AOS: Adaptive Overwrite Scheme for Energy-Efficient MLC STT-RAM Cache

Xunchao Chen[1], Navid Khoshavi[1], Jian Zhou[1], Dan Huang[1], Ronald F. DeMara[1], Jun Wang[1]
Wujie Wen[2], Yiran Chen[3]
[1]Department of ECE, University of Central Florida
[2]Department of ECE, Florida International University
[3]Department of ECE, University of Pittsburgh
{xchen, nkhoshavi}@eecs.ucf.edu, {ronald.demara, jun.wang}@ucf.edu

ABSTRACT

Spin-Transfer Torque Random Access Memory (STT-RAM) has been identified as an advantageous candidate for on-chip memory technology due to its high density and ultra low leakage power. Recent research progress in Magnetic Tunneling Junction (MTJ) devices has developed Multi-Level Cell (MLC) STT-RAM to further enhance cell density. To correct the write disturbance in MLC strategy, data stored in the soft bit must be restored back immediately after the hard bit switching is completed. However, frequent restores are not only unnecessary, but also introduce a significant energy consumption overhead. In this paper, we propose an Adaptive Overwrite Scheme (AOS) which alleviates restoration overhead by intentionally overwriting selected soft bit lines based on RRD (Read Reuse Distance). Our experimental results show 54.6% reduction in soft bit restoration, delivering 10.8% decrease in overall energy consumption. Moreover, AOS promotes MLC to be a preferable L2 design alternative in terms of energy, area and latency product.

Keywords

MLC; STT-RAM; Write Disturbance; Prediction

1. INTRODUCTION

In order to meet the ever-growing demand for performance and energy efficiency, a large portion of modern processors is occupied by on-chip multi-level SRAM caches. However, the significant leakage power and cell area of SRAM impedes its future deployment in energy critical applications, such as mobile platforms. Despite several remarkable research advances to reduce the leakage power, it is inevitable to encounter additional SRAM energy consumption as CMOS technology continues to scale down. Recently, emerging non-volatile memory technologies, such as Spin-Transfer Torque RAM (STT-RAM), have been identified as promising alternatives to SRAM. STT-RAM is ideal for on-chip caches due to its near-zero leakage and high cell density. In fact, companies like Qualcomm, already advanced their research in using STT-RAM to revamp the memory hierarchy [1].

By storing two or more bits in a single cell, Multi-Level Cell (MLC) designs boost data density and have been adopted in commercial products, such as MLC NAND flash. In light of this, MLC STT-RAM has been explored to enhance cache capacity [2][3]. Compared to the Single Level Cell (SLC) structure, one more Magnetic Tunnel Junction (MTJ) is placed into a single MLC STT-RAM cell. These two MTJs, whose feature sizes are maintained to meet certain Tunneling Magneto-Resistance (TMR), can be implemented either in plane or perpendicularly. Various bit to cache line mapping strategies are enabled due to the proposal of MLC designs.

In order to read data without flipping MTJs, a very small sensing current is applied to the bit line in MLC. While in a write operation, switching current for a large MTJ (hard bit) is able to change the magnetization direction of the corresponding small MTJ (soft bit), which is called Write Disturbance (WD). To rectify WD, on every write access to a hard bit, data stored in the soft bit of the same STT-RAM cell must be sensed out first and restored back after the writing is completed. Although the contemporary restore scheme guarantees data integrity, it introduces extra reads and writes and is energy inefficient in some scenarios. For instance, if a soft bit cache line is not subsequently read prior to its eviction, immediately restoring this line is not only unnecessary, but also brings an energy overhead.

In this paper, we propose an energy efficient restore scheme, Adaptive Overwrite Scheme (AOS), for MLC STT-RAM based cache. AOS chooses to overwrite the soft bit line when it is less likely to be read in the near future. For the sake of this, first we define the concept of Read Reuse Distance (RRD), which is the timing distance between two consecutive read accesses. We also develop the RRD predicator inspired by an existing cache line reuse distance prediction design [4][5]. The RRD predictor samples memory access streams (hashed values of program counter) to calculate RRD at run time, and the RRD predication table is updated for higher accuracy upon the incoming pairs of PC and RRD. Furthermore, to determine if it is harmless to overwrite, a threshold RRD with high coverage is adopted and compared with the Estimated Distance to the next Read (EDR). Our experimental results show AOS fully release the potential of MLC to be the favored L2 alternative.

2. BACKGROUND AND MOTIVATION

DAC '16, June 05-09, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2897937.2897987

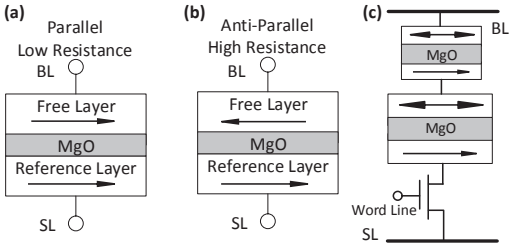

Figure 1: (a) Low Resistance State (b) High Resistance State (c) Serial MLC STT-RAM

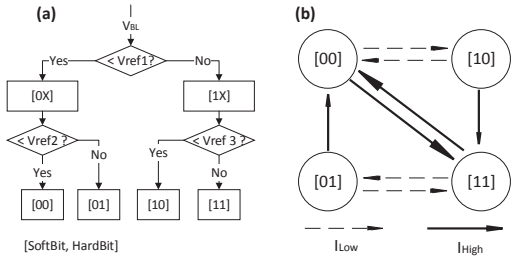

Figure 2: MLC STT-RAM (a) Read Scheme (b) Write Scheme

STT-RAM is an emerging non-volatile memory which can provide SRAM-like read speed, DRAM-like density and near-zero leakage power. Each MTJ consists of two ferromagnetic layers (free and reference layer) and an oxide barrier (MgO) sandwiched between them. As shown in Fig. 1(a) & Fig. 1(b), the magnetization directions of two ferromagnetic layers can be tuned to either parallel or anti-parallel, indicating whether the MTJ is in a low resistance state (logical 0) or a high resistance state (logical 1), respectively.

In the write operation, a high voltage is applied between the source line (SL) and the bit line (BL), generating a current across the MTJ and switching the magnetization direction of the free layer. When reading data from a STT-RAM cell, a small sensing current is injected to generate a bit line voltage (V_{BL}). This V_{BL} is then compared with a reference voltage in order to decide whether a logical 1 or a logical 0 is stored in the cell.

To further improve the density of STT-RAM, MLC designs have been introduced and studied recently. There are two varieties of MLC structures, namely, serial MLC and parallel MLC. In this paper, we use the serial MLC STT-RAM structure as shown in Fig. 1(c) by default, for the serial MLC has been proven to be more reliable and easier to fabricate. However, our proposed technique is also applicable to the parallel structure. The serial MLC stacks two MTJs vertically in a single memory cell. These two MTJs must occupy different cell areas so that four differential resistance states can be achieved. We call the small and large MTJ as the soft bit and hard bit respectively. Under constant resistance-area product, the soft bit, with larger resistance, is easier to be flipped than the hard bit because the soft bit requires a smaller switching current.

The read and write schemes of MLC STT-RAM have been well studied in [6]. To read a 2-digit value from a MLC, it takes two comparisons as shown in Fig. 2(a). Recall that the sensing current passing through MTJs will produce a V_{BL}. This V_{BL} is compared with the reference voltage V_{ref1} first to decide the value of the soft bit, then a second comparison with V_{ref2} or V_{ref3} is done to decide the hard bit. Programming a MLC needs up to two steps as shown in Fig. 2(b). If the target 2-digit number has the same value on soft and hard bits, then a strong current I_{High} is used to switch both bits. Otherwise, a weak current I_{Low}, which will only affect the small MTJ, can be applied to switch the soft bit.

As we have described, the switching current I_w flows through an MTJ and changes its magnetization direction. The write current value is proportional to its MTJ area [7][8] as defined in Equation (1):

$$I_w = A \cdot (J_{c0} + \frac{C}{{T_\omega}^\alpha}) \quad (1)$$

where J_{c0} is the critical current density at zero temperature; T_ω is the switching current duration; C and α denotes fitting parameters. In a serial MLC, switching the large MTJ (hard bit) requires higher current according to Equation (1), which will overwrite the value stored in the small MTJ (soft bit). This is also known as Write Disturbance (WD). To deal with WD, upon each hard bit write request, data stored in the soft bit has to be read out first, then restored back after the hard bit update is completed. As astute readers may point out, this inefficient, yet necessary immediately restore scheme will incur significant performance and energy consumption overheads.

To be more specific, we illustrate the energy overhead caused by restoration using cell split mapping MLCs [2] which will be presented in the next section. Assume E_{PC} is the dynamic energy consumed by cache peripheral circuitry (e.g. address decoder) per access, E_{WSBL} and E_{RSBL} are the average write and read energy of Soft Bit Line (SBL) per request, while N_{WHBL} is the write access number of Hard Bit Line (HBL). The energy overhead spent on immediately restore operations is:

$$E_{IRS} = (E_{PC} + E_{WSBL} + E_{RSBL}) \cdot N_{WHBL} \quad (2)$$

Fig. 3 demonstrates the dynamic energy consumption breakdown of read, write, and restore operations for PARSEC benchmarks (see Section 4.1 for detailed experiment settings). Restoration consumes 23% on average and up to 27% more dynamic energy, which brings a significant overhead to MLC STT-RAM based L2.

3. TECHNICAL APPROACH

3.1 Cell Split Mapping MLC STT-RAM

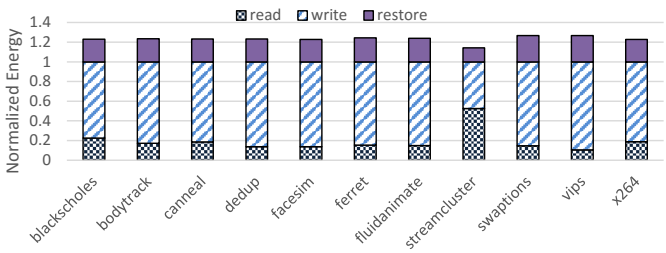

Figure 3: Dynamic Energy Breakdown of MLC STT-RAM

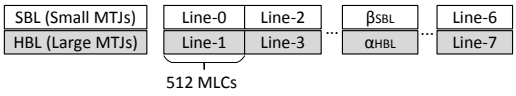

Figure 4: A 8-way Cell Split Mapping Cache Set

There are three practical bit mapping strategies for MLC STT-RAM cache, namely, direct mapping, cell split mapping, and interleaved mapping [2][9]. Direct mapping neglects the fact that it is easier to access soft bits than hard bits, and interleaved mapping requires additional address decomposition to support mixed block modes. Therefore, we choose cell split mapping strategy in our design, which maps an entire data block to the favourable SBL. Fig. 4 depicts an 8-way associative cell split mapping cache set. 512 MLC cells construct two 64-byte cache lines, where all the hard bits form an HBL and the corresponding soft bits in the same cells compose an SBL.

3.2 Adaptive Overwrite Scheme

The intuition behind saving energy for MLC is to avoid unnecessary restore operations. In light of this, Adaptive Overwrite Scheme (AOS) is proposed as shown in Fig. 5. By overwriting selected SBLs, a large portion of restorations can be reduced. Here α_{HBL} is a cache data block stored in a HBL, β_{SBL} is another block saved in the corresponding SBL of α_{HBL} as shown in Fig. 4, both of which are in L2. A separated tag- and data- array implementation as well as an non-inclusive cache hierarchy are adopted here. We will elaborate the design of read reuse predictor in Section 3.3 and the adoption of threshold RRD value in Section 4.2. AOS works as follows:

- If β_{SBL} is invalid (V bit is '0'), it can only serve write access, but not read access. Overwriting it will not introduce any cache miss. The dirty bit of α_{HBL} is set when the write request is a dirty cache block write-back from higher level. When the request is a new write allocation from main memory due to L2 cache miss (i.e., cache fills), the dirty bit remains '0' .
- If β_{SBL} has a replica in L1, overwriting it will not incur extra cache miss. Recall that in the non-inclusive hierarchy, upon a cache miss in L2, the missing block is retrieved from the main memory to all cache levels. However, the copy stored in L2 will not be accessed until its eviction from L1. Thus, immediately restoring β_{SBL} will not benefit cache read hit. Also, since L1 maintains the most updated copy of the cache block, there is no need to write back β_{SBL} at the time of overwrite even if it is dirty.
- If β_{SBL} has no copy in L1, overwriting and invalidating this block may lead to a future read miss. Therefore, it is necessary to predict the read reuse behavior of β_{SBL} and calculate when the next time this block is read (EDR). When EDR is shorter than a threshold value (RRD_{th}), the block is considered to be involved in subsequent reads soon. Conventional immediately restore scheme is applied to handle read miss.
- Supposing β_{SBL} has no copy in L1 (i.e., no write-back) and will not be read in the near future (i.e., no read fetch) according to the prediction result, β_{SBL} can be overwritten in the write access of α_{HBL}. Note here the dirty bit of β_{SBL} needs to be checked first. If it is '1', this block is written back to the main memory and is then overwritten.

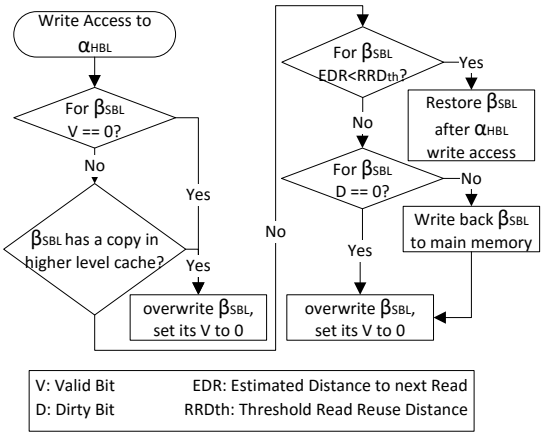

Figure 5: Adaptive Overwrite Scheme Flowchart

3.3 Read Reuse Distance Prediction

In contrast to the reuse distance measurement considering both read and write [10], RRD is the interval between two successive reads to the same block. In other words, RRD is a notion that quantifies data block read reuse frequency. Here, we use the number of intervening L2 access to represent it. EDR is defined as the interval from the time an HBL is written to the next time its corresponding SBL is read. Previous work [4][5] proposed a PC-based reuse distance predictor to optimize cache replacement policy. In light of this design, we exploit the instructions of cache accesses to predict block read reuse behavior.

Similar to prior work [4][11], there are two parts in our RRD predictor, namely, "read sampler" and "RRD prediction table". Read sampler aims to calculate the corresponding RRD of a given PC. The sampling FIFO buffer is scanned for a matching PC on each read access. Simultaneously, read sampler samples the access stream and stores into the sampling buffer. To calculate RRD, read sampler simply multiplies the sample period by its relative tail pointer position. For example, in Fig. 6, the sampling period is set to 2, therefore PCs associated with RdA, RdC are sampled into the buffer from the access stream. WrtB is stored as a "stall" regardless of its PC particularly. Then the WrtC request comes into cache followed by the RdC. WrtC is skipped in the PC matching process. While triggered by the next read access RdC, the sampling buffer is scanned for a match with the PC associated with block C (PC2). Since PC2 exists in the buffer already, a match is found and the RRD paired to this PC is computed as 4. The second part of our RRD predictor, RRD prediction table, is similar to the one in [4], which is indexed by hashed PC and holds correlated RRD and Confidence Counter (CC). Upon every incoming pair of PC and RRD from the read sampler, the CC of the corresponding pair in RRD prediction table is updated. Only if the CC reaches a certain threshold, the RRD can be taken for prediction. EDR is calculated using the last read access timestamp stored in the cache line, current time, and predicted RRD.

Our RRD predictor differs from previous designs in the

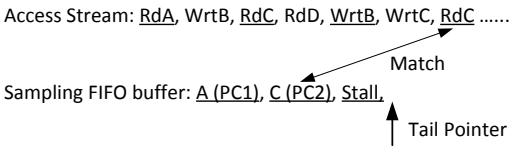

Figure 6: Read Sampler Operation

following aspects. First, at the sampling stage, if a write request is sampled, a "stall" will be stored in the FIFO buffer as a placeholder without its PC. However, in [4], the PC associated with a sampled write request will be stored into the FIFO buffer for matching. Second, at the PC matching stage, only an incoming read request can trigger PC matching. This is due to AOS objective of estimating the read reuse distance instead of the general reuse distance in [4]. Lastly, in [4], a write-back access filter is needed, which significantly increases the design overhead. This is because evictions from L1 appear as write accesses at L2. However, these write-backs are not associated with any instructions. Failure to consider this will substantially degrade prediction accuracy. Read sampler explicitly avoids irrelevant PCs brought by evictions without the need for a write-back filter. Therefore, our predictor is more lightweight, but delivers even better performance over a range of workloads.

Regarding the memory size overhead incurred by RRD predictor, for example, a 512-entry RRD prediction table brings 512 entries × 39 bits (32 bits PC + 5 bits bucket + 2 bits confidence counter) per entry = 2.4 KB overhead, which is only 0.06% of a 4 MB cache. Also, for the 10-bit timestamp in each cache line, it only consumes 10 bits per 64 B = 1.95% of cache line size.

4. EVALUATION

4.1 Experiment Setup

The evaluation was conducted by using the cycle-accurate simulator MARSSx86 [12]. We modified its cache controller module to realize the proposed function. The simulator mimics the computer architecture as shown in Table 1. Eleven different benchmarks from PARSEC 2.1 suite [13] were used for the experiment, executing 500 million instructions starting at the Region Of Interest (ROI) after warming up the cache with 5 million instructions. The simsmall input sets are selected for all benchmarks.

We adopted the serial MLC STT-RAM cell design from [3] and scaled it under 32nm technology node [14]. The MTJ pillar is configured as a 32nm×64nm elliptical shape. We used the NVSim and CACTI [15][16] to obtain the key design parameters as shown in Table 2. MLC STT-RAM model is integrated into NVSim by modifying configurations, such as set/ reset currents, nMOS transistor sizes, etc. The energy contributions from peripheral circuits are also included.

Table 1: Baseline Configuration

CPU	4 cores, 3.3 GHz, Fetch/ Exec/ Commit width 4
L1	private, 32 KB, I/D separate, 8-way, 64 B, SRAM, WB
L2	private, 4 MB, unifed, 8-way, 64 B, STT-RAM, WB
Main Memory	8 GB, 1 channel, 4 ranks/ channel, 8 bank/ rank

4.2 RRD Predictor Configuration

Table 2: Comparsion of 4 MB SLC STT-RAM, MLC STT-RAM and SRAM

	SLC STT-RAM	MLC STT-RAM	SRAM
Array Area (mm^2)	1.86	1.01	7.28
Read Latency ($Cycles$)	9.08	S: 6.73 H: 9.80	7.43
Write Latency ($Cycles$)	25.58	S: 25.31 H: 56.50	5.78
Read Energy (nJ)	0.216	S: 0.22 H: 0.43	0.161
Write Energy (nJ)	0.839	S: 0.843 H: 2.502	0.156
Leakage (mW)	18.39	7.02	295.58

RRD Theshold Value: In order to determine whether restoration can be postponed or not, EDR of an SBL needs to be compared with a threshold RRD value. This value is adopted from the analysis of various memory accesses. Note here RRD value is presented by the power of two as this can help us define RRD access counter size. Fig. 7 shows cache blocks whose RRD is greater than 2^4 account for 62% of the total number on average. In other words, more than half of the cache blocks have a distant RRD and are eligible for overwrite. Moreover, the number of reads associated with these blocks takes only 13% of total read accesses as described in Fig. 8. That means overwriting these blocks is less likely to incur read misses. Hence, we adopted 2^4 as the threshold RRD. Blocks with RRD greater than the threshold (i.e., non read-intensive) are selected to skip restorations.

Read Sampler Sampling Period: To reduce the storage requirement for RRD prediction table, a reasonable sampling period needs to be adopted. We conducted an experiment to explore the impact of sampling frequency on the hit rate of L2 cache. We found when the sampling period increases from 2^3 to 2^7, the average hit rate barely changes across PARSEC benchmarks. However, from 2^7 to 2^{10}, it started to fluctuate and decreased significantly at 2^{10}. This is because a large sampling period is not able to capture the locality in the access stream. Since a small sampling period introduces intensive PC comparison and table updates, which impairs predictor performance in turn, we chose a sampling period of 2^7 in our final evaluation. The size of sampling buffer can be determined via the equation:

$$Sampling\_FIFO\_size = \frac{Max\_RRD}{Sampling\_Period}$$

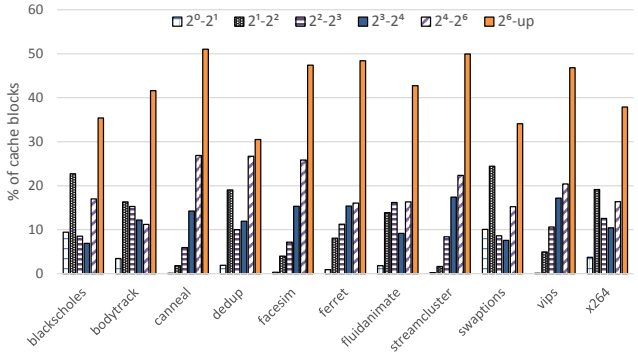

Figure 7: Percentage of total cache line for six RRD ranges

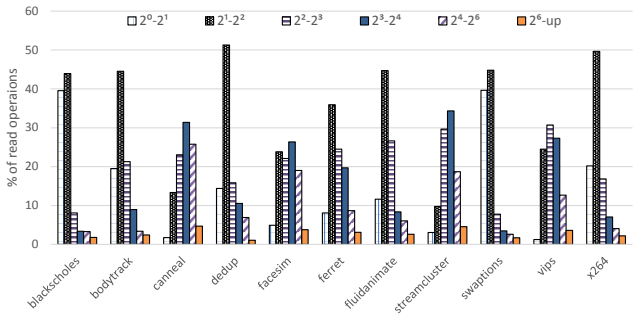

Figure 8: Percentage of total read operation for six RRD ranges

The maximum RRDs we observed in memory traces are generally under 2^{10}, and the sampling period is set as 2^7. Thus we selected 2^3 as the FIFO size.

4.3 Energy Area Latency (EAT) Product Comparison

There are a few other technologies that can be employed as L2, for instance, SLC STT-RAM and conventional SRAM. SLC almost doubles the cell area of MLC with the same capacity, whereas MLC suffers from asymmetric read and write performance and needs restoration. On the other hand, SRAM consumes significantly more leakage power and size in contrast to MTJ-based technologies. We use Energy Area Latency (EAT) product [17] as metrics to not only find the energy efficiency AOS can help to improve, but also to evaluate the preferred L2 candidate.

In the following evaluations, the baseline is the MLC with conventional immediately restore scheme. No SBL restoration can be deducted in the baseline.

Restoration Reduction: The normalized numbers of restoration reduced by AOS are shown in Fig. 9. The baseline MLC has 100% restoration. When applying AOS, 54.6% of the total restore operations are avoided on average; in some benchmarks like streamcluster, where there are a large portion of distant read intervals, AOS can save up to 62.7% of restoration.

Energy Comparison: Fig. 10 compares the dynamic energy consumption of four L2 candidates. SRAM consumes the least dynamic power, as switching transistors from "on" to "off" and "off" to "on" are symmetrical. Writes are not as power hungry as they are in STT-RAM. With AOS, 10.2% of dynamic energy in MLC is reduced on average.

To calculated the leakage energy of four candidates, we used the execution time of each benchmark and the unit leakage power. The overall energy consumption equals the sum of dynamic and leakage energy as shown in Fig. 11.

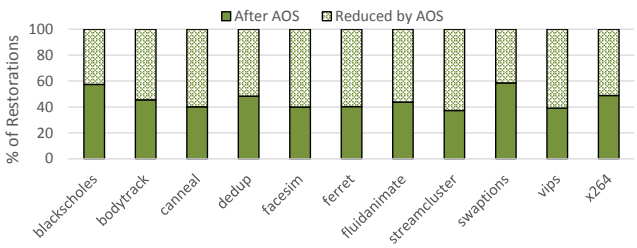

Figure 9: Restoration Reduction with AOS

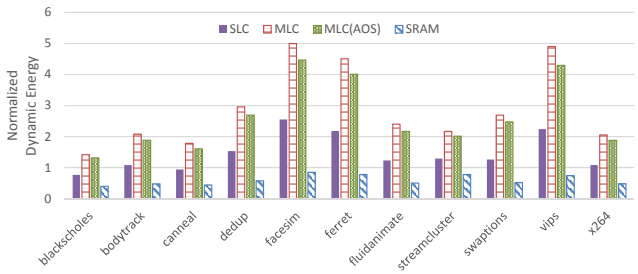

Figure 10: Dynamic energy comparison among different L2 candidates

SRAM consumes considerably more leakage energy than other candidates, making the overall energy consumption enormous. AOS can mitigate 10.8% of the total energy for MLC on average. In write intensive benchmark like vips, AOS can save up to 13.0% energy.

Latency Comparison: We used the read and write numbers and unit latency to calculate cache access latency. AOS reduces unnecessary restorations and decreases MLC latency by up to 14.85% and 11.72% on average as shown in Fig. 12. SRAM is the fastest among four candidates due to its symmetrical rapid access speed. Note here the overhead incurred by the peripheral circuits are also included.

EAT Product: Fig. 13 shows that MLC with AOS has the preferable EAT in most cases. Compared to SLC, which tends to be considered as the preferred L2 candidate by intuition, MLC with AOS decreases EAT by 4.6% on average. In the read intensive workload like streamcluster, EAT is reduced by 20.3%. This is because SLC almost doubles the size of MLC, and besides, the read latency of SLC is very close to that of the hard bit in MLC. AOS also reduces the baseline MLC EAT by 10.1%.

5. RELATED WORK

Previous research on MLC STT-RAM mainly focused on leveraging the performance disparity of two MTJs. For instance, Jiang et al. [18] proposed to promote frequently written data into write-fast-read-slow soft bit lines, and frequently read data into read-fast-write-slow hard bit lines within parallel MLC structured STT-RAM. Similarly, based on series MLC, Bi et al. [2] advocated a bit mapping strategy constructing general fast- and slow- lines which is used in our paper. Wang et al. [9] proposed to dynamically disable the hard bits in cache line while keep the number of associativity. Chi et al. [19] presented an efficient local checkpointing method by storing working data in soft bits and checkpoint data in hard bits.

Meanwhile, the reliability issue of STT-RAM is gaining

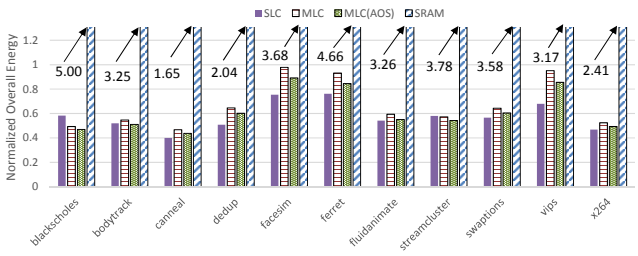

Figure 11: Overall energy consumption among different L2 candidates

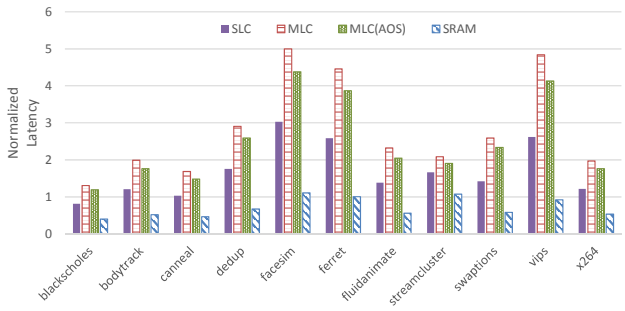

Figure 12: Cache latency comparison among different L2 candidates

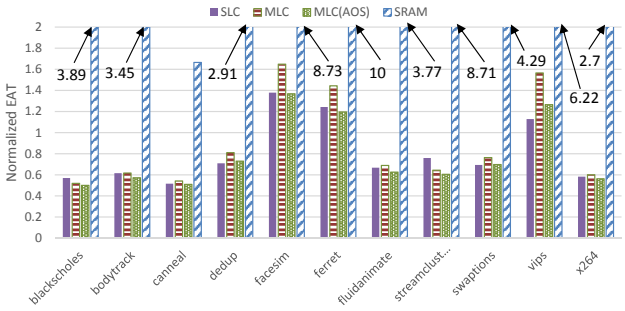

Figure 13: EAT comparison among different L2 candidates

significant research interest. Recently, Wang et al. [8] proposed a selective restore method to mitigate the overhead brought by read disturbance correction in SLC STT-RAM. They also added three more flag bits in each cache line to assist their scheme. Our work shares some similarities with this one, but our research focuses on the write disturbance issue in MLC. In [20][21], ternary coding technique is proposed to remove the most error-prone state and trade MLC capacity for reliability. Wen et al. considered the nonuniform ECC design requirement for MLC and advocated a tri-region mapping strategy to reduce the write pressure on hard bit lines [22].

6. CONCLUSION

In this paper, we proposed AOS to mitigate the energy overhead caused by restores in MLC STT-RAM. We first introduce the concept of RRD, which exploits the number of memory access, to quantify the timing distance between two successive reads to the same cache block. We also developed an RRD predictor consisting of a read sampler and RRD prediction table to provide trustworthy RRD for each cache block. Furthermore, upon a write request to an HBL, the EDR of the corresponding SBL is compared with a threshold value to determine if restores can be postponed or not. We conducted memory access analysis to adopt a threshold value with high confidence. Our experimental results show that AOS effectively reduces both dynamic and overall system energy dissipation, and decreases EAT product.

Acknowledgment

This work is supported in part by the US NSF Grant CCF-1527249, CCF-1337244 and NSF CAREER Award 0953946.

7. REFERENCES

[1] S. H. Kang. Embedded stt-mram for mobile application: Enabling advanced chip architectures. In *Non-Volatile Memories Workshop*, 2010.

[2] X. Bi et al. Unleashing the potential of mlc stt-ram caches. In *ICCAD*, 2013.

[3] Y. Zhang et al. Multi-level cell stt-ram: Is it realistic or just a dream. In *ICCAD*, 2012.

[4] G. Keramidas et al. Cache replacement based on reuse-distance prediction. In *ICCD*, 2007.

[5] P. Petoumenos et al. Instruction-based reuse-distance prediction for effective cache management. In *SAMOS*, 2009.

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

[7] K. C. Chun et al. A scaling roadmap and performance evalution of in-plane and perpendicular mtj based stt-mrams for high-density cache memory. In *JSSC*, 2013.

[8] R. Wang et al. Selective restore: an energy efficient read disturbance mitigation scheme for future stt-mram. In *DAC*, 2015.

[9] J. Wang et al. Optimizing mlc-based stt-ram caches by dynamic block size reconfiguration. In *ICCD*, 2014.

[10] A. Jaleel et al. High performance cache replacement using re-reference interval prediction (rrip). In *ACM SIGARCH Computer Architecture News*, 2010.

[11] P. Petoumenos et al. Instruction-based reuse distance prediction replacement policy. In *JWAC-1@ISCA*, 2010.

[12] A. Patel et al. Marss: a full system simulator for multicore x86 cpus. In *DAC*, 2011.

[13] C. Bienia et al. The parsec benchmark suit: characterization and architectural implications. In *PACT*, 2008.

[14] Predictive technology model (ptm). In *http://ptm.asu.edu/*.

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

[16] S. Thoziyoor et al. A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies. In *ISCA*, 2008.

[17] B. Bass. A low-pwer, high-performance, 1024-point fft processor. In *JSSC*, 1999.

[18] L. Jiang et al. Constructing large and fast multi-level cell stt-mram based cache for embedded processors. In *DAC*, 2012.

[19] P. Chi et al. Using multi-level cell stt-ram for fast and energy-efficient local checkpointing. In *ICCAD*, 2014.

[20] W. Wen et al. State-restrict mlc stt-ram designs for high-reliable high-performance memroy system. In *DAC*, 2014.

[21] S. Hong et al. Ternary cache: Three-valued mlc stt-ram caches. In *ICCD*, 2014.

[22] W. Wen et al. A holistic tri-region mlc stt-ram design with combined performance, energy, and reliability optimizations. In *DATE*, 2016.