Cache Revive: Tuning Retention times of STT-RAM Caches for Enhanced Performance in CMPs.

Anonymous MICRO submission

Paper ID 761

Abstract

Spin-Transfer Torque RAM (STT-RAM) is a CMOS compatible emerging non-volatile memory (NVM) technology that has the potential to replace the conventional on-chip SRAM caches for designing a more efficient memory hierarchy for future multicore architectures. However, it's high write latency and dynamic write energy are major obstacles for being competitive with the SRAM-based cache hierarchy. On the other hand, STT-RAM technology has another adaptable feature that it is possible to reduce its write latency by reducing its retention time, thereby making it volatile. In this paper, we exploit this volatile property of the STT-RAM for designing an efficient L2 cache architecture. The paper addresses several critical design issues such as how do we decide a suitable retention time for last level cache, what is the relationship between retention time and write latency, and how do we architect the cache hierarchy with a volatile STT-RAM. In this paper, we propose to trade off the non-volatility (data retention time) for better write performance/energy in STT-RAM cache design. We study two data-retention relaxation cases, one with data retention time of 1 second, which satisfies the lifetime requirement of typical cache blocks; and the other one with data retention time of 1ms, which is a more aggressive design for better performance/energy gains but a data refreshing mechanism is needed. In the aggressive data retention time relaxation design, for the rest of the cache blocks that have a higher inter-write time than the STT-RAM retention time, we propose an architectural solution to identify these blocks with a per block 2 bit counter, temporarily save a limited number of MRU blocks in a buffer, and write-back the rest of the dirty blocks to avoid any data loss. Our experiments with 4 and 8-core architectures with an SRAM-based L1 cache and STT-RAM-based L2 cache indicate that not only we can eliminate the high write overhead of an NVM STT-RAM, but can provide on an average 10-12% improvement in IPC compared to the traditional SRAM-based design, while reducing the energy consumption significantly

1. Introduction

Designing an efficient memory hierarchy for multicore architectures is a critical but challenging problem. As the number of cores on a chip increases with technology scaling, the demand on the on-chip memory would increase significantly, thereby worsening the memory wall problem [5]. The memory wall problem is critical both from the performance (memory density) and power perspectives. Thus novel technology, circuit and architectural techniques are currently being explored to address the memory wall problem for many core systems.

Spin-Transfer Torque RAM (STT-RAM) is a promising memory technology that delivers on many aspects desirable of an universal memory. It has the potential to replace the conventional on-chip SRAM caches because of its higher density, competitive read times and lower leakage power consumption compared to SRAM. However, the high write latencies and write energy are key drawbacks of this technology for providing competitive or better performance compared to the SRAM-based cache hierarchy. Consequently, recent efforts have focused on masking the effects of high write latencies and write energy at the architectural level []. In contrast to these architectural approaches, a recent work explored the feasibility of relaxing STT-RAM data retention times to reduce both write latencies and write energy []. This adaptable feature of tuning the data retention time can be exploited in several dimensions. The focus of this paper is to tune this data retention time to closely match the required lifetime of the last level cache line blocks to achieve significant performance and energy gains. In this context, the paper addresses several design issues such as how to decide an appropriate retention time for the last level cache, what is the relationship between retention time and write latency, and how do we architect the cache hierarchy with a volatile STT-RAM.

The non-volatile nature and non-destructive read ability of STT-RAM provides a key difference with regard to traditional on-chip cache design with SRAM technology. However, for many applications, it is sufficient if the data stored in the last level of a cache hierarchy remains valid for a few tens of milliseconds. Consequently, the duration of data retention in STT-RAM is an obvious candidate for device optimization for cache design. We, therefore, conduct an application-driven study to analyze the inter-write times of the L2 cache blocks to decide a suitable data retention time. Although lifetime analysis of cache lines has been the conducted earlier to improve performance and reduce power consumption [], we revisit this topic with a different intention - correlating STT-RAM data retention time to cache life time. An extensive analysis of PARSEC and SPEC benchmarks using the M5 simulator [] indicates that the average inter-write times for most of the L2 cache blocks is around 10ms, and thus, we advocate our STT-RAM design with this retention time.

We conduct a detailed device level analysis of the STT-RAM cells to analyze the write current versus write pulse width tradeoffs, cell area analysis, and retention time stability analysis to capture the relationship between area, read/write latency and leakage power as a function of the retention time. Our observations, in contrast to the results reported in [] indicate that retention times in the range of ms are probably more achievable than in the micro second range. Thus, using this ms range retention time model, we then propose effective architectural techniques to avoid any data loss due the volatile STT-RAM-based cache hierarchy.

A key challenge in determining a suitable data retention times for the STT-RAM is to balance the reduced write latency of cells with lower retention time with the overhead for data refresh or write back of cache lines with longer lifetimes. In this paper, we compare 3 different STT-RAM based cache designs: (1) STT-RAM cache without retention time relaxation (¿10 years of data retention time); (2) STT-RAM cache with retention time of 1 second, which is long enough for the lifetime of majority of the cache lines and therefore no refreshing overhead is incurred; (3)STT-RAM cache with retention time of 10ms, which is a more aggressive design with better performance/energy gain but a data refreshing technique is needed for correct operations since cache lines that have lifetimes exceeding 10ms are likely to loose data. Thus, we propose simple extensions to the L2 cache design for avoiding any data loss. This include a simple 2-bit counter like the victim cache [] to keep track of the lifetime of all the cache blocks and a small buffer to temporarily store the blocks whose time has exceeded the retention time. We conduct execution-driven analysis of our proposed techniques using the M5 simulator and a suite of PARSEC and SPEC benchmarks. The main contributions of this work are the following:

Detailed characterization of STT-RAM volatile property: We present a detailed device characterization of data retention tunability in STT-RAM Cells providing insight to the underlying principles enabling these tradeoffs. We believe the design in [?] is very aggressive and may not be feasible considering the state-of-the-art in device technology. On the other hand, if the micro second level retention time is feasible, then our proposed architectural solution would be even more beneficial.

An application-driven study to determine retention time: We analyze the time between writes or replacements to a cache line for various multi-threaded and multi-programmed workloads. Our characterization augments the prior body of work that analyzes cache lifetimes mainly in single processor and single program configurations. Based on the L2 cache behavior, we propose to design STT-RAMs with retention time in the range of 10ms.

Architectural solution to handle STT-RAM volatility: We present a simple buffering mechanism to ensure the integrity of programs given the volatile nature of our tuned STT-RAM cells. Experimental results with PARSEC and SPEC benchmarks on a four-core and eight-core multicore platform compared to a base case 1MB SRAM per core and the ideal 4MB SRAM per core indicate that the proposed solution is attractive both from performance and power perspectives. (Summarize the results here)

The rest of this paper is as follows:

2. STT-RAM Design

2.1. Preliminary on STT-RAM

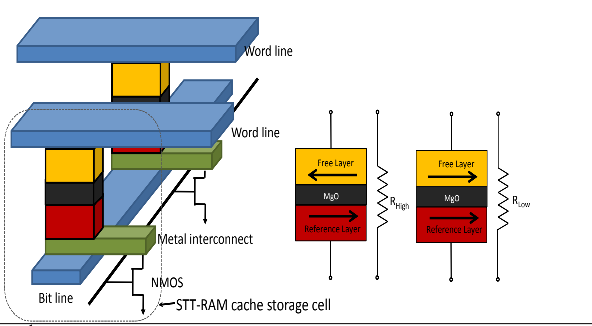


Figure 1. **Demonstration of three switching phases: thermal activation, dynamic reversal and precessional switching.**

STT-RAM uses Magnetic Tunnel Junction (MTJ) as the memory storage and leverages the difference in magnetic directions to represent the memory bit. As shown in Fig. 1, MTJ contains two ferromagnetic layers. One ferromagnetic layer is has fixed magnetization direction and it is called the reference layer, while the other layer has a free magnetization direction that can be changed by passing a write current and it is called the free layer. The relative magnetization direction of two ferromagnetic layers determines the resistance of MTJ. If two ferromagnetic layers have the same directions, the resistance of MTJ is low, indicating a “1” state; if two layers have different directions, the resistance of MTJ is high, indicating a “0” state.

As shown in Fig. 1, when writing “0” state into STT-RAM cells, positive voltage difference is established between SL and BL; when writing “1” state, vice versa. The current amplitude required to reverse the direction of the free ferromagnetic layer is determined by the size and aspect ratio of MTJ and the write pulse duration.

2.2. Write current versus write pulse width trade-off

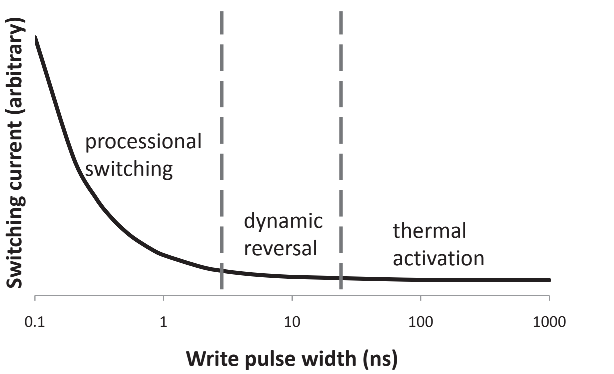


Figure 2. **(a) Structural view of of STT-RAM Cache Cell (b) Anti Space Parallel (High Resistance, Indicating “1” state (c) Parallel (Low Resistance, Indicating “0” state**

The current amplitude required to reverse the direction of the free ferromagnetic layer is determined by a lot of factors such as material property, device geometry and importantly the write pulse duration. Generally, the longer the write pulse is applied, the less the switching current is needed to switch the MTJ state. Three distinct switching modes were identified [7] according to the operating range of switching pulse width τ: thermal activation ($\tau > 20ns$), processional switching ($\tau < 3ns$) and dynamic reversal ($3ns < \tau < 20ns$).

The relationship between switching current density J_c and write pulse width τ was characterized by an analytical model in [15]. The equations are listed as follows,

$$J_{c,TA}(\tau) = J_{c0}\{1 - (\frac{k_B T}{E_b}) ln(\frac{\tau}{\tau_0})\} \quad (1)$$

$$J_{c,PS}(\tau) = J_{c0} + \frac{C}{\tau^\gamma} \quad (2)$$

$$J_{c,DR}(\tau) = \frac{J_{c,TA}(\tau) + J_{c,PS}(\tau) e^{-k(\tau-\tau_c)}}{1 + e^{-k(\tau-\tau_c)}} \quad (3)$$

where $J_{c,TA}$, $J_{c,PS}$, $J_{c,DR}$ are the switching current densities for thermal activation, precessional switching and dynamic reversal respectively. J_{c0} is the critical switching current density, k_B is the Boltzmann constant, T is the temperature, E_b is the thermal barrier, and τ_0 is inverse of the attempt frequency. C, γ, k, and τ_c are fitting constants. Based on the observation from Fig. 2 and analysis of the analytical model, we found very different switching characteristics in the three switching modes. For example, in thermal activation mode, the required switching current increases very slowly even we decrease the write pulse width by orders of magnitude, thus short write pulse width is more favorable in this regime because reducing write pulse can reduce both write latency and energy without much penalty on read latency and energy. While in processional switching, write current goes up rapidly if we further reduce write pulse width, therefore minimum write energy of the MTJ is achieved at some particular write pulse width in this regime. Consequently, this paper will focus on the exploration of write pulse width in processional switching and dynamic reversal to optimize for different design goals.

2.3. STT-RAM Modeling

To simulate the performance of STT-RAM cache, it is important to estimate its cell area first. As mentioned before, each 1T1J STT-RAM cell is composed of one NMOS and one MTJ. The NMOS access device is connected in series with the MTJ. The size of NMOS is constrained by both SET and RESET current, which are inversely proportional to the writing pulse width. In order to estimate the current driving ability of MOSFET devices, a small test circuit using HSPICE with PTM 45nm HP model [18] is simulated. The BL-to-SL current and SL-to-BL current are obtained by assuming typical TMR (120%) and LRS ($3k\Omega$) value [14] and bursting wordline voltage to be 1.5V (the optimal value is extracted from [6]). And we over size the access transistor width to guarantee enough write current provided to MTJ using the methodology discussed in [17]. To achieve high cell density, we model the STT-RAM cell area by referring to DRAM design rules [11]. As a result, the cell size of a STT-RAM cell is calculated as follows,

$$\mathrm{Area_{cell}} = 3\,(W/L+1)(F^2) \tag{4}$$

Table 1. 16-way L2 Cache Simulation Results

			Area (mm^2)	Read Latency (ns)	Write Latency (ns)	Read Energy (nJ)	Write Energy (nJ)	Leakage Power (mW)
$1MB$ SRAM			2.612	1.012	1.012	0.578	0.578	4542
$4MB$ STT-RAM	$t = 10yr$	Leakage Opt.	2.628	2.434	4.919	0.635	0.663	1399
		Latency Opt.	3.003	0.998	10.61	1.035	1.066	2524
	$t = 1s$	Leakage Opt.	2.203	2.044	3.552	0.589	0.616	1388
		Latency Opt.	2.904	0.973	5.571	1.015	1.036	2235
	$t = 100ms$	Leakage Opt.	2.181	1.994	3.432	0.575	0.608	1250
		Latency Opt.	2.902	0.963	3.002	1.008	1.021	2230
	$t = 10ms$	Leakage Opt.	2.167	1.956	3.390	0.571	0.601	1151
		Latency Opt.	2.901	0.959	2.598	1.002	1.028	2227

2.4. Impact of MTJ Retention Time on STT-RAM

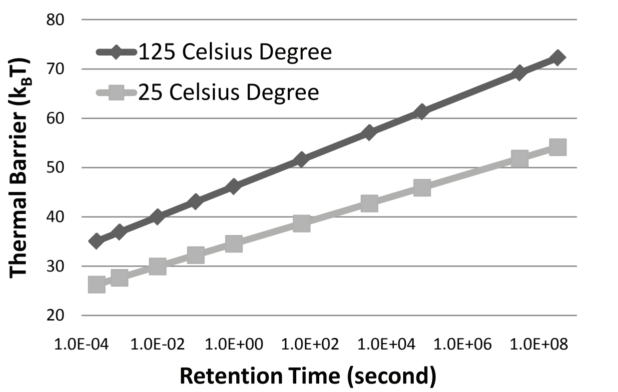


Figure 3. **MTJ thermal stability requirement for different retention time**

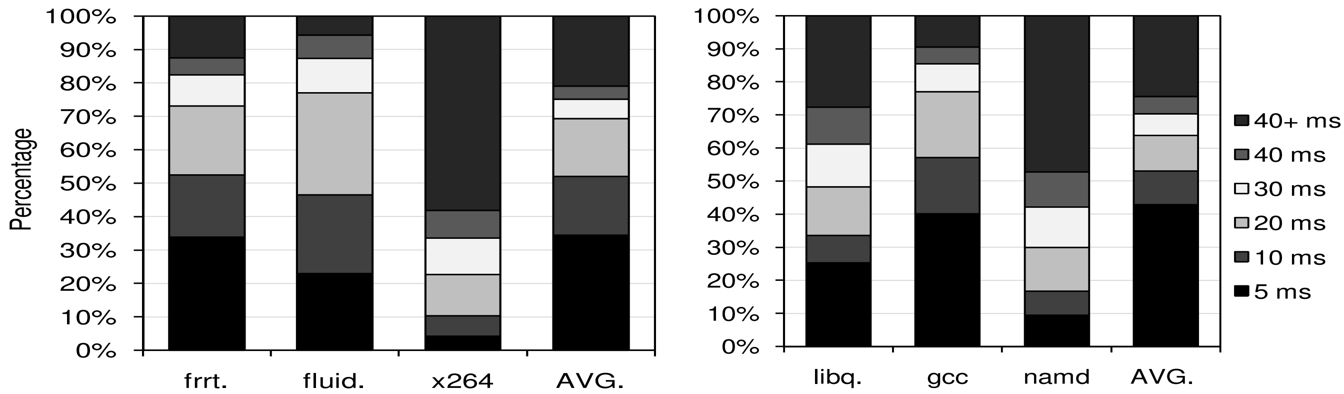
The retention time of a MTJ is largely determined by the thermal stability of the MTJ. The relation between retention time and thermal barrier is captured in Figure 3, which can be modeled as $t = C \times e^{k\Delta}$, where t is the retention time and Δ is the thermal barrier while C and k are fitting constants. Thermal stability of the free layer in an MTJ does not only have impact on retention time of STT-RAM memory cell but also on the write current. It was found in [12] that the switching current of MTJ increases almost linearly with thermal barrier when thermal barrier is $< 70k_BT$, where k_B is the Boltzman constant and T is temperature. Here we combine this observation with the write current versus write time trade-off described in Section 2.2, which essentially means that once the thermal barrier of a MTJ is lowered we are able to achieve faster write speed or/and smaller write current/energy. The most straightforward way to reduce thermal barrier is to tune device geometry such as planar area, thickness of free layer and aspect ratio of the elliptic MTJ.

2.5. STT-RAM Cache Simulation Setup

We simulate SRAM-based caches and STT-RAM-based caches with a tool called NVsim [8], which is a circuit-level performance, energy, and area simulator based on CACTI for emerging non-volatile memories. All the models described in this Section has been integrated in NVsim. The simulation results are listed in Table 1. We can see that the leakage-optimized $4MB$ non-volatile STT-RAM cache has almost the same area with $1MB$ SRAM. This is consistent with previous work [9]. By relaxing retention time of STT-RAM with lower thermal barrier, the leakage-optimized STT-RAM cache can have smaller area, faster write latency and less leaky peripheral circuity. However, as retention time is exponentially related with thermal barrier and thermal barrier is extremely sensitive to process variation and temperature, the benefit of decreasing write latency by relaxing the retention time in the same order (i.e. from $50ms$ to $10ms$) is so small which may be offset by slight variation in device geometry or environment temperature. Moreover, the intrinsic fluctuations in CACTI make it very difficult to observe that small benefit as well. Another point worth mentioning is that the read latency of leakage-optimized $4MB$ STT-RAM cache is significantly larger than $1MB$ SRAM cache because sensing the state of STT-RAM cell takes longer and fast SRAM sensing. Thus, we reduce the array size to improve the latency of STT-RAM cache. As can be seen in Table 1, the latency-optimized STT-RAM cache has noticeable better read and write latency with $14\% - 34\%$ area overhead about 40% dynamic energy overhead compared to leakage-optimized STT-RAM cache with the same retention time.

3. An Application-driven Approach to Determining Retention Time

In order to utilize the volatile STT-RAM as the last level cache in designing an effective cache hierarchy, we need to know what should be the ideal/feasible retention time. Ideally, the STT-RAM write latency should be competitive to SRAM latency and the cache retention time should be high. However, as discussed in the previous section, since the write latency is inversely propositional to the retention time, we need to find a feasible tradeoff based on the STT-RAM device characteristics. Thus, we attempt to decide an ideal retention time by analyzing the characteristics of a last level cache in a multiprogrammed environment. The idea is to understand the distribution of the inter-write interval and thus the average inter-write time to a last level cache and use this time as the STT-RAM retention time. This section describes our application-driven study to estimate the retention time.



(a) PARSEC (b) SPEC 2006

Figure 4. **Distribution of Blocks Showing Different Revival Times**

3.1. Relating Application Characteristics to Retention Time

Application characterization gives the basis for evaluating the impact of retention time on the overall system performance. In order to do this characterization, the first step is to find an ideal time for which the cache block should retain the data. A cache block is only refreshed when the block is written. Thus, we record intervals between two successive writes (refreshes) to the same L2 cache block. We define this interval to be *revival time*. While collecting these results, we ensure that if a block gets invalidated in between two consecutive writes, we do not consider the time in between the invalidation and the next write. Previous work [13] has done similar type of revival time analysis, but for the L1 cache. Figure 4 shows the distribution of L2 cache blocks having different revival time intervals. These results are obtained by running multi-threaded (PARSEC [3]) and multi-programmed (SPEC 2006 [1]) applications on the M5 Simulator [4] that models a 2GHz processor consisting of 4 cores, with 4MB L2 cache. Table 3 contains additional details of the system configuration. Figures 4 (a) and (b) show the results of three PARSEC and SPEC benchmarks along with the averages across 12 PARSEC and 14 SPEC benchmarks, respectively. We observe from the figures that, on an average, approximately 50% of the cache blocks get refreshed within 10 ms, this is in contrast to the microsecond reuse for L1 case [13]. About 20% of blocks remain in the cache for more than 40 ms and rest of the blocks have intermediate revival times. Blocks which stay longer than the retention time in the cache without being refreshed are assumed to be not available, and would affect the application performance the most. This distribution

Table 2. Retention and Write Latencies for STT-RAM L2 Cache

Retention Time	10years	1sec	10ms
Write Latency (Latency Optimized) @2GHz	22 cycles	12 cycles	6 cycles
Write Latency (Leakage Optimized)@2GHz	10 cycles	8 cycles	8 cycles

also gives us the basis on which we can choose the optimal retention time. Reducing the retention time too much will make the cache highly volatile leading to degraded performance, while increasing the retention time would affect the write latency.

3.2. Low Retention STT-RAM Characteristics

Table 2 summarizes the retention time and write latency tradeoffs based on the analysis of Section 3 ffor a 2GHz processor. The results are shown for three different settings and for latency and leakage optimized configurations. The results indicate that there is significant reduction in write latency with reduction in retention time. We want to clarify from device fabrication perspective that these retention times are the most stable designs possible. As we lower the retention times of these STT-RAM cells in the range of *ms*, it becomes much harder to precisely mark a STT-RAM cell with a fixed retention time. For the sake of correctness and preciseness, we discuss these designs only in the paper. Later in the Section 7, it will be clear, that our design assumptions have no affect on the generality of the results.

To analyze the tradeoffs between retention time and overall system performance, let us consider an utopian cache with 10 year retention time having minimum write latency and energy. To bridge the gap between a feasible state-of-the-art design and the utopian cache, we need to reap the benefits of both application characteristics and emerging device technology. From the application side, it is best to choose a retention time which minimizes the number of unrefreshed blocks and from the technology side it is ideal to choose the STT-RAM with minimum write latency and energy. From Table 2, we choose 10 ms as the optimal retention time that balances both the sides.

4. Architecting Volatile STT-RAM

We observe from figure 4 that on an average, approximately 50% blocks will expire after 10 ms, if no action is taken. This expiration of blocks will not only result in additional cache misses but also would

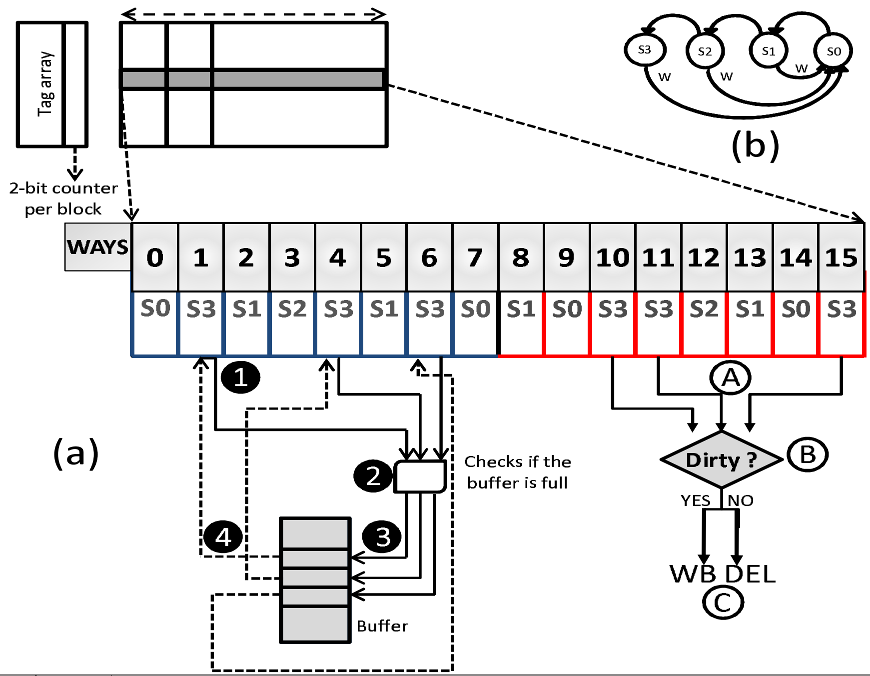


Figure 5. **A modified 16-way L2 cache architecture with a 2-bit counter and a small buffer**

result in data loss, if they were dirty. In this section, we propose our architectural solutions starting with a naive scheme of writing back all the dirty blocks to a more sophisticated scheme, where we minimize the number of refreshes and write backs.

4.1. Volatile STT-RAM

In this naive design, we write back all the unfreshed dirty blocks which become volatile after the retention time. To identify these blocks, we maintain a counter per cache block. To understand the working of the counter, let us consider an *n* bit counter. We assume the time between transitions (T) from one state to another equals to the retention time divided by the number of states, where the number of states is 2^n . A block strats in state S_0 when it is first brought to the cache. After every transition time (T), the counter of each block is incremented. When a block reaches state S_{n-1}, it indicates that it is going to expire in time T. We define this time as the *leftover time* and the block in state S_{n-1} as the diminishing block. Increasing the value of *n*, will decrease the leftover time at the cost of increased overhead of checking the blocks at a finer granularity. For example, if we use a 2-bit counter, the leftover time is 2.5 ms and for a 3-bit it is 1.25ms. A large counter decreases the *leftover time* and allows more time for a block to stay in the cache before applying any refreshing techniques. The down side is the overhead of designing and maintaing a large counter.

Our experimental results show that a 2-bit counter, similar to the one used in [**?**], is sufficient enough to detect the expiration time of the blocks without significantly aafecting the performance. With a 2-bit counter a block can be in one of the four states as shown in figure 5 (b). A block moves from state S_0 to state S_3 in steps on 2.5ms and any time the block is refreshed, it goes back to the initial state. The Counter bits are kept as a part of the SRAM tag array. The overhead of the 2-bit counter is 0.4% for one L2 cache bank.

This scheme has negative impact on the performance for two reasons: (1) There will be a large number of write backs to the main memory for all the dirty blocks at the end of the retention time. (2) The expired block could have been frequently read and losing it will incur additional read misses. We evaluate the results of this design in Section 7.

4.2. Revived STT-RAM Scheme

In order to minimize the write back overhead of the expired blocks at the end of retention time, we propose a different technique, where we use a small buffer to hold a subset of expired blocks at the end of the retention time. We call this design the *revived STT-RAM* scheme. These dirty blocks are thus not written back to the main memory. They are simply written to the temporary buffer and written back to the cache to start another freash cycle. Figure 5 a) shows the schematic diagram of the proposed scheme. The main components of this design are a small buffer and a buffer controller.

Buffer: It is a per bank small storage space with a fixed number of entries made up of low-retention time STT-RAM cells. We use these entries to temporarily store the diminished blocks. We estimate the optimal buffer size later in the section.

Buffer Controller: The buffer controller consists of a $\log_2$N bit buffer overflow detector, where N is the buffer size. The buffer overflow detector is incremented when a diminishing block is copied to one of the buffer slots. The overflow detector is first checked to see the occupancy of the buffer, when a diminishing block is directed to the buffer. The block is copied to one of the empty buffer entries along

Table 3. **Baseline processor, cache, memory and network configuration**

Processor Pipeline	2 GHz processor, 64-entry instruction window, Fetch/Exec/Commit width 8
L1 Cache (SRAM)	32 KB per-core (private) I/D cache, 4-way set associative, 64B block size, write-back, 10 MSHRs
L2 Cache (SRAM or STT-RAM)	1MB (SRAM) or 4MB (STT-RAM) bank, shared, 16-way set associative, 64B block size, 10 MSHRs
Network	Ring network, one router per bank, 3 cycle router and 1 cycle link latency
Main Memory	4GB DRAM, 400 cycle access

with the set and way id, if there is buffer space. If the buffer is full, the dirty blocks are written back to the main memory; otherwise they are invalidated.

Implementation Details: Figure 5 (a) shows a 16-way set associative cache bank with the associated tag array. Counter bits are also placed in tag array. We show the working of our scheme using a 2-bit counter. One of the sets, is shown in detail to clarify the details of the scheme. All the blocks in a set are marked with their current state. Each bank is associated with a buffer and the buffer controller. Let us consider that we are using the buffering scheme for eight MRU slots. Later in this section, we will justify this decision. In Section 7, we will vary the number of slots to see the effects on performance. In Figure 5 (a), ❶ shows that three blocks in first eight MRU slots are diminishing and directed to the buffer. ❷ checks the occupancy of the buffer and if it is not full, each of the diminishing blocks is copied to one of the entries of ❸ along with way and set id. Way and set id are again used by the ❷ to copy back the blocks to the same place in the L2 cache. Ⓐ shows the blocks which are not in MRU slots, but are diminishing. We check these blocks in Ⓑ to see whether they are dirty or not. If they are dirty, we write back those blocks as shown in Ⓒ. If they are not dirty, they are invalidated.

Choosing Optimal Buffer Size and MRU Slots: In order to calculate the optimal MRU slots for buffering, we collected statistics of MRU positions of diminishing blocks by running various PARSEC and SPEC Benchmarks on the M5 Simulator. Figure 6 shows the average cumulative distribution of expired blocks per bank as a function of the number of ways in a set. We observe that the number of diminished blocks becomes stable after first eight MRU ways. The mean number of blocks corresponding to the first eight ways is 2048 (3.16% overhead over per L2 cache bank), which is a good initial choice for the buffer size. In sensitivity analysisi, we will fine tune the buffer size to minimize buffer overflows.

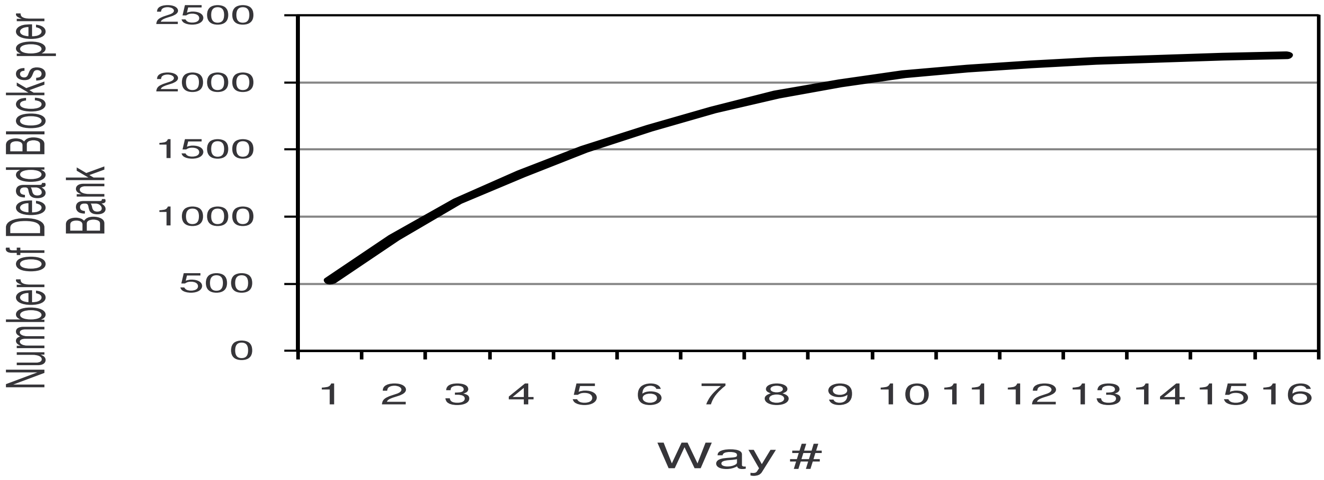


Figure 6. **Cumulative Distribution of Dead Blocks per Bank with number of ways.**

5. Experimental Evaluation

We evaluate our designs using a modified ALPHA M5 Simulator [4] . We operate the M5 Simulator in Full System (FS) mode for PARSEC applications and in the System Emulation (SE) Mode for the SPEC 2K6 Multiprogammed mixes. We model a 2GHz processor with four out of order cores. The memory instructions are modeled through M5 detailed memory hierarchy. We modified the M5 simulator to model L2 cache banks composed of tunable retention time STT-RAM cells. A fixed 400 cycles main memory latency is used for all our simulations. Table 3 details our experimental system configuration.

We report results of 12 multithreaded PARSEC applications and 14 multiprogrammed mixes of 4 SPEC 2K6 applications. Multiprogrammed mixes are chosen randomly from the set of 13 SPEC 2K6 applications. Table **??** shows the properties of PARSEC and SPEC 2K6 applications. We use sim-small input for PARSEC applications and report the results of only Region of Interest (ROI) after warming up the caches for 500M instructions and skipping the initialization and termination phases (except facesim, where we report results for only 2B instructions of ROI). For the SPEC multiprogrammed mixes, we fast forward 1B Instructions, warm up caches for 500M instructions and then report results for 1B instructions.

Design Choices: We report the results for the following designs:

- **S-1MB:** This is our baseline scheme, where all L2 cache banks are composed of SRAM cells. Capacity of each bank is 1MB.

- **S-4MB:** This is our ideal case, where all L2 cache banks are composed of SRAM cells. Capacity of each bank is 4MB and each bank has same read and write latency as that of S-1MB.
- **M-4MB:** This is our baseline scheme for STT-RAM design, where all L2 cache banks are composed of 10 year retention time STT-RAM cells. Capacity of each bank is 4MB.
- **Volatile M-4MB(1sec):** This design is used to evaluate our Volatile STT-RAM Scheme described in **??**, where all L2 cache banks are composed of 1 sec retention time STT-RAM cells.
- **Volatile M-4MB(10ms):** This design is similar to Volatile M-4MB(1sec) except that, now the retention time of STT-RAM cells is 10 ms.
- **Revived M-4MB(10ms):** This design is used to evaluate our Revived STT-RAM Scheme described in **??**, where all L2 cache banks are composed of 10 ms retention time STT-RAM cells. All the results shown with design uses 8 MRU Slots and 2048 Buffer Slots.

Performance Metrics: For the multithreaded PARSEC applications, we assume 4 threads are mapped to our modeled processor with four cores. We report normalized speedup for these applications, which is defined as the improvement over the slowest thread. For the multiprogrammed SPEC applications, we report Instruction throughput and Weighted Speedup. We define instruction throughput (IT) to be sum total of all the number of instructions committed per cycle in the entire CMP (Eq. (5)). The weighted speedup (WS) is defined as the slowdown experienced by each application in a multiprogram mix, compared to its run under the same configuration when no other application is running on the other cores.(Eq.(6)).

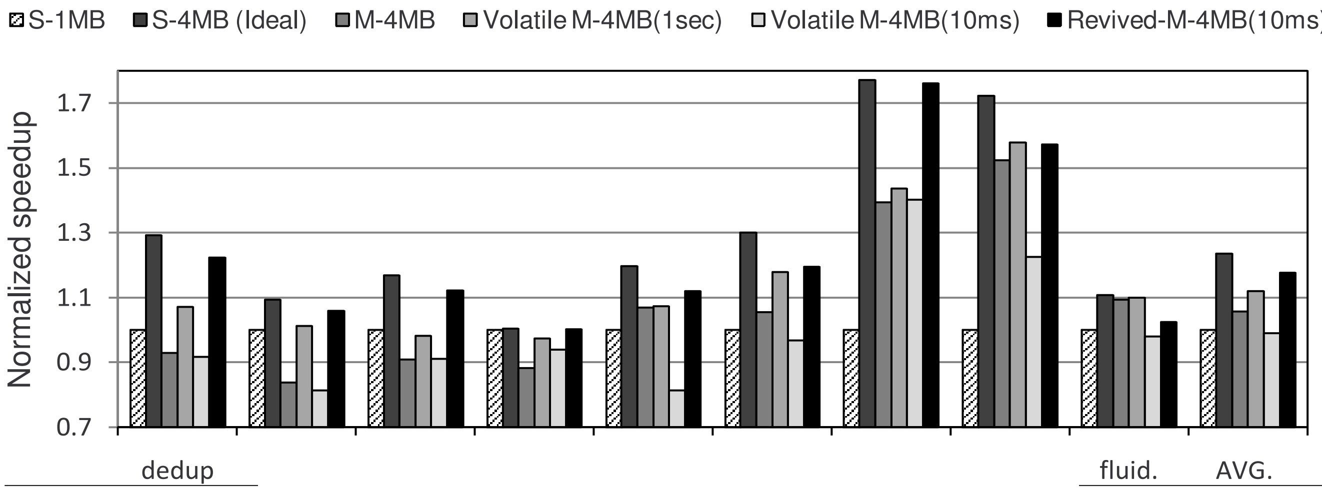
$$Instruction\ throughput = \sum_i IPC_i \quad \textbf{(5)}$$

$$Weighted\ speedup = \sum_i \frac{IPC_i^{shared}}{IPC_i^{alone}} \quad \textbf{(6)}$$

6. Results

In this section, we will discuss all the experimental results associated with our design choices.

Performance: Figure **??** shows speedup improvements of a subset of PARSEC multithreaded applications along with the average (taken across 12 PARSEC applications listed in Table **??**). All speedup



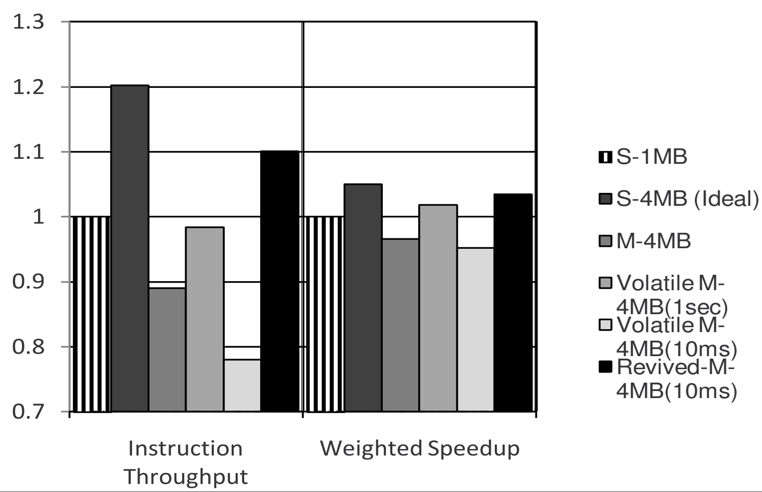


Figure 8. **Normalized Average Instruction Throughput(IT) and Weighted Speedup(WS) for SPEC 2006 multiprogrammed mixes.**

numbers are normalized to S-1MB. The applications to the right of x264 application (including) are showing speedup improvements over S-1MB when M-4MB is used, inspite of high write latency of M-4MB. The reason for this is multi-fold. These applications are benefiting from 4x capacity increase and also because of the presence of write buffers. To see the benefits of write buffer, consider fluidanimate and vips applications. Even though they have high number of writes to the L2 cache, they are benefiting from M-4MB because of the staggered writes which help write buffers in amleirorating increased write latency of M-4MB.The applications to the left of x264 are write intensive applications. In all these applications we see degradation in speedup because of the high write latency. On an average, traditional 10 year STT-MRAM gives 3

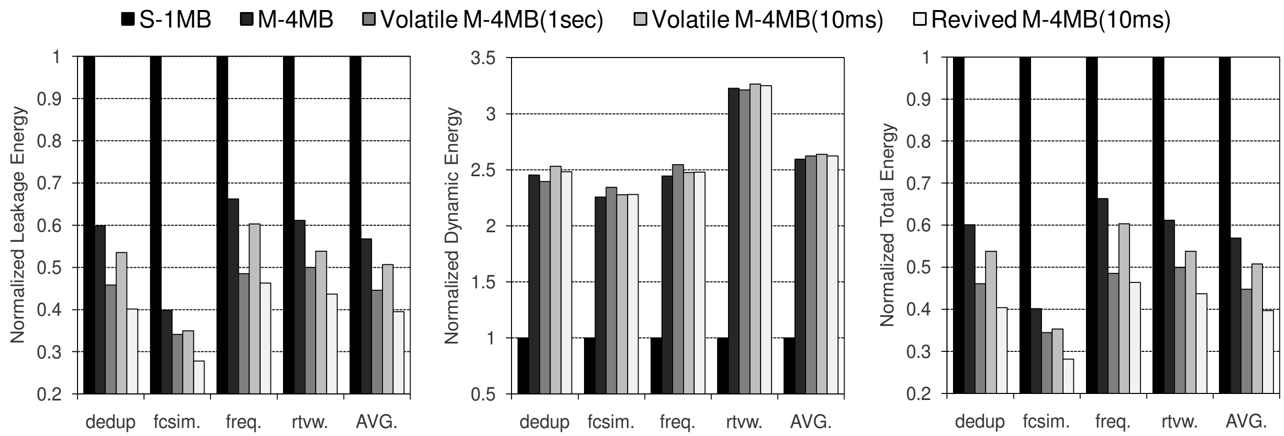
Consider Volatile M-4MB(1sec) design. This design has no refreshing scheme, but since 1 sec time covers almost all the refreshes as discussed in **??**, one can expect benefits of reduced write latency. We observe benefits over M-4MB in all the applications. Volatile M-4MB(10ms) design also doesn't have any refreshing scheme but the retention time of STT-RAM cells used is 10ms. Although this design has very less write latency, no refreshing scheme triggers large number of write backs. For example in the case of vips, there is about 20M-4MB, because of the large number of write backs shown in 10. It is interesting to see the case of ferret. Although there is increase in number of writebacks,

The benchmarks to the right of x264 (including) are showing improvements when S-1MB is replaced by M-4MB. This improvement is on account of increase in capacity. Although, fluidanimate is write intensive, 11x increase does not hamper in getting performance gains of increased capacity. Also the writes in this benchmark are staggered and write buffers are able to ameliorate write latency to a certain extent. We observe that on an average our scheme (Revived M-4MB(10ms)) is giving 11% speedup over traditional non-volatile STT-RAM on account of reduced write latency and sophisticated refresh mechanisms. The naive technique (Volatile-M-4MB(10ms)) performs worse in almost all cases because of the high number of write backs. In ferret it performs better because reduced write latency and comparatively less number of write backs. Volatile-M-4MB(1sec) is on an average gives 7% speedup over M-4MB, because of the reduced write latency latency and no write backs as almost all blocks are refreshed within 1second. Volatile-M-4MB has no architectural overheads, but using our revival technique we are getting 4% more improvement and getting closer to ideal case of S-4MB.

Figure 10 shows number of write backs normalized to M-4MB. We observe that there are very high number of write backs in (Volatile-M-4MB(10ms)) as expected. In ferret,

Choosing correct buffer slots

Number of bits for counter There is no observable difference in performance and energy by increase in the number of bits of the counter.



(a) Leakage Energy (b) Dynamic Energy (c) Total Energy

Figure 9. **Energy of Benchmarks Normalized to that of S-1MB**

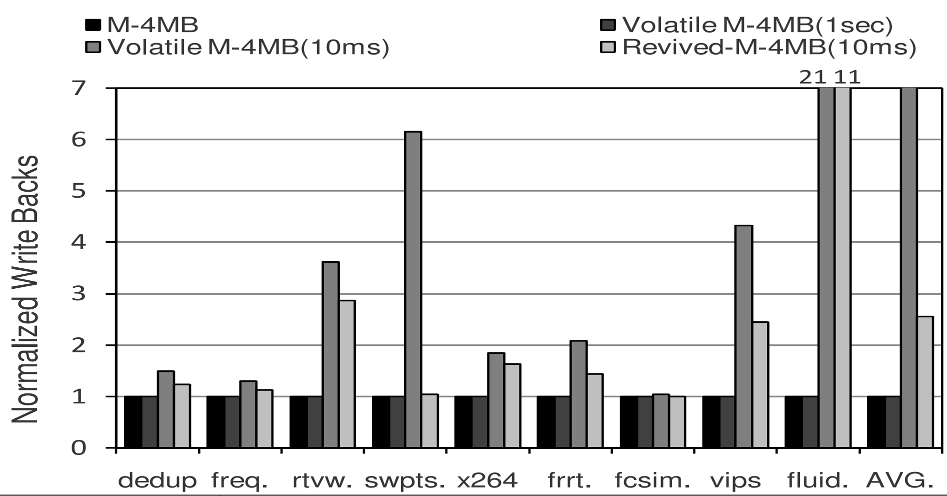


Figure 10. **Number of Write backs normalized to M-4MB**

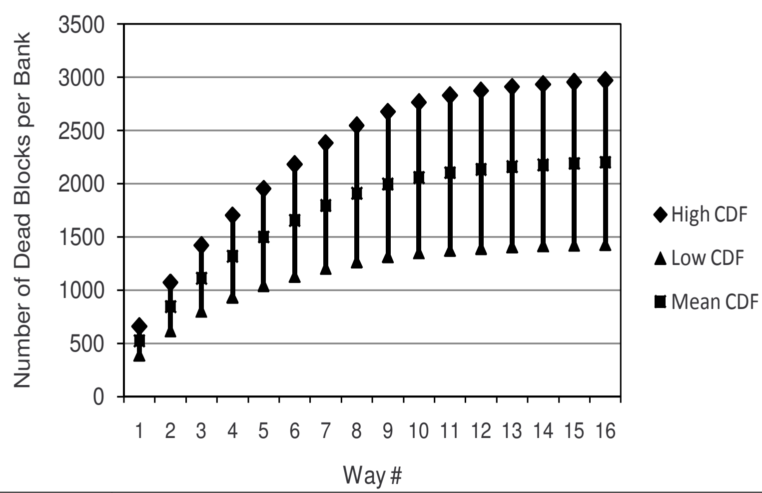


Figure 11. **Confidence Intervals of Dead Blocks for each Way**

6.1. Scaling Effects

7. Prior Work

[ADWAIT, MORE STT-RAM RELATED WORK other than HPCA 2011 paper ESPECIALLY FOR STT-RAM CACHE DESIGN should be put here]

Moreover, in [16] the authors relax retention time of STT-RAM from $10years$ to $56\mu s$ by reducing the planar area of MTJ from $32F^2$ to $10F^2$. However, the scope of this work is limited by addressing practical device parameters and their variabilities. First, the retention time of MTJ is exponentially proportional to the thermal barrier, which makes the retention time of individual STT-RAM device extremely sensitive to any factor that has impact on thermal barrier, particularly device geometry. Thus it's important to take practical values of device geometry such as MTJ planar area and take their process variations into consideration. We get these parameters and corresponding variabilities from fabricated STT-RAM published in recent years [12, 2, 14, 10]. These state-of-the-art MTJs has much smaller baseline planar area that is around $2F^2$. Therefore there is not too much room to reduce retention time by aggressively reduce MTJ planar area. In this paper, we focus on the MTJ with worst-case retention time larger than millisecond and optimize STT-RAM cache correspondingly. Our analysis in this paper reveals the granularities at which a device designer can reliably tune the data retention times.

8. Conclusions

Spin-Transfer Torque RAM (STT-RAM) is a promising candidate for future on-chip cache design due to its high-density, low leakage, and immunity to soft errors. However, it's high write latency and dynamic write energy are the disadvantages compared to SRAM-based cache design. In this paper, we propose to trade off the non-volatility (data retention time) for better write performance/energy in STT-RAM cache design. We study two data-retention relaxation cases, one with data retention time of 1 second, which satisfies the lifetime requirement of typical cache blocks; and the other one with data retention time of 1ms, which is a more aggressive design for better performance/energy gains but a data refreshing mechanism is needed. Our experimental results show that.....

References

[1] Systems Performance Evaluation Cooperation,SPEC Benchmarks, www.spec.org/. 9

[2] P. Amiri, Z. Zeng, P. Upadhyaya, G. Rowlands, H. Zhao, I. Krivorotov, J.-P. Wang, H. Jiang, J. Katine, J. Langer, K. Galatsis, and K. Wang. Low write-energy magnetic tunnel junctions for high-speed spin-transfer-torque MRAM. *IEEE Electron Device Letters*, 32(1):57 –59, 2011. 19

[3] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *Proceedings of the 17th Intl. Conf. on Parallel Architectures and Compilation Techniques*, 2008. 9

[4] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. The m5 simulator: Modeling networked systems. *IEEE Micro*, 26:52–60, 2006. 9, 14

[5] D. Burger, J. R. Goodman, and A. Kägi. Memory bandwidth limitations of future microprocessors. In *ISCA*, 1996. 2

[6] S. Chatterjee, M. Rasquinha, S. Yalamanchili, and S. Mukhopadhyay. A scalable design methodology for energy minimization of STTRAM: a circuit and architecture perspective. *IEEE Transactions on Very Large Scale Integration*, PP(99):1 –9, 2010. 6

[7] Z. Diao, Z. Li, S. Wang, Y. Ding, A. Panchula, E. Chen, L.-C. Wang, and Y. Huai. Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory. *Journal of Physics: Condensed Matter*, 19(16):165209, 2007. 5

[8] X. Dong, N. P. Jouppi, and Y. Xie. PCRAMsim: System-level performance, energy, and area modeling for phase-change RAM. In *Proceedings of the International Conference on Computer-Aided Design*, pages 269–275, 2009. 7

[9] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, et al. Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) as a Universal Memory Replacement. In *Proceedings of the Design Automation Conference*, pages 554–559, 2008. 8

[10] A. Driskill-Smith. Latest Advances in STT-RAM. In *2nd Annual Non-Volatile Memories Workshop*, 2011. 19

[11] F. Fishburn, B. Busch, J. Dale, D. Hwang, et al. A 78nm $6F^2$ DRAM technology for multigigabit densities. In *Proceedings of the Symposium on VLSI Technology*, pages 28–29, 2004. 6

[12] T. Kishi, H. Yoda, T. Kai, T. Nagase, E. Kitagawa, M. Yoshikawa, K. Nishiyama, T. Daibou, M. Nagamine, M. Amano, S. Takahashi, M. Nakayama, N. Shimomura, H. Aikawa, S. Ikegawa, S. Yuasa, K. Yakushiji, H. Kubota, A. Fukushima, M. Oogane, T. Miyazaki, and K. Ando. Lower-current and fast switching of a perpendicular TMR for high speed and high density spin-transfer-torque MRAM. In *Proceedings of International Electron Devices Meeting*, pages 1 –4, 2008. 7, 19

[13] X. Liang, R. Canal, G. yeon Wei, and D. Brooks. Process variation tolerant 3t1d-based cache architectures. In *International Symposium on Microarchitecture*, pages 15–26, 2007. 9

[14] C. Lin, S. Kang, Y. Wang, K. Lee, X. Zhu, W. Chen, X. Li, W. Hsu, Y. Kao, M. Liu, Y. Lin, M. Nowak, N. Yu, and L. Tran. 45nm low power CMOS logic compatible embedded STT MRAM utilizing a reverse-connection 1T/1MTJ cell. In *Proceedings of International Electron Devices Meeting*, pages 57 –59, 2009. 6, 19

[15] A. Raychowdhury, D. Somasekhar, T. Karnik, and V. De. Design space and scalability exploration of 1T-1STT MTJ memory arrays in the presence of variability and disturbances. In *Proceedings of International Electron Devices Meeting*, pages 707–710, 2009. 5

[16] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *Proceedings of the International Symposium on High Performance Computer Architecture*, pages 50–61, 2011. 19

[17] W. Xu, H. Sun, X. Wang, Y. Chen, and T. Zhang. Design of last-level on-chip cache using spin-torque transfer RAM (STT RAM). *IEEE Transactions on Very Large Scale Integration*, 19(3):483 –493, 2011. 6

[18] W. Zhao and Y. Cao. New generation of predictive technology model for sub-45 nm early design exploration. *IEEE Transactions on Electron Devices*, 53(11):2816 –2823, nov. 2006. 6