Selective Restore: an Energy Efficient Read Disturbance Mitigation Scheme for Future STT-MRAM

Rujia Wang†, Lei Jiang†, Youtao Zhang§, Linzhang Wang‡, Jun Yang†
†Electrical and Computer Engineering Department, University of Pittsburgh
§Computer Science Department, University of Pittsburgh
‡State Key Laboratory of Novel Software Technology,
Department of Computer Science and Technology, Nanjing University
{ruw16,lej16,youtao,juy9}@pitt.edu,lzwang@nju.edu.cn

ABSTRACT

STT-MRAM (Spin-Transfer Torque Magnetic RAM) has recently emerged as one of the most promising memory technologies for constructing large capacity last level cache (LLC) of low power mobile processors. With fast technology scaling, STT-MRAM read operations will become destructive such that post-read restores are inevitable to ensure data reliability. However, frequent restores introduce large energy overheads. In this paper, we propose Selective Restore (SR), an energy efficient scheme to mitigate the restore overheads. Given a L2 cacheline disturbed from a read operation, SR postpones its restore till the cacheline being evicted from the upper level cache L1. Based on the status of the line at the eviction time, SR selectively restores the disturbed cells to achieve energy efficiency. Our experimental results show that SR improves system performance by 5% and reduces dynamic energy consumption by 62%.

Categories and Subject Descriptors

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

General Terms

Design

Keywords

STT-MRAM, Read Disturbance, Energy

1. INTRODUCTION

To meet the increasing demand for low power and high performance, modern mobile processors integrate multiple cores onto one single die. For instance, the latest high-end mobile processors that have four or eight cores, such as Intel Atom [9] and Qualcomm Snapdragon [19], have been adopted in commercial tablet [14] and smartphone [7] products. Multi-core mobile processors require large on-chip caches to achieve scalable performance.

Many modern mobile processors integrate less than 512KB SRAM last level caches (LLCs) [19, 17], as a large capacity SRAM cache not only consumes large power but also occupies large die area. As technology scales in deep sub-micron regime, SRAM suffers from low cell density, large leakage power, and physical reliability problems [13]. Recent studies are looking for promising alternatives, such as Phase-Change Memory (PCM) [26] and Spin-Transfer Torque magnetic RAM (STT-MRAM) [3], to traditional SRAM for onchip LLCs. Among different memory technologies, STT-MRAM has been identified as one of the most promising candidates. STT-MRAM has zero cell leakage, high cell density and close-to-SRAM read latency, which are attractive properties for constructing large and low power onchip LLCs.

The STT-MRAM write current reduces quickly with technology scaling. This is due to shrunk cell area and magnetic material improvement [8]. However, the STT-MRAM read current does not scale well. It is difficult to apply a small current to STT-MRAM cells to sense their stored bits reliably [1]. The STT-MRAM read current stays almost the same over different technology nodes in deep sub-micron regime [18]. Starting from 32nm, the amplitudes of read and write currents are so close [23] such that the being-read cells have high possibility to get disturbed (i.e., modified). With further technology scaling, a post-read restore operation becomes inevitable to ensure the data reliability of STT-MRAM caches. However, the restore-after-read approach [23, 21] introduces extra cell writes, resulting in large write energy consumption.

In this paper, we propose Selective Restore (SR), an energy efficient scheme to mitigate the restore overheads. Given a L2 cacheline disturbed from a read operation, SR postpones its restore till the time that this line is evicted from the upper level cache L1. Based on the status of the line at the eviction time, SR selectively restores the disturbed cells to achieve energy efficiency. Our experimental results show that SR improves system performance by 5% and reduces energy consumption by 62%.

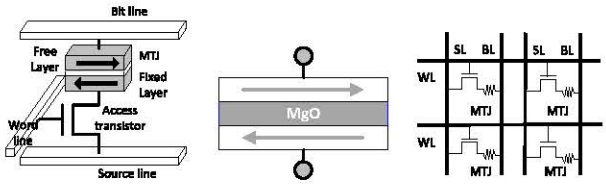
In the rest of paper, Section 2 briefly describes the STT-MRAM background. Section 3 elaborates the design details. We present the experiment methodology in Section 4 and analyze the simulation results in Section 5. More related work is discussed in Section 6. Section 7 concludes the paper.

DAC'15 June 07 - 11, 2015, San Francisco, CA, USA

http://dx.doi.org/10.1145/2744769.2744908

2. BACKGROUND AND MOTIVATION

STT-MRAM (Spin-Transfer Torque Magnetic RAM) is an emerging nonvolatile memory technology, which is deemed as one of promising alternatives to SRAM to implement onchip caches. As shown in Figure 1(a), a STT-MRAM cell adopts one MTJ (Magnetic Tunnel Junctions) to store data [6]. Figure 1(b) shows one example of MTJ, which consists of two ferromagnetic layers separated by an oxide barrier layer (MgO). The magnetization direction of one ferromagnetic layer is fixed (reference layer) while the other (free layer) can be changed by injecting 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) Cell (b) MTJ (c) Array

Figure 1: STT-MRAM cell and array structure.

A typical STT-RAM cell array is shown in Figure 1(c). An NMOS connected to word line (WL) is used to select a row of cells. To read and write cell, a voltage is applied between bit line (BL) and source line (SL).

- To write bit '1', a positive voltage is applied between SL and BL, creating a current flow from SL to BL. To write bit '0', a negative voltage is applied to create a current flow in the opposite direction [12].
- To read a cell, a small voltage is applied between SL and BL [5]. The amplitude of current flowing through the cell depends on the resistance of MTJ, which is sensed by a sense amplifier to identify the stored data. The read operation is similar to the write operation on STT-MRAM, except that the read current is smaller than the write current and the duration is shorter [12].

  While read can be done by applying current in two directions, existing designs often choose the direction of writing '0', which is more reliable [11].

2.1 Destructive Read

STT-MRAM has shown good scalability by shrinking cell size beyond $22nm$ [2]. Equation (1) shows that, given a STT-MRAM cell, its write current scales proportionally with the MTJ area [2].

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

where I_w is the STT-MRAM write current; A denotes the MTJ area; J_{c0} denotes the critical current density at zero temperature; T_w denotes the write current duration; C and α are fitting parameters. Since the MTJ area shrinks exponentially with decreasing feature sizes, the STT-MRAM write current reduces fast — halving feature size results in 25% MTJ area and 75% write current reduction.

STT-MRAM read operation is similar to write operation, except that the voltage is smaller than that of write [5]. At 130nm, the read current is much smaller than the write current and thus reads are reliable. However, scaling read current is challenging as it is very difficult to build STT-MRAM sense amplifiers that can sense the correct data using below $20\mu A$ current [28]. Figure 2.1 compares the read and write currents at different technology nodes. From the figure, the read current stays about the same in deep submicron regime. At 32nm node, the read and write currents are so close such that some reads may disturb (i.e., write) their being-read cells [23]. This is referred to as *read disturbance* in STT-MRAM.

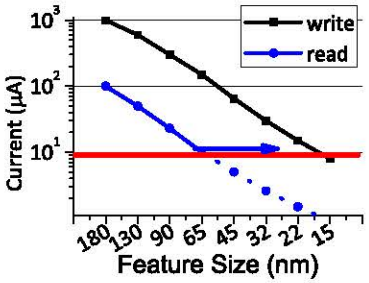


Figure 2: The scaling of read and write currents [23].

The read disturbance can modeled [22] with Equation (2).

$$P = 1 - exp\{-\frac{t}{\tau} exp[-\Delta_0(1 - \frac{I}{I_{c0}})]\} \quad (2)$$

where P is the read disturbance rate; I denotes the read current; t denotes the read pulse width; τ denotes the inverse of the attempt frequency; Δ_0 denotes the magnetic memorizing energy without any current and magnetic field; and I_{c0} denotes the critical switching current at 0K. We choose t, Δ and I_{c0} by following the widely adopted STT-MRAM model in [22], a model that has been validated against commodity SST-MRAM products. To compute the disturbance rates at different technology nodes, we adopt the STT-MRAM scaling model in [2], and model the σ of process variation as 10% [15].

tech(nm)	45	32	22	15	11
BER	1.38E-8	3.38E-7	3.07E-6	2.16E-5	1.2E-4
LER	7.05E-6	1.72E-4	1.57E-3	1.1E-2	6E-2
LER ECC	6.98E-6	1.61E-4	1.54E-3	1.01E-2	5.8E-2

Table 1: The read disturbance rate.

Table 1 summarizes BER (the raw bit error rate), LER (the raw line error rate for a 64B cacheline), and LER with ECC (the line error rate after adopting a 5EC6ED BCH code that can detect 6 errors and correct 5 errors) at different technology nodes. From the table, the LER with ECC at 32nm node is 1.6E-4, which is much larger than 2.5E-11, the acceptable error rate of DRAM [20]; the LER with ECC at 15nm node is larger than 1E-3, the acceptable error rate for onchip caches [25]. Given that read disturbance is becoming severe, a recent chip demonstration [23] adopts restore-after-read approach to eliminate read disturbance. Restore-after-read was also studied in the research community [21].

Unfortunately, restore-after-read introduces large write energy overhead. Figure 3 categorizes the energy consumption when adopting restore-after-read on STT-MRAM based L2 caches. The experimental settings can be found in Section 4. On average, restore consumes 68% more dynamic energy,

making it a big obstacle to deploy STT-MRAM in mobile computing platforms.

In this paper, we consider restore-after-read as the baseline and device innovations to achieve energy efficiency.

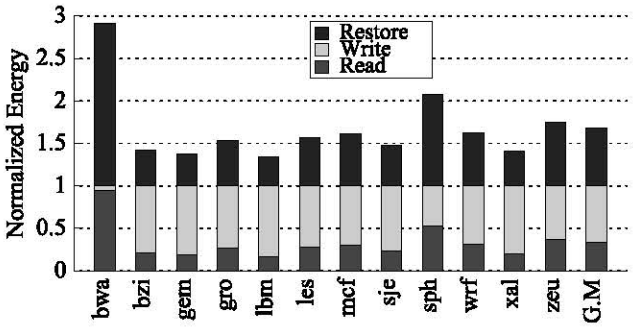


Figure 3: The energy overhead of restore operations.

3. THE DESIGN DETAILS

3.1 The baseline and restore-after-read

To facilitate the discussion, we elaborate the activities, in particular, the restore-after-read activities, in accessing a STT-MRAM based L2 cache (Figure 4).

When there is a cache miss, L1 chooses one line as victim and, if it is dirty, writes it back to L2 (❶). L1 also sends a read to find the missed data in L2 (❷). If it is a L2 hit, the data is fetched and placed in the location of the victim line (❸); otherwise, the main memory is accessed to return the requested data (❹ and ❺). The data returned from main memory is sent to both L2 and L1 (❻) [29].

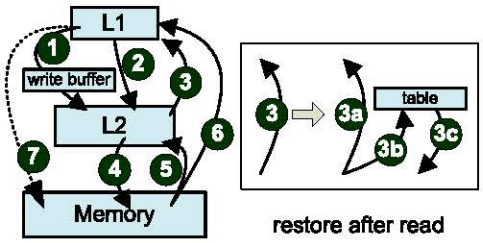


Figure 4: The cache access and restore-after-read.

Recently, Wang *et al.* proposed to reduce write backs activities (to L2) by predicting if an evicted cacheline from L1 may be used in the future [24]. A negative prediction evicts the dirty cacheline from L1 to main memory directly (❼), which helps to reduces writes to STT-MRAM based L2 and thus improves its lifetime.

When STT-MRAM read disturbance manifests, the cells in L2 may be disturbed after read. A simple restore-after-read scheme [23] is to restore (i.e., write) the data back to the STT-MRAM cells. To mitigate the performance and write energy overhead, [21] proposed two optimizations: (i) it integrates a 4- to 8- entry table to buffer the read data and restore when the cache bank is idle. This helps to reduce the performance degradation (❸b). (ii) it restores only 1s since only 1s may be disturbed during read (when using the same current direction as writing '0'), which helps to reduce the restore energy.

3.2 The design details

Figure 5 presents an overview of our proposed SR (selective restore) design. It consists two schemes — *delayed-restore* and *read-before-restore*. *Delayed-restore* delays the restore operation till L1 eviction time, which helps to merge and eliminate many unnecessary restores. When there is a need to restore, *read-before-restore* reads the disturbed line using inverted reading current, which helps to identify the disturbed cells. SR only restores the disturbed cells.

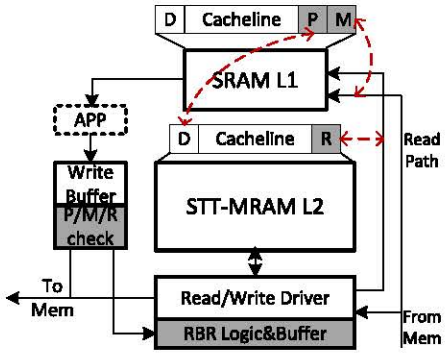


Figure 5: The proposed system architecture.

3.2.1 *Delayed restore*

Given L1 keeps the most up-to-date data after loading a cacheline from L2, it is not immediately necessary to restore the disturbed device line in L2. SR takes advantage of this observation to delay the restore operation till the corresponding line is evicted from L1. Based on the line status at the eviction time, SR decides how to restore the data.

To enable *delayed-restore*, SR attaches two bits to each cacheline in L1 — a 'M' bit indicates if the line is loaded from main memory directly; a 'P' bit indicates if the line is dirty in L2 when it is loaded. In addition, SR attaches one bit to each L2 line — a 'R' bit indicates whether the line has been read disturbed. SR strives to minimize its impact on L1/L2 cache miss rates by following the same replacement strategy. It works as follows.

- Servicing an L2 read request may result in a hit or a miss. An L2 hit loads the cacheline data to L1, saves its L2 dirty bit to the 'P' bit in L1, and clears the 'M' bit in L1. The read hit also sets the 'R' bit in L2.

  An L2 miss triggers main memory access, which returns data to both L1 and L2. It sets the 'M' flag bit and clears the 'P' bit.

- When a cacheline in L1 is chosen as a victim to be evicted, L1 tests its status. If the victim L1 line is dirty, SR writes it back to L2, and sets the dirty bit of the L2 line.

- If the victim L1 line is clean and its 'M' bit is set, the line is evicted without restore. The 'M' bit being set indicates that this line was loaded to L1 directly without disturbing L2. A restore is thus not necessary.

- If the victim L1 line is clean and its 'M' is not set, the line needs to be restored to L2. We differentiate two cases.

  The simple case is that the line to be restored can be

found in L2. A simple or an optimized restore operation can then be performed directly.

The other case, i.e., the line to be restored cannot be found in L2, is more complicated. The fact that we loaded the data from L2 (as 'M' is not set) but cannot find the line in L2 at L1 eviction time indicates that the line has been evicted from L2. In the baseline, evicting a L2 line triggers a write back to main memory if the line is dirty. To avoid that a disturbed line writes back to main memory, in our scheme, we skip writing back a read disturbed L2 line ('R' bit is set) no matter if it is dirty, and directly clear it. Afterwards, to ensure data integrity in main memory, we need to write back the evicted L1 line to main memory directly if 'P' bit is clear, and skip restore if 'P' is set.

SR helps to reduce many unnecessary restores, e.g., restoring a line that later becomes dirty is a waste. By eliminating the total number of restores, SR reduces the competition for cache banks and improves the memory performance.

3.2.2 *Read before restore*

A STT-MRAM cell writes '1' by applying SL to BL current and writes '0' by applying BL to SL current. Similarly, reading a STT-MRAM cell can be *write-1-like* or *write-0-like*. The former applies a small SL to BL read current while the latter applies a small BL to SL read current. It has been observed that *writing-0-like* read current may only disturb cells that store '1' as vice verse.

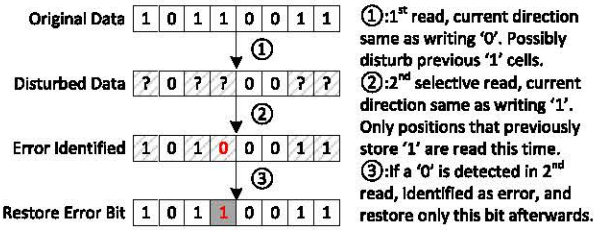


Figure 6: Identifying disturbed cells through inverted read current.

To reduce the restore energy, SR performs a selective read operation with inverted read current, which helps to identify all disturbed cells. SR then restore only disturbed cells. As shown in Figure 6, a normal STT-MRAM applies write-0-like read current such that it may disturb some cells that store '1'. Note that the read-out bit is correct, even though the cell itself gets disturbed.

When we decide to restore such a line, SR inverts the read current and applies write-1-like read current to read all '1' cells. The decision is based on the read-out values of the 1st read, e.g., we only read the 1st, 3rd, 4th, 7th and 8th cells in the example. The read-out value of the 2nd read indicates that the 4th cell is disturbed. The rest of cells are reliable. During the 2nd read, all disturbed cells in the 1st read are identified, and no more cells get disturbed.

After identifying the disturbed cells, SR restores only these cells rather than (i) restoring all cells when using a naive restore scheme; or (ii) restores all 1s with simple optimization [21].

Architecture enhancement. To enable read-before-restore, SR needs to enhance the read circuit to perform bidirectional read. The modified read path circuit, shown in Figure 7, is based on [23] and [11]. The read current direction is selected with *RM*. When doing a read operation, SL is usually connected to ground. Thus, bidirectional read also requires two reference cells voltage as input to SA.

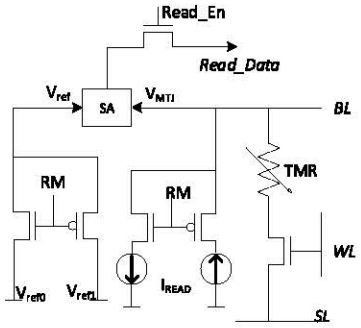


Figure 7: The enhancement to read circuit.

3.3 Integrated with Access Pattern Predictor

To reduce the number of write accesses to STT-MRAM and thus extend its lifetime, Wang *et al.* proposed access pattern predictor (APP) to predict whether a to-be-written-back cacheline from L1 may be used soon and, if not, save the line directly to memory [24]. Adopting *APP* in SR can further reduce the number of restores. In addition to the write back reduction as shown in [24], SR can skip restoring a clean L1 line even if reading this line has disturbed its clean L2 copy.

3.4 Overhead Analysis

SR adds 2 bits per L1 cache line and 1 bit per L2 cache line. That is 16KB extra storage, or 0.19% of total cache space. This is negligible. Read-before-restore needs a small modification on read path. Since STT-MRAM write drivers are bidirectional, the circuit revision is minimal.

4. EXPERIMENTAL METHODOLOGY

4.1 Settings

We used an in-house simulator to simulate a 4-core in-order processor running at 2GHz. Table 2 summarizes the details of the baseline configuration. The simulator simulates the entire L1/L2/Memory hierarchy with timing. We adopted the dynamic and leakage power of STT-MRAM from [10, 4]. We used the SPEC2006 benchmark suite and used the PIN tool [16] to collect the memory access traces after skipping the warming-up phase of each benchmark.

CPU	4-core single issue in-order CMP, 2GHz
L1	private, I/D separate, 32KB per core, 64B line
STT L2	private, 8MB per core, 16-way LRU, 64B line write back, write: 20 cycles, read: 5 cycles
Main Memory	8GB, 1 channel, 2 ranks, 8 banks per rank, 100 cycles access time

Table 2: Baseline Configuration

The performance metric *Normalized Speedup* is defined as $Speedup = IPC_{tech}/IPC_{ideal}$ [10], where IPC_{ideal} and IPC_{tech} are the instructions per cycle from the ideal scheme and the proposed scheme to be evaluated.

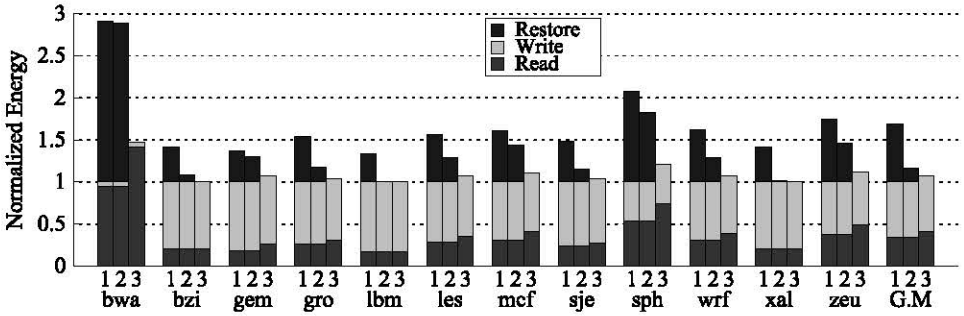


Figure 8: Comparing the dynamic energy of different schemes (1:RAR; 2:DR; 3:DR+RBR).

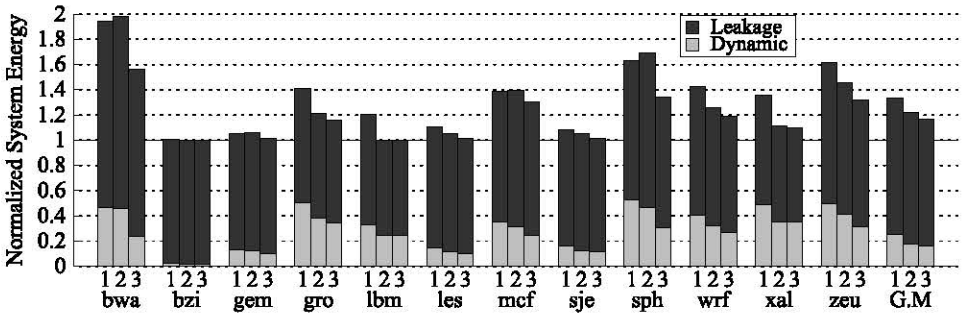


Figure 9: Comparing the system energy of different schemes (1:RAR; 2:DR; 3:DR+RBR).

4.2 Compared Schemes

We implemented and compared the following schemes.

- `Ideal` — This scheme assumes there is no read disturbance.
- `RAR(1)` — This scheme mitigates read disturbance by restoring after each read. The optimized restore-after-read design [21] is used.
- `DR(2)` — This scheme mitigates read disturbance by adopting delayed restore only.
- `RBR(3)` — In addition to `DR`, this scheme reads (selectively, with inverted current, and before restore) to identify disturbed cells and restores only disturbed cells.
- `APP` — In addition to `RBR`, this scheme adopts access pattern predictor to further reduce the number of restores to STT-MRAM based L2.

5. RESULTS

5.1 Energy Evaluation

Figure 8 compares the breakdown of the dynamic energy consumption in different schemes. All results are normalized to `Ideal`. In `RAR`, the restore activities introduces 69% more dynamic energy consumption. While the numbers of L2 reads and restores are the same, a restore operation consumes more energy (even by restoring only 0s).

With delayed restore, `DR` reduces the number of restores by skipping unnecessary ones, e.g., restoring a line that later on becomes dirty is a waste. `DR` reduces the restore energy overhead to 16% of `Ideal`. `DR+RBR` reduces the dynamic energy consumption further by eliminating restores to those un-disturbed cells. However `RBR` needs extra reads, which increase read energy consumption. On average, `DR+RBR` increases read energy by 6.5%, and reduces 68.8% restore energy of `Ideal`. It consumes only 6% more dynamic energy comparing to `Ideal`.

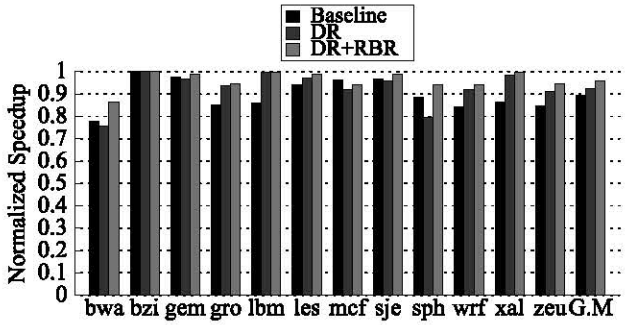


Figure 10: The performance comparison.

Figure 9 compares the breakdown of the system energy consumption in different schemes. All results are normalized to `Ideal`. Comparing to `RAR`, on average, `DR` and `DR+RBR` save 11% and 17% total energy, respectively. From the figure, the total savings come from both dynamic and leakage energy reductions. The leakage reduction is due to short execution time as discussed next.

5.2 Performance Comparison

Figure 10 compares the performance of different schemes. `RBR` integrates a small restore table, which helps to reduce the slowdown from 20% (not shown in the figure) to around 10% performance degradation. `DR` helps to reduce the number of restore operations, and thus reduce the contention of cache banks. For *sph*, `DR` is worse than `RBR` because clustered restores at eviction time may hurt L2 read performance and lead to larger degradation. On average, `RBR` and `DR` achieves comparable overall performance.

`DR+RBR` has the smallest performance degradation — on average, `DR+RBR` is 5% slower than `Ideal`. While `RBR` introduces extra read operations, we observed that many checks result in zero disturbed cells. Restore operation can be skipped if there is no disturbed cell, which helps to improve the overall performance.

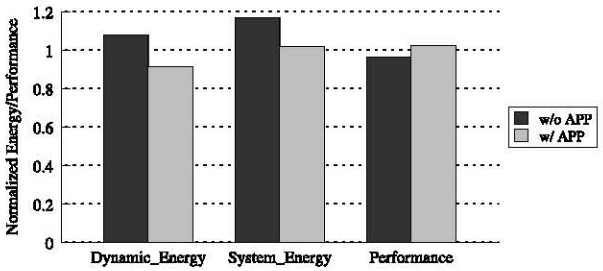


Figure 11: Integrating SR and APP.

5.3 Integrated with APP

As discussed in Section 3.3, *APP* (access pattern predictor) reduces the number of write operations to STT-MRAM cache and thus can be adopted to further optimize our SR scheme. Figure 11 presents the energy saving and performance improvement by integrating APP with SR. The dynamic energy of L2 is reduced to 91% of `Ideal` because not only write backs but also clean restores are reduced. The system energy and the overall performance are close to ideal case too.

6. RELATED WORK

The reliability issues on STT-MRAM, such as write / read disturbance and write / read failure were discussed in [15]. Previous work [27] showed that the read current needs to be scaled with write current to keep the safe read margin, otherwise read disturbance errors shall manifest at small technology nodes. A recent chip demonstration [23] ($45nm$) of STT-MRAM treats reads as destructive and restores the original data after each read. [21] proposes two write modes under read disturbance — one performs restore-after-read to ensure data integrity; the other does not if having high error tolerance. As discussed in the paper, the energy overhead due to restore is still high with optimizations in [21].

7. CONCLUSION

In this paper, we propose SR (selective restore) to mitigate read disturbance in STT-MRAM caches. SR consists of two designs — delayed-store helps to delay the restore operation till the L1 eviction time such that many unnecessary restores are eliminated; read-before-restore further reduces the energy consumption of each restore by reading the disturbed line with inverted read current, detecting the disturbed cells, and restoring only disturbed cells. Our experimental results showed that SR greatly reduces both dynamic and system energy consumption. On average, it reduces the dynamic energy overhead to 6% of the ideal case (i.e., with no read disturbance).

8. ACKNOWLEDGMENTS

This work is supported by National Science Foundation 1422331, 0747242, and National Natural Science Foundation of China No.91318301, 61328491, 61472179. The authors thank all the anonymous reviewers for their advice and comments.

9. REFERENCES

[1] C.-T. Cheng, *et al.*, "A high-speed current mode sense amplifier for Spin-Torque Transfer Magnetic Random Access Memory," in *MWSCAS*, 2010.

[2] K. C. Chun, *et al.*, "A Scaling Roadmap and Performance Evaluation of In-Plane and Perpendicular MTJ Based STT-MRAMs for High-Density Cache Memory," *JSSC*, 48(2), 2013.

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

[4] A. Driskill-Smith, *et al.*, "Latest Advances and Roadmap for In-Plane and Perpendicular STT-RAM," in *IMW*, 2011.

[5] D. Halupka, *et al.*, "Negative-resistance read and write schemes for STT-MRAM in 0.13 μm CMOS," in *ISSCC*, 2010.

[6] M. Hosomi, *et al.*, "A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-RAM," in *IEDM*, 2005.

[7] HTC, "Desire 820," , 2014, `http://blog.htc.com/2014/09/htc-desire-820/`.

[8] Y. Huai, "Spin-transfer torque MRAM (STT-MRAM): Challenges and prospects," *AAPPS Bulletin*, 18(6):33–40, 2008.

[9] Intel, "Atom Z3795," , 2014, `http://ark.intel.com/products/80267`.

[10] L. Jiang, *et al.*, "Constructing Large and Fast Multi-level Cell STT-MRAM Based Cache for Embedded Processors," in *DAC*, 2012.

[11] T. Kawahara, *et al.*, "2Mb Spin-Transfer Torque RAM (SPRAM) with Bit-by-Bit Bidirectional Current Write and Parallelizing-Direction Current Read," in *ISSCC*, 2007.

[12] T. Kawahara, *et al.*, "2 Mb SPRAM (SPin-Transfer Torque RAM) With Bit-by-Bit Bi-Directional Current Write and Parallelizing-Direction Current Read," *JSSC*, 43(1), 2008.

[13] S. V. Kumar, *et al.*, "Impact of NBTI on SRAM Read Stability and Design for Reliability," in *ISQED*, 2006.

[14] Lenovo, "Thinkpad10," , 2014, `http://shop.lenovo.com/us/en/tablets/thinkpad/thinkpad-10/`.

[15] J. Li, *et al.*, "Variation-tolerant Spin-Torque Transfer (STT) MRAM array for yield enhancement," in *CICC*, 2008.

[16] C.-K. Luk, *et al.*, "Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation," in *PLDI*, 2005.

[17] MediaTek, "MT5692," , 2013, `http://event.mediatek.com/_en_octacore/`.

[18] K. Ono, *et al.*, "A disturbance-free read scheme and a compact stochastic-spin-dynamics-based MTJ circuit model for Gb-scale SPRAM," in *IEDM*, 2009.

[19] Qualcomm, "Snapdragon 615," , 2013, `https://www.qualcomm.com/products/snapdragon/processors/615`.

[20] B. Schroeder, *et al.*, "DRAM Errors in the Wild: A Large-scale Field Study," in *SIGMETRICS*, 2009.

[21] Z. Sun, H. Li, *et al.*, "A dual-mode architecture for fast-switching STT-RAM,", in *ISLPED*, 2012.

[22] C. W. Smullen, IV, *et al.*, "The STeTSiMS STT-RAM Simulation and Modeling System," in *ICCAD*, 2011.

[23] R. Takemura, *et al.*, "Highly-scalable disruptive reading scheme for Gb-scale SPRAM and beyond," in *IMW*, 2010.

[24] Z. Wang, *et al.*, "Adaptive placement and migration policy for an STT-RAM-based hybrid cache," in *HPCA*, 2014.

[25] C. Wilkerson, *et al.*, "Reducing Cache Power with Low-cost, Multi-bit Error-correcting Codes," in *ISCA*, 2010.

[26] X. Wu, *et al.*, "Hybrid cache architecture with disparate memory technologies," in *ISCA*, 2009.

[27] Y. Zhang, *et al.*, "The Prospect of STT-RAM Scaling From Readability Perspective," *TMAG*, 48(11), 2012.

[28] W. Zhao, *et al.*, "High Speed, High Stability and Low Power Sensing Amplifier for MTJ/CMOS Hybrid Logic Circuits," *IEEE Trans. on Magnetics*, 45(10), 2009.

[29] Mekhiel, Nagi N, "Multi-level cache with most frequently used policy: A new concept in cache design," *CAINE*, 1995