ROSS: A Design of Read-Oriented STT-MRAM Storage for Energy-Efficient Non-Uniform Cache Architecture

Jie Zhang, Miryeong Kwon, Chanyoung Park, Myoungsoo Jung, and Songkuk Kim
School of Integrated Technology,
Yonsei Institute Convergence Technology,
Yonsei University

jie@yonsei.ac.kr, mkwon@camelab.org, chanpark@yonsei.ac.kr, m.jung@yonsei.ac.kr, songkuk@yonsei.ac.kr

Abstract

Spin-Transfer Torque Magnetoresistive RAM (STT-MRAM) is being intensively explored as a promising on-chip last-level cache (LLC) replacement for SRAM, thanks to its low leakage power and high storage capacity. However, the write penalties imposed by STT-MRAM challenges its incarnation as a successful LLC by deteriorating its performance and energy efficiency. This write performance characteristic unfortunately makes STT-MRAM unable to straightforwardly substitute SRAM in many computing systems.

In this paper, we propose a hybrid non-uniform cache architecture (NUCA) by employing STT-MRAM as a read-oriented on-chip storage. The key observation here is that many cache lines in LLC are only touched by read operations without any further write updates. These cache lines, referred to as singular-writes, can be internally migrated from SRAM to STT-MRAM in our hybrid NUCA. Our approach can significantly improve the system performance by avoiding many cache read misses with the larger STT-MRAM cache blocks, while it maintains the cache lines requiring write updates in the SRAM cache. Our evaluation results show that, by utilizing the read-oriented STT-MRAM storage, our hybrid NUCA can better the performance of a conventional SRAM-only NUCA and a dead block aware STT-MRAM NUCA by 30% and 60% with 45% and 8% lower energy values, respectively.

1 Introduction

Last-level cache (LLC) is utilized as a shared resource in most commercial multicore products and co-processor architectures [6], making it a pivotal design component in determining both system performance and energy-efficiency. While many emerging applications enjoy the massive parallelism and high computational power of multicore systems, they are required to manage large datasets presenting challenge of scale, which in turn leads to the demand for integrating larger LLCs in modern computing system [2].

Static Random Access Memory (SRAM) has been the prevailing technology for on-chip cache, successfully catering to the latency demands of the high performance processors. However, SRAM's cell design constitutes of a large number of transistors (usually six, at least four), making it a low-density device with considerably high leakage power. With increasing demand for larger caches, SRAM is struggling to keep up with the density and energy-efficiency requirements set by state-of-the-art system designs.

Thanks to the device-level advantages of STT-MRAM such as high-density structure, zero leakage current, and very high endurance, it comes out as an excellent candidate to replace age-old SRAM for LLC design. However, the performance of STT-MRAM is critically sensitive to write frequency due to its high write latency and energy values. Therefore, an impulsive replacement of SRAM with high-density STT-MRAM, simply for increasing LLC capacity, can deteriorate cache performance and introduce poor energy consumption behavior.

Extensive research is taking place to address these challenges. [8] proposed a 3D-stacked STT-MRAM, which intends to hide the long-latency of STT-MRAM on writes by adding a small SRAM write buffer to each STT-MRAM bank. [9] discussed a memory request scheduling scheme on a 3D multicore environment integrating STT-MRAM, which aims to resolve the STT-MRAM write issues at the on-chip network level. This approach promoted an idea of re-routing subsequent requests to idle cache banks, instead of the write-busy bank. While these proposed schemes can partially alleviate the write overhead problem imposed by STT-MRAM, the quest for an energy-efficient cache still remains unconquered. In addition, the hybrid cache using SRAM-based write buffer is unaware of application behaviors, and its benefits are restricted by the amount of available buffer at runtime.

We observe that performance and energy-efficiency of a hybrid cache critically depend on its internal data-movement trend, and by studying such trends we can modify the cache architecture accordingly to get the best out of it. Specifically, we observe that nearly 90% of the data in a LLC can actually be written only once during its lifetime, which has been reported as "deadwrites" in [13]. One of the potential approaches to eliminate STT-MRAM's disadvantages, being aware of this deadwrite

characteristics, is to bypass the writes associated to the deadwrites to the underlying main memory. However, we also observe that, about 60% of total read misses results from such bypassed write data.

In this paper, we propose a ***R**ead-**O**riented **S**TT-MRAM **S**torage (ROSS) based non-uniform cache architecture (NUCA)*, that can avail combined benefits of a hybrid design and the data movement/usage trends in a LLC. We built upon the standard NUCA architecture, and modified it to develop a high-capacity hybrid cache where designated SRAM banks are replaced with STT-MRAM banks. This inclusion of high-density STT-MRAM blocks enables the cache to attain larger storage capacity, as well as higher energy efficiency. In our proposed architecture, cache data with singular-write characteristics (i.e., data is written only once and is not re-referenced by any latter write operations) are migrated from the SRAM cache blocks to the STT-MRAM blocks inside the LLC, rather than bypassing them to the main memory. As a result, our proposed scheme allows the cache controller to free up sufficient SRAM blocks for write intensive data, without sustaining the read operation overheads imposed by off-chip memory accesses. Our comprehensive evaluation results show that, our ROSS cache significantly improves the overall LLC performance, incurring only a minor overhead from the data migration process. The main **contributions** of this work can be summarized as follows:

• *Evaluation of data movement trend in the LLC.* We extensively evaluate data read and write trend in the LLC for a wide range of memory-intensive workloads. This motivational evaluation allows us to better understand the actual trend of data movement in the LLC, and to modify our architecture accordingly, for optimum performance and energy-efficiency.

• *Development of ROSS based NUCA.* Motivated from the observations, we propose two novel hybrid ROSS based non-uniform cache architectures. The first one is hybrid ROSS (HB-ROSS) that takes advantage of SRAM's low write latency and small write power as well as STT-MRAM's high density and energy efficiency by detecting and retiring data with singular-write from SRAM to STT-MRAM. Second, to further optimize energy efficiency, early retirement ROSS (ER-ROSS) is designed to reduce SRAM capacity, while maintaining high performance. Differing from HB-ROSS, ER-ROSS retires all potential singular-write data sets at early stage.

• *Comprehensive system-level evaluation.* We evaluate and compare our proposed ROSS-based NUCA with a SRAM-only NUCA and a dead block aware STT-MRAM based NUCA, and show that our proposed ROSS caches show significant improvement in terms of both cache performance and energy-efficiency. Specifically, compared to the prior work, our singular-write aware ROSS cache improves LLC performance by upto 60%, and consumes upto 50% less energy at system level.

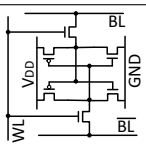
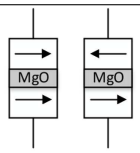
	SRAM	STT-MRAM
Cell Structure		
Cell Area (F^2)	50-120	6-40
Leakage (mW)	75.7	6.6
Read Latency (ps)	397	238
Write Latency (ps)	397	6932
Read Energy (pJ)	35	13
Write Energy (pJ)	35	90

Table 1: Features of 32KB SRAM and STT-MRAM caches.

2 Background

SRAM Cache. Table 1 lists the technology features of SRAM and STT-MRAM. In practice, SRAM has a six-transistor structure and has high static power due to sub-threshold and gate leakage currents. It exhibits excellent latency values and low dynamic energy consumption due to the simple read/write operations and its latch-like storage mechanism.

STT-MRAM Cache. Compared to SRAM, STT-MRAM is comprised of a magnetic tunnel junction (MTJ) and a single access transistor. As a result, STT-MRAM caches have higher density (3x $\sim$ 4x) and lower leakage power. However, for the write operation, STT-MRAM involves physical rotation of the MTJ free layer, which requires high write latency and write power consumption. High write overhead becomes a serious obstacle for STT-MRAM to completely replace SRAM.

NUCA. In modern cache architectures, large LLC is employed to relieve the burden of frequent accesses to underlying main memory. However, global wire delay followed by increased cache area size, has emerged as a dominant factor for increasing cache access time and more access contentions. The non-uniform cache architecture (NUCA) is proposed to reduce the penalty of long wire delay by dividing the cache into smaller banks. Each bank has non-uniform latency based on bank locations, smaller than what it would be if the whole cache was a uniform cache.

3 Design and Implementation of ROSS

In this section, we first describe the motivation behind our proposed ROSS based NUCA architecture, and then discuss about the LLC topology, NUCA baseline and

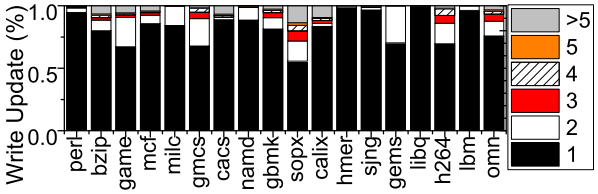


Figure 1: Last Level Cache eviction trend.

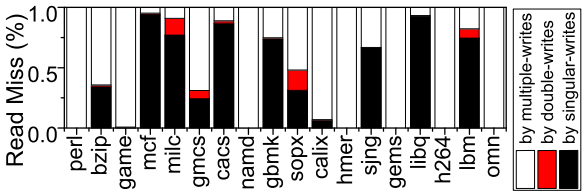


Figure 2: Read-miss distribution.

ROSS scheme. At last, we provide the implementation details of ROSS.

3.1 Motivation

Figure 1 shows cache eviction trend for the tested workloads and "1" $\sim$ ">5" denote the fraction of cache blocks which have different write times. One can observe from this figure that, nearly 90% of the written data are actually never re-written before their eviction, referred to *singular-writes*. Singular-writes can be generated by different situations such as data fill by a single read-miss, a redundant write just before a write-back, or an access sequence to a cache block ending with a write-back request. In addition, we make another critical observation from our motivational evaluation regarding singular writes. Specifically, Figure 2 shows the distribution of read-misses for evicted data that experience different times of write operations. In the figure, each column presents the total number of read-misses. One can observe from this figure that the number of read misses caused by evicting data which has singular writes accounting for more than 60% of the total read misses.

Based on these observations, we propose our ROSS-based NUCA where the data blocks with multiple writes are assigned to SRAM blocks to take advantage of SRAM's low write latency, while the data blocks with singular writes which are less impacted by longer write latency, can be migrated in STT-MRAM, and can reap the benefit of huge capacity brought by STT-MRAM.

3.2 An Overview of ROSS Architecture

As the capacity of LLC keeps increasing, the conventional single-bank cache architecture exposes significant latency overhead, due to the high RC delay of long wordlines and bitlines. To mitigate such high latency, we divide LLC into heterogeneous (SRAM and STT-MRAM)

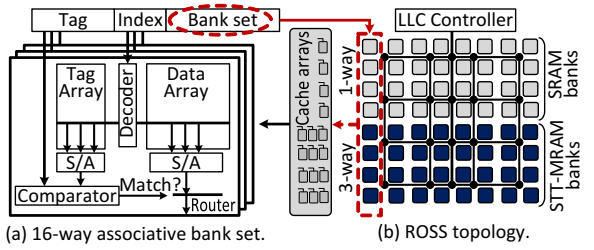


Figure 3: An overview of our proposed ROSS.

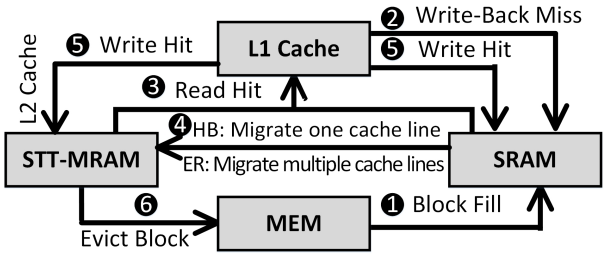


Figure 4: Cache management policy for ROSS.

cache banks to model the bottom line of our hybrid cache (referred to as "ROSS"). Figure 3b shows the topological overview of our proposed ROSS-based NUCA. Our proposed hybrid cache consists of 64 cache banks, which are connected by a point-to-point mesh network and managed by our LLC controller. The LLC controller is responsible for coordinating cache accesses inside LLC, communicating with memory and lower-level cache, and making decision on data replacement. On the other hand, each cache bank is built up by either SRAM array or STT-MRAM array. To take advantage of SRAM's shorter access latency, the SRAM-based cache bank is placed close to the LLC controller, which can significantly reduce the wire latency overhead, while STT-MRAM banks are placed off the LLC controller.

3.3 Read-Oriented Hybrid NUCA

Traditional NUCA designs [1] are not optimized as a platform for ROSS based LLC. In this work, we propose a well-tailored NUCA and its corresponding cache management policy, which are shown in Figure 4.

Tailored NUCA. Considering the data movement overhead (high write latency and energy consumption) in STT-MRAM banks, we employ S-NUCA which statically maps data to STT-MRAM banks and does not allow data migration later on. On the other hand, SRAM banks follow the rule of D-NUCA strategy [1], which adjusts hot data sets among SRAM banks to achieve best access latency [1]. In addition, due to the limited SRAM capacity, SRAM banks are also designed to migrate singular-write data to STT-MRAM banks. Consequently, internal data migration only happens between SRAM banks or from SRAM to STT-MRAM banks.

Cache access workflow in ROSS. Figure 4 shows the

basic cache management policy for our ROSS cache. If one data request misses in L1 cache, that request continues to inquire the cache banks in LLC. In case of cache hit in either SRAM bank or STT-MRAM bank, that bank will serve the requested data directly to the host ❸❺. On the other hand, if cache misses in LLC, memory fill will be done to SRAM blocks on read miss and data block will be written to SRAM banks on write-back miss ❶❷. Considering multicore system is sensitive to latency, such cache miss strategy is optimized to provide fast write response. When more space is required in SRAM banks, SRAM banks will migrate cache blocks to STT-MRAM banks ❹. The specific migration policy will be described later on. Finally, when STT-MRAM is full, it will evict data blocks to underlying memory based on pseudo LRU policy ❻.

Address mapping policy. Multibanked cache shows substantial flexibility for mapping lines to banks, as data can be placed in any cache line of any cache bank. However, such mapping policy incurs overwhelming memory and latency overheads to maintain a big address mapping table. To balance the trade off between bank utilization and overhead, we applied *simple mapping* [1], which is shown in Figure 3a. According to *simple mapping*, 4 SRAM banks and 4 STT-MRAM banks are grouped as a *bank set*. The *bank set* can be regarded as an associative cache structure with each *cache set* spreading across multiple banks. Each bank consists of several pairs of tag arrays and data arrays as one or more ways of the cache set (c.f. Figure 3a). Based on the modified cache structure, the search address has been partitioned into *Tag*, *Index*, and *Bank set*. Any incoming cache request will be mapped to one bank set based on *Bank set* field.

Data search policy. As shown in Figure 3a, when searching for a data block in cache, cache controller decides which bank set the data belongs to, according to the *bank set*. Then, the remaining address information is sent to each pair of tag array and data array in banks of the matched bank set. Specifically, the *Index* field is used by row decoder to activate rows in tag arrays and *tag* field is used for comparison.

Data migration policy. In our hybrid ROSS (HB-ROSS) design, whenever a SRAM bank is full and should clean up space for incoming data, one cache block which was written for one time (singular-write candidate) is selected to move to a STT-MRAM bank. If there are multiple candidates, the oldest data block among them is retired. We assume that the oldest data that only experiences an one-time write has the highest possibility to be a singular-write data during its lifetime, and therefore should be migrated to a STT-MRAM bank.

Unfortunately, HB-ROSS cannot sufficiently relieve the network burden of data migration from SRAM to STT-MRAM banks. For better performance, we introduce an aggressive **E**arly **R**etirement **ROSS** (ER-ROSS) cache design. In ER-ROSS, instead of passively migrating single cache block to STT-MRAM on cache fill, cache controller searches for all qualified cache blocks and migrates them to STT-MRAM when network is not busy. While this aggressive migration cannot guarantee all the cache blocks are singular-write, the penalty can be mitigated with significant advantages such as lower leakage from less SRAM banks, and relieving of potential traffic burst by offering more free space in SRAM.

3.4 Design Details of ROSS

To support the functionality of ROSS scheme, we incorporate a few components inside the cache controller. Firstly, multiple status registers are organized as a bitmap table for every cache line to indicate the number of write requests accessed by each data. Status "0" represents data with one-time write, while status "1" means the corresponding cache line is accessed by multiple write operations. These status registers are initialized or updated after cache register handles incoming read/write requests. In addition, we add 8-bit timing counters for every cache line in SRAM banks to cooperate with status registers by storing more cache lines' information. The timing counter has three operations:

Initialize: Whenever a cache line is updated with a new data block, the corresponding timing counter will be reset by the cache controller. For example, as for singular-write data, the timing counter only reset when its cache line get filled. On the other hand, as for multi-write data, the timing counter is reset when cache line is accessed by a write request. Note that timing counter cannot replace status registers, as timing counter cannot distinguish between singular-write data and multi-write data.

Poll: We introduce a polling mechanism to increment every timing counter periodically, unless a timing register has reached its maximum. Another responsibility of the polling mechanism is to check if any timing counter has reached the predefined threshold (e.g. 200). if one cache line exceeds the threshold which means no write requests access the cache line for a long time, the cache line would be evicted to a STT-MRAM bank.

Evict: In HB-ROSS, evicting a cache line from SRAM banks to STT-MRAM banks follows two criteria: "Status register = 0" and "Max of all timing counters", which decide the "oldest singular-write data". On the other hand, in ER-ROSS, "Status register = 0" and "timing counter reaches threshold" are the criterion to evict cache lines.

4 Evaluation Methodology

Simulation Setup. For our evaluation, we have used the gem5 simulator [11], one of the most recognized archi-

Processor	1 core, OoO execution, SE mode
Frequency	3.2GHz
L1 Cache	16KB/core, 2-way, 2 cycles
SRAM L2	32KB/bank, R/W : 20 cycles
STT-MRAM L2	96KB/bank, R/W : 20/60 cycles
Network	Wormhole Switching, 2 cycles
Off-chip Mem	4GB DDR3 DRAM FR-FCFS
Benchmark	SPEC2006
Workloads	PerlBench, mcf, milc, libquantum, lbm, cactusADM, sjeng, and gobmk

Table 2: Major simulation parameters and workloads.

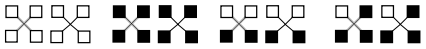
	ND-NUCA	DB-NUCA	HB-ROSS	ER-ROSS
STT-MRAM / SRAM				
STT-MRAM Bank Fraction	0%	100%	50%	75%
Total Storage	2MB	6MB	4MB	5MB
Associativity	8	24	16	20
Capacity Constitution	32KBx64	96KBx64	32KBx32 +96KBx32	32KBx16 +96KBx48

Table 3: Comparative details of 4 cache configurations.

tectural simulator. We modified the cache segment of the simulator to evaluate the STT-MRAM cache and our proposed hybrid configurations. Table 2 depicts the main simulation parameters.

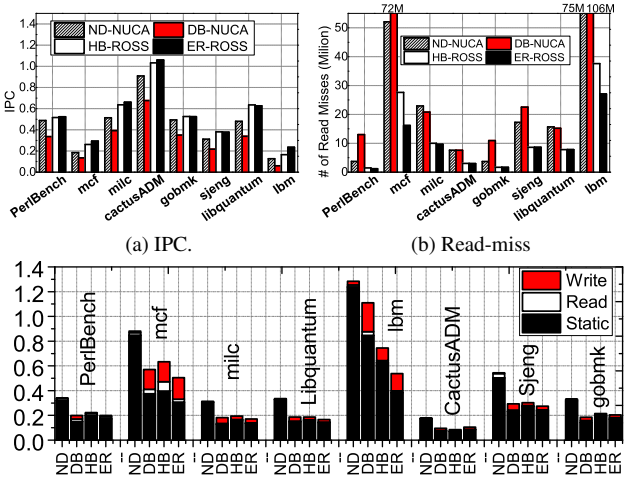
Simulated Configurations. In our evaluation, we compared two variations of our proposed ROSS architecture with SRAM-only NUCA and dead block aware STT-MRAM-only NUCA. Table 3 shows the details of the four architectures.

• *Normal Non-Uniform Cache Architecture (ND-NUCA).* In this model, the whole on-chip cache is constructed with traditional SRAM in D-NUCA model [1].

• *Dead Block aware Non-Uniform Cache Architecture (DB-NUCA).* The entire cache banks are made up of STT-MRAM. The DB-NUCA is also aware of dead-blocks, and bypasses dead blocks directly to the off-chip memory in an attempt to avoid the write penalties brought by STT-MRAM. We mimic the implementation of dead block detection in [13].

• *Hybrid Read-Oriented STT-MRAM Storage (HB-ROSS).* In this configuration, half of SRAM cache banks are replaced with STT-MRAM. HB-ROSS migrates singular-writes only when the SRAM banks have no more room.

• *Early Retirement aware Read-Oriented STT-MRAM Storage (ER-ROSS).* This configuration represents our complete scheme for designing a high-performance albeit energy-efficient cache architecture. Here, six-eighth of the SRAM cache blocks are replaced with STT-MRAM blocks, greatly increasing the capacity of the cache. It also has the ability to retire multiple data blocks at an early stage.



(a) IPC. (b) Read-miss

(c) Energy Consumption of Cache.

Figure 5: Performance and energy analysis.

5 Evaluation Results

Figure 5 shows the evaluation result of the six workloads we evaluated, which including the system performance and energy comparison of the two ROSS cache configurations with the DB-NUCA and ND-NUCA.

Raw IPC. In the figure 5a, for all eight workloads, HB-ROSS outperforms the baseline ND,DB-NUCA by 20% and 50%, respectively. ER-ROSS outperforms the baseline ND/DB-NUCA by 30% and 60%, respectively. This result supports our expectation that, the high storage capacity and better utilization of SRAM blocks in ROSS cache enable better performance from the cache. In addition, ER-ROSS provides, on average, 10% better performance than HB-ROSS. For the three workloads libquantum, sjeng and gobmk, HB-ROSS performs slightly better than ER-ROSS. However, it should be noted that, in addition to the core cache architecture, IPC performance is also dependent on various other system parameters and workload characteristics.

LLC Read-miss Rate. Figure 5b shows the read-miss rate in the last-level cache (LLC). DB-NUCA shows a higher read-miss rate than ND-NUCA by 67%, on average; which is unexpected considering that DB-NUCA enjoys a much higher capacity STT-MRAM LLC. We believe this degradation in read-miss is caused by the 'bypass all singular-write to memory' policy, since 60% of those singular-write has to be retrieved by read operations. Importantly, for all eight workloads, both HB/ER-ROSS outperforms the ND-NUCA by 55%, on average. This result supports our claim that, ROSS can substantially improve cache performance by not migrating singular-writes to the off-chip memory. In addition, for all eight workloads ER-ROSS provides, on average, 5% better performance than HB-ROSS. We believe this is because, ER-ROSS, compared to HB-ROSS, terminates many more defunct cache line at an early stage and keeps

the LLC free for future read operations.

Cache energy consumption. Figure 5c shows the energy consumption by the four cache architectures for the eight workloads. As expected, the cache energy component is dominated by static leakage energy. From the figure one can see that, the baseline ND-NUCA, with its all-SRAM design, shows the worst performance in terms of cache energy consumption. DB-NUCA reduces leakage energy by replacing SRAM with STT-MRAM, and improves the baseline performance by 40%, on average. Both HB/ER-ROSS displays similar cache energy consumption as DB-NUCA, and shows performance improvement of 45%, on average, over ND-NUCA. Specifically, for the lbm workload, ER-ROSS improves energy consumption compared to the baseline ND-NUCA by 60%. Finally, ER-ROSS improves energy consumption of HB-ROSS by 5%, on average.

6 Acknowledgement

This research is supported in part by MSIP "ICT Consilience Creative Program" IITP-R0346-16-1008, NRF-2015M3C4A7065645, NRF 2016R1C1B2015312 DOE grant DE-AC02-05CH1123 and MemRay grant (2015-11-1731). M. Jung has an interest in being supported for any type of engineering or costumer sample product on emerging NVM technologies (e.g., PRAM, X-Point, ReRAM, STT-MRAM etc.).

7 Conclusion

In this paper, we proposed a hybrid NUCA using Read-Oriented STT-MRAM Storage (ROSS), which is able to fully utilize the benefits of STT-MRAM by detecting and deploying singular-write data in STT-MRAM banks. The evaluaion results show that, our proposed ROSS caches exhibit upto 60% and 50% improvement in terms of performance and energy-efficiency, compared to prior work.

References

[1] Kim, Changkyu, et al. "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches." In Acm Sigplan Notices, 2002.

[2] Lin, Junmin, et al. "Understanding the memory behavior of emerging multi-core workloads." In Proc. of ISPDC, 2009.

[3] Khvalkovskiy, A. V., et al. "Basic principles of STT-MRAM cell operation in memory arrays." In Journal of Physics, 2013.

[4] Zhao, W. S., et al. "Failure and reliability analysis of STT-MRAM." In Microelectronics Reliability, 2012.

[5] Dorrance, Richard. "Modeling and Design of STT-MRAMs." PhD diss (2011).

[6] Molka, Daniel, et al. "Memory performance and cache coherency effects on an Intel Nehalem multiprocessor system." In Proc. of PACT, 2009.

[7] Guo, Xiaochen, et al. "Resistive computation: avoiding the power wall with low-leakage, STT-MRAM based computing." In ACM SIGARCH Computer Architecture News, 2010.

[8] Sun, Guangyu, et al. "A novel architecture of the 3D stacked MRAM L2 cache for CMPs." In Proc. of HPCA, 2009.

[9] Mishra, Asit K., et al. "Architecting on-chip interconnects for stacked 3D STT-RAM caches in CMPs." In ACM SIGARCH Computer Architecture News, 2011.

[10] Rizzo, N. D., et al. "Thermally activated magnetization reversal in submicron magnetic tunnel junctions for magnetoresistive random access memory." In Applied Physics Letters, 2002.

[11] Binkert, Nathan, et al. "The gem5 simulator." In ACM SIGARCH Computer Architecture News, 2011.

[12] Smullen, Clinton W., et al. "Relaxing non-volatility for fast and energy-efficient STT-RAM caches." In Proc. of HPCA, 2011.

[13] Junwhan Ahn, et al. "DASCA: Dead Write Prediction Assisted STT-RAM Cache Architecture". In Proc. of HPCA, 2014.

[14] Jin, Youngbin, Mustafa Shihab, and Myoungsoo Jung. "Area, Power, and Latency Considerations of STT-MRAM to Substitute for Main Memory." Proc. ISCA. 2014.