the most significant bit (MSB) is the hard bit.

Unlike SLC STT-MRAM, MLC STT-MRAM may have an endurance problem. With wear leveling [Wang et al. 2013] and error correction [Chen et al. 2011; Xu et al. 2009] based techniques, MLC STT-MRAM-based caches have already obtained a reasonable lifetime in a general-purpose computing platform with everyday workload write traffic.

2.4. MLC Read and Write Operations

Two-Step MLC Read. It requires a two-step operation to read the value stored in a 2-bit MLC STT-MRAM cell [Chen et al. 2010]. As shown in Figure 3(a), the read operation senses the resistance of an MLC MTJ and compares it to two of three resistance references. Each reference is generated by using two reference memory cells. The sense amplifier first compares the cell with Ref_0 to determine the hard bit and then compares it with either Ref_1 or Ref_2 to determine the soft bit. The read latency of an MLC cell is 1.5 times that of an SLC cell [Chen et al. 2010].

Two-Step MLC Write. Writing a 2-bit value into the cell is more complicated. In particular, two MTJs are controlled by one access transistor so that the write current always passes through both MTJs at runtime. Since the current required to switch the hard bit is higher than that of the soft bit (Figure 3(b)), changing the hard bit (MSB) always changes the soft bit (LSB) to have the same magnetic field, that is, either '00' or '11' is written. In the case that LSB is not the same as MSB, a smaller write current is required in the second step to switch the soft bit. Since the time spent writing each bit is comparable to that of SLC, the latency of a 2-bit MLC write roughly doubles that of an SLC write. MLC cell write can terminate early in some cases: (i) if the LSB is the



(a) Core # over years.

(b) Cache capacity over years.

(c) Cache capacity per core.

(d) Performance with various caches.

Fig. 4. Cache capacity per core and performance improvement with various caches.

same as the MSB, then the second step can be skipped; (ii) if the MSB is not changed and only the LSB (soft bit) needs to be changed, then the first step can be skipped [Chen et al. 2010]. However, when writing a cache line (64B), it is less likely that the write can be terminated early, as in most cases at least one cell requires both steps.

2.5. Design Motivation

Multicore processor designs are widely adopted for mobile computing. This is because simply increasing the frequency of one single core not only complicates the design but also leads to significant dynamic power consumption. Figure 4(a) shows the core number increases over years. In the figure, we select two mobile processor families, ARM Cortex A15 core-based CPU Snapdragon [Qualcomm 2013] and X86-based Atom [Intel 2014] as representative examples. Both Qualcomm and Intel have commercialized 8-core mobile processors in 2014. To achieve salable performance, the capacity of on-chip last-level cache (LLC) is also enlarged, as shown in Figure 4(b). These designs adopted SRAM to build large LLCs.

However, while it is well known that large-capacity caches often lead to better system performance, modern mobile processors only strive to keep the same ratio of LLC capacity and core number, that is, per-core LLC capacity. As shown in Figure 4(c), the per-core LLC capacity stays stable in both Atom and Snapdragon processor families. This is mainly due to the large leakage power and weak scalability of traditional SRAM, that is, integrating bigger LLCs may lead to significant power consumption at

runtime. Even though the total LLC capacity increases over years, each individual core gets limited benefits from the enlarged LLC.

In addition, the system performance depends not only on capacity but also on how the cache is constructed. In Figure 4(d), we compare the performance of a set of benchmark programs using SRAM, SLC STT-MRAM, and MLC STT-MRAM caches (the configuration and benchmark details are in Table II and Table IV, respectively). For the die area that integrates 512KB SRAM cells, SLC STT-MRAM and MLC STT-MRAM can integrate 20M and 16M cells, that is, 2.5MB data and 4MB data, respectively. On average, a 2.5MB SLC STT-MRAM LLC improves performance by 28% over the 512KB SRAM LLC design. A 4MB MLC STT-MRAM shows a surprisingly lower improvement, that is, 26% over the baseline. This is mainly due to the slow read and write operations in accessing MLC STT-MRAM cells. Therefore, it is challenging to architect a large and fast MLC STT-MRAM-based LLC.

3. PROPOSED TECHNIQUES

In this section, we elaborate the design details of *line pairing* (LP) and *line swapping* (LS). We then explain their limitations and propose the *dynamic LP/LS enabling* (DLE) mechanism to maximize benefits and reduce the effective access latency of MLC STT-MRAM caches.

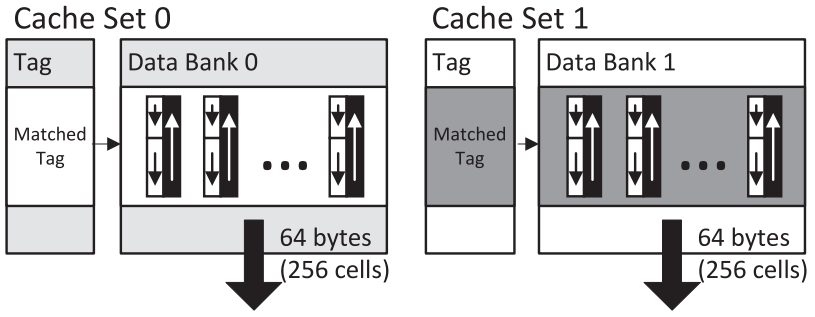
3.1. Line Pairing

Given a 64B cacheline, one access to the data array activates 256 2-bit MLC cells in one bank. Each 2-bit MLC cell consists of one hard bit and one soft bit. The hard bit is fast to read but slow to write, while the soft bit is fast to write but slow to read. The access latency of a cacheline is determined by finishing the read or write operations on the slow bits.

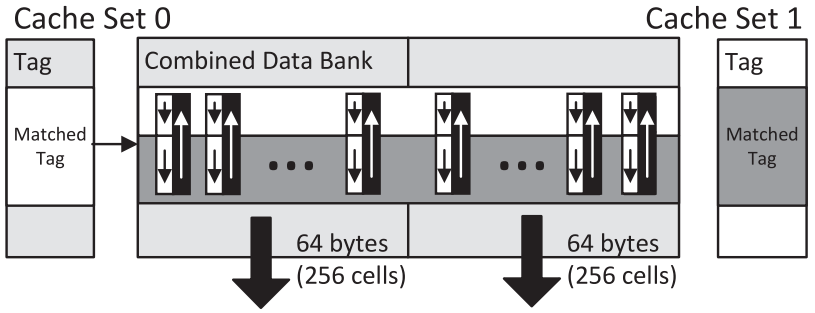
Here we use the cache parameters in later tables for discussion purposes. The design is applicable to caches with different cache sizes and line sizes. In this article, we assume the two-phase cache access scheme that is widely adopted for highly associative caches, that is, a parallel tag matching phase followed by a data array access phase (only to the matched line). Two-phase access achieves better trade-off between performance and energy consumption.

To exploit different read/write characteristics of hard and soft bits, we propose the line pairing (LP) scheme to combine cells from two physical cache lines (having the same access latency) and rearrange them to construct one **RFWS** line (*read fast and write slow line*) and one **RSWF** line (*read slow and write fast line*). As shown in Figure 5, while each physical line has both hard and soft bits, a logical cache line has either all hard bits or all soft bits, but not mixed. The cache line that only has soft bits is referred to as an RSWF line; the cache line that only uses hard bits is referred to as an RFWS line. Within each cache set, half of the cache lines are RSWF, while others are RFWS. A one-bit flag per cacheline is introduced in the tag array to indicate if the line is RSWF or RFWS. After tag access, LP knows the type of cache line.

By constructing RSWF lines, we can greatly speed up write operations when they fall in these lines. The latency is 19 cycles, which is only $\sim$50% of the write latency in baseline design (Table III). However, LP penalizes the writes to RFWS lines. As discussed in Section 2.4, the current required to write a hard bit always destroys the soft bit in the same MLC cell. Therefore, with LP, directly writing a hard-bit line erases the data of its colocated soft-bit line. To prevent such losses, LP first reads the soft-bit line, merges the data from both lines, and then writes both lines into the MLC cells. It takes 42 cycles(=5 cycles read + 37 cycles write) to write a hard-bit RFWS line. It is 14% slower than the baseline.



(a) Each cacheline contains both hard bits and soft bits.



(b) After line pairing, each cacheline contains either hard bits or soft bits.

Fig. 5. Employ line pairing (LP) to reduce hit latency. (After LP, each cache set is stored across two banks but requires half the original storage space in each bank.)

In this design, the read latencies for RFWS lines and RSWF lines are 3 cycles and 5 cycles, respectively. This is because we read the hard bits only for RFWS lines and the soft bits only for RSWF lines.

To reduce the hardware cost, LP uses the same number of ports (i.e., read/write circuits) as the baseline has. Since one bank has only 256 read/write circuits, to finish one cache access in one round, LP spreads the cells from one logical cache line into two banks. As shown in Figure 5, *cache set 0* and *set 1* are stored in *data bank 0* and *bank 1* in the baseline design. In LP, *cache set 0* stores its lines in both banks but uses one bit per cell. Activating 512 cells from two banks enables the access of one 64B cacheline in one round.

LP improves cache performance if many accesses fall in the RSWF lines. Assuming writes are evenly distributed among RSWF and RFWS lines, the average write latency is 31 cycles ($= (19 + 42)/2$ cycles), which is better than the baseline 37 cycles. However, LP activates two banks per access and thus reduces the number of parallelizable accesses. On the one hand, if the baseline issues two accesses to two banks in parallel, these accesses can finish in 37 cycles. Instead, LP has to sequentialize and finish two accesses in 38 cycles (two fast accesses), or up to 84 cycles (two slow accesses). On the other hand, LP can finish one write access in 19 cycles and thus resume following reads earlier to improve performance. We will study its impact on different types of mobile processors.

3.2. Line Swapping

To maximize the benefits provided by LP, we further propose dynamically promoting write-intensive data to RSWF lines and read-intensive data to RFWS lines, referred to as line swapping (LS).

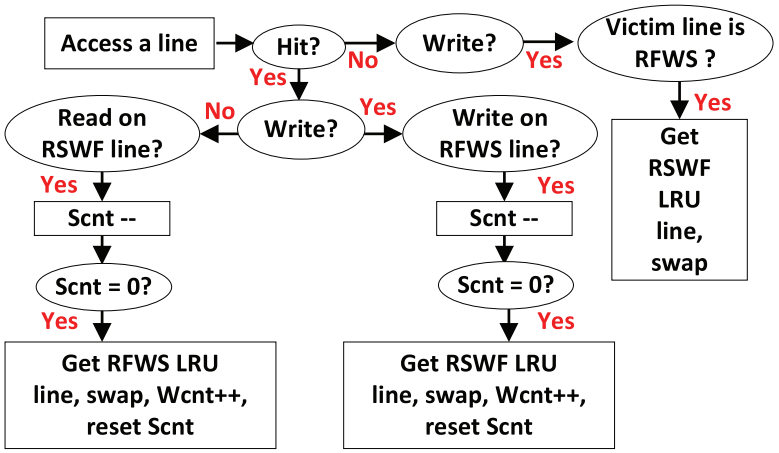


Fig. 6. The line swapping (LS) algorithm.

For a simple implementation of LS, a write-intensive line may be placed in an RFWS line initially; when it is identified as write intensive, it is always swapped to an RSWF line. A read-intensive line may be swapped from an RSWF line to an RFWS line as well. Clearly, too many such swaps increases the number of write operations, incurs more bank contentions, and consumes more energy. In addition, a line that is both read intensive and write intensive may keep thrashing between RSWF and RFWS lines. To achieve better swapping effectiveness and to mitigate thrashing, we introduce two counters—one swap counter `Scnt` and one weight counter `Wcnt`. `Scnt` is introduced to indicate whether a line should be swapped: the one added to RSWF (or RFWS) line indicates whether the line becomes read intensive (or write intensive, respectively). `Wcnt` is introduced to prevent thrashing. A line with large weight value is less likely to be swapped.

Figure 6 illustrates the line swapping (LS) algorithm. `Wcnt` is initialized to be 1, increments when a swap happens, and saturates when it reaches `M`. `Scnt` is initialized to be `Wcnt`×`N` and decrements when an RFWS line has a write hit or an RSWF line has a read hit. Here `M` is the maximal value that the hardware counter can represent; `N` is the swap threshold to be studied in the experiments. When `Scnt` reaches zero, LS swaps either the current RSWF line with an LRU (least recently used) RFWS line, or the current RFWS line with an LRU RSWF line. If a miss happens and the victim line is an RFWS line, LS moves the LRU RSWF line here and loads the new data to the RSWF line location.

By moving write-intensive data into RSWF lines, LP also pushes read-intensive data naturally into RFWS lines, which speeds up both read and write operations.

3.3. The Limitation of LP and LS

While LP and LS reduce the latency of individual cache accesses, they need to access more banks for each access and thus hurt the cache bank-level parallelism. In the case that there are many concurrent requests falling into LLC, LP and LS may degrade the overall system performance. The examples in Figure 7 show that LP and LS may and may not improve the overall performance. Here we assume that read operations always fall into RFWS lines and write operations always fall into RSWF lines. LP takes 3 (or 19) cycles while the baseline needs 5 (or 37) cycles to finish one read (or write) operation. For one access, LP accesses two banks instead of one bank in the baseline, which sacrifices memory-level parallelism for access latency reduction.

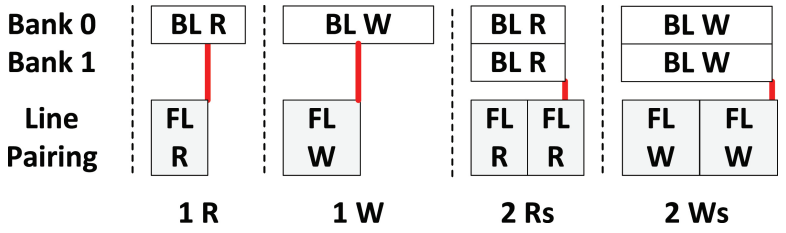


Fig. 7. The gain and loss of LP (BL: baseline; FL: fast line; R: read; W: write). (Not to scale.)

In the case that two read requests arrive at two associated banks, respectively, at the same time, LP spends 3 cycles to finish each read and return data at cycle 3 and cycle 6, respectively. Comparing against the baseline that returns data for both reads at cycle 5, LP speeds up the first read but slows down the second. The average read latency of two reads in LP is shorter than that of our baseline (4.5 cycles v.s. 5 cycles). If two write requests arrive simultaneously, LP finishes them at cycle 19 and cycle 38, respectively, while the baseline finishes both at cycle 37.

From the preceding discussion, the effectiveness of LP and LS heavily depends on the access pattern to the MLC STT-MRAM-based LLC. If the LLC is not very busy, that is, accessing two banks for one access does not block many other accesses, LP and LS can improve the overall system performance by almost halving the cache access latency. However, if the LLC is busy, LP and LS may speed up half the number of accesses while slowing down the other half, resulting in a similar average cache access latency as that of the baseline. Given that LP and LS introduce additional cache-line management overhead at runtime, they tend to degrade the performance slightly if the LLC is busy.

3.4. Dynamic LP/LS Enabler (DLE)

To prevent LP and LS from degrading the system performance, we propose DLE, a dynamic monitoring mechanism that detects the number of concurrent accesses to the LLC and enables LP and LS only if the LLC becomes less busy. We integrate it within an Out-of-Order (OoO) processor [MediaTek 2013; nVIDIA 2013; Qualcomm 2013], because the large instruction issue window and OoO execution style significantly increase memory-level parallelism, that is, there are many concurrent read and write requests sent to the LLC. On the contrary, an in-order processor is less likely to keep the LLC busy, as each core has at most one outstanding read request sent to the LLC.

Figure 8 illustrates the DLE hardware. An OoO processor records outstanding L1 cache misses in miss information/status holding registers (MSHRs). By counting the number of read and write misses in MSHRs, we can quantify the current memory-level parallelism in LLC and enable LP/LS only if the sum of misses is smaller than a threshold. Each cacheline is tagged with one bit flag in its tag array. Two banks form a group, and their corresponding lines are associated such that one cache line can be spread into two banks if the tag is set, that is, LP and LS are enabled.

Figure 9 illustrates how DLE works at runtime. When an access miss occurs in the L1 cache, we check the MSHRs and calculate how many outstanding requests are being serviced concurrently in the LLC. If the number of outstanding requests is no larger than a threshold value (*THD*), we enable both LP and LS in the LLC. If an access that falls into a cache line is not LP-managed, that is, the tag is not set and the line should be accessed the same as in our baseline, we read this line and its buddy line in the associated bank, reorganize two lines as one RSWF line and one RFWS line, and rewrite the data into these two LP-style cache lines. When the number of outstanding requests is bigger than *THD*, we disable LP and LS. In this case, if the access falls into

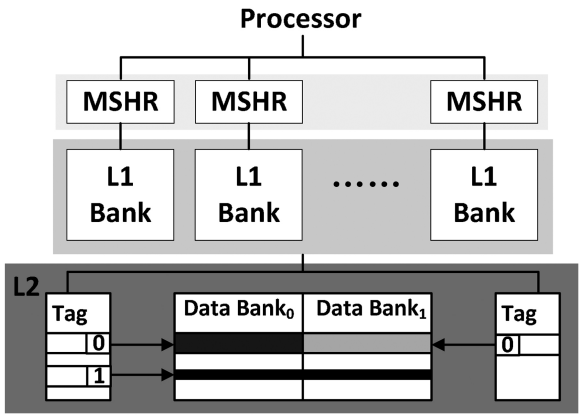


Fig. 8. The hardware of the dynamic LP/LS enabler (DLE).

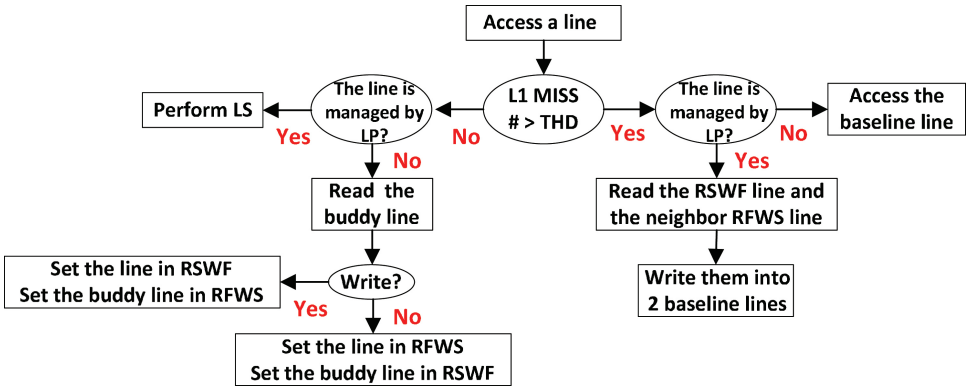


Fig. 9. The workflow of the DLE mechanism.

an LP-managed line, we read this line (an RSWF line or an RFWS line) and its buddy LP-managed line, and rewrite them as two baseline cachelines into the LLC.

3.5. Design Overhead

The hardware cost of our proposed schemes is very modest. Our results show that using a total of 7 bits per line (2 bits for `Wcnt`, 4 bits for `Scnt`, and 1 bit for en/disabling LP and LS) can effectively track frequently-used lines. To enable LS, we add one 512-bit SRAM buffer for each bank. The storage overhead is ∼1% (42 tag bits and 512 data bits).

The control circuit is relatively simple when adopting the design. Based on the decision of whether LP or LS is enabled, the hardware accesses either two banks or one bank to service a cache request. In LP, the control is determined based on the location of data (in either RFWS or RSWF lines) and the type of request (read and write operations). In LS, updating a counter may trigger a swap, which stalls the cache bank and performs line swapping.

4. EXPERIMENT METHODOLOGY

Simulator. We evaluated our proposed schemes using the Virtutech Simics [Magnusson et al. 2002] simulation environment. We upgraded the Simics *g-cache* cache module to

Table II. Baseline Chip Configuration

Processor	Intel Atom, 2-core, 1.8GHz
In-oder Core	single-issue, in-order
OoO Core	4-issue width, bi-mode branch predictor 40 ROB entries, maximal 16 MSHRs in L1
Private L1	32KB I & 32KB D caches; 4-way, write-back SRAM-Based, 32B line, 1-cycle hit latency
Shared L2	16-way, 64B line, write-back, 4 banks **single** read/write port per bank
Main Memory	2GB, 300-cycle latency for the critical block

simulate bank conflicts, bus contention, RFWS/RSWF MLC STT-MRAM cache lines, line pairing (LP), line swapping (LS), and dynamic LP/LS enabler (DLE).

Baseline Modeling. Table II summarizes the core and cache configuration that we simulated in this work. We adopted the Intel Atom processor [Intel 2013] as our simulation processor prototype. It has two in-order single-issue cores. Each core has separate 32KB I-L1 and 32KB D-L1 write-back caches. The L1 caches are private, and the L2 cache is shared. MESI snoopy protocol is deployed to maintain the cache coherence. If an L1 miss happens, the host cache queries the other private L1 caches and waits until receiving a response. We overlapped L2 cache tag lookup with snooping. Based on the type of memory used in the L2 cache, we summarized L2 cache tag lookup latency, read/write hit latency, and energy consumption in Table III. In order to study the impact of our techniques on modern OoO mobile processors, we modeled an OoO version of the Intel Atom processor to mimic ARM Cortex-A15 [ARM 2012], a typical OoO mobile core widely adopted in modern smartphones and tablets [nVIDIA 2013; MediaTek 2013; Qualcomm 2013]. We did not directly model the ARM core, since we want to follow the X86 ISA and make a fair comparison. The OoO core has four-issue width, bimode branch predictor, and 40 ROB entries. This OoO core can concurrently support 16 misses in its L1 cache.

L2 Cache Modeling. Intel Atom is equipped with 1MB shared SRAM L2 cache [Intel 2013]. We modeled the cache using CACTI [HP 2010] and obtained the baseline L2 cache area overhead: $5.1mm^2$. By assuming an SLC STT-MRAM cell size of $14F^2$ [Chung et al. 2010], we calculated that 5MB SLC can fit in a $5.1mm^2$ area overhead under $45nm$ technology. Since the area of an STT-MRAM cell is dominated by its access transistor, the size of the MLC cell is slightly bigger than that of the SLC cell [Chen et al. 2011]: 32M cells (i.e., 8MB data) MLC STT-MRAM can fit within $5.1mm^2$ area. We adopted SLC STT-MRAM for the tag array of our MLC caches. Compared to SRAM and MLC tag arrays, SLC tag array provides a better trade-off between area overhead and access latency. We also modeled an eDRAM L2 cache for comparison. We included a 4MB eDRAM L2 cache, as the density of eDRAM is slightly larger than that of SLC STT-MRAM [Wu et al. 2009].

We used CACTI [HP 2010] to model STT-MRAM dynamic energy based on the reported results [Chen et al. 2010, 2011; Zhou et al. 2009]. The write energy of SLC STT-MRAM is close to that of a soft bit for MLC STT-MRAM.

Benchmarks. We compiled a set of benchmark programs with different memory access characteristics to evaluate our proposed schemes. Since high-end embedded chips usually have GPUs [nVIDIA 2012; Intel 2013], we did not constrain our selections in multimedia benchmarks. Instead, we picked up various workloads from SPEC-CPU2006,

Table III. L2 Cache Configuration

Type	Latency (cycles)	Dynamic Energy (64B) (nJ)	Leakage Power (W)	Area (mm^2)
SRAM (1MB)	tag lookup: 1 data hit: 3	0.31	1.354	5.10
eDRAM (4MB)	tag lookup: 3 data hit: 5	0.51	0.396 refresh	4.96
SLC STTMRAM (5MB)	tag lookup: 2 R-hit: 3 W-hit: 19	read: 0.32 write: 1.29	0.156	5.09
MLC STTMRAM (8MB)	tag lookup: 3 R-hit: 5 W-hit: 37	read :0.32 write: 1.58	0.152	5.13
MLC STTMRAM (8MB) with LP	hard R-hit: 3 soft R-hit: 5 soft W-hit: 19 hard W-hit: 42	hard-R: 0.34 soft-R: 0.38 hard-W: 1.93 soft-W: 1.28	0.152	5.19

Table IV. Simulated Benchmarks

Benchmark	Description	L1 Read MPKI	L1 Write MPKI
art	SPEC-OMP2001	106.4	40.1
bzip2	SPEC-CPU2006	463.4	90.3
gromacs	SPEC-CPU2006	10.1	4.2
hmmer	SPEC-CPU2006	21	10.79
tonto	SPEC-CPU2006	18.7	8.76
zeusmp	SPEC-CPU2006	36.3	11.4
dedup	PARSEC	14.6	8.13
ferret	PARSEC	18.7	13.6
vips	PARSEC	16.1	7.58
X264	PARSEC	35.3	4.6

SPEC-OMP2001, and PARSEC to study memory accesses for embedded cores. Table IV lists the workload details.

5. EXPERIMENTAL RESULTS

We compared four schemes as follows.

—`Baseline` indicates the MLC STT-MRAM cache without LP and LS.
—`LP` means the scheme using LP only.
—`LP+LS` is the scheme using both LP and LS.
—`LP+LS+DLE` is the scheme adopting LP, LS, and DLE.
—`Ideal` represents an ideal but unrealistic setting that defines the upper bound. In this setting, the read and write latencies of MLC are the same as those of SLC.

5.1. Performance Improvement

We report the weighted speedup that is computed as follows [Tullsen and Brown 2001].

$$Weighted\ Speedup = \sum_{thread} \frac{IPC_{tech}}{IPC_{baseline}}. \quad (1)$$

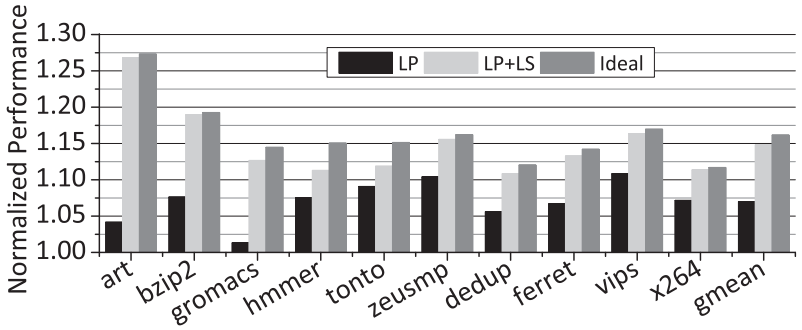


Fig. 10. In-order core performance improvement (normalized to `baseline`).

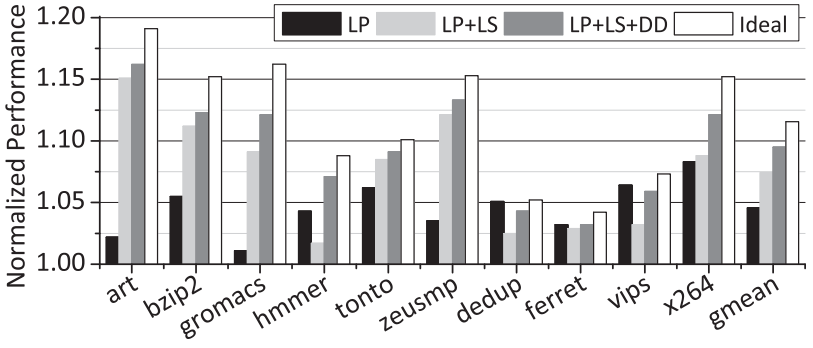
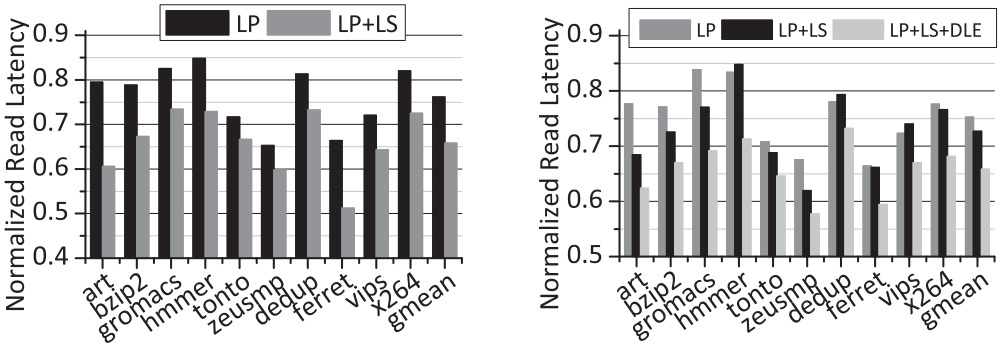


Fig. 11. OoO core performance improvement (normalized to `baseline`).

For Single-Issue In-Order Core. Figure 10 compares the performance using different schemes. On average, `LP` and `LP+LS` achieve 7% and 14.8% performance improvement, respectively, over `Baseline`. From Figure 10, `LP+LS` always outperforms `LP`, indicating that dynamically promoting write-intensive data into RSWF lines and read-intensive data into RFWS lines is effective. `LP+SW` is only ∼1.6% worse than `Ideal`, indicating that `LP+LS` has explored most opportunities—most reads fall in RFWS lines while most writes fall in RSWF lines.

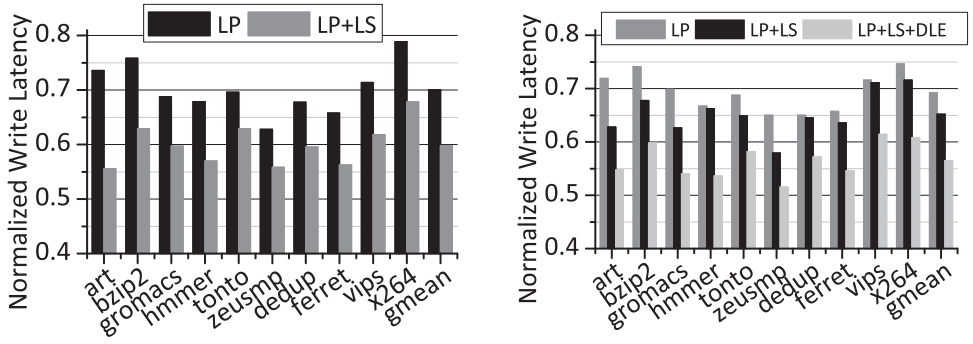
We have discussed in Section 3 that `LP` trades the maximal parallelizable accesses for short hit latency. Since single-issue in-order core does not have large memory request parallelism, our `LP+LS` substantially improves the cache performance. From Figure 10, we observe that memory-intensive workloads with large L1 MPKIs tend to gain large performance improvements. For example, *art* has 40.1 write MPKI and achieves 27.3% performance improvement. The results suggest that for L2 cache accesses, it is more critical to speed up per access.

For Out-of-Order Core. Figure 11 shows the performance comparison using different schemes. Due to OoO execution, reducing long read latency has less impact on system performance. On average, our `LP` only slightly improves performance by 4.5%. Even though we have many reads hitting in RSWF lines and writes falling into RFWS lines, the system performance still benefits from `LP`. This is because the working set of our benchmarks is smaller than 4MB per core, which is only one half of our MLC STT-MRAM-based cache capacity. `LP+LS` further improves the system performance to 7.4% on average.



(a) In-order. (b) OoO.

Fig. 12. Comparing read latency reduction (normalized to `baseline`).



(a) In-order. (b) OoO.

Fig. 13. Comparing write latency reduction (normalized to `baseline`).

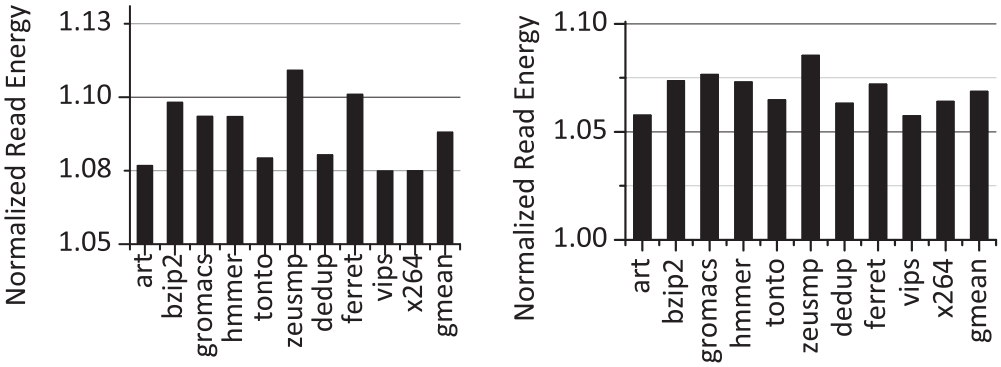
The benefits of `LP+LS` diminish when a benchmark program exhibits large LLC access parallelism. For some benchmark programs, like *hmmer*, *dedup*, *ferret*, and *vips*, `LS` even hurts the overall performance. `LP+LS+DLE` ameliorates the performance by 9.5%, which is only 1.9% smaller than `ideal`. The overheads of converting lines between LP-managed mode and non LP-managed mode cannot be ignored for benchmark programs (e.g., *dedup* and *vips*) that exhibit large parallelism in the LLC.

5.2. Latency Reduction

To evaluate the effectiveness of different schemes, we compared their reduction on read and write latencies.

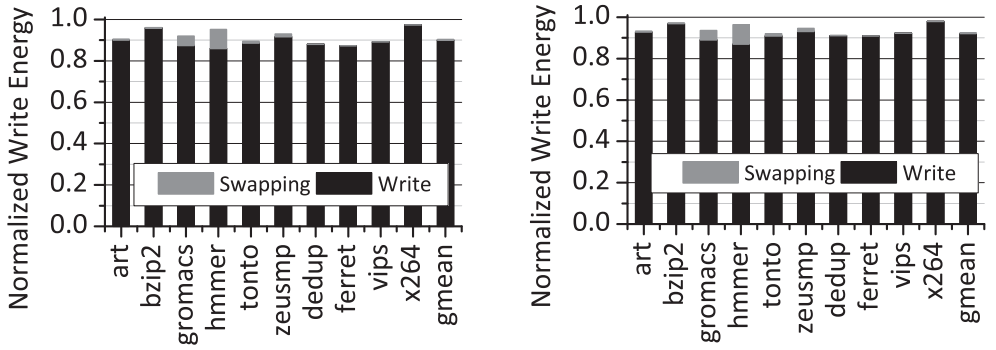
Figure 12 summarizes the read latency reduction using different schemes. On average, for in-order cores, `LP` and `LP+LS` reduce the read latency by 23.9% and 34.2%, respectively, over `Baseline`; for OoO cores, `LP`, `LP+LS`, and `LP+LS+DLE` reduce the read latency by 24.8%, 27.3%, and 34.2%, respectively, over `Baseline`. Since OoO cores have more memory-level parallelism, adopting `LP+LS+DLE` in OoO cores achieves larger percentage reduction than that in in-order cores. However, OoO cores are less sensitive to cache performance, resulting in `LP+LS+DLE` achieving smaller performance improvement on OoO cores.

Figure 13 summarizes the write latency reduction using different schemes. Compared to the reduction of read latency, our techniques achieve larger percentage



(a) In-order. (b) OoO.

Fig. 14. Read energy comparison (normalized to `baseline`).



(a) In-order. (b) OoO.

Fig. 15. Write energy comparison (normalized to `baseline`).

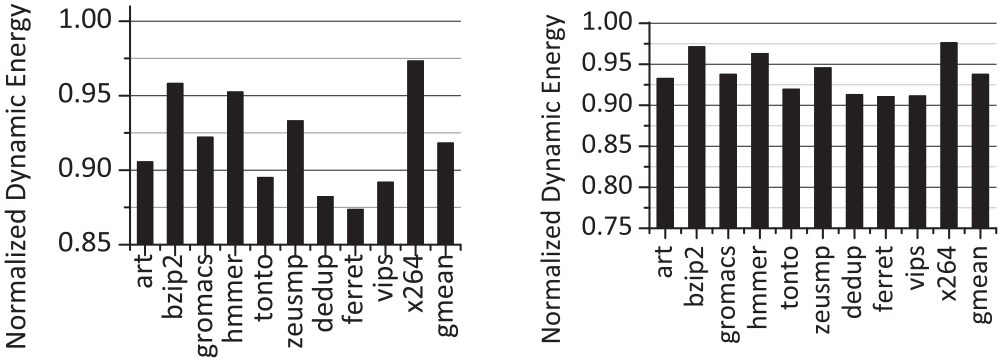
reductions on write latency. For in-order cores, `LP+LS` reduces the write latency by 40.1%, while for out-of-order cores, `LP+LS+DLE` reduces the write latency by 43.5%.

5.3. Energy Reduction

We next studied the energy consumption of different schemes. Given that `LP` activates two banks per cache access and penalizes more if write requests fall into RFWS lines, we expected to observe increased dynamic energy consumption. In the experiments, we used CACTI [HP 2010] to model the bank activation power and summarize the details in Table III.

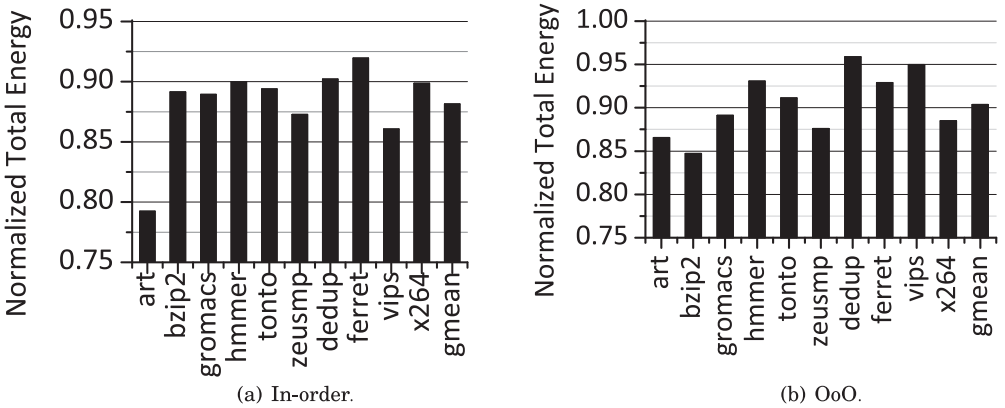
Read Energy. For in-order core, since `LP` activates two banks for each access, the read energy increases: on average, `LP+LS` consumes 9% more read energy, as shown in Figure 14(a). For out-of-order core, Figure 14(b) shows that `LP+LS+DLE` only increases the read energy by 6.9% on average. This is because fewer cache lines are managed by `LP`.

Write Energy. Table III reports the required energy for two types of write operations—RFWS line write and RSWF line write. The former is more expensive due to its extra operations. For in-order core, Figure 15(a) shows that on average `LP+LS` reduces 8% write energy over `baseline`. With `LS`, most writes fall in RSWF lines such that write energy consumption is greatly reduced. For out-of-order core, Figure 15(b) shows that



(a) In-order. (b) OoO.

Fig. 16. Dynamic energy comparison (normalized to `baseline`).



(a) In-order. (b) OoO.

Fig. 17. Total energy comparison (normalized to `baseline`).

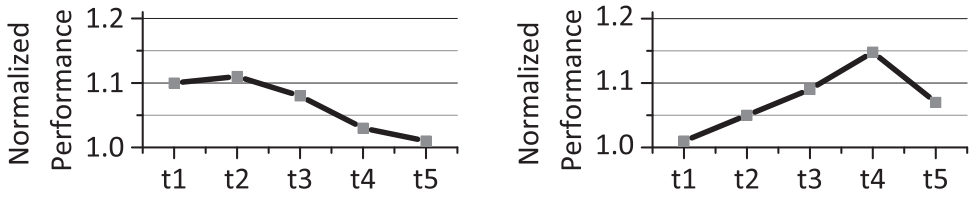
`LP+LS+DLE` only decreases write energy by 6.2% over `baseline`. This is because fewer cache lines are managed by `LP` such that many writes work in a more energy-consuming way, which is the same as in our `baseline`.

Dynamic Energy. For in-order core, Figure 16(a) reports that `LP+LS` saves 8% dynamic energy over `baseline`. This is because write energy dominates the dynamic energy consumption in STT-RAM-based caches. This observation has also been reported in recent studies [Zhou et al. 2009; Smullen et al. 2011]. For out-of-order core, from Figure 16(b), `LP+LS+DLE` reduces dynamic energy by 6.3%.

Total Energy. Given that leakage energy dominates total energy consumption, and our techniques spend fewer CPU cycles in executing the same number of instructions, we observed large energy reduction. For in-order core, Figure 17(a) shows that on average, `LP+LS` reduces 12% total energy consumption. For out-of-order core in Figure 17(b), `LP+LS+DLE` only reduces 9.7% total energy. This is because our techniques achieve less performance improvement on OoO cores.

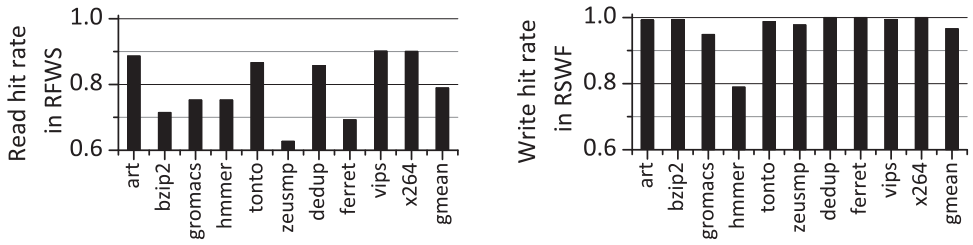
5.4. Line-Swapping Overhead

Next we studied the `LS` overhead and adjusted the values of `Scnt` and `Wcnt`. We only used in-order core in the evaluation, as out-of-order core shares the same access sequence.



(a) Write threshold. (b) Read threshold.

Fig. 18. Performance geomean comparison when using different `Scnt` swap thresholds (normalized to `baseline`).



(a) Read in RFWS lines. (b) Write in RSWF lines.

Fig. 19. Access hit rate in fast cachelines.

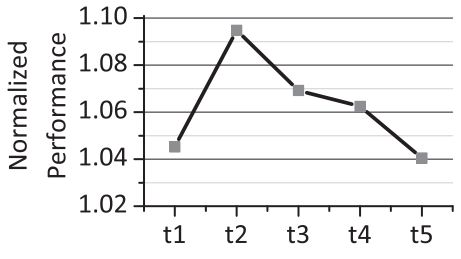
Given that `LS` introduces both latency and energy overheads, we prefer fewer swaps at runtime. The number of swaps depends on how effective we can find read-/write-intensive lines and whether these lines are stable. We adjusted the swap threshold value and studied read- and write-triggered swaps independently. Read-triggered swap happens when `Scnt` of RSWF line reaches zero, while write-triggered swap happens when `Scnt` of RFWS line reaches zero.

Figure 18(a) summarizes the performance improvement when adjusting the initial value of `Scnt` from 1 to 5. Initializing `Scnt` of RFWS lines to 2 achieves the best performance, which shows that the space locality of writes is strong—a swap should be conducted if the RFWS line gets the second write hit. We then studied read-triggered swaps. Figure 18(b) shows that initializing `Scnt` of RSWF lines to 4 achieves the best performance improvement. This is because a swap operation has a long latency. It is not worth paying the overhead if a line to be swapped will be read only once or twice in the future. If the data has been read four times, it is very likely that it will receive more reads. Swapping such kinds of data into RFWS lines reduces both the effective read latency and the number of unnecessary swaps.

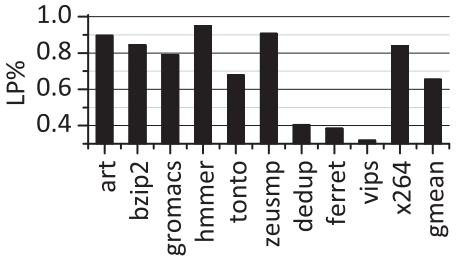
We thus set `Scnt` to be 4 for RSWF lines and 2 for RFWS lines. We also studied `Wcnt` and found that incrementing `Wcnt` per swap effectively eliminates almost all thrashing in our tested programs. We used two bits for `Wcnt` and let it saturate at 3. Accordingly, we need four bits for `Scnt`. In total, we need 6 bits per line to support `LS`. With this setting, Figure 19(b) reports that 97% of writes hit in RSWF lines. Figure 19(a) shows the read hit rate in RFWS lines: on average, 79% of reads occur in RFWS lines. The read hit rate in RFWS lines is lower because using a large `Scnt` threshold has more reads occurring in RSWF lines.

5.5. Dynamic LP/LS Enabling Overhead

Since each in-order core in our setting is single issued and has no more than one outstanding read request, we selected out-of-order core to evaluate the best value for

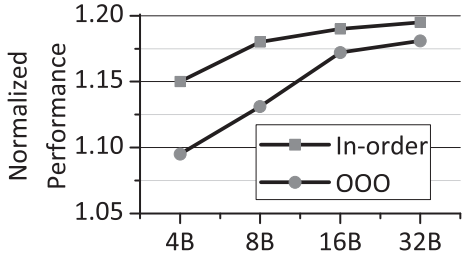


(a) DLE threshold (normalized to `baseline`).

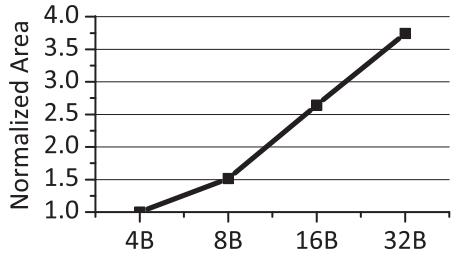


(b) The percentage of LP-managed lines.

Fig. 20. Design-space exploration on DLE.



(a) Performance (normalized to `baseline`).



(b) Cache area (normalized to 4 banks).

Fig. 21. Design-space exploration on bank number.

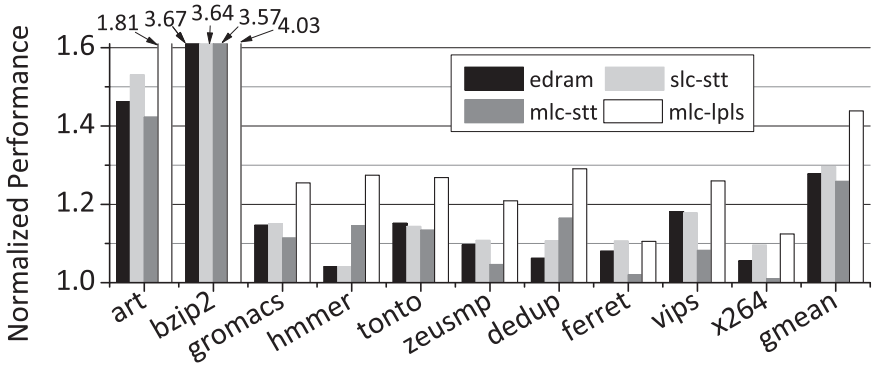
the threshold value (*THD*) in DLE. When the number of concurrent LLC accesses is bigger than *THD*, LP+LS may degrade the overall system performance such that DLE needs to switch following LLC accesses to the baseline mode.

We evaluated the cache performance with THD values from one to five and summarized the results in Figure 20(a). When THD is zero, all cache lines work as in baseline. When THD increases, more cache lines are switched to be managed by LP+LS. From the figure, the performance peaks when THD is two. Since there are two cores and four banks in total, having a THD bigger than two indicates that these requests cannot be serviced at the same time if all cachelines are LP managed. With a larger THD, DLE can not effectively reduce the bank busy time, while, with a smaller THD, DLE triggers huge cache line swapping overhead. In the article, we set THD to two in the evaluation.

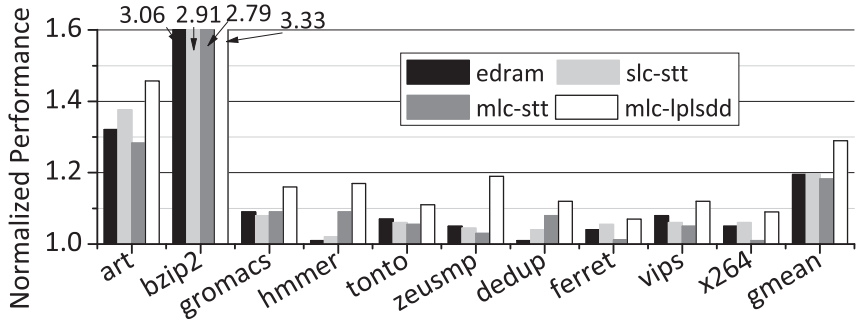
Figure 20(b) shows the percentage of LP-managed lines in the LLC. From the figure, on average, 65.5% of cache lines are managed by our LP+LS+DLE. We observed that *dedup*, *ferret*, and *vips* do not benefit from LP, as most cache lines are accessed the same as in our baseline. They show slightly worse performance improvement than LP (as shown in Figure 11). This is due to the line mode switch overhead introduced by DLE.

5.6. Design-Space Exploration on Bank Number

Since LP trades the access parallelism in LLC for per-access latency reduction, it depends on the number of LLC banks. Figure 21(a) reports system performance with different bank numbers. From the figure, our scheme achieved larger performance improvement with increasing numbers of LLC banks. This is because more requests can be serviced with larger numbers of banks. Given the same benchmark program, more banks increase the bank idle time, which leaves room for LP management. The performance improvements saturate when the bank number reaches 32.



(a) In-order.



(b) OoO.

Fig. 22. Performance comparison (normalized to SRAM performance).

Mobile processors may not have many LLC banks due to their tight power and chip area constraints. Increasing the number of LLC banks often leads to higher fabrication cost and lower yield. Figure 21(b) shows the area comparison with different numbers of LLC banks. We calculated the cache area through CACTI [HP 2010]. A doubled bank number represents 50∼70% cache area increase.

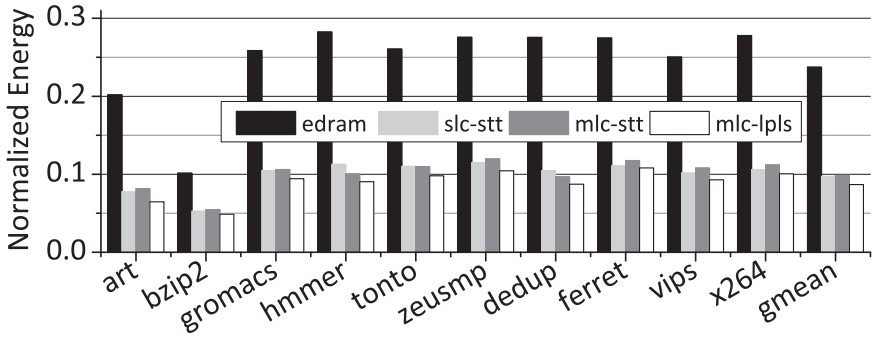
In this article, we report the results using four banks in the baseline configuration.

5.7. Various Memory Technologies

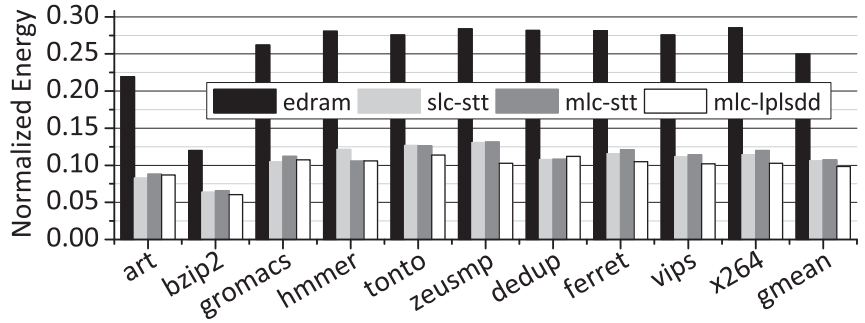
The last experiment that we conducted compared SRAM, eDRAM, SLC STT-MRAM, and MLC STT-MRAM caches under the same die area constraint. The configuration details are in Table III.

Performance. For in-order core, Figure 22(a) summarizes the performance comparison of L2 caches using SRAM, eDRAM, SLC STT-MRAM, MLC STT-MRAM, and MLC STT-MRAM with LP and LS. The results are similar to the findings in Wu et al. [2009]. For most benchmark programs that we simulated, 512KB per-core SRAM cache can not hold their working sets. Too many off-chip accesses significantly hurt the performance of SRAM-based caches. eDRAM has similar performance as SLC STT-MRAM. With a larger capacity and similar access latency, MLC STT-MRAM with LP and LS achieves much better performance than SLC STT-MRAM.

For out-of-order core. Figure 22(b) shows the performance comparison using different memory technologies. Because of large instruction issue window and out-of-order



(a) In-order cores.



(b) OoO cores.

Fig. 23. Energy comparison (normalized to SRAM energy).

execution fashion, the system performance is less sensitive to LLC cache access latency. From the figure, LP+LS+DLE still improves the system performance of MLC STT-MRAM-based LLC by 7∼8% over other types of LLCs.

Total Energy. For in-order core, Figure 23(a) compares the total energy when using SRAM, eDRAM, and SLC STT-MRAM-based caches. For all configurations, leakage energy dominates total energy. It is not surprising that the SRAM L2 cache consumes the highest total energy. Due to refreshes, eDRAM spends 23% of SRAM total energy on average. The energy consumptions on SLC and MLC STT-MRAM are similar, which are about 10% of SRAM total energy. MLC STT-MRAM with LP and LS always obtains the smallest leakage energy among all benchmarks because of its shortest execution time. On average, MLC STT-MRAM with LP and LS consumes only 8% of the total energy of SRAM when executing the same number of instructions. The energy comparison result for out-of-order core can be found in Figure 23(b). LP+LS+DLE spends 10% of the total energy of SRAM in executing the same benchmark applications.

5.8. Compatibility with Read-Preemptive Write Buffer

To reduce the negative impact of long write latency of STT-MRAM, previous work [Sun et al. 2009] has proposed a write buffer such that an ongoing write may be preempted by an incoming read request and than resumes after servicing the read. Since reads are on the critical path, this technique can substantially improve system performance. To exam the compatibility of our techniques with the existing read-preemptive write buffer, we added 4-entry write buffer to our L2 cache.

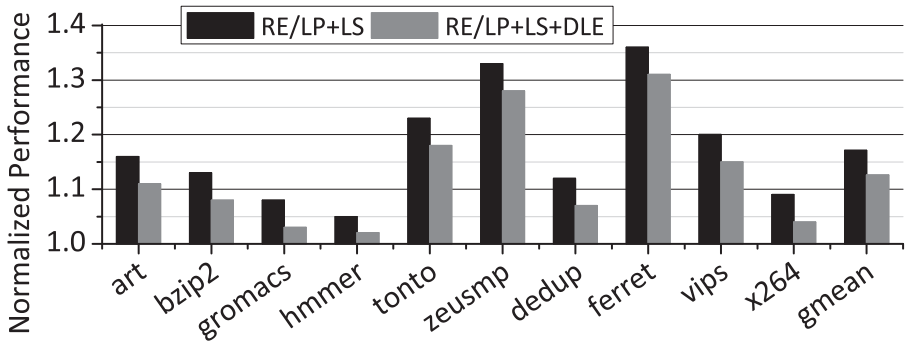


Fig. 24. Read-preemptive write buffer performance improvement (normalized to LP+LS and LP+LS+DLE, respectively).

Figure 24 shows the performance improvement of read-preemptive (RE) write buffer on both in-order core and out-of-order core platforms. The results are normalized to LP+LS for the in-order cores and LP+LS+DLE for OoO cores, respectively. From the figure, the combined scheme achieves smaller but decent improvement, which indicates that the read-preemptive write buffer is still effective for STT-MRAM caches.

6. RELATED WORK

As a promising nonvolatile memory, STT-MRAM has been proposed as SRAM or DRAM replacement for on-chip caches [Sun et al. 2012; Li et al. 2013]. Previous works [Wu et al. 2009; Dong et al. 2008; Sun et al. 2009] here presented a 3D-stacked processor architecture with on-chip SLC STT-MRAM cache and compared SLC STT-MRAM to other different memory technologies for on-chip caches. Zhou et al. design early write termination to save write energy when the bit value is not changed [2009].

Multilevel cell (MLC) STT-MRAM is a promising approach for improving the density of STT-MRAM [Chen et al. 2010, 2011; Ishigaki et al. 2010; Lou et al. 2008]. Ishigaki et al. proposed serial MLC MTJ and studied two-step read and write schemes [2010]. Lou et al. presented parallel MLC MTJ and compared it to serial MLC STT-MRAM [2008]. Sbiaa et al. proposed a perpendicular MTJ with MLC capability [2011]. Chen et al. optimized read and write circuits and algorithms for parallel MLC STT-MRAMs [2010]. Energy-efficient encoding and a wear-level scheme [Chen et al. 2011] are proposed to make MLC STT-MRAM-based cache more practical. Schemes reorganizing soft bits and hard bits into fast and slow lines can significantly accelerate the creation of on-chip cache checkpoints [Chi et al. 2014].

Recent studies of MLC flash devices have also found that some MLC flash pages are as fast as SLC pages, while others are slow [Grupp et al. 2009]. Flash manufacturers utilize FTL (flash translation layer) support to pack bits from two different pages into each MLC cell. The line pairing (LP) scheme proposed in this article shares similarities except that LP is performed at line granularity. A big difference between our design and flash designs is that we developed line swapping (LS) which can greatly improve system performance and reduce energy overhead.

Another type of MRAM cell structure storing multiple bits in one cell is the domain wall memory (DWM) [Sun et al. 2013]. DWM records multiple bits in one magnetic domain along a nanoscopic permalloy wire and passes current through the wire to move the domains so that each domain can be read or written by peripheral circuits. By combining MLC and DWM structures, the cache capacity is substantially enlarged [Sharad et al. 2013].

7. CONCLUSIONS

Integrating a large and fast on-chip L2 cache is a simple and effective way to mitigate the memory wall on high-performance embedded processors. As technology scales, MLC STT-MRAM shows many advantages over SRAM, eDRAM, and SLC-MRAM in constructing on-chip caches. However, MLC STT-MRAM suffers from long read- and write-access latencies. In this article, we proposed three novel schemes, line pairing (LP), line swapping (LS), and dynamic LP/LS enabling (DLE), to reduce the average cache hit latency of MLC STT-MRAM-based L2 caches. Our experimental results show that these schemes are effective and on average achieve 9–15% performance improvement and 14–21% energy reduction over the baseline design.

REFERENCES

Mohammad Alizadeh, Adel Javanmard, Shang-Tse Chuang, Sundar Iyer, and Yi Lu. 2012. Versatile refresh: Low complexity refresh scheduling for high-throughput multi-banked eDRAM. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems*. 247–258.

ARM. 2012a. Cortex-A15. http://www.arm.com/products/processors/cortex-a/cortex-a15.php.

ARM. 2012b. Cortex-A7. http://www.arm.com/products/processors/cortex-a/cortex-a7.php.

ARM. 2011. ARM big.LITTLE technology. http://www.arm.com/products/processors/technologies/biglittleprocessing.php.

Xiuyuan Bi, Mengjie Mao, Danghui Wang, and Hai Li. 2013. Unleashing the potential of MLC STT-RAM caches. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 429–436.

Mu-Tien Chang, Paul Rosenfeld, Shih-Lien Lu, and Bruce Jacob. 2013. Technology comparison for large last-level caches (L^3Cs): Low-leakage SRAM, low write-energy STT-RAM, and refresh-optimized eDRAM. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture*. 143–154.

Yiran Chen, Xiaobin Wang, Wenzhong Zhu, Hai Li, Zhenyu Sun, Guangyu Sun, and Yuan Xie. 2010. Access scheme of multi-level cell spin-transfer torque random access memory and its optimization. In *Proceedings of the IEEE International Midwest Symposium on Circuits and Systems*. 1109–1112.

Yiran Chen, Weng-Fai Wong, Hai Li, and Cheng-Kok Koh. 2011. Processor caches built using multi-level spin-transfer torque RAM cells. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 73–78.

Ping Chi, Cong Xu, Tao Zhang, Xiangyu Dong, and Yuan Xie. 2014. Using multi-level cell STT-RAM for fast and energy-efficient local checkpointing. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 301–308.

Suock 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. Au, J.-U. Yi, Y.-H. Seo, D.-H. Jung, M.-S. Lee, S.-H. Cho, J.-N. Kim, G.-J. Park, J. Gyuan, 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. I. Hong. 2010. Fully integrated 54nm STT-RAM with the smallest bit cell dimension for high density memory application. In *Proceedings of the IEEE International Electron Devices Meeting*. 12–7.

Xiangyu Dong, Xiaoxia Wu, Guangyu Sun, Yuan Xie, H. Li, and Yiran Chen. 2008. Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement. In *Proceedings of the ACM/IEEE Design Automation Conference*. IEEE, 554–559.

Fujitsu. 2012. LOOX. http://solutions.us.fujitsu.com/LOOX/.

Preston Gralla. 2011. *Motorola Xoom: The Missing Manual*. O'Reilly Media, Inc.

Laura M. Grupp, Adrian M. Caulfield, Joel Coburn, Steven Swanson, Eitan Yaakobi, Paul H. Siegel, and Jack K. Wolf. 2009. Characterizing flash memory: Anomalies, observations, and applications. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture*. 24–33.

M. Hosomi, H. Yamagishi, T. Yamamoto, K. Bessho, Y. Higo, K. Yamane, H. Yamada, M. Shoji, H. Hachino, C. Fukumoto, H. Nagao, and H. Kano. 2005. A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-RAM. In *Proceedings of the IEEE International Electron Devices Meeting Technical Digest*. 459–462.

HP. 2010. CACTI. http://www.hpl.hp.com/research/cacti/.

HTC. 2014. Desire 820. http://blog.htc.com/2014/09/htc-desire-820/.

Intel. 2013. Atom C2000. http://ark.intel.com/products/71269.

Intel. 2014. Atom Z3795. http://ark.intel.com/products/80267.

Intel. 2015. Core i7-5557U. http://ark.intel.com/products/84993/.

T. Ishigaki, T. Kawahara, R. Takemura, K. Ono, K. Ito, H. Matsuoka, and H. Ohno. 2010. A multi-level-cell spin-transfer torque memory with series-stacked magnetotunnel junctions. In *Proceedings of the Symposium on VLSI Technology*. 47–48.

Sanjay V. Kumar, Chris H. Kim, and Sachin S. Sapatnekar. 2006. Impact of NBTI on SRAM read stability and design for reliability. In *Proceedings of the IEEE International Symposium on Quality Electronic Design*. 210–218.

Jianhua Li, Liang Shi, Qingan Li, Chun Jason Xue, Yiran Chen, Yinlong Xu, and Wei Wang. 2013. Low-energy volatile STT-RAM cache design using cache-coherence-enabled adaptive refresh. *ACM Trans. Des. Automat. Electron. Syst.* 19, 1 (2013), 5:1–5:23.

Xiaohua Lou, Zheng Gao, Dimitar V. Dimitrov, and Michael X. Tang. 2008. Demonstration of multilevel cell spin transfer switching in MgO magnetic tunnel junctions. *Appl. Phys. Lett.* 93, 24 (2008), 242502–242503.

Peter S. Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, Gustav Hallberg, Johan Hogberg, Fredrik Larsson, Andreas Moestedt, and Bengt Werner. 2002. Simics: A full system simulation platform. *Computer* 35, 2 (2002), 50–58.

MediaTek. 2013. MT5692. http://event.mediatek.com/_en_octacore/.

nVIDIA. 2012. Tegra 2. http://www.nvidia.com/object/tegra-superchip.html.

nVIDIA. 2013. Tegra 4. http://www.nvidia.com/object/tegra-4-processor.html.

Qualcomm. 2013. Snapdragon 615. https://www.qualcomm.com/products/snapdragon/processors/615.

R. Sbiaa, R. Law, S. Y. H. Lua, E. L. Tan, T. Tahmasebi, C. C. Wang, and S. N. Piramanayagam. 2011. Spin transfer torque switching for multi-bit per cell magnetic memory with perpendicular anisotropy. *Appl. Phys. Lett.* 99, 9 (2011).

Mrigank Sharad, Rangharajan Venkatesan, Anand Raghunathan, and Kaushik Roy. 2013. Multi-level magnetic RAM using domain wall shift for energy-efficient, high-density caches. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 64–69.

Clinton W. Smullen, Vidyabhushan Mohan, Anurag Nigam, Sudhanva Gurumurthi, and Mircea R. Stan. 2011. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture*. 50–61.

Guangyu Sun, Huazhong Yang, and Yuan Xie. 2012. Performance/thermal-aware design of 3D-stacked L2 caches for CMPs. *ACM Trans. Des. Autom. Electron. Syst.* 17, 2 (2012), 13:1–13:20.

Guangyu Sun, Xiangyu Dong, Yuan Xie, Jian Li, and Yiran Chen. 2009. A novel architecture of the 3D stacked MRAM L2 cache for CMPs. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture*. 239–249.

Zhenyu Sun, Wenqing Wu, and Hai Li. 2013. Cross-layer racetrack memory design for ultra high density and low power consumption. In *Proceedings of the IEEE/ACM Design Automation Conference*. 1–6.

Dean M. Tullsen and Jeffery A. Brown. 2001. Handling long-latency loads in a simultaneous multithreading processor. In *Proceedings of the ACM/IEEE International Symposium on Microarchitecture*. 318–327.

Jue Wang, Xiangyu Dong, Yuan Xie, and Norman P. Jouppi. 2013. i^2WAP: Improving non-volatile cache lifetime by reducing inter-and intra-set write variations. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture*. 234–245.

Xiaoxia Wu, Jian Li, Lixin Zhang, Evan Speight, Ram Rajamony, and Yuan Xie. 2009. Hybrid cache architecture with disparate memory technologies. In *Proceedings of the International Symposium on Computer Architecture*. ACM, New York, NY, USA, 34–45.

Wei Xu, Yiran Chen, Xiaobin Wang, and Tong Zhang. 2009. Improving STT MRAM storage density through smaller-than-worst-case transistor sizing. In *Proceedings of the ACM/IEEE Design Automation Conference*. 87–90.

Bo Zhao, Jun Yang, Youtao Zhang, Yiran Chen, and Hai Li. 2013. Common-source-line array: An area efficient memory architecture for bipolar nonvolatile devices. *ACM Trans. Des. Autom. Electron. Syst.* 18, 4 (2013), 57:1–57:18.

Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. 2009. Energy reduction for STT-RAM using early write termination. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*. 264–268.

Received January 2015; revised April 2015; accepted n/a