Constructing Large and Fast Multi-Level Cell STT-MRAM based Cache for Embedded Processors *

Lei Jiang †, Bo Zhao †, Youtao Zhang ‡, Jun Yang †
† Electrical and Computer Engineering Department
University of Pittsburgh
Pittsburgh, PA 15261, USA
†{lej16,boz6,juy9}@pitt.edu

‡ Computer Science Department
University of Pittsburgh
Pittsburgh, PA 15260, USA
‡zhangyt@cs.pitt.edu

ABSTRACT

MLC STT-MRAM (Multi-level Cell Spin-Transfer Torque Magnetic RAM), an emerging non-volatile memory technology, has become a promising candidate to construct L2 caches for high-end embedded processors. However, the long write latency limits the effectiveness of MLC STT-MRAM based L2 caches. In this paper, we address this limitation with two novel designs: Line Pairing (LP) and Line Swapping (LS). LP forms fast cachelines by re-organizing MLC soft bits which are faster to write. LS dynamically stores frequently written data into these fast cachelines. Our experimental results show that LP and LS improve system performance by 15% and reduce energy consumption by 21%.

Categories and Subject Descriptors

B.3.2 [**Memory Structures**]: Design Styles

General Terms

Design, Performance

Keywords

MLC, STT-MRAM, LLC

1. INTRODUCTION

Modern embedded processors, in order to meet the fast growing computation and communication demands from end users, often integrate multiple cores with high computing capability and low power consumption. For example, high-end embedded processors such as nVIDIA Tegra [1] and Intel Atom [2] have been integrated in commercial tablet [3] and smartphone [4] products. Multi-core embedded processors require large on-chip caches to achieve scalable performance.

It is challenging to construct large caches that achieve both high performance and low power consumption. Many high-end embedded processors have 256KB∼2MB [1, 2] L2 SRAM caches. Choosing SRAM for future larger caches is less appealing due to SRAM's low density, serious leakage and reliability problems at nanoscale. Studies have been conducted to integrate emerging memory technologies, such as Phase-Change Memory (PCM) [5] and Spin-Transfer Torque magnetic RAM (STT-MRAM) [6], in the cache hierarchy. Among different memory technologies, STT-MRAM has been identified as a promising candidate for on-chip caches. STT-MRAM has no leakage from the memory cell and a comparable read speed as SRAM.Recently proposed Multi-level cell (MLC) STT-MRAM [7, 8, 9, 10] can store multiple bits per cell, reduce per-bit cost, and make STT-MRAM even more attractive for constructing large and low power onchip caches.

However, while MLC STT-MRAM improves density, it has almost doubled read and write latencies. This degrades the cache performance and diminishes the benefits gained from enlarged capacity. In this paper, we propose two novel designs, Line Pairing (LP) and Line Swapping (LS), to address the latency limitation and improve the effectiveness of MLC STT-MRAM based caches. The existing fabrication of a 2-bit MLC STT-MRAM cell includes a *hard-bit* and a *soft-bit* [7, 9]. The hard-bit is fast to read but slow to write, while the soft-bit is fast to write but slow to read. LP re-organizes two physical cachelines into a *read slow write fast* (RSWF) soft-bit cacheline and a *read fast write slow* (RFWS) hard-bit cacheline. By promoting frequently written data into soft-bit lines and frequently read data into hard-bit lines, LS greatly reduces L2 hit latency and improves the overall performance. We evaluate LP and LS using a set of different benchmark programs. Experimental results show that LP and LS can improve performance by 15% and reduce energy by 21% over the naive MLC STT-MRAM based L2 cache design on a high-end embedded processor platform.

In the rest of paper, 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 the paper in Section 7.

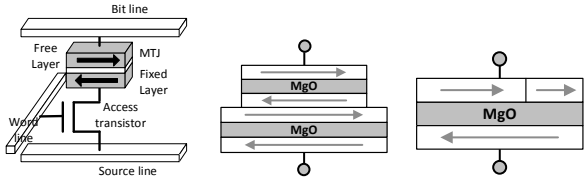
2. BACKGROUND

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 Junctions) to store binary information [11]. A MTJ consists of two ferromag-

*This work was supported in part by NSF CSR #1012070 and NSF CAREER #0747242.

DAC 2012, June 3-7, 2012, San Francisco, California, USA.

netic 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 a 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'.



(a) SLC STT Cell (b) Serial MLC (c) Parallel MLC

Figure 1: STT-MRAM cell structure of SLC and 2-bit MLC cells.

2.1 STT-MRAM MLC Structure

The advances in device research [7, 9] have enabled the fabrication of multi-level cell (MLC) STT-MRAM. There are two types of MLC cells in the literature (Figure 1(b) and 1(c)). A 2-bit *serial* MLC MTJ (Figure 1(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, e.g., Hitachi has announced the successful tapeout of serial MLC MTJs [7]. However, serial MTJ requires a large critical switching current [9], which increases onchip dynamic power of caches. Therefore, Seagate proposed another type of MLC MTJ, *parallel* MLC MTJ (Figure 1(c)), which utilizes a single MgO MTJ whose free layer has two fields [9]. Parallel MLC STT-MRAM has been adopted in onchip caches [8, 10]. In this paper, we assume parallel MLC STT-MRAM is used.

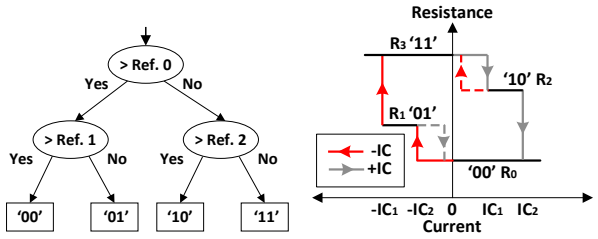
For both types of MLC cells, the two magnetic fields are switched at different spin-polarized currents. Their combinations form multiple resistance levels to represent multi-bit values. The field that requires large current to switch is referred to as *hard-bit*; the field that requires smaller current to switch is referred to as *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 endurance problem. Wear leveling and error correction based techniques [10, 12] have been proposed for MLC STT-MRAM based caches.

2.2 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 [8]. As shown in Figure 2(a), the read operation senses the resistance of a MLC MTJ and compares it to two of three resistance references. Each reference is generated by using two reference memory cells [7]. 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 a MLC cell is 1.5 times that of a SLC cell [8].

Two-step MLC write: Writing a 2-bit value into the cell is more complicate. In particular, two MTJs are controlled by one access transistor so that the write current



(a) two-step read (b) two-step write

Figure 2: Accessing MLC STT-MRAM cells.

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 2(b)), changing the hard-bit (MSB) always changes the soft-bit (LSB) to have the same magnetic field, i.e., either '00' or '11' is written. In the case if the 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 in writing each bit is comparable to that of SLC, the latency of a 2-bit MLC write roughly doubles that of a SLC write. MLC cell write can terminate early in some cases: (i) if the LSB is the 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 [8]. However, when writing a cacheline (64B), it is less likely that the write can be terminated early as most likely at least one cell requires both steps.

3. PROPOSED TECHNIQUES

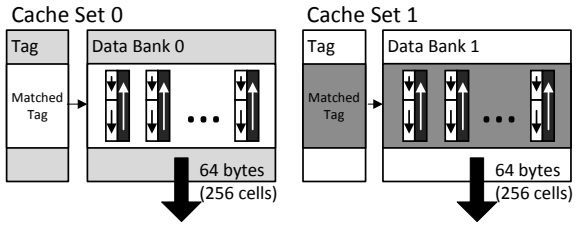
In this section, we elaborate the design details of *Line Pairing* (LP) and *Line Swapping* (LS). Our goal is to reduce the effective hit latency of MLC STT-MRAM caches.

3.1 Line Pairing

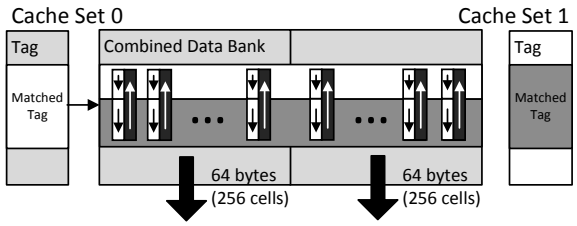
Given a 64B cacheline, one access to the data array activates 256 2-bit MLC cells in one bank. Each 2-bit MLC cell consists of one hard-bit and one soft-bit. The hard-bit is fast to read but slow to write while the soft-bit is fast to write but slow to read. The access latency of a cacheline is determined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

Here we use the cache parameters in Table 1 and 2 for discussion purpose. The design is applicable to caches with different cache sizes and line sizes. In this paper, we assume the two-phase cache access scheme that is widely adopted for highly associative caches, i.e., 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 Line Pairing (LP) scheme to combine cells from two physical cachelines (having the same access latency) and re-arrange 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 3, while each physical line has both hard- and soft- bits, a logical cacheline has either all hard-bits or all soft-bits, but not mixed. The cacheline that only has soft-bits is referred to as a RSWF line; the cacheline that only uses hard-bits is referred to as a RFWS line. Within each cache set, half of cachelines are



(a) each cacheline contains **both hard bits and soft bits**



(b) after Line Pairing, each cacheline contains **either hard bits or soft bits**

Figure 3: Employ Line Pairing (LP) to reduce hit latency. (After LP, each cache set is stored across two banks but requires half of the original storage space in each bank)

RSWF while others are RFWS. A one-bit flag per cacheline is introduced in the tag array to indicate if the line is a RSWF or RFWS line. After tag access, LP knows the type of the cacheline.

By constructing RSWF lines, we can greatly speedup write operations when they fall in these lines. The latency is 19 cycles, which is only ∼50% of the write latency in baseline design (Table 2). However, LP penalizes the writes to RFWS lines. As discussed in Section 2.2, 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 co-located 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.

To reduce the hardware cost, LP uses the same number of port (i.e., read/write circuits) as 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 cacheline into two banks. As shown in Figure 3, *cache set 0* and *set 1* are stored in *data bank 0* and *bank 1* before LP. After 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 studied the overall impact and the results (as shown in Section 5) showed that L2 caches are frequent but do not have many overlaps such that LP speedups the overall performance.

3.2 Line Swapping

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

For a simple implementation of LS, a write-intensive line may be placed in a RFWS line initially; when it is identified as write intensive, it is always swapped to a RSWF line. A read-intensive line may be swapped from a RSWF line to a RFWS line as well. Clearly, too many such swaps increase the number of write operations, incur more bank contentions, and consume 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 if a line should be swapped: the one added to RSWF (or RFWS) line indicates if 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 4 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 a RFWS line has a write hit, or a 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 a LRU (least recently used) RFWS line, or the current RFWS line with a LRU RSWF line. If a miss happens and the victim line is a RFWS line, LS moves the LRU RSWF line here and loads the new data to the RSWF line location.

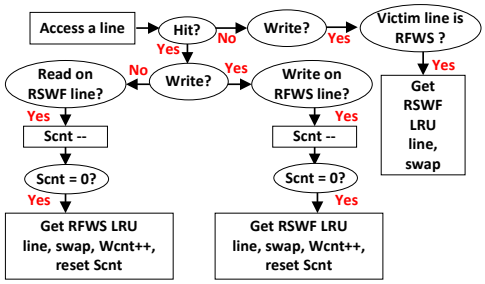


Figure 4: The line swapping (LS) algorithm.

The hardware cost is very modest. Our results showed that using a total of 6 bits per line (2 bits for `Wcnt` and 4 bits for `Scnt`) makes most accesses fall in corresponding fast lines. The storage overhead is ∼1% (42 tag bits and 512 data bits). Note that supporting line swapping does not require extra buffer to hold extracted lines — the baseline has two 512-bit buffers for two banks, which are enough to hold two lines to enable LS.

4. EXPERIMENT METHODOLOGY

Simulator: We evaluated our proposed schemes using Virtutech Simics [13] simulation environment. We upgraded Simics *g-cache* cache module to simulate bank conflicts, bus contention, RFWS/RSWF MLC STT-MRAM cache lines, Line Pairing (LP) and Line Swapping (LS).

Baseline Modeling: Table 1 summarizes the core and cache configuration that we simulated in this paper. We modeled Intel Atom D525 processor [2] that 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 L2 cache is shared. MESI snoopy protocol is deployed to maintain the cache coherence. If a 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 L2 cache, we summarized L2 cache tag lookup latency, read/write hit latency and energy consumption in Table 2.

L2 Cache Modeling: Intel Atom D525 is equipped with 1MB shared SRAM L2 cache [2]. We modeled such a cache by CACTI [14] and obtained the baseline L2 cache area overhead — $5.1mm^2$. By assuming a SLC STT-MRAM cell size of $14F^2$ [15], we calculated that 5MB SLC can fit in a $5.1mm^2$ area overhead under $45nm$ technology. Since the area of a STT-MRAM cell is dominated by its access transistor, the size of MLC cell is slightly bigger than that of SLC cell [10] — 4M cells (or 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. Comparing 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 [5].

We used CACTI [14] to model STT-MRAM dynamic energy based on the reported results [8, 10, 16]. The write energy of SLC STT-MRAM is close to that of a soft bit for MLC STT-MRAM.

Processor	Intel Atom, 2-core, 1.8GHz, single-issue, in-order
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

Table 1: Baseline Chip Configuration.

Type	Latency (cycles)	Dynamic Energy (64B) (nJ)	Leakage Power (W)
SRAM (1MB)	tag lookup: 1 data hit: 3	0.31	1.354
eDRAM (4MB)	tag lookup: 3 data hit: 5	0.51	0.396 refresh
SLC STTMRAM (5MB)	tag lookup: 2 R-hit: 3 W-hit: 19	read: 0.32 write: 1.29	0.156
MLC STTMRAM (8MB)	tag lookup: 3 R-hit: 5 W-hit: 37	read :0.32 write: 1.58	0.152
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

Table 2: L2 Cache Configuration.

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 [1, 2], we did not constrain our selections in multimedia benchmarks. Instead, we picked up various workloads from SPEC-CPU2006, SPEC-OMP2001 and PARSEC to study memory accesses for embedded cores. Table 3 lists the workload details.

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

Table 3: Simulated Benchmarks.

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.
- `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

Figure 5 compares the performance using different schemes. We report the weighted speedup that is computed as follows [17].

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

On average, `LP` and `LP+LS` achieve 7% and 14.8% performance improvements, respectively, over `Baseline`. From Figure 5, `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 opportunties — most reads fall in RFWS lines while most writes fall in RSWF lines.

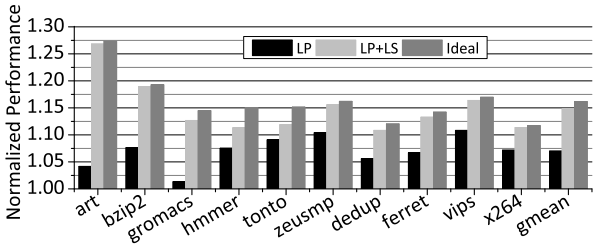


Figure 5: Performance improvement (Normalized to `baseline`).

We discussed in Section 3.1 that `LP` trades the maximal parallelizable accesses for short hit latency. From Figure 5,

we observed 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.

5.2 Energy Reduction

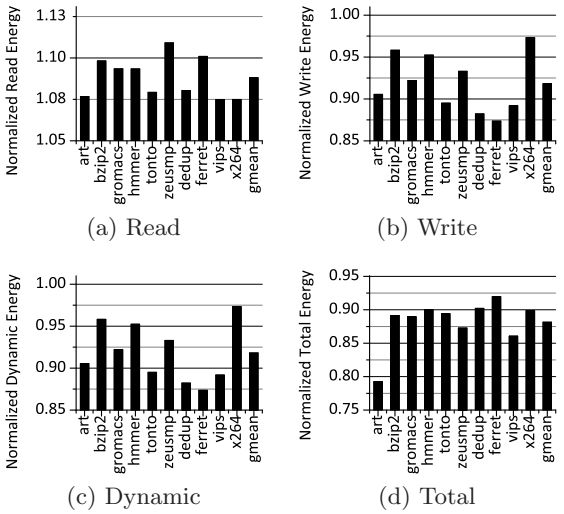
We next studied energy consumption of different schemes. As `LP` needs to activate more banks per cache access, and penalize more if writes fall into RFWS lines, our design may consume more energy. We used CACTI [14] to model the bank activation power with details reported in Table 2.

Read Energy: 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 6(a).

Write Energy: Table 2 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. Figure 6(b) shows that on average `LP+LS` reduces 8% write energy over `baseline`. With `LS`, most writes happen on RSWF lines such that write energy consumption is greatly reduced.

Dynamic Energy: Figure 6(c) 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 was also reported in recent studies [16, 18].

Total Energy: `LP+LS` spends less CPU cycles in executing the same number of instructions, which results in less leakage energy. Given leakage energy dominates the total energy consumption, Figure 6(d) shows that on average `LP+LS` reduces 12% total energy consumption.



(a) Read (b) Write

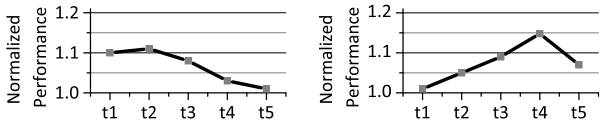
(c) Dynamic (d) Total

Figure 6: Energy comparison (Normalized to `baseline`).

5.3 Line Swapping Overhead

Since `LS` introduces both latency and energy overheads, we prefer less swaps at runtime. The number of swaps depends on how effective we can find read-/write-intensive lines and if 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.

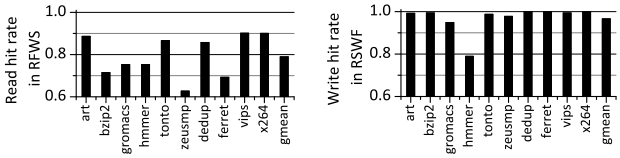
Figure 7(a) summarizes the performance improvement when adjusting the initial value of `Scnt` from 1 to 5. Initializing `Scnt` of RFWS lines to 2 achieves the best performance, which shows that the space locality of writes is strong — a swap should be conducted if the RFWS line gets the second write hit. We then studied read-triggered swaps. Figure 7(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 worthy to pay the overhead if a line to be swapped will be read only once or twice in the future. If the data has been read for 4 times, it is very likely for it to receive more reads. Swapping such kind of data into RFWS lines reduces both the effective read latency and the number of unnecessary swaps.



(a) Write Threshold. (b) Read Threshold.

Figure 7: Performance geomean comparison when using different `Scnt` swap thresholds (Normalized to `baseline`).

We thus set `Scnt` to be 4 for RSWF lines and to be 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 bit 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 8(b) reports that 97% writes hit in RSWF lines. Figure 8(a) shows the read hit rate in RFWS lines — on average, 79% 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.



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

Figure 8: Access hit rate in fast cachelines.

5.4 Various Memory Technologies

The last experiment that we conducted was to compare SRAM, eDRAM, SLC STT-MRAM and MLC STT-MRAM caches, under the same die area constraint. The configuration details are in Table 2.

Performance: Figure 9 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 [5]. For most benchmark we simulated, 512KB per core SRAM cache can not hold their working sets. Too many offchip 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.

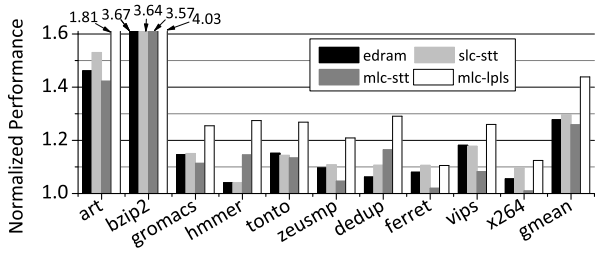


Figure 9: Performance comparison (Normalized to SRAM performance).

Total energy: Figure 10 compares the total energy when using SRAM, eDRAM, SLC STT-MRAM based caches. For all configurations, leakage energy dominates the total energy. It is not surprising that 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.

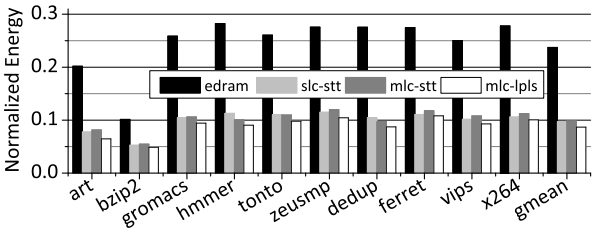


Figure 10: Total energy comparison (Normalized to SRAM total energy).

6. RELATED WORK

As a promising non-volatile memory, STT-MRAM has been proposed as SRAM or DRAM replacement for onchip caches. Previous works [5, 6, 19] presented a 3D-stacked processor architecture with onchip SLC STT-MRAM cache and compared SLC STT-MRAM to other different memory technologies for onchip caches. Zhou *et al.* designed early write termination to save write energy when the bit value is not changed [16].

Multi-level cell (MLC) STT-MRAM is a promising approach to improve the density of STT-MRAM [8, 7, 9, 10]. Ishigaki *et al.* proposed serial MLC MTJ and studied two-step read and write schemes [8]. Lou *et al.* presented parallel MLC MTJ and compared it to serial MLC STT-MRAM [9].

Recent studies of MLC flash devices have also found that some MLC flash pages are as fast as SLC pages while others are slow [20]. 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 paper shares similarity except that LP is performed at line granularity. A big difference between our design and Flash designs is that we developed Line Swapping (LS) that can greatly improve system performance and reduce the energy overhead.

7. CONCLUSIONS

Integrating a large and fast onchip 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 onchip caches. However, MLC STT-MRAM suffers from long read and write access latencies. In this paper, we proposed two novel schemes: line pairing (LP) and line swapping (LS), to reduce average cache hit latency of MLC STT-MRAM based L2 caches. Our experimental results showed that these schemes are effective and on average achieve 15% performance improvement and 21% energy reduction over the baseline design.

8. REFERENCES

[1] nVIDIA Tegra. http://www.nvidia.com/object/tegra-superchip.html.

[2] Intel Atom D525. http://ark.intel.com/products/49490.

[3] P. Gralla. Motorola Xoom: The Missing Manual. In *O'Reilly Media*, 2011.

[4] Fujitsu LOOX. http://solutions.us.fujitsu.com/LOOX/.

[5] X. Wu et al. Hybrid Cache Architecture with Disparate Memory Technologies. In *ISCA*, 2009.

[6] X. Dong et al. Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) as a Universal Memory Replacement. In *DAC*, 2008.

[7] T. Ishigaki et al. A Multi-level-cell Spin-transfer Torque Memory with Series-stacked Magnetotunnel Junctions. In *Symposium on VLSI Technology*, 2010.

[8] Y. Chen et al. Access Scheme of Multi-Level Cell Spin-Transfer Torque Random Access Memory and Its Optimization. In *IEEE International Midwest Symposium on Circuits and Systems*, 2010.

[9] X. Lou et al. Demonstration of multilevel cell spin transfer switching in mgo magnetic tunnel junctions. *Applied Physics Letters*, 2008.

[10] Y. Chen et al. Processor Caches with Multi-level Spin-transfer Torque RAM Cells. In *ISLPED*, 2011.

[11] M. Hosomi et al. A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-ram. In *IEDM*, 2005.

[12] W. Xu et al. Improving STT MRAM Storage Density through Smaller-than-worst-case Transistor Sizing. In *DAC*, 2009.

[13] Virtutech Simics. http://www.virtutech.com.

[14] HP CACTI. http://www.hpl.hp.com/research/cacti/.

[15] S. Chung et al. Fully integrated 54nm STT-RAM with the Smallest Bit Cell Dimension for High Density Memory Application. In *IEDM*, 2010.

[16] P. Zhou et al. Energy Reduction for STT-RAM using Early Write Termination. In *ICCAD*, 2009.

[17] D. Tullsen and J. Brown. Handling Long-latency Loads in a Simultaneous Multithreading Processor. In *MICRO*, 2001.

[18] C. Smullen et al. Relaxing Non-volatility for Fast and Energy-efficient STT-RAM Caches. In *HPCA*, 2011.

[19] G. Sun et al. A Novel Architecture of the 3D Stacked MRAM L2 Cache for CMPs. In *HPCA*, 2009.

[20] L. Grupp et al. Characterizing Flash Memory: Anomalies, Observations, and Applications. In *MICRO*, 2009.