Architecting Volatile STT-RAM Cache for Enhanced Performance in CMPs

Anonymous HPCA submission

Paper ID 58

Abstract

Spin-Transfer Torque RAM (STT-RAM) is an 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 multi-core architectures. Although the high density, low leakage and high endurance are attractive features of STT-RAM, its long write latency and high dynamic write energy are major obstacles for being competitive with the SRAM. Our study shows that the non-volatility feature with years of data-retention time for STT-RAM technology is not necessary for its usage in on-chip cache, since the refresh times of cache data are usually in μs (for L1 cache) or ms (for L2 cache) range. Consequently, we exploit such observation for designing an efficient L2 cache architecture, and propose to trade off the non-volatility (data-retention time) for better write performance/energy in STT-RAM cache design. The paper addresses several critical design issues such as how we decide a suitable retention time for last level cache, what the relationship between retention time and write latency is, and how we architect the cache hierarchy with a volatile STT-RAM. We study two data-retention time relaxation cases, one with data-retention time of $1sec$, which satisfies the refresh time requirement of typical cache blocks; and the other one with data-retention time of $10ms$, which is a more aggressive design for better performance/energy gains, but a data refreshing mechanism is needed. In the aggressive $10ms$

retention time design, for the rest of the cache blocks that have a higher refresh 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 a four-core architecture with an SRAM-based L1 cache and volatile STT-RAM-based L2 cache indicate that not only we can eliminate the long write latency overhead of an NVM STT-RAM, but can provide on an average* 10-12% *improvement in performance 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 increase with technology scaling, the demand on the on-chip memory would increase significantly, further 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 a 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 static-RAM (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 [23, 20]. 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 [19]. 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

refresh time of the Last Level Cache (LLC) 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 LLC, what the relationship between retention time and write latency is, and how 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, as our analysis will show, 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 ms in contrast to μs for the L1 cache [14]. Consequently, the duration of data retention in STT-RAM is an obvious candidate for device optimization for the cache design. We, therefore, conduct an application-driven study to analyze the inter-write times (refresh times) of the L2 cache blocks to decide a suitable data retention time. Although lifetime analysis of cache lines has been conducted earlier to improve performance and reduce power consumption [12, 14], we revisit this topic with a different intention - correlating STT-RAM data-retention time to cache lifetime. An extensive analysis of PARSEC 2.1 [3] and SPEC2K6 [1] benchmarks using the M5 simulator [4] indicates that the average inter-write times for most of the L2 cache blocks is close to $10ms$, 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 demonstrate that the retention times in the range of ms are more practical compared to the retention time in the μs range as described in prior work [19]. Consequently, using this ms range retention time model, we propose effective architectural techniques to avoid any data loss due to the volatile STT-RAM-based cache hierarchy.

A key challenge in determining a suitable data-retention time for the STT-RAM is to balance the reduced write latency of STT-RAM cells with lower retention time against 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 $1sec$, which is long enough for the inter-write time 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 inter-write times exceeding $10ms$ are likely to lose data. Thus, we propose simple extensions to the L2 cache design for avoiding any data loss. This includes a simple 2-bit counter (similar to one proposed in [12]) to keep track of the inter-write times of all the cache blocks and a small buffer to temporarily store the blocks whose inter-write time has exceeded the retention time. We conduct execution-driven analysis of our proposed techniques using the M5 simulator and a suite of PARSEC 2.1 and SPEC 2K6 benchmarks.

The main **contributions** of this paper and the **differences** from the closest work [19] are 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. The MTJ designs (10-32F^2) used in [19] are not realistic and give an inaccurate picture of the current technology. We feel that there is not much scope to further reduce the baseline area of the state-of-the-art MTJ designs of 2-3F^2 [10] and hence, we focus on other optimization parameters to get optimal operating points.

An application-driven study to determine retention time: Our characterization of the time between writes or replacements to a LLC block augments the prior body of work that analyzes cache lifetimes mainly in single processor and single program configurations. This exhaustive application-level analysis is done with 25 multi-threaded and multi-programmed applications, as compared to only 5 applications in [19], shows that a very aggressive μs level retention time advocated in [19] is unsuitable for the LLC and we propose to be in the range of $10ms$. We feel that analyzing this broad spectrum of applications and architecting the whole cache hierarchy with single retention time STT-RAM cells is a non-trivial task. We do acknowledge the fact that there could be a possibility of designing LLC with STT-RAM banks of varying retention times and moving dying blocks from lowest retention time bank to the highest one, but fabricating such LLC is extremely challenging as it requires different types of STT-RAM cells with different retention times to be part a of cache hierarchy.

Architectural solution to handle STT-RAM volatility: We present a simple buffering mechanism to ensure the integrity of the programs given the volatile nature of our tuned STT-RAM cells. Our refresh scheme is simple, yet very area/power/performance efficient compared to the DRAM style refreshing proposed in [19]. Experimental results with PARSEC 2.1 and SPEC 2K6 benchmarks on a four-core 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. On an average, we find 18% speedup improvement with PARSEC 2.1 benchmarks, 10%/4% instruction throughput/weighted speedup improvement with SPEC 2K6 benchmarks, and an average energy saving of 60%, over 1MB SRAM across our entire application suite.

The rest of this paper is as follows: In Section 2, we discuss the volatile STT-RAM design to parameterize the retention time and write latency behavior. Section 3 presents a retention time estimation study from application perspective. Following this, the design of a volatile STT-RAM-based last level cache architecture is given in Section 4; the experimental platform details in Section 5; results in Section 6, and description of related work in Section 7. The last section provides concluding remarks.

2. STT-RAM Design

In this section, we first discuss STT-RAM preliminaries. This is followed by a detailed discussion of STT-RAM models which we have developed to guide us in our exploration of suitable STT-RAM device for LLC hierarchies. Based on this, we show three stable/feasible STT-RAM cell designs suitable for the LLC hierarchies of multi-core architectures.

2.1. Preliminary on STT-RAM

STT-RAM uses Magnetic Tunnel Junction (MTJ) as the memory storage and leverages the difference in magnetic directions to represent a memory bit ("0"/"1" state). As shown in Figure 1, an MTJ contains two ferromagnetic layers. One ferromagnetic layer has a fixed magnetization direction and it is called the reference layer. The second layer's magnetic direction can be changed by passing a write current, and, thus it is called the free layer. The relative magnetization direction of two ferromagnetic layers

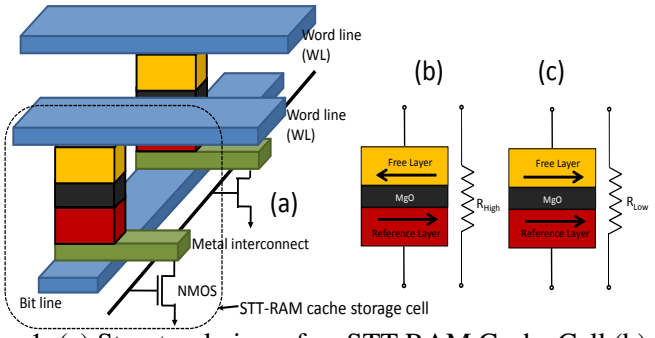


Figure 1. (a) Structural view of an STT-RAM Cache Cell (b) Anti Space Parallel High Resistance, Indicating "0" state (c) Parallel Low Resistance, Indicating "1" state

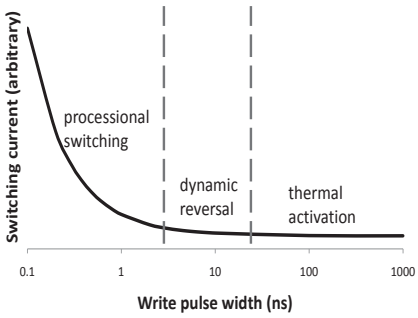


Figure 2. Demonstration of three switching phases: thermal activation, dynamic reversal and precessional switching

determines the resistance of MTJ. If two ferromagnetic layers have different directions, the resistance of MTJ is high, indicating a "0" state; if two layers have the same directions, the resistance of MTJ is low, indicating a "1" state. When writing a "0" state into STT-RAM cells, positive voltage is applied on bitline and when writing a "1" state a negative voltage is applied. The current amplitude required to reverse the direction of the free ferromagnetic layer is determined by the size and the aspect ratio of MTJ, and the write pulse duration [7, 15].

2.2. Write current versus write pulse width trade-off

The current amplitude required to reverse the direction of the free ferromagnetic layer is determined by many factors like material property, device geometry and most importantly the write pulse duration. Generally, the longer the write pulse is applied, the lesser the switching current is needed to switch the MTJ state. Three distinct switching modes were identified in [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 in the three operating ranges was characterized by an analytical model in [18]. The equations are listed as follows:

$$J_{c,TA}(\tau) = J_{c0}\{1 - (\frac{k_B T}{E_b})ln(\frac{\tau}{\tau_0})\} \quad \textbf{(1)} \quad J_{c,PS}(\tau) = J_{c0} + \frac{C}{\tau^\gamma} \quad \textbf{(2)} \quad J_{c,DR}(\tau) = \frac{J_{c,TA}(\tau) + J_{c,PS}(\tau)e^{-k(\tau-\tau_c)}}{1+e^{-k(\tau-\tau_c)}} \quad \textbf{(3)}$$

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 Figure 2 and analysis of the analytical model, we found vastly different switching characteristics in the three switching modes. For example, in the thermal activation mode, the required switching current increases very slowly even if we decrease the write pulse width by orders of magnitude, and thus, short write pulse width is more favorable in this region because reducing write pulse can reduce both write latency and energy without much penalty on read latency and energy. In the processional switching mode, write current goes up rapidly if we further reduce the write pulse width, therefore, minimum write energy of the MTJ is achieved at some particular write pulse width in this region. Based on this analysis, 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

Before simulating the performance characteristics of an STT-RAM cache, we first introduce models to determine the area of a STT-RAM cell. Area of each STT-RAM cell would determine the area of a cache bank composed of these cells and in turn influence the read/write latency of the bank. As shown earlier in Figure 1, each 1T1J STT-RAM cell is composed of an 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 [22] is simulated. The driving current is obtained by assuming typical TMR (120%) and LRS ($3k\Omega$) value [15] and wordline voltage to be 1.5V (the optimal value is extracted from [6]). Further, we oversize the access transistor width to guarantee enough write current is provided to MTJ using the methodology discussed in [21]. 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 given as:

$$\text{Area}_{\text{cell}} = 3\,(W/L + 1)(F^2) \quad \textbf{(4)}$$

where, W and L are the channel width and length of the access NMOS transistor respectively.

2.4. Impact of MTJ Retention Time on STT-RAM

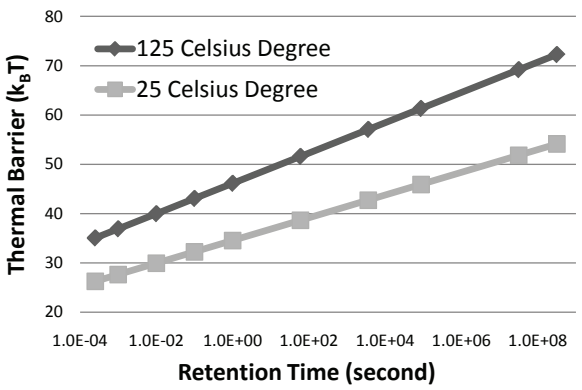


Figure 3. MTJ thermal stability requirement for different retention time

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 shown 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 not only has impact on retention time of STT-RAM memory cell, but also on the write current. It was found in [13] that the switching current of MTJ decreases as thermal barrier is reduced. 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 baseline MTJ in our study is a $2F^2$ in-plane MTJ with a thermal barrier of $72k_BT$, where k_B is the Boltzman constant and T is the temperature. The non-volatile MTJ can hold data more than 10 years under a worst-case temperature of 125 celsius degree. Unlike [19], the optimized $2F^2$ cell does not have much room for only reducing planar area. Hence, reduction in thermal barrier was obtained by decreasing the thickness of free layer and lowering the saturation magnetization. For example, we are able to get two volatile MTJs with reduced thermal barriers of $46k_BT$ and $40k_BT$, which are corresponding for retention times of 1 second and 10 millisecond under 125 celsius degree separately. Raw experimental data were obtained from our device collaborator and we further did curve fitting by using the equations (1)-(3). The results are plotted in Figure 4. Next we are going to show how the volatile MTJs are optimized for both latency and energy. The $10ns$ write pulse width is used as the practical operating point A($10ns$, $114\mu A$) in our baseline MTJ similar to previous work [9]. Simply by fixing the write pulse width at $10ns$, we are able to lower the write currents of the two volatile MTJs which operate at B($10ns$, $73\mu A$) and C($10ns$, $40\mu A$). Then we apply the write current versus write time trade-off on the operating points of the volatile MTJs to reduce the write latency. Particularly, we operate the MTJ with 1sec-retention at point B'($5ns$, $82\mu A$) and the

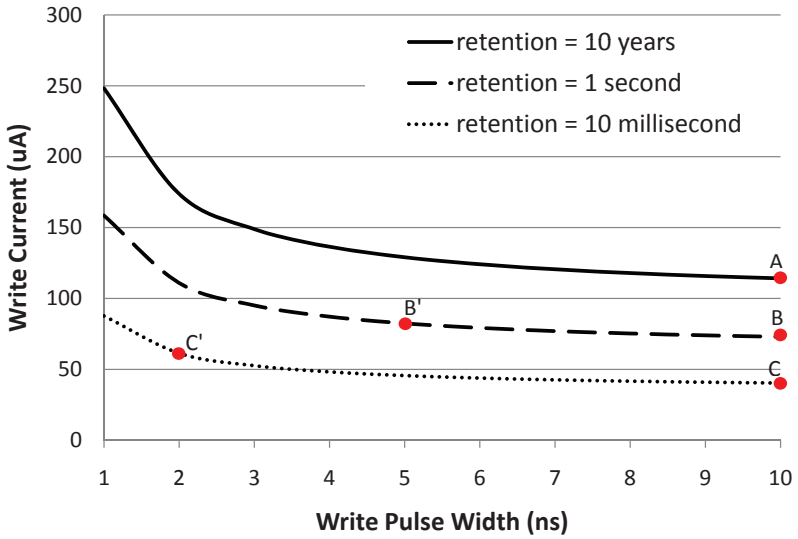


Figure 4. Write current versus write pulse width for three MTJs with 10yr-, 1s, and 10ms- retention at 125 celsius degree

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	$t = 10yr$	3.003	0.998	10.61	1.035	1.066	2524
	$t = 1s$	2.904	0.973	5.571	1.015	1.036	2235
	$t = 10ms$	2.901	0.959	2.598	1.002	1.028	2227

write current is 20% lower than that of baseline A. Moreover, we operate the MTJ with 10ms-retention at C'($2ns$, $61\mu A$) and the write current is 20% lower than that of B'.

2.5. STT-RAM Cache Simulation Setup

Based on the above device level characterizations and analytical models, we simulate SRAM-based caches and STT-RAM-based caches with a tool called NVsim [8]. NVsim which is a circuit-level performance, energy, and area simulator based on CACTI for emerging non-volatile memories. We integrated all the models described in the previous sub-sections in NVsim. Table 1 shows the architectural parameters based on our STT-RAM models. It shows the read, write times and energy numbers of three stable operating points A, B', and C' for MTJs with different retention times. We find that a 4MB NVM STT-RAM cache occupies similar chip area as 1MB SRAM. This is consistent with previous work [9]. For the leakage simulation, we assume there is no power gating techniques for all cache banks. The results show that STT-RAM consume almost half leakage power of SRAM. That is basically because

half of STT-RAM die area is occupied by peripheral circuitry which means half of the chip are leaky.

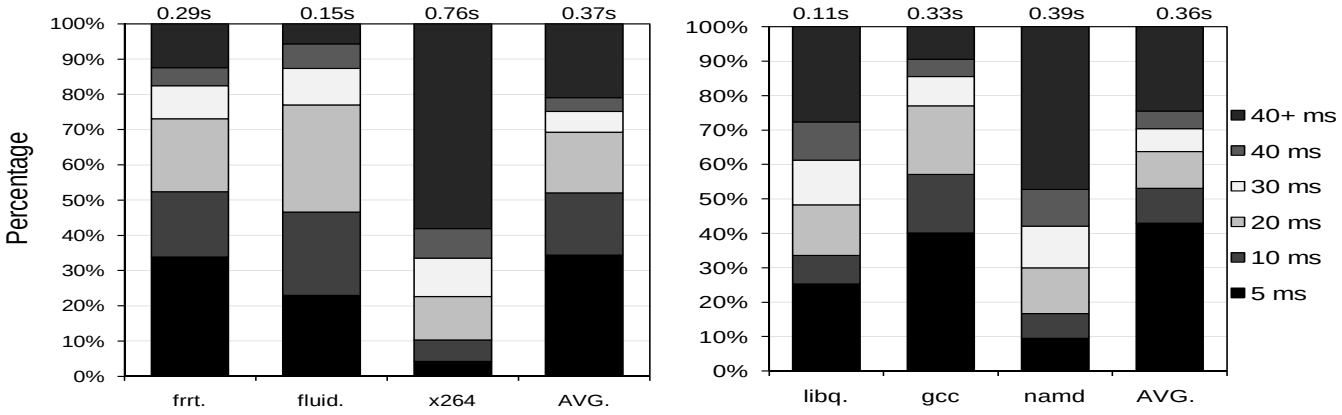
By relaxing retention time of STT-RAM with lower thermal barrier, the STT-RAM cache can have smaller area, lower write latency and less leaky peripheral circuitry. However, since retention time is exponentially related with thermal barrier, and thermal barrier is highly sensitive to process variation and temperature, the benefit of decreasing write latency by relaxing the retention time in the same order of magnitude is so small that it can be easily offset by slight variation in device geometry or environment temperature. For example, the difference of thermal barrier for 10*ms* and 50*ms* retention MTJ is only about 5%. Based on this analysis, we propose to use 10 year, 1*sec* and 10*ms* STT-RAM devices in our cache banks.

3. An Application-driven Approach to Determine Retention Time

In order to leverage the volatile STT-RAM properties at the LLC hierarchy, we need to know what the ideal/feasible retention time should be. 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 proportional to the retention time, we need to find a feasible trade-off based on the STT-RAM device characteristics. Thus, we attempt to decide an ideal retention time by analyzing the retention times of LLC blocks in a multithreaded PARSEC 2.1 [3] and multiprogrammed SPEC2K6 [1] environment. The idea is to understand the distribution of the inter-write intervals to a LLC and use average of these intervals as the STT-RAM retention time. The next sub-section describes our application-driven study to estimate the retention time.

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 investigate the duration 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



(a) PARSEC 2.1 (b) SPEC 2K6

Figure 5. Distribution of blocks showing different revival times. Values on the top of the bar shows maximum revival time for that distribution.

invalidation and the next write. Previous work [14] has done similar type of revival time analysis, but it was for the L1 cache. Figure 5 shows the distribution of L2 cache blocks having different revival time intervals. These results are obtained by running multithreaded(PARSEC 2.1) and multiprogrammed (SPEC 2K6) applications on the M5 Simulator [4] that models a 2GHz processor consisting of 4 cores, with 4MB L2 cache. Table 4 contains additional details of the system configuration. Figures 5 (a) and (b) show the results of three PARSEC 2.1 and SPEC 2K6 benchmarks along with the averages across 12 PARSEC 2.1 and 14 SPEC 2K6 benchmarks, respectively. We observe from the figures that, on an average, approximately 50% of the cache blocks get refreshed within $10ms$, this is in contrast to the microsecond reuse for L1 case [14]. About 20% of blocks remain in the cache for more than $40ms$ 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 expired, and would affect the application performance the most. This distribution 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 negatively 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 2 for a 2GHz processor. The results indicate that there is significant reduction in write latency with reduction

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

Retention Time	10years	1sec	10ms
Write Latency @2GHz	22 cycles	12 cycles	6 cycles

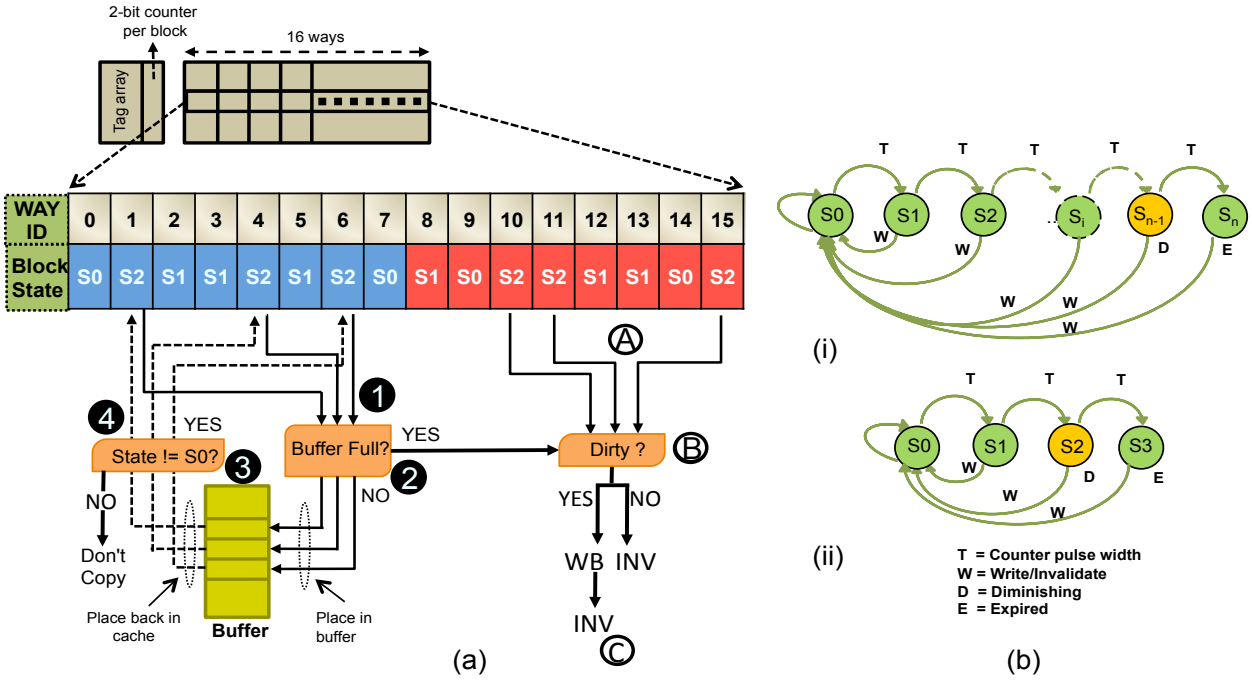


Figure 6. A modified 16-way L2 cache architecture with a 2-bit counter and a small buffer

in retention time. Note that one can possibly lower the retention time further beyond the ms ranges, but it becomes much harder to control the variations and the benefits of performance/energy diminish. For the sake of correctness and preciseness, we discuss these designs only in the paper.

To analyze the tradeoffs between retention time and overall system performance, let us consider an utopian cache with $10year$ 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 $10ms$ as the optimal retention time that balances both the sides.

4. Architecting Volatile STT-RAM

We observe from Figure 5 that on an average, approximately 50% of the blocks will expire after $10ms$, if no action is taken. This expiration of blocks will not only result in additional cache misses, but would also 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 design, we write back all the unrefreshed dirty blocks which become volatile after the retention time. To identify these blocks, we maintain a counter per cache block. Each cache block uses an *n*-bit counter (shown in Figure 6(b)(i)). 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 starts 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 $2.5ms$ and for a 3-bit counter it is $1.25ms$. A large bit counter decreases the leftover time and allows more time for a block to stay in the cache before applying any refreshing techniques. This gives the block more opportunity to stay in the cache. The down side is the overhead of designing and maintaining a large bit counter.

Our experimental results show that a 2-bit counter, similar to the one used in [12], suffices to detect the expiration time of the blocks without significantly affecting the performance. With a 2-bit counter, a block can be in one of the four states as shown in Figure 6 (b)(ii). A block moves from state S_0 to state S_3 in steps of $2.5ms$ and at any time the block is written/invalidated, 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 a 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 6.

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 diminishing blocks. 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 fresh cycle. Figure 6(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 diminishing blocks. In order to calculate the optimal MRU slots for buffering, we collected statistics of MRU positions of diminishing blocks by running various PARSEC 2.1 and SPEC 2K6 Benchmarks on the M5 Simulator. Figure 7 shows the average cumulative distribution of diminishing blocks per bank as a function of the number of ways in a set. We observe that the number of diminishing blocks becomes stable after first eight MRU ways. The mean number of blocks corresponding to the first eight ways is 1900 (3% overhead over per L2 cache bank), which is a good initial choice for the buffer size. In sensitivity analysis, we will fine tune the buffer size to minimize buffer overflows.

Buffer Controller: The buffer controller consists of a $\log_2$N bit buffer overflow detector, where N is the buffer size. The overflow detector is first checked to see the occupancy of the buffer, when a diminishing block is directed to the buffer and the buffer overflow detector is incremented. The block is copied to one of the empty buffer entries along 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 6 (a) shows a 16-way set associative cache bank with the associated 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

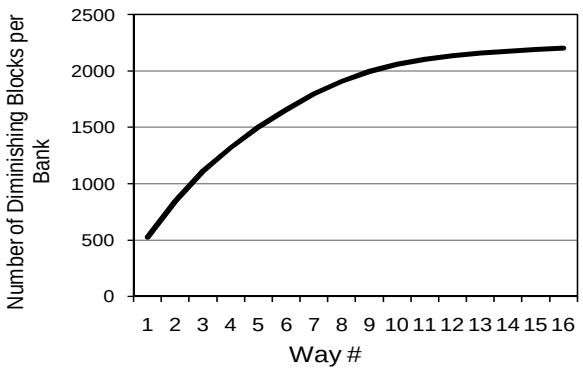


Figure 7. Cumulative distribution of diminishing blocks per bank with number of ways.

associated with a buffer and the buffer controller. Let us consider that the buffering scheme incorporates eight MRU slots, based on the cumulative distribution of MRU slots shown in Figure 7. In Section 6, we will vary the number of slots to see the effects on performance. In Figure 6(a), ❶ shows that three blocks in first eight MRU slots are diminishing and directed to the buffer. Please note that we apply our scheme only to the diminshing (to-be-expired) blocks, which gives enough time for the scheme to be completed before actual data loss happens. 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 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 Ⓒ and then invalidate. If they are not dirty, they are just invalidated. During this whole refresh process, if a read request for the cache block comes, it will be sucessfully completed as the data is still valid during that time. If a write/invalidate request comes, the cache blocks goes back to its original state. To make sure we don't copy the stale data from the buffer, we do perform state check before copying back as shown in ❹. Moreover, our implementation assumes the worst-case temperature of (125C) and hence there is no possibility of early expiration of block because of sudden reduction in retention time of STT-RAM cells.

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 2.1 applications and in the System Emulation (SE) Mode for

Table 3. Application characteristics: *Read%*:Denotes the percentage of reads to the L2 cache out of the total L2 accesses, *Write%*:Denotes the percentage of writes to the L2 cache out of the total L2 accesses, *Intensity*: Read/Write intensive based on read%/write%.

#	PARSEC 2.1	Read%	Write%	Intensity	#	SPEC 2K6	Read%	Write%	Intensity
1	blackscholes	91.9	8.1	Read	13	bzip2	86.2	13.8	Read
2	bodytrack	92.2	7.8	Read	14	gcc	99.4	0.6	Read
3	dedup	73.8	26.2	Write	15	mcf	94.5	5.5	Read
4	facesim (fcsim.)	78.7	21.3	Read	16	leslie3d	70.7	29.3	Write
5	ferret (frrt.)	46.2	53.8	Write	17	namd	92.7	7.3	Read
6	fluidanimate (fluid.)	82.4	17.6	Read	18	soplex	59.6	40.4	Write
7	freqmine (freq.)	72.1	27.9	Write	19	hmmer	63.6	36.4	Write
8	rtview (rtvw.)	64.1	35.9	Write	20	sjeng	76.6	23.4	Write
9	streamcluster	98.4	1.6	Read	21	libquantum(libq.)	100.0	0.0	Read
10	swaptions (swpts.)	49.9	50.1	Write	22	lbm	15.7	84.3	Write
11	vips	75.0	25.0	Write	23	GemsFDTD	99.2	0.8	Read
12	x264	95.5	4.5	Read	24	omnetpp	97.7	2.3	Read
					25	h264ref	57.8	42.2	Write

Table 4. 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

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 4 details our experimental system configuration.

We report results of 12 multithreaded PARSEC 2.1 applications and 14 multiprogrammed mixes of 4 SPEC 2K6 applications. Multiprogrammed mixes are chosen randomly from the set of 13 SPEC 2K6 applications. Table 3 shows the properties of PARSEC 2.1 and SPEC 2K6 applications. We use sim-small input for PARSEC 2.1 applications and report the results of only Region of Interest,(ROI)(except facesim, where we report results for only 2B instructions of ROI) after warming up the caches for 500M instructions and skipping the initialization and termination phases. 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 L2 cache configurations:

- **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 an ideal case, where all L2 cache banks are composed of SRAM cells. Capacity of each bank is 4MB and each bank has the same read and write latency as that of S-1MB. This case is analyzed to see the potential benefit of having a 4x improvement in cache capacity at 4x area density, while still having read/write latencies comparable to SRAM. This ideal but hypothetical design has the capacity, area and leakage properties of a STT-RAM but the read/write latencies of an SRAM cache.

• **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 and each bank occupies the same area as that of an SRAM bank.

• **Volatile M-4MB(1sec):** This design is used to evaluate our Volatile STT-RAM Scheme described in Section 4, where all L2 cache banks are composed of 1 sec retention time STT-RAM cells.

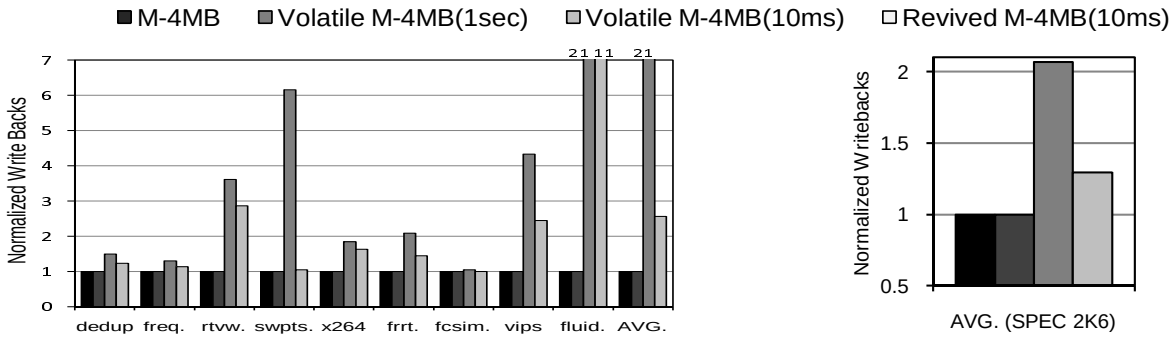
• **Volatile M-4MB(10ms):** This design is similar to Volatile M-4MB(1sec) except that the retention time of STT-RAM cells is 10 ms.

• **Revived M-4MB(10ms):** This design is used to evaluate our Revived STT-RAM Scheme described in Section 4, where all L2 cache banks are composed of 10 ms retention time STT-RAM cells. All the results are for the design with 8 MRU Slots and 1900 Buffer Slots.

All 4MB STT-RAM banks occupy the same area as that of a 1MB SRAM bank (because STT-RAM cells are 4x denser), but have varying write latencies due to varying retention times.

Evaluation Metrics: For the multithreaded PARSEC 2.1 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 the sum 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)). For analyzing the energy behavior, we measure the leakage energy, dynamic energy and total energy for all designs.

HPCA #58



(a) PARSEC 2.1 Applications (b) Average across Multiprogrammed Mixes.

Figure 8. Number of Write backs normalized to M-4MB.

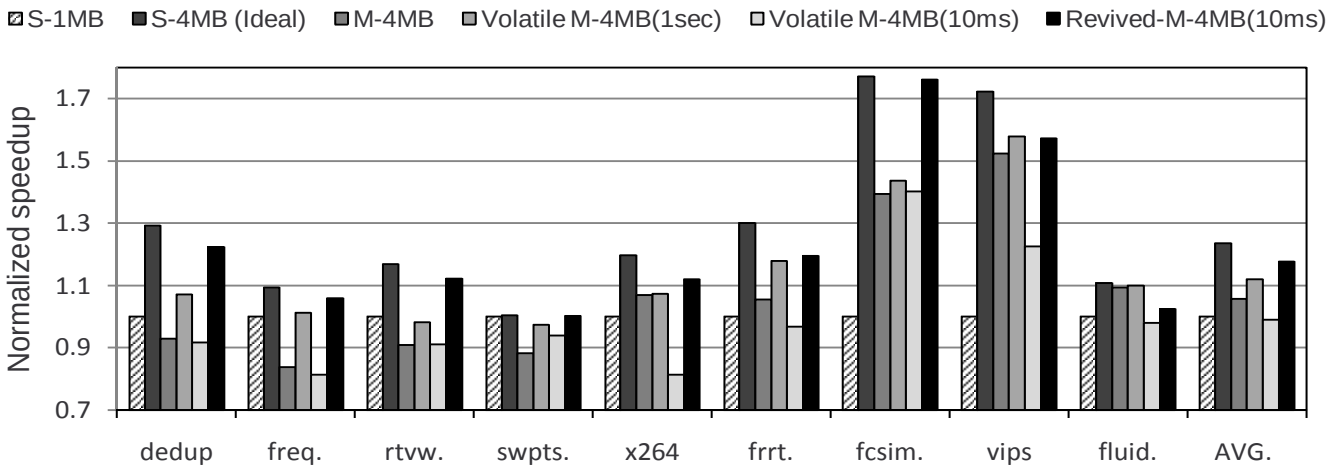


Figure 9. Normalized speedup for PARSEC 2.1 applications.

$$Instruction\ throughput = \sum_i IPC_i \quad \textbf{(5)} \qquad Weighted\ speedup = \sum_i \frac{IPC_i^{shared}}{IPC_i^{alone}} \quad \textbf{(6)}$$

6. Analysis of Results

In this section, we provide a comparative analysis of the performance and energy results of the six designs. We also discuss the sensitivity of several architectural parameters.

6.1. Performance Comparison

Figure 9 shows speedup results with multithreaded PARSEC 2.1 applications along with the average improvements. Only 9 applications are shown to reduce clutter in the plots, however, the average numbers are computed across the entire suite. All speedup numbers are normalized to S-1MB.

Speedup when replacing a SRAM with an ideal STT-RAM cache: When aggregated across the entire PARSEC suite, we find that this hypothetical design has an average speedup of 23% over the SRAM cache. This is the maximum performance that any scheme can provide.

Speedup when replacing a SRAM with 4MB, $10year$ retention time STT-RAM: We find that, for the M-4MB design, all applications to the right of $x264$ (including $x264$) exhibit speedup improvements over S-1MB. Most of these applications are read intensive applications (details in Table 3) and thus, they benefit from not only the 4x capacity increase of STT-RAM, but also from the presence of a write buffer for the L2 cache. On an average, we find 6% improvement in speedup for these applications. Although $ferret$ and $vips$ are write-intensive applications, they benefit with a 4x improvement in capacity when going from a S-1MB to a M-4MB. This is because, the write request to L2 cache banks in these two applications are staggered in time. Thus, a 10-entry write buffer proves to be sufficient to hide the long write latencies of a STT-RAM bank.

All applications to the left of $x264$ are write-intensive and have bursty requests arriving at L2 cache banks. For these applications, we observe significant degradation in speedup (on an average 11% degradation) because of the high write latency of STT-RAM. Overall, when averaged across the entire PARSEC suite, a traditional $10year$ STT-RAM gives a minimal 5% speedup improvement over S-1MB. However, a $10year$ STT-RAM cache organization has 14% lower performance when compared to the ideal design S-4MB and with write-intensive applications this difference is 21%. It is this gap that our proposal seeks to bridge by tuning the retention time.

Speedup when replacing a SRAM with 4MB, $1sec$ retention time STT-RAM: With such a STT-RAM cache bank, no refreshing schemes are employed. As shown earlier, almost all blocks get refreshed within a $1sec$ time interval. Reducing the retention time from $10year$ to $1sec$ reduced the write latency of a cache bank by 10 cycles (from 22 cycles to 12 cycles). This leads to significant speedup improvements over a $10year$ retention time STT-RAM cache organization. On an average, this reduction in 10 cycles lead to 6% performance improvement (14% for write intensive applications). However, this design is still 9% (11% for write intensive applications) lower in performance than the ideal case. Further, when compared to the baseline S-1MB SRAM design, most of the write intensive and bursty application have

no speedup gain compared to Volatile-M-4MB(1sec).

Speedup when replacing a SRAM with 4MB, $10ms$ retention time STT-RAM without refreshing: This volatile M-4MB(10ms) design also does not have any refreshing scheme, but the retention time of STT-RAM cells used is $10ms$. Using this STT-RAM device, after $10ms$ the STT-RAM device will lose its data and hence to keep the data integrity forces a large number of write-backs from the last-level cache to the main-memory before the actual expiration happens. Figure 8 shows number of write backs of all the designs normalized to M-4MB. We observe that the 4MB, $10ms$ retention time STT-RAM design, on an average, has 21% more write backs than the traditional STT-RAM design. This leads to significant performance degradation across most applications when compared to simply using a $10year$ retention time STT-RAM cache (8% performance degradation on average). For instance, in vips, there is about 20% speedup degradation over M-4MB. It is interesting to see the case of $swaptions$, where there is a slight improvement in speedup over M-4MB, although there is an increase in the number of write backs. The reason for this improvement is due to the fact that the majority of blocks that are not refreshed within $10ms$ interval, are not accessed in future as well leading to a low number of read misses. This helps in reaping benefits from the reduced write latency.

Speedup when replacing a SRAM with 4MB, $10ms$ retention time STT-RAM with refreshing: This is our proposed scheme (annotated as Revived-M-4MB(10ms) in the figures), which incorporates refreshing of dirty blocks beyond the $10ms$ retention time. Use of this scheme significantly improves performance when compared to all realistic design scenarios evaluated. On an average, the proposed revived scheme is better than the conventional SRAM design (S-1MB) by 18%, traditional $10year$ STT-RAM by 15% and over Volatile M-4MB(1sec) by 4.5%. The write latency of this STT-RAM cache bank is 6 cycles and when compared to a $1sec$ retention time STT-RAM, the difference of 6 cycles reduction in L2 cache bank access time helps in improving performance. This performance improvement is in spite of the increase in the number of write-backs (which increase by over 2x) compared to an SRAM cache. $Fluidanimate$ is the only application we found that does not show a gain in performance with this type of STT-RAM device. With $fluidanimate$ application, all the dirty blocks are almost equally distributed among all the ways, and hence, our scheme of considering only the first eight MRU slots is

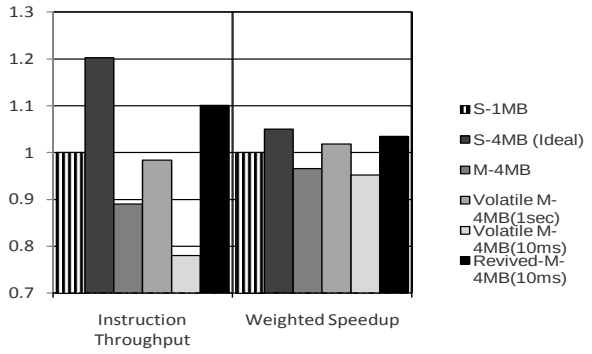


Figure 10. Normalized Average Instruction Throughput(IT) and Weighted Speedup(WS) for SPEC 2K6 multiprogrammed mixes.

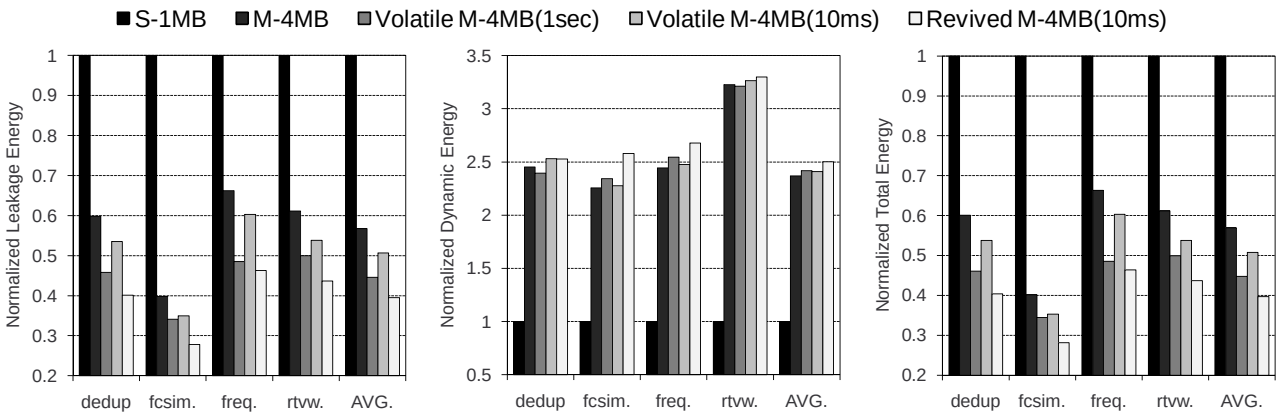
insufficient in reducing many writebacks (as shown in Figure 8, with $fluidanimate$, the write-backs become 11x with this volatile STT-RAM).

The Revived-M-4MB(10ms) scheme is closest to the ideal S-4M case and is within 4% of it, showing the benefits of our scheme in making the STT-RAM device a choice for universal memory.

Analysis with SPEC 2K6: In general, the observations with SPEC benchmarks are consistent with those made with PARSEC 2.1 applications. Figure 10 shows instruction throughput and weighted speedup with the SPEC 2K6 multiprogrammed mixes. Simply replacing a SRAM bank with 4MB STT-RAM would lead to 11% degradation in instruction throughput, and 4% degradation in weighted speedup. However, employing our refreshing scheme on a $10ms$ volatile STT-RAM can lead to an average 22%, 11% and 10% improvement over M-4MB, Volatile M-4MB(1sec) and S-1MB,respectively. With a very write intensive mix ($bzip2$, gcc, lbm and $hmmer$) this improvement can be as high as 35% over the baseline SRAM design. Figure 8(b) depicts the normalized write backs for the SPEC 2.1 benchmarks.

Weighted speedup also follows a similar trend as instruction throughput and we find that our proposed refreshing scheme on a $10ms$ retention time volatile STT-RAM shows the best results. Overall, our proposed design is within 9% (2%) of the ideal device in terms of instruction throughput (weighted speedup).

HPCA #58



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

Figure 11. Energy of applications normalized to that of S-1MB.

6.2. Energy Usage Comparison

Figure 11 shows normalized cache leakage, total of dynamic read and write energy, and total energy for a subset of PARSEC applications. While computing the energy numbers, we take into account the overheads of our proposed cache block refreshing (revived) schemes. We observe that on an average, there is 44% improvement in total energy going from S-1MB to M-4MB designs. This improvement is mainly because of the drastic reduction in leakage energy (43%). In general, all volatile STT-RAMs reduce the energy envelope of the LLC. With $1sec$ volatile STT-RAM, total energy is reduced because of reduction in leakage energy and also nominal performance improvement. However, when comparing Volatile M-4MB(10ms) with Volatile M-4MB(1sec), because of larger number of write-backs with $10ms$ retention time STT-RAMs, the performance degrades and thus, leakage energy increases.

With our proposed cache block refreshment scheme, although there is an increase in the dynamic energy on account of additional back and forth writes to the buffer and cache lines, we consistently observe improvement in total energy (due to both reduction in leakage power and improvement in performance). On an average, we find 11% energy benefits of using revived M-4MB design over Volatile-1sec and 30% improvement over the baseline STT-RAM design. We observe similar benefits with SPEC 2K6 multiprogrammed applications, but for the sake of brevity we are not showing them.

In conclusion, we find that our proposed revived scheme is significantly better in terms of performance and energy when compared to a traditional non-volatile STT-RAM and even a volatile STT-RAM with

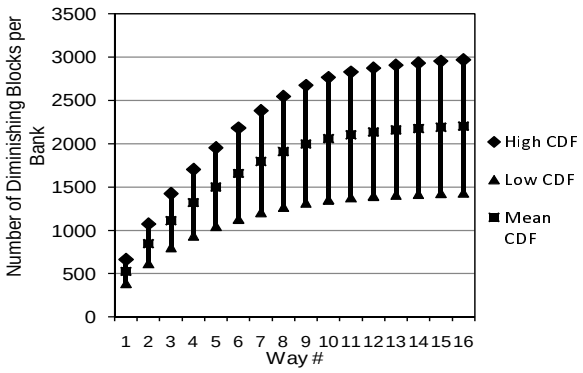


Figure 12. 95% Confidence Intervals of diminishing blocks for each way

$1sec$ retention time. Further, the proposed $10ms$ retention time STT-RAM with revival schemes, is very close to the ideal LLC architecture in performance and significantly better in energy conservation.

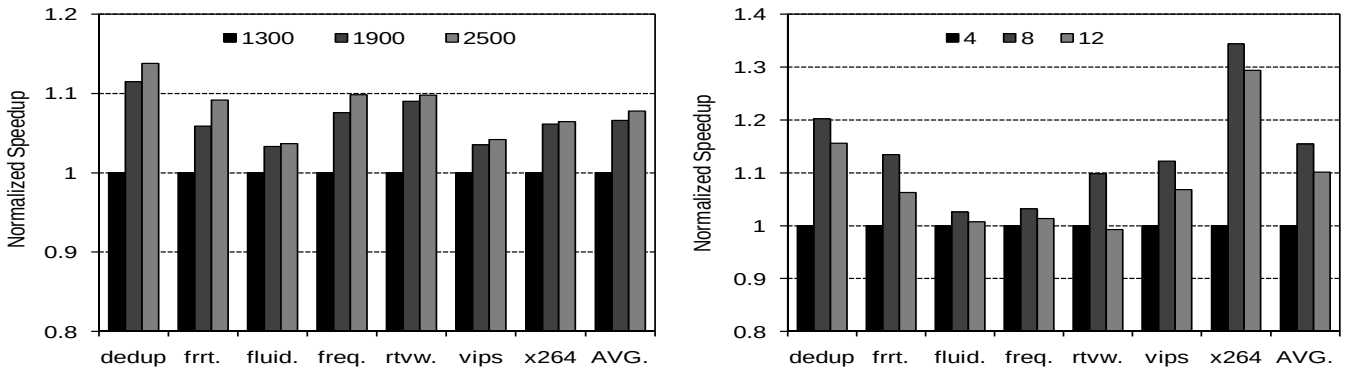
6.3. Sensitivity Analysis

In this subsection, we study the sensitivity of various architectural parameters we chose in proposing the cache revival architecture.

Sensitivity to number of buffer entries: The number of buffer entries can affect the performance of the Revived-M-4MB scheme in two ways: increasing the buffer size will accommodate more diminishing blocks at a particular instance and decreasing the buffer size will leads to more buffer overflows. With increasing the buffer size, leading to fewer buffer overflows, the reduction comes at a cost of increase in buffer area and consequent revival overheads. Decreasing the buffer size, eventually leads to additional write backs (discussed in Section 4).

To find the optimal buffer size, we calculate the 95% Confidence Intervals (CIs) for the cumulative distribution of diminishing blocks per bank. This is shown in Figure 12. We observe that, for the first 8 MRU slots, the mean value of the buffer entries is 1900 blocks, which corresponds to a 3% area overhead per L2 cache bank. Upper limit of the 95% CI corresponds to 2500 blocks, which represents 4% area overhead per L2 cache bank. The lower limit of 95% CI corresponds to 1300 blocks (2% area overhead).

Figure 13 shows the normalized speedup to 1300 blocks, with a subset of PARSEC applications by varying the number of buffer entries. On an average, varying the number of buffer entries from 1900 to 2500,results in only 1% speedup improvement. Hence, in all our results, we used 1900 buffer entries



(a) Buffer Entries (b) MRU Slots.

Figure 13. Showing effects on speedup by varying number of Buffer Entries and MRU Slots

(resulting in a 3% area overhead)with the best possible performance per area-overhead.

Sensitivity to number of MRU slots: Choosing the optimal number of MRU slots to buffer was discussed briefly in Section 4. Figure 7 shows the mean cumulative distribution of diminishing blocks per way across all PARSEC benchmarks. We find that after 8 MRU slots, the number of diminishing blocks saturates, which suggests that the optimal number of MRU slots is 8. Figure 13 shows speedup of a subset of PARSEC applications along with the average across 12 PARSEC applications, with varying number of MRU slots. Buffer size is kept constant at 1900 per bank. We see degradation in performance when we decrease slots from 8 to 4, since buffering 8 MRU slots would have covered more frequently used blocks and hence, reducing write backs of useful blocks. We also see degradation in performance by increasing the number of slots from 8 to 12. This is because, with a constant buffer size, 12 MRU slots increase the probability of buffer overflows, which increases the write backs leading to performance degradation.

Sensitivity to number of bits of the counter: As discussed in Section 4, increasing the number of bits of the counter decreases the left over time at the cost of incrementing counters at a finer granularity. Our experiments showed that there is no observable difference in performance and energy by increasing/decreasing the number of bits of the counter.

7. Prior Work

This section summarizes the circuit and architectural techniques proposed for enhancing the STT-RAM write performance.

The work that is most closely related to ours is [19]. Here, 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 $10F^2$. The scope of their 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 is 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 [13, 2, 15, 10]. These state-of-the-art MTJs have much smaller baseline planar area that is around 2-3F^2. Therefore, there is not too much room to reduce the retention time by aggressively reducing MTJ planar area. Thus, in this paper, we focus on the MTJ with worst-case retention time larger than ms and optimize STT-RAM cache correspondingly. Further, analysis of retention times of LLC blocks of actual applications show that the retention times are in the order of ms, thus making our proposal much more amenable to implement. Hence, compared to [19], our scheme will provide significant performance improvement.

Apart from this recent work, few other prior works have also proposed architectural and circuit level solutions to handle this long write latency problem in STT-RAMs. Architectural techniques such as early write termination [23], hybrid SRAM/STT-RAM architecture [20, 17] and read-preemptive write-buffer designs have been shown to mitigate write latency/energy. The circuit level techniques such as eliminating redundant bit-writes [23] and data inverting technique [20] have also been shown to be effective in hiding the long write latency. In contrast to all these prior works that attempt to *hide* the write latency, our scheme investigates techniques to *actually* reduce the write latency of STT-RAM banks and make their write latency comparable to SRAM banks. When compared to Zhou et al.'s work [23] that require additional gates for detection and termination of writes inside *each STT-RAM sub-bank*, our

techniques are simpler to implement since our proposal works at a much coarser granularity.

Sun et al. [20] showed that write buffers can be helpful in hiding the long write latencies of STT-RAM banks. Our analysis shows that, if an application is bursty, write-buffers fail to hide this latency and are rendered in-effective. Out of 25 applications, we found 12 applications to be write intensive and bursty and hence, write-buffering is ineffective for these applications. Moreover, all our results are conservative since we have already assumed a 10-entry (as used in [20]) write-buffer at every STT-RAM bank and our results would be significantly better without the presence of write-buffers.

In a recent work [16], the authors have proposed a network level solution to hide the write latency of STT-RAM banks. This solution requires complex busy/idle bank detection followed by prioritization mechanisms in the network. On a qualitative basis, the network level solution to hide write latency in [16] was shown as the most promising technique compared to any other techniques. The application level performance improvement with this scheme was about 2-4% higher compared to the write buffering technique of Sun et al. [20]. Contrasting this to our work here, our scheme provides about 15%/4%(PARSEC IPC/SPEC weighted-speedup) improvement over 10yr traditional STT-RAM, on top of the write buffering scheme, thereby making it more attractive compared to [16]. Overall, we believe that no prior work makes a case for tuning the retention time of STT-RAM banks that is based on profiling retention duration of last-level cache blocks of applications, which our proposal does.

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, its 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. In this context, we conduct an application-driven study to characterize the life time of LLC with the intention of using this time as the ideal retention time for the STT-RAM. Execution-driven experiments with several PARSEC and SPEC benchmarks indicate that at least 50% of the cache blocks are updated in $10ms$ and thus, chose $10ms$ as an optimal retention time by analyzing the STT-

RAM retention time and write time trade-offs. We investigate two design alternatives for avoiding the data loss due to the volatile nature of the STT-RAM. The first approach writes back all the dirty blocks in the cache at the end of the retention time and the second approach uses a buffering scheme to refresh the cache blocks that are not refreshed during the retention time.

We analyze three different scenarios for designing the L2 cache: one with $1sec$ retention time with write back, second with $10ms$ retention time with write back and the third with $10ms$ retention time with buffering, called Revived-M-4MB. Compared to a base case design of 1MB per core SRAM design, the traditional non-volatile STT-RAM cache with 4 times the SRAM capacity but high write latency, and the volatile STT-RAM with simple write back policy, the proposed revive scheme shows both performance and power benefits across the application benchmarks studied in this paper. The results not only indicate that it is possible to get up to 18% improvement in speedup for PARSEC benchmarks and 60% reduction in total energy consumption over S-1MB design, but also show that the proposed design can be within 4% of the ideal case with an equal capacity SRAM configuration, while being more energy efficient. Furthermore, compared to the prior schemes that are aimed at hiding the high write latency of STT-RAMs, the approach to reduce its write latency seems a better solution for designing a performance and power efficient memory hierarchy for multi-cores.

References

[1] Systems Performance Evaluation Cooperation,SPEC Benchmarks, www.spec.org/. 3, 10

[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. 25

[3] C. Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011. 3, 10

[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. 3, 11, 15

[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. 7

[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. 6

[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. 9

[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, 9

[10] A. Driskill-Smith. Latest Advances in STT-RAM. In *2nd Annual Non-Volatile Memories Workshop*, 2011. 4, 25

[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. 7

[12] S. Kaxiras, Z. Hu, and M. Martonosi. Cache decay: exploiting generational behavior to reduce cache leakage power. In *ISCA*, 2001. 3, 4, 13

[13] 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. 8, 25

[14] X. Liang, R. Canal, G. yeon Wei, and D. Brooks. Process Variation Tolerant 3T1D-Based Cache Architectures. In *MICRO*, 2007. 3, 11

[15] 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, 7, 25

[16] A. K. Mishra, X. Dong, G. Sun, Y. Xie, N. Vijaykrishnan, and C. R. Das. Architecting On-Chip Interconnects for Stacked 3D STT-RAM Caches in CMPs. In *ISCA*, 2011. 26

[17] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. Scalable High Performance Main Memory System Using Phase-Change Memory Technology. In *36th ISCA*, 2009. 25

[18] 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. 6

[19] 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. 2, 3, 4, 5, 8, 25

[20] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen. A Novel Architecture of the 3D Stacked MRAM L2 Cache for CMPs. In *15th HPCA*, 2009. 2, 25, 26

[21] 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. 7

[22] 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. 7

[23] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. Energy reduction for STT-RAM using early write termination. In *ICCAD*, 2009. 2, 25