Building Energy-Efficient Multi-Level Cell STT-RAM Caches with Data Compression

Liu Liu[*], Ping Chi, Shuangchen Li, Yuanqing Cheng, Yuan Xie
Department of Electrical and Computer Engineering
University of California, Santa Barbara
{liu_liu, pingchi, shuangchenli, yuanqing, yuanxie}@ece.ucsb.edu

ABSTRACT

Spin-transfer torque magnetic random access memory (STT-RAM) technology has emerged as a potential replacement of SRAM in cache design, especially for building large-scale and energy-efficient last level caches. Compared with single-level cell (SLC), multi-level cell (MLC) STT-RAM is expected to double cache capacity and increase system performance. However, the two-step read/write access schemes incur considerable energy consumption and performance degradation. In this paper, we propose two techniques using data compression to optimize MLC STT-RAM cache design. The first technique tries to compress a cache line and fit it into only the soft-bit region of the cells, so that reading or writing this cache line takes only one step which is fast and energy-efficient. We introduce a second technique to increase the cache capacity by enabling the left hard-bit region to store another compressed cache line, which can improve the system performance for memory intensive workloads. The experimental results show that, compared with a conventional MLC STT-RAM last level cache design, our overhead minimized technique reduces the dynamic energy consumption by 38.2% on average with the same system performance, and our capacity augmented technique boosts the system performance by 6.1% with 19.2% dynamic energy saving on average, across the evaluated multi-programmed benchmarks.

1. INTRODUCTION

In modern chip multiprocessors, on-chip caches account for a large portion of chip area and power consumption. SRAM has been the common embodiments of different levels of on-chip caches thanks to its fast access speed. However, as technology scales, its increasing leakage power and reduced reliability have attracted increasing concern. In recent years, various emerging nonvolatile memory (NVM) technologies have been developed and studied by the industry and the academia, such as phase change memory, metal-oxide resistive memory, and spin-transfer torque magnetic random access memory (STT-RAM), which have zero leakage power. Among these NVM technologies, STT-RAM has demonstrated a great potential to replace SRAM for building next-generation on-chip caches.

Compared with SRAM, STT-RAM has similar read access speed and read energy consumption, 3-4 times higher density, zero leakage power, and better scalability [10]. However, it suffers from longer write latency and higher write energy. Most of the previous work focused on single level cell (SLC) STT-RAM based cache design [16]. With multi-level cell (MLC) technology, each STT-RAM cell can store two bits, which enhances its density, making STT-RAM more attractive in constructing large-scale and energy-efficient last level caches (LLC) [6].

However, MLC STT-RAM requires two-step read and write operations, and therefore has longer access latency and higher access energy than SLC. It deteriorates the write performance and write energy consumption of STT-RAM, and can potentially offset the benefits from the enlarged capacity and even degrade system performance. Between the parallel and series MLC designs, the series one has been considered to be more feasible [18]. In series MLC design, each cell stores a soft-bit and a hard-bit; accessing the soft-bit takes one step, which is fast and energy-efficient; however, accessing the hard-bit is two-step, which is slow and consumes a lot of energy. Leveraging the differences between soft-bits and hard-bits, we propose an optimized MLC STT-RAM LLC design using data compression in this work. With data compression, we can compress a cache line into half the size and fit it into only the soft-bit region of the cells, reducing the number of slow and energy-consuming two-step accesses. We propose two techniques that operate data at the granularity of cache ways. The first one is overhead optimized, which involves negligible hardware and scheduling overhead. The experimental results show that it significantly reduces the energy consumption of MLC STT-RAM LLC without performance loss. The second technique enhances the effective cache capacity by utilizing the left hard-bit region of a compressed cache line to store an additional compressed line if possible. Besides saving energy, this technique also improves system performance due to the increased cache capacity and the reduced off-chip memory accesses.

2. BACKGROUND AND RELATED WORK

In this section, we first introduce the basics of STT-RAM, and then present the related work.

2.1 The Basics of STT-RAM

STT-RAM represents data by the resistance of the magnetic tunnel junction (MTJ) in each cell. Figure 1(a) shows the most popular one-transistor-one-MTJ (1T1J) cell structure, in which the transistor is used to select the MTJ by activating the word line. An MTJ consists of three layers: two ferromagnetic layers called free layer and reference layer respectively, and one oxide barrier layer in the middle. The magnetization direction of the reference layer is fixed, while that of the free layer can be changed by applying a spin polarized current through the MTJ. The resistance of an MTJ is determined by the relative magnetization directions of the two ferromagnetic layers. As shown in Figure 1(b), when they are in parallel, the MTJ has low resistance and

[*]This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Award number DE-SC0013553 with disclaimer at http://seal.ece.ucsb.edu/doe/. It was also supported in part by NSF 1500848 and 1533933, and a grant from Qualcomm.

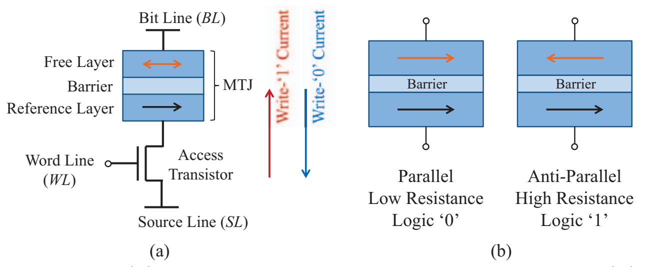


Figure 1: (a) 1T1J SLC STT-RAM cell structure; (b) in-plane MTJs in low and high resistance states.

represents logic '0'; in contrast, when they are anti-parallel, the MTJ is in a high resistance state and denotes logic '1'. The direction of the applied switching current determines to write '0' or '1', as shown in Figure 1(a). The magnetization directions of the MTJs in Figure 1 are in-plane, and they can also be made perpendicular [11]. As the technology node scales, perpendicular MTJ based STT-RAM demonstrates better characteristics than in-plane MTJ based STT-RAM. It requires a lower switching current and can maintain high thermal stability for a longer retention time.

The cell in Figure 1 stores a single logic bit, called single level cell (SLC). To enhance the density of STT-RAM, multi-level cell (MLC) structures have been proposed, holding multiple logic bits in a single cell. There are two categories of MLC MTJ designs, parallel [13] and series [12]. In parallel MLC MTJs, the free layer has two domains to achieve four resistance states by the combinations of their magnetization directions; while series MLC STT-RAM stacks two MTJs to represent two logic bits. A recent study demonstrates that series MLC STT-RAM is more feasible than parallel, because parallel MLC STT-RAM design is only applicable to in-plane MTJ technology while series design is compatible with advanced MTJ technologies such as perpendicular MTJ and has overwhelming advantage in read and write reliability [18]. This work focuses on series MLC STT-RAM.

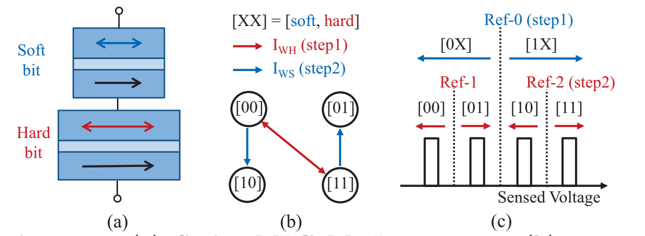


Figure 2: (a) Series MLC MTJ structure; (b) 2-step write operation; (c) 2-step read operation.

Figure 2(a) depicts the series MLC MTJ structure. The two MTJs have different areas in order to distinguish two logic bits. The bits stored in the smaller and bigger MTJs are called soft-bit and hard-bit, respectively. Given both constant resistance-area product and critical switching current density (I_C), the soft-bit has a larger resistance so that it is the more significant bit, and it requires a smaller switching current than the hard-bit, i.e. $I_{C,soft} < I_{C,hard}$. Figure 2(b) and (c) illustrate the write and read operations of series MLC STT-RAM. They both have two steps. As shown in Figure 2(b), in a write operation, first a large current I_{WH} ($I_{WH} > I_{C,hard}$) is applied to switch both the hard-bit and the soft-bit; next a smaller current I_{WS} ($I_{C,soft} < I_{WS} < I_{C,hard}$) is used to flip only the soft-bit. As shown in Figure 2(c), a read operation based on voltage sensing requires three reference voltages (Ref-0, Ref-1, and Ref-2) and two comparisons. The soft-bit is first detected by comparing the sensing voltage with Ref-0; then based on

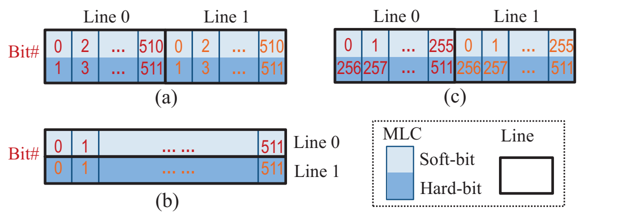


Figure 3: MLC data mapping methods for a 64-byte (512-bit) cache line: (a) direct mapping; (b) cell split mapping; (c) interleaved mapping.

the result, we read the hard-bit by comparing the sensing voltage with either Ref-1 or Ref-2.

Because of more complex read and write operations, MLC STT-RAM has longer read and write latencies and consumes higher read and write energy than SLC STT-RAM, which may degrade system performance and energy efficiency despite its capacity benefits. We notice that, regardless of the hard-bit, reading the soft-bit takes only one step; moreover, writing the soft-bit requires only a small switching current which will not flip the hard-bit. Therefore, MLC STT-RAM can perform like SLC by working only on the soft-bits, which improves the access speed but reduces the capacity.

2.2 Data Mapping for MLC Based Cache

To adopt MLC STT-RAM for cache, one key design issue is how to map the data bits of a cache line to MLCs. A straight-forward method is *direct mapping (DM)*, as shown in Figure 3(a). Given a 64B (512-bit) cache line, it uses 256 2-bit MLCs to hold a line and each cell stores two adjacent bits. Thanks to its simplicity, DM has been adopted in some previous work [5, 7]. However, since DM does not differentiate soft-bit and hard-bit regions, the read and write latencies are the worst-case two-step latencies no matter which word of a line is accessed.

To take advantage of the fact that soft-bit region is fast, Bi *et al.* proposed *cell split mapping (CSM)* [3] as shown in Figure 3(b), in which a cache line is stored either in all the soft-bits or in all the hard-bits of the cells. The soft-bit line and the corresponding hard-bit line are a fast way and a slow way respectively of the same set. To make best use of fast ways, a data migration mechanism is designed including an inter-cell swapping scheme and the migration policies. To further improve the access speed, they also propose an application-aware speed enhancement (ASE) mode for MLC STT-RAM based caches, in which MLC STT-RAM can work as fast SLC at a set level of granularity when applications do not benefit much from the large capacity of each set offered by MLC. Each cache set can dynamically halve or double the number of ways by switching between MLC mode and ASE mode according to the result of a mode-predictor.

Interleaved mapping (IM) is another data mapping method [15]. As illustrated in Figure 3(c), the lower half of the data bits of a line are stored in the soft-bit region while the higher half are put in the corresponding hard-bit region. Based on IM, Wang *et al.* proposed a dynamic block size technique (DBS) to optimize MLC STT-RAM based caches. Each cache set can operate in two modes: large block mode (LBM) and small block mode (SBM), as shown in Figure 4(a). LBM is a normal mode, utilizing both the soft-bit region and the hard-bit region to store each line, while SBM uses only the soft-bit region to store the hot data chunks of a line which has half the size of a line. Therefore, SBM is faster and consumes less energy than LBM. The dynamic block size mechanism can reconfigure the block sizes of each cache set at

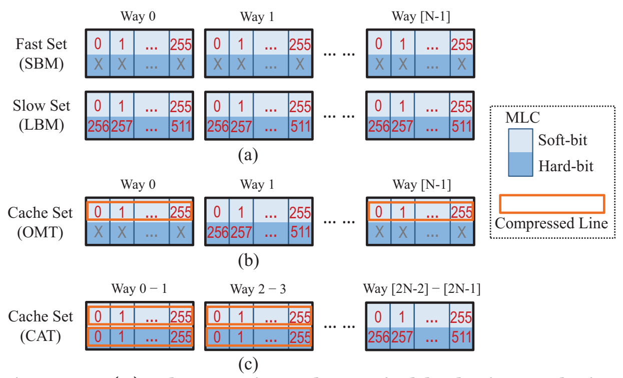


Figure 4: (a) The previous dynamic block size technique (DBS); (b) the proposed overhead minimized technique (OMT); (c) the proposed capacity augmented technique (CAT).

runtime according to their proposed block size reconfiguration policy. Their experiment results show that, the dynamic LBM-SBM switching mechanism based on IM surpasses the dynamic MLC-ASE mode switching mechanism based on CSM [3], achieving about 1% more IPC improvement and 5% more energy saving over the LBM only baseline.

3. THE PROPOSED DESIGN

This section first gives an overview of our design, and then introduces data compression used in our design. Next, the details of two proposed techniques are presented, respectively.

3.1 Design Overview

In our MLC STT-RAM cache design, we adopt the interleaved mapping method (Figure 3 (c)) for data mapping. However, unlike the previous work using dynamic block sizes [15], this work fixes the data block size, and employs data compression to fit a compressible line into only the soft-bit region or the hard-bit region of the cells.

We propose two techniques based on data compression. One is the Overhead Minimized Technique (OMT) as shown in Figure 4(b), in which a compressible line is put only into the soft-bit region and the corresponding hard-bit region is not used so that the change of the cache management and the modification of the tag arrays are minimized. The other is the Capacity Augmented Technique (CAT) as shown in Figure 4(c), in which a compressible line can be put into the soft-bit region as well as the hard-bit region so that the capacity of the cache is enhanced thanks to data compression. Compared with the dynamic block size technique (DBS) as shown in Figure 4(a), OMT supports a finer granularity of fast and slow ways. It does not require one whole cache set to operate in either fast way mode (SBM) or slow way mode (LBM); however, each way in a cache set can be a fast way as long as the data can be compressed into half the size and fit into the soft-bit region. Moreover, OMT is more lightweight than DBS by avoiding the hardware and scheduling overheads of dynamic block size reconfiguration. Compared with DBS, OMT and CAT provide a larger cache capacity, because the soft-bit region can store all the data of a line in a compressed fashion and in CAT the left hard-bit region can be used to store another compressible line.

3.2 Data Compression

The effectiveness of our techniques depends on how many cache lines are compressible. In this work, a line is "compressible" specifically means that it can be compressed into half its size so that it can be fit into only the soft-bit region or the hard-bit region. A number of cache compression schemes have been proposed which exploit various data compression methods to expand the effective cache capacity. For example, Zero-Content Augmented caches represent zero-value lines in a very compact way [9]. Also, Frequent Value Compression [17] and Frequent Pattern Compression [1] have been proposed and utilized in cache designs. Additionally, Base-Delta-Immediate (BΔI) Compression [14] has been widely adopted as a practical data compression method for on-chip caches thanks to its high compression ratio, low decompression latency, and modest hardware complexity. Huffman coding based statistical compression has also been explored for cache compression [2].

Our proposed techniques for MLC STT-RAM cache design do not rely on any specific compression method. In this work, we take the state-of-the-art BΔI compression as an example, and integrate it into our cache design. BΔI compression leverages the fact that the values within a cache line have a low dynamic range, and therefore presents a cache line using one or multiple base values and an array of differences ("deltas") whose combined size is smaller than the original cache line [14]. We use two bases, one of which is default zero, according to the suggested best option. Assuming that the cache line size is 64 bytes, we define our BΔI encoding in Table 1.

Our compressor consists of three compression units: one zero compression unit, one 8-byte-base 1-byte-delta (Base8-Δ1) compression unit, and one 8-byte-base 2-byte-delta (Base8-Δ2) compression unit. Therefore, the encoding needs 2 bits, as shown in Table 1, and they are attached to the corresponding tag for each cache line. All these compression units operate in parallel, and a selection logic chooses the optimal one if multiple compression options are available for a cache line. In the previous BΔI compressor design [14], it contains other BΔI compression units of different base and delta sizes, *e.g.* 8-byte-base 4-byte-delta, 4-byte-base 1-byte-delta, and so on. The reasons why we only choose Base8-Δ1 and Base8-Δ2 are: i) they can compress a cache line into half the size; ii) they can achieve almost the best compression ratio of BΔI compression; and iii) the hardware overhead is greatly reduced. In Table 1, the compressed cache line size for Base8-Δ1 is 17 bytes, including one 8-byte base, eight 1-byte deltas, and one byte each bit of which indicates the base for each segmentation (or delta) is either the default zero base or the other base. Similarly, the compressed cache line size for Base8-Δ2 is 25 bytes.

We simulated a 3-level cache hierarchy as described in Table 2, and profiled the data in the last level cache over the SPEC CPU2006 benchmarks. Figure 6 shows the percentages of the compressible lines and the breakdowns of the compression schemes across 15 selected benchmarks. On average, 42% of the target cache line data are compressible. From the breakdown results we can find that, for some benchmarks such as *zeusmp* and *GemsFDTD*, more lines are compressed by the "Zeros" scheme encoded as "00", and for some other benchmarks such as *milc* and *lbm*, more lines are compressed by the BΔI schemes including Base8-Δ1 and Base8-Δ2 encoded as "01" and "10".

3.3 The Overhead Minimized Technique

As shown in Figure 4(b), with OMT, each way in a cache set can be a fast way or a slow way, depending on whether the data line can be compressed into half the size. As men-

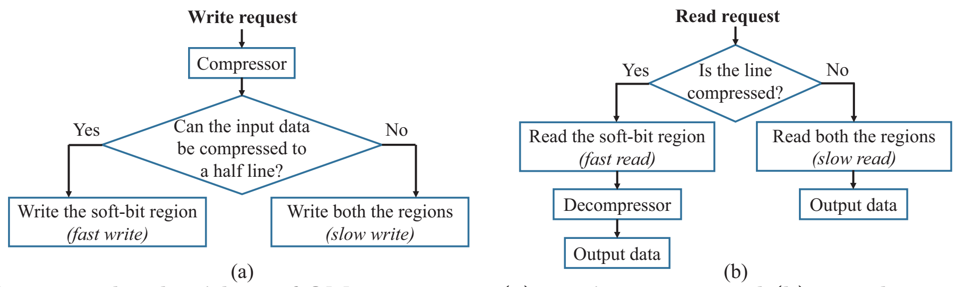

Write request
Compressor
Can the input data be compressed to a half line?
Yes
No
Write the soft-bit region (fast write)
Write both the regions (slow write)
(a)
Read request
Is the line compressed?
Yes
No
Read the soft-bit region (fast read)
Read both the regions (slow read)
Decompressor
Output data
Output data
(b)

Figure 5: The algorithms of OMT to process (a) a write request and (b) a read request.

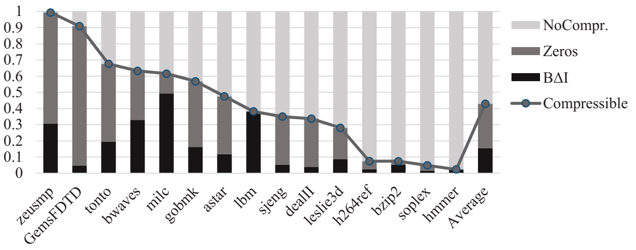


Figure 6: The profiling results over SPEC CPU2006 benchmarks: the percentages of compressible lines and the breakdowns of compression schemes.

Table 1: BΔI encoding.

Name	Base	Δ	Size	Encoding
Zeros	1B	0B	1B	00
Base8-Δ1	8B	1B	17B	01
Base8-Δ2	8B	2B	25B	10
No Compression	N/A	N/A	64B	11

tioned above, we attach a 2-bit compression scheme code to the tag for each cache line which indicates whether the line is compressed and which compression scheme is used.

Figure 5 describes how OMT processes write and read requests. For a write request, as shown in Figure 5(a), first the input data are processed by the compressor. Then, based on the compression scheme code, it is determined whether the input data have been compressed. If yes, the compressed input data are written into the soft-bit region, and it is a fast and energy-efficient write operation; otherwise, the input data are written into both the soft-bit and the hard-bit regions, and it is a slow and energy-consuming write operation. The corresponding compression scheme code in the tag arrays is also updated. For a read request, as shown in Figure 5(b), first the compression scheme code in the tag arrays is checked to determine whether the line is compressed or not. If yes, the compressed data are read out from the soft-bit region through a fast read operation, and then decompressed by the decompressor according to the compression scheme code before output; otherwise, the data are read out from both the soft-bit and the hard-bit regions through a slow read operation, and then output.

3.4 The Capacity Augmented Technique

As presented above, OMT takes advantage of data compression to reduce the access latency and energy to MLC STT-RAM. However, it does not leverage the capacity benefits from data compression. We further propose CAT, which increases the cache capacity by using the saved hard-bit regions to accommodate more compressible lines of data. As shown in Figure 4(c), two compressible lines can reside in the soft-bit region and the hard-bit region of the cells simultaneously. Therefore, the maximum number of ways that each cache set can hold doubles.

To fully utilize the capacity benefits requires doubling the tags, which incurs a considerable overhead. To reduce the tag overhead, we may limit the maximum number of ways of each cache set. According to our profiling results of the percentage of compressible lines as shown in Figure 6, increasing the tags by 50% is a reasonable choice. We design a simple tag-data mapping based on the conventional one, in which the original tags are directly mapped to the original data positions while the added tags can be mapped to any hard-bit region of the cache set. Therefore, each added tag needs to store its data position. For example, it requires 4 additional bits per added tag to store 16 possible hard-bit region positions in an originally 16-way cache. Moreover, the cache replacement policy needs to be modified since some ways are compressed while others are not. We implement a very simple least recently used (LRU) policy, which chooses the LRU compressed line or the LRU in-compressed line for replacement according to whether the incoming line is compressible or not. We leave the study of more intelligent cache policies for the future work.

Table 2: Processor and Memory Configurations

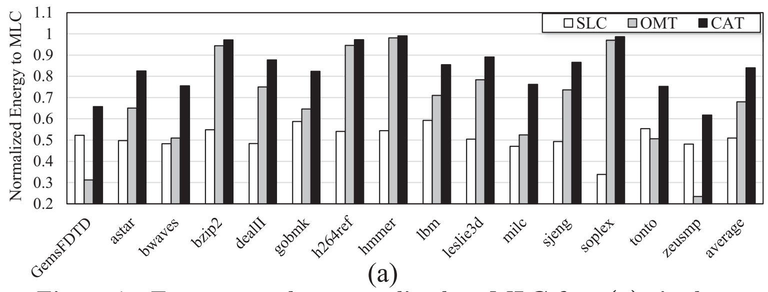
Processor	1-/4-core, alpha ,4GHz, out-of-order, 8-issue
L1 cache SRAM	private, 32KB I/D, 64B line, 4-way 2-cycle latency
L2 cache SRAM	private, 256KB, 64B line, 8-way 5-cycle latency
L3 cache STT-RAM	shared, 64B line, 16-way 4MB SLC, 8MB MLC
Main memory DRAM	8GB, DDR3-1600, 64bit I/O, 8 banks tCL-tRCD-tRP-tWR: 11-11-11-12

4. EVALUATIONS

In this section, we first describe our experimental methodology and then evaluate the system performance and the energy reduction achieved by the proposed scheme.

4.1 Experimental Methodology

In this work, our evaluation is based on the gem5 full system simulator [4] with modification to the cache implementation to simulate the architectural design. Our modification to gem5 includes an asymmetric cache read/write latency model and integrated (de)compression unit. The processor and memory configurations are summarized in Table 2. The proposed designs are applied on L3 cache, which uses SRAM for tag array and STT-RAM for data array. We simulate both single-core and four-core systems with DDR3-1600 as the referenced memory model. We choose 15 benchmarks from the SPEC CPU2006 benchmark suite for single-core simulation. In addition, we select 10 benchmarks with a wide range of different compression ratio and mix them to 10 groups of 4 for multi-programmed workload simulation as shown in Table 4. We use accesses per kilo instructions (APKI) as a metric to measure the access behavior of LLC, and list the APKI of different workloads in Table 4 We fast-forward one billion instructions, and execute 5 billion instructions. Note that we select at least one



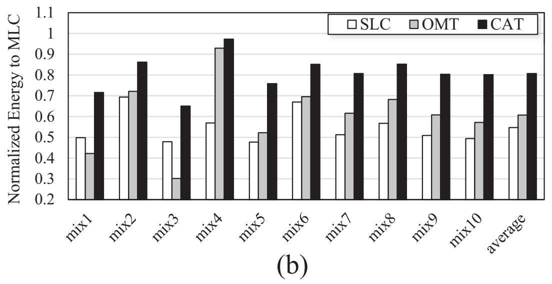
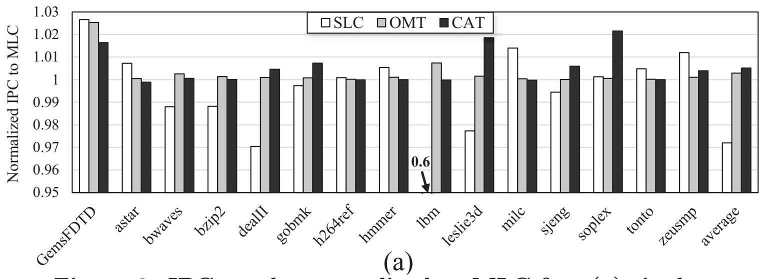


Figure 7: Energy results normalized to MLC for: (a) single-programmed and (b) multi-programmed workloads.



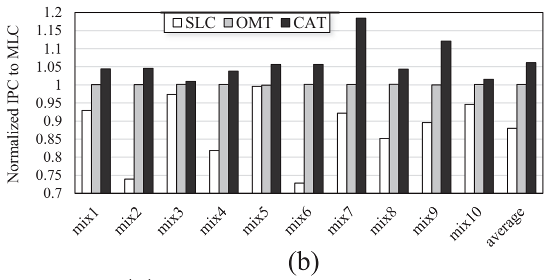
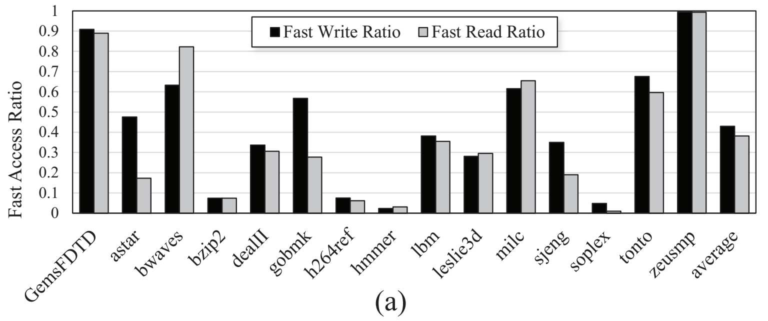


Figure 8: IPC results normalized to MLC for: (a) single-programmed and (b) multi-programmed workloads.



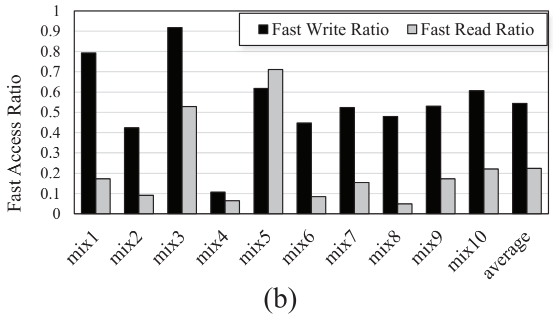


Figure 9: Fast access ratio results for: (a) single-programmed and (b) multi-programmed workloads.

benchmark with high compressibility for multi-programmed workloads. We compare four STT-RAM LLC designs, **SLC**: the SLC STT-RAM cache; **MLC**: the baseline conventional MLC STT-RAM cache; **OMT/CAT**: the proposed MLC STT-RAM cache design with overhead-minimized technique and capacity-augmented technique. The STT-RAM LLC configurations are summarized in Table 3, where the circuit-level parameters of latency, energy, and leakage power are generated from NVSim [8].

4.2 Energy Comparison

Figure 7 shows the energy consumption normalized to MLC design for single-programmed workloads and multi-programmed workloads respectively. As indicated, SLC design has the lowest energy consumption due to its one-step operation. As shown in Figure 7(a), the OMT scheme reduce system energy by 32.0% on average and up to 70.0% for benchmarks with high compressibility and APKI, e.g. *zeusmp* and *GemsFDTD*. Figure 7(b) shows that the energy consumption for multi-programmed workloads is further reduced by 39.4% on average. This is because applications running on a multi-core system cause more accesses on LLC. Fortunately, OMT avoids compressible cache lines being written into the hard-bit region, which leads to significant energy savings. While CAT design reduce energy by 19.3% on average, since we trade-off energy saving for potential better performance. Note that we consider dynamic energy only since the major difference of energy consumption comes from dynamic cache accesses and different cache designs share the similar peripheral circuit.

A major source of energy saving is the write energy reduction from two-step operation to one-step soft-bit only operation. As shown in Figure 9, we define Fast Write Ratio as the percentage of saved one-step write accesses over total write accesses on LLC and demonstrate that our design avoids on average 54.5% of normal two-step writes to one-step programming for shared LLC.

4.3 Performance Comparison

We use Instruction per Cycle (IPC) as the metric to evaluate system performance, and employ overall throughput ($\sum$IPC) for multi-programmed workloads. We first demonstrate the effectiveness of MLC design over SLC design. As shown in Figure 8(a), for single-programmed simulation, the SLC design with half of the capacity than MLC design incurs on average 2.8% performance degradation compared with the baseline MLC design, especially for workloads sensitive to LLC capacity such as *lbm*. While the performance benefit of employing MLC design differs among different applications. Compared with SLC design, the system performance of some benchmarks, e.g. *omnetpp* and *hmmer*, actually degrades due to the increased access latency of MLC design, and these benchmarks have a smaller working set on LLC and cannot take much advantage of the increased cache capacity. On the other hand, Figure 8(b) shows that for multi-programmed workloads, the performance of SLC design degrades by 12.1% on average, which comes from the heavier pressure on LLC. Therefore, even though the SLC design consumes the lowest energy, it potentially degrades system performance due to the reduced capacity.

Table 3: STT-RAM LLC Parameters

	SLC	MLC	OMT
Read latency (cycle)	13	19	S: 14, H: 20
Write latency (cycle)	49	90	S: 50, H: 95
Read energy (nJ)	0.415	0.424	S: 0.427, H: 0.579
Write energy (nJ)	0.876	1.859	S: 1.084, H: 2.653
Leakage power (mW)	80.8		

Table 4: Evaluated workloads

No	Benchmark	APKI	No	Benchmark	APKI
1	GemsFDTD	19.38	14	tonto	0.73
2	astar	0.96	15	zeusmp	11.50
3	bwaves	16.66	mix1	4,5,11,15	13.59
4	bzip2	9.52	mix2	3,6,8,13	9.69
5	dealII	1.73	mix3	5,11,14,15	10.17
6	gobmk	2.92	mix4	4,6,8,13	10.77
7	h264ref	1.92	mix5	3,9,11,14	13.39
8	hmmer	4.52	mix6	4,5,8,15	9.89
9	lbm	55.63	mix7	3,6,9,13	23.73
10	leslie3d	11.97	mix8	3,5,8,11	9.56
11	milc	9.42	mix9	4,6,13,15	25.69
12	sjeng	0.60	mix10	3,4,5,14	10.68
13	soplex	18.52			

Our proposed OMT design has the system performance benefit of larger capacity enabled by MLC structure over SLC design. Compared with the baseline MLC design, OMT reduces energy consumption as discussed in Section 4.2 but incurs no performance degradation. We observe slightly performance improvement (up to 2.5%) of OMT design since certain amount of read/write accesses to hard-bit region are eliminated. The performance improvement is insignificant because our target cache is the level-3 cache, which is insensitive to write latency, and the optimization on read latency only has limited impact on system performance. However, we demonstrate that on average 22.5% (38.2% for single-programmed workloads) of total read accesses are optimized to one-step operation as the Fast Read Ratio results show in Figure 9. We expect better performance improvement when OMT is applied at higher level of cache hierarchy or the system is loaded with cache intensive workloads.

Figure 8 shows that our CAT design improves system performance by 0.5% for single-programmed benchmarks and by 6.1% for multi-programmed benchmarks, both on average. The effectiveness of the CAT design relies on both compressibility and APKI of different workloads, as well as intensive cache accesses of multi-programmed workloads. For workloads with high compressibility and APKI, e.g. *mix7* and *mix9*, we observe that the performance is improved by over 10.0%. The effective cache capacity is increased when equipped with CAT since the saved hard-bit region can store additional cache lines. Consequently, it reduces memory accesses and improves system performance. While other workloads show little impact on performance, which is due to either low compressibility or insensitive access latency impact on LLC.

4.4 Overhead Estimation

The hardware overhead of our proposed techniques using data compression include the compressor and the decompressor. We implemented our BΔI compressor and decompressor using Verilog, and synthesized them with Synopsys Design Compiler at $45nm$ technology node. The estimated area cost is $0.0018mm^2$, which is negligible to the entire chip area. Besides the compressor and decompressor overhead, OMT and CAT also incur some tag overheads. As discussed above, OMT has very small tag overhead, only 2-bit compression scheme code for each cache line; CAT uses 50% more tags to support a larger effective cache capacity. Based on the area estimation on NVSim, the proposed techniques incur 0.18% and 0.45% area overhead for OMT and CAT, respectively.

Data compression also increases the cache access latency and energy. With our system simulation configurations (as shown Table 2), compression increases the read latency by 1 cycle, and decompression increases the write latency also by 1 cycle. Moreover, compressing a cache line consumes $0.003nJ$ on average, and decompressing a cache line consumes $0.001nJ$ on average.

5. CONCLUSIONS

Multi-level cell (MLC) STT-RAM has been widely studied as a replacement of SRAM for constructing large-scale and energy-efficient on-chip cache. However, the two-step read/write accesses of MLC cause energy overhead and system performance degradation. In this paper, we propose an architectural design for MLC-based STT-RAM last-level cache. Our OMT scheme avoids slow and energy-inefficient two-step accesses on both hard-bit and soft-bit region to one-step accesses on soft-bit region only by applying data compression. The proposed CAT design can further increase effective cache capacity by fit a compressible cache line into the saved hard-bit region of STT-RAM cells. The experimental results of the evaluated multi-programmed workloads show that the proposed OMT design reduces dynamic energy consumption on LLC by 38.2% on average without system performance degradation. While combined with CAT, our design improves system performance by 6.1% and reduces energy consumption by 19.3%.

6. REFERENCES

[1] A. R. Alameldeen et al. Adaptive cache compression for high-performance processors. In *ISCA*, 2004.

[2] A. Arelakis et al. Sc2: A statistical compression cache scheme. In *ISCA*, 2014.

[3] X. Bi et al. Unleashing the potential of MLC STT-RAM caches. In *ICCAD*, 2013.

[4] N. Binkert et al. The gem5 simulator. *SIGARCH Comput. Archit. News*, 2011.

[5] Y. Chen et al. Access scheme of multi-level cell spin-transfer torque random access memory and its optimization. In *MWSCAS*, 2010.

[6] Y. Chen et al. Processor caches built using multi-level spin-transfer torque ram cells. In *ISLPED*, 2011.

[7] P. Chi et al. Building energy-efficient multi-level cell STT-MRAM based cache through dynamic data-resistance encoding. In *ISQED*, 2014.

[8] X. Dong et al. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *TCAD*, 2012.

[9] J. Dusser et al. Zero-content augmented caches. In *ICS*, 2009.

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

[11] S. Ikeda et al. A perpendicular-anisotropy CoFeB-MgO magnetic tunnel junction. *Nature Materials*, 2010.

[12] T. Ishigaki et al. A multi-level-cell spin-transfer torque memory with series-stacked magnetotunnel junctions. In *VLSIT*, 2010.

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

[14] G. Pekhimenko et al. Base-delta-immediate compression: Practical data compression for on-chip caches. In *PACT*, 2012.

[15] J. Wang et al. Optimizing MLC-based STT-RAM caches by dynamic block size reconfiguration. In *ICCD*, 2014.

[16] X. Wu et al. Design exploration of hybrid caches with disparate memory technologies. *ACM Trans. Archit. Code Optim.*, 7(3), 2010.

[17] J. Yang et al. Frequent value compression in data caches. In *MICRO*, 2000.

[18] Y. Zhang et al. Multi-level cell STT-RAM: Is it realistic or just a dream? In *ICCAD*, 2012.