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 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 future multi-core architectures. However, it's long write latency and high dynamic write energy are major obstacles for being competitive with the SRAM-based cache hierarchy. On the other hand, the non-volatility feature with years of data retention time for STT-RAM technology is not necessary for the usage of STT-RAM as on-chip cache, since the life time of cache data are usually within us or ms. 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 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. 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 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 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 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 lifetime of the last level cache 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 caches, 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, 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 milliseconds. 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 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 [12, 14], 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 2006 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, in contrast to the results reported in [19] indicate that retention times in the range of milliseconds (*ms*) are probably more achievable than in the microseconds (μs) 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 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 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 (similar to one proposed in [12]) 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 2006 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 [19] is very aggressive and may not be feasible considering the state-of-the-art in device technology. Moreover, as our analysis shows, a very aggressive μs level retention time is unsuitable for last level cache block.

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 multi-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 XX% IPC improvement with PARSEC benchmarks, XX%/XX% instruction throughput/weighted speedup improvement with SPEC 2006 benchmarks, and an average energy saving of XX% across our entire application suite.

The rest of this paper is as follows: In Section **??**, we discuss the volatile STT-RAM design to parameterize the retention time and write latency behavior. Section **??** 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 **??**; the experimental platform details in Section **??**, results in Section **??**, and description of related work in Section **??**. The last section provides concluding remarks.

2. STT-RAM Design

2.1. Preliminary on STT-RAM

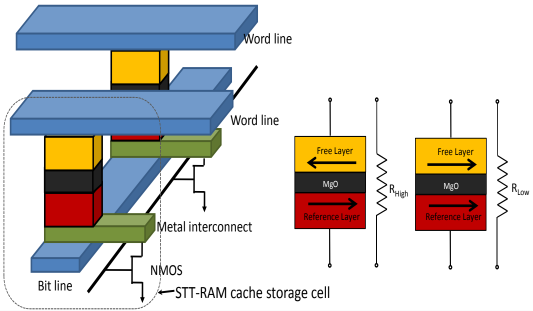


Figure 1. **(a) Structural view of of STT-RAM Cache Cell (b) Anti Space Parallel (High Resistance, Indicating “1” state (c) Parallel (Low Resistance, Indicating “0” state**

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

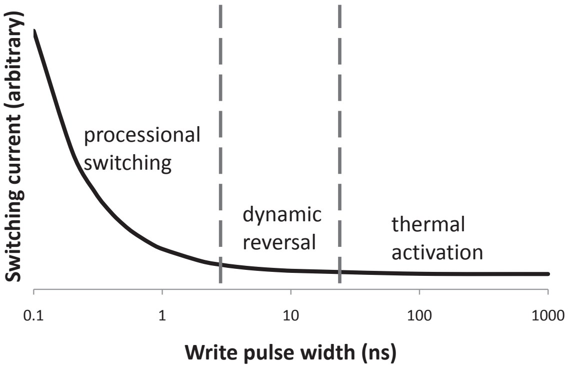


Figure 2. **Demonstration of three switching phases: thermal activation, dynamic reversal and precessional switching**

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 [18]. The equations are listed as follows,

$$\begin{aligned} J_{c,TA}(\tau) &= J_{c0}\{1-(\frac{k_BT}{E_b})ln(\frac{\tau}{\tau_0})\} && (1)\\ J_{c,PS}(\tau) &= J_{c0}+\frac{C}{\tau^\gamma} && (2)\\ J_{c,DR}(\tau) &= \frac{J_{c,TA}(\tau)+J_{c,PS}(\tau)e^{-k(\tau-\tau_c)}}{1+e^{-k(\tau-\tau_c)}} && (3)\end{aligned}$$

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 [22] is simulated. The BL-to-SL current and SL-to-BL current are obtained by assuming typical TMR (120%) and LRS ($3k\Omega$) value [15] 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 [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 calculated as follows,

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

2.4. Impact of MTJ Retention Time on STT-RAM

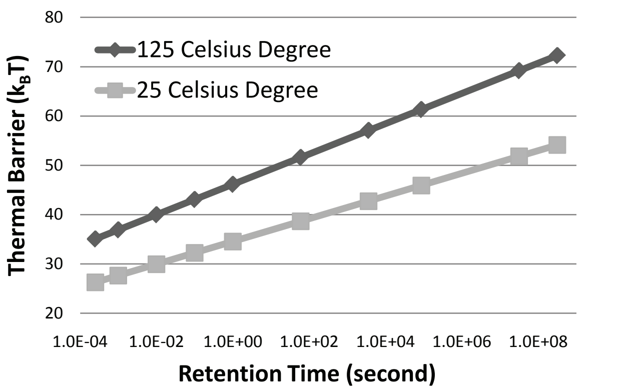


Figure 3. **MTJ thermal stability requirement for different retention time**

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 [13] 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.

Table 1. 16-way L2 Cache Simulation Results

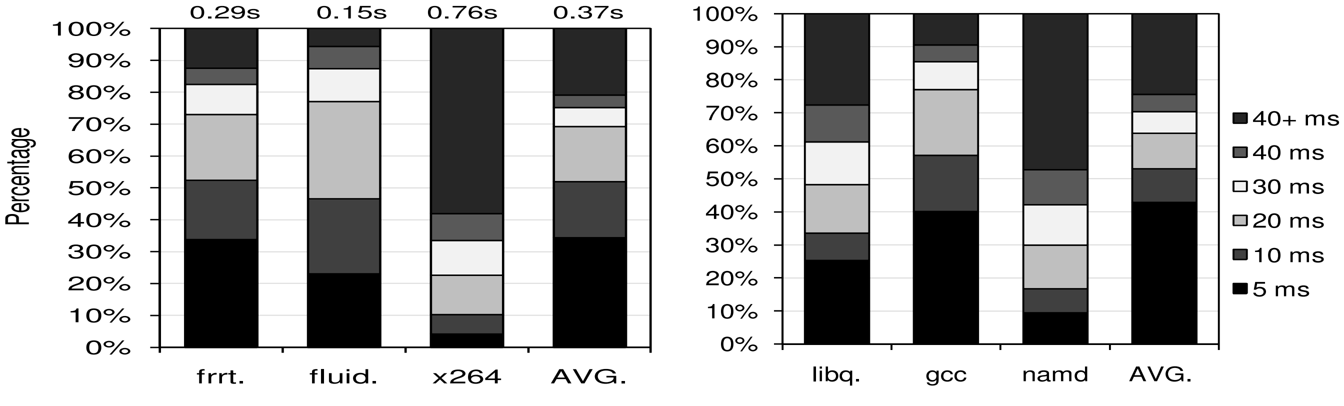
			Area (mm^2)	Read Latency (ns)	Write Latency (ns)	Read Energy (nJ)	Write Energy (nJ)	Leakage Power (mW)
$1MB$ SRAM			2.612	1.012	1.012	0.578	0.578	4542
$4MB$ STT-RAM	$t = 10yr$	Leakage Opt.	2.628	2.434	4.919	0.635	0.663	1399
		Latency Opt.	3.003	0.998	10.61	1.035	1.066	2524
	$t = 1s$	Leakage Opt.	2.203	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

2.5. 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. 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 than that of SRAM. 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 and 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 leverage 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



(a) PARSEC

(b) SPEC 2006

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

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.

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 [?] 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

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

L2 cache. Table 4 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 [**?**]. 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 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

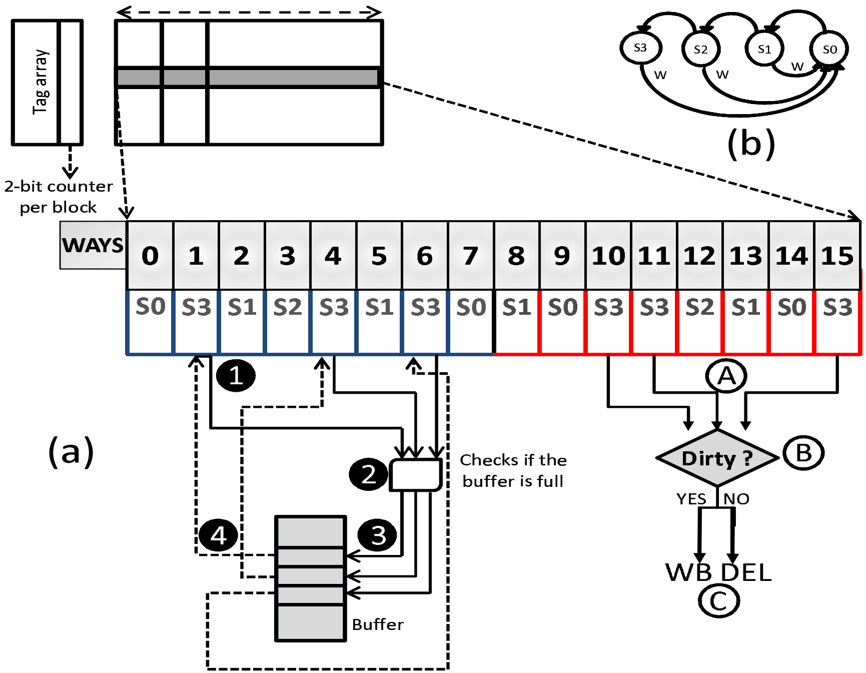


Figure 5. **A modified 16-way L2 cache architecture with a 2-bit counter and a small buffer**

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 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 unrefreshed 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 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 2.5 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 maintaining a large counter.

Our experimental results show that a 2-bit counter, similar to the one used in [12], is sufficient enough 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 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 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 (3.125% 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.

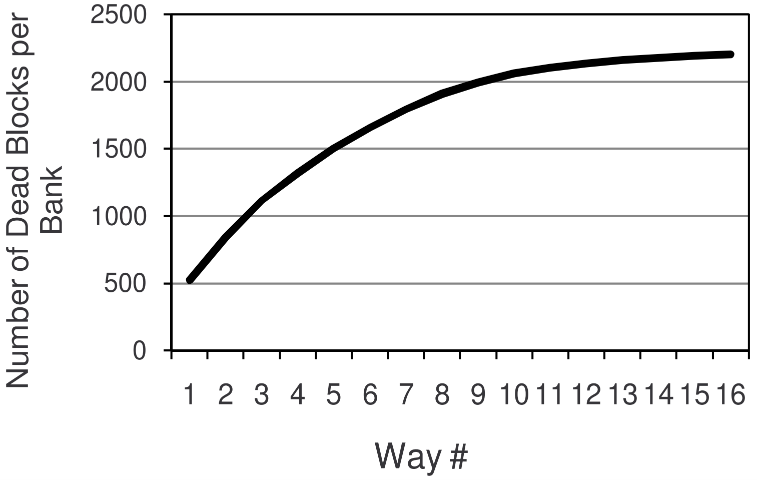


Figure 6. **Cumulative Distribution of Dead Blocks per Bank with number of ways.**

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	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	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	Read	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

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 4 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 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 our 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.
- **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 are for the design with 8 MRU Slots and 2048 Buffer Slots, shown in Figure 5..

Evaluation 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

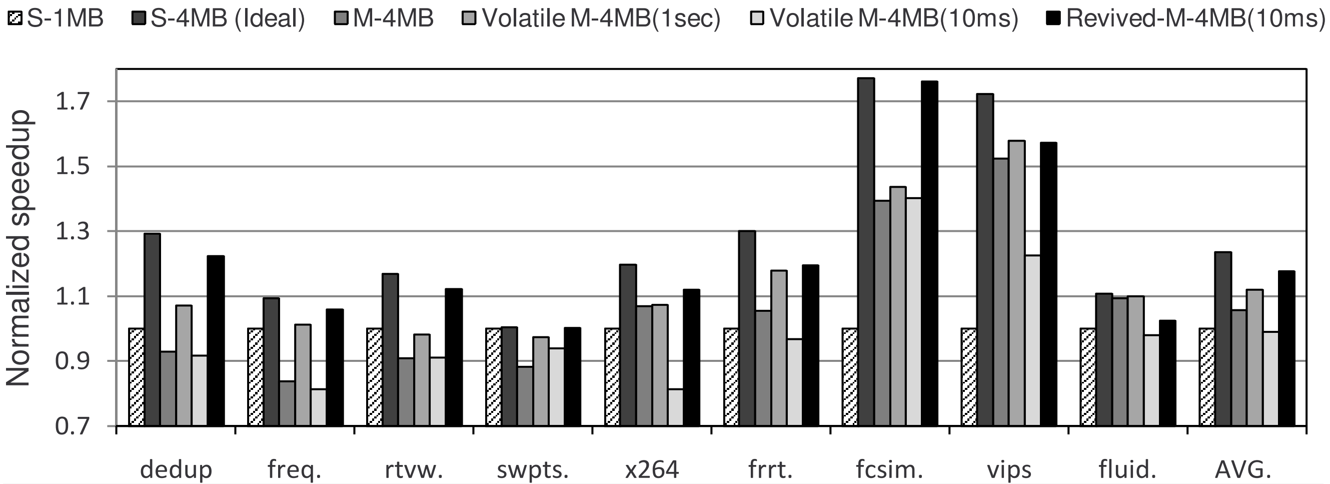


Figure 7. **Normalized speedup for PARSEC Applications**

report Instruction throughput and Weighted Speedup. We define instruction throughput (IT) to be 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)).

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

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

For analyzing the energy behavior, we measure the leakage energy, dynamic energy and total energy for all designs.

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 of several architectural parameters.

6.1. Performance comparison

Figure 7 shows speedup improvements of a subset of PARSEC multithreaded applications along with the average (taken across 12 PARSEC applications listed in Table **??**). All speedup numbers are normalized to S-1MB. For the M-4MB design, the applications to the right of x264 (including x264) exhibit speedup improvements over S-1MB because these are read intensive applications as shown in Table 3.

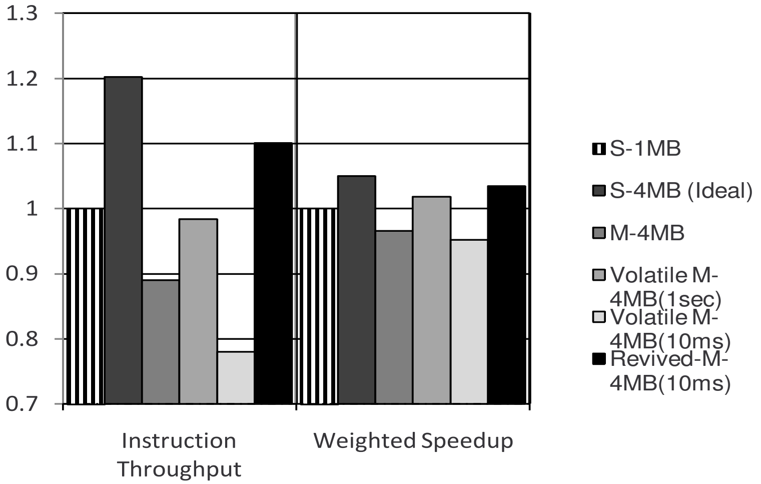


Figure 8. **Normalized Average Instruction Throughput(IT) and Weighted Speedup(WS) for SPEC 2006 multiprogrammed mixes.**

The read intensive applications not only benefit from the 4x capacity increase of STT-RAM, but also because of the presence of a write buffer for the L2 cache. To see the benefits of write buffer, let us consider fluidanimate and vips applications. Even though they have high number of writes to the L2 cache, the writes are staggered, which help the write buffer in ameliorating increased write latency. The applications to the left of x264 are write intensive applications and thus, we see degradation in speedup because of the STT-RAM high write latency. On an average, traditional 10 year STT-RAM gives 3% speedup improvement over S-1MB because most of the read intensive applications show considerable improvement in speedup (maximum speedup observed was YY% for application XX).

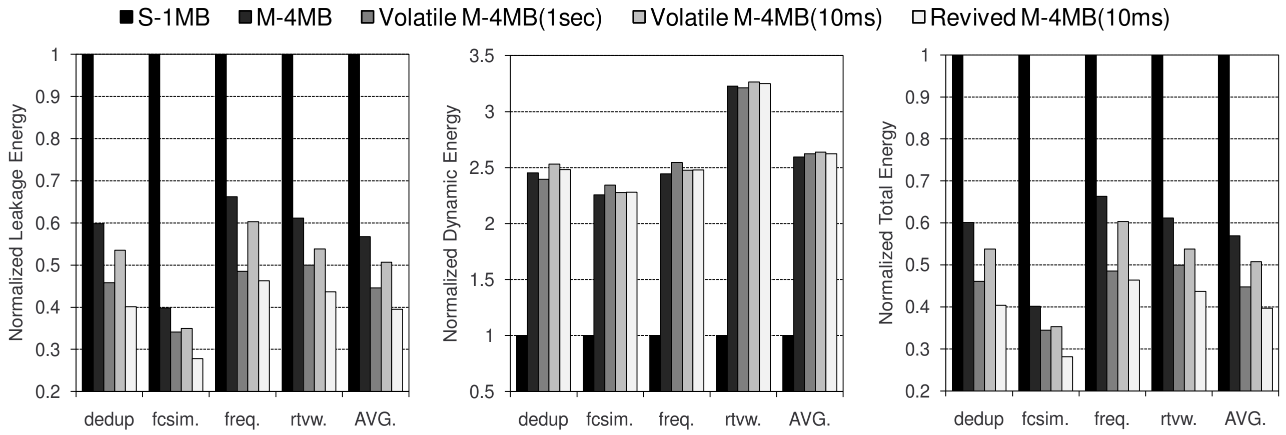
Next, let us consider Volatile M-4MB(1sec) design. This design has no refreshing scheme, but since within 1 sec interval, almost all the blocks are refreshed inherently as per application characteristics discussed in Section **??**, we observe speedup improvements of this design over M-4MB in all the applications because of the reduced write latency (12 cycles compared to 22 cycles). Again for a few of the write intensive application like swaptions, M-4MB(1sec) has no speedup gain compared to the base case S-1MB SRAM cache design. Volatile M-4MB(10ms) design also does not have any refreshing scheme, but the retention time of STT-RAM cells used is 10ms, which triggers large number of write backs. Figure 10 shows number of write backs of all the designs normalized to M-4MB. We observe that this design, on an average, has 21% more write backs than the traditional STT-RAM design. For this reason, 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 increase in 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.

Our scheme revived-M-4MB(10ms), which incorporates refreshing of dirty blocks beyond the 10ms retention time, depicts speedup improvements in almost all applications except for fluid, in which all the dirty blocks are almost equally distributed among all the ways. Hence, our scheme of the considering only first eight MRU slots does not prevent in stopping many writebacks. This observation can also be made from Figure 10, where write backs in our scheme are also very high. On an average, the proposed revived scheme is better than the conventional SRAM design (S-1MB) by 18%, traditional 10yr STT-RAM by 15% and over Volatile STT-RAM (1sec) by 4.5% (In case of facesim, our scheme is better than Volatile STT-RAM (1 sec) by 22.7%). The revived-M-4MB(10ms) scheme is closest to the ideal S-4M case with difference of only 4%.

Figure 8 shows instruction throughput and weighted speedup for the SPEC multiprogram mixes. We observe that our scheme revived-M-4MB gives 22% improvement in instruction throughput over M-4MB, 11% improvement over Volatile STT-RAM (1 sec), and 10% improvement over the base line SRAM cache. (although not shown in the Figure due to brevity, in case of the mix of bzip2, gcc, lbm, hmmer, the improvement is 15%). The weighted speedup improvement over M-4MB is 4% and over Volatile STT-RAM(1 sec) is 2%. (NEED TO EXPLIAN WHY annd more insight).

6.2. Energy comparison

Figure 9 shows normalized leakage, total of dynamic read and write energy, and total energy. The number of reads and writes to L2 cache are only considered for the calculation of dynamic energy. We observe that on an average there is 44% improvement in total energy going from S-1MB to M-4MB designs. The improvement is mainly because of the drastic reduction in leakage energy. Volatile M-4MB(1sec) leakage benefits over M-4MB correlates with the performance improvement. On an average, this design consume more dynamic energy than M-4MB. The dynamic energy fluctuations among differ-



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

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

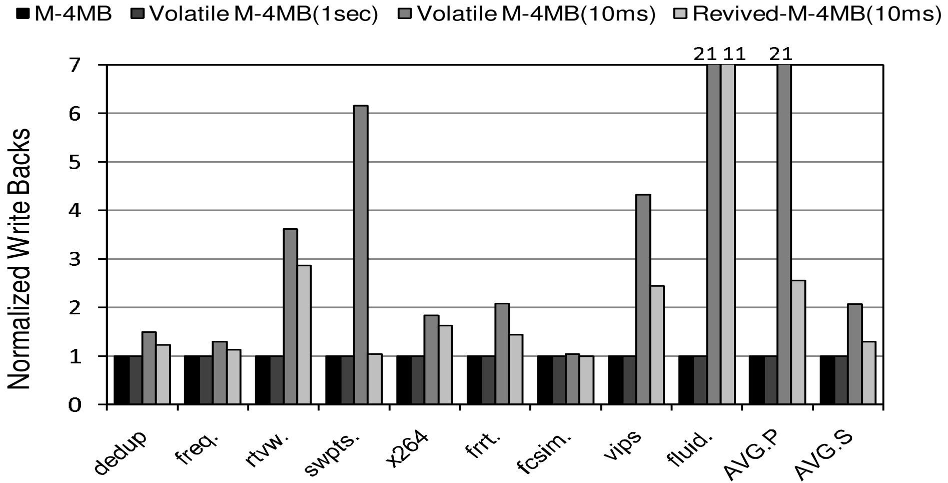
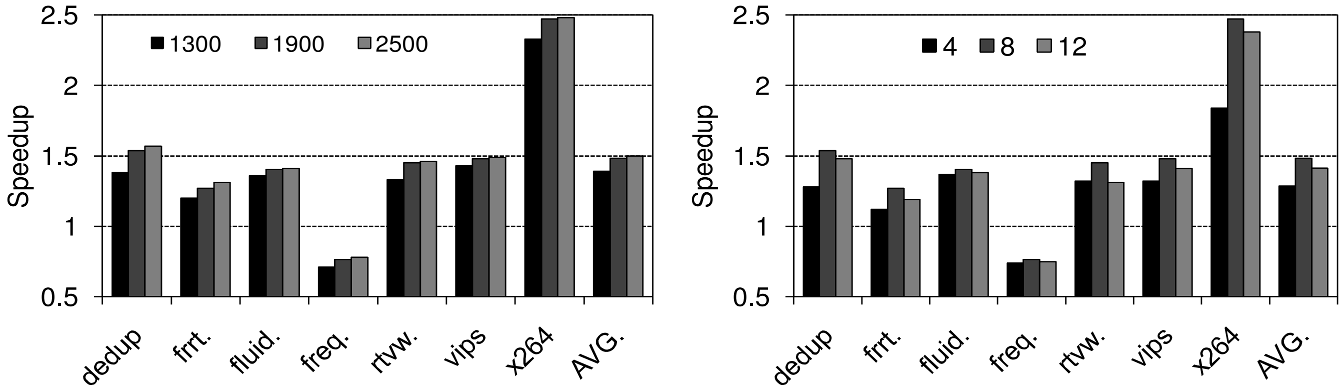


Figure 10. **Number of Write backs normalized to M-4MB**

ent applications are on account of changes in number of read and writes. Additional write backs triggers read misses which ultimately lead to additional writes to L2 cache. Write energies of 1sec design is more than the 10ms designs, which makes the fluctuations depend on the number of reads/writes.

We see 11% energy benefits of using revived M-4MB design over Volatile-1sec and 30% improvement over M-4MB designs.The energy numbers of this scheme covers all the overheads of the buffer design. We observe that our scheme is better in terms of both performance and energy over Volatile-4MB(1sec) and M-4MB designs.



(a) Buffer Entries

(b) MRU Slots

Figure 11. **Showing effects on speedup by varying number of Buffer Entries and MRU Slots**

6.3. Sensitivity Analysis

Sensitivity to number of Buffer entries: The number of buffer entries can tune the performance of Revived-M-4MB scheme. Increasing the buffer size will accommodate more diminishing blocks at a particular instance, leading to fewer buffer overflows. This reduction in overflows, is at the cost of increase in buffer area and energy overheads. Decreasing buffer size leads to more buffer overflows, which ultimately lead to additional write backs as discussed in 4

To find the optimal buffer size, we calculate 95% confidence intervals for the cumulative distribution of dead blocks per bank as shown in 12. We observe that, for first 8 MRU slots, mean value of the buffer entries is 1900 blocks, which corresponds to 3% area overhead over L2 cache bank. Upper limit to the 95% confidence interval corresponds to 2500 blocks (4% overhead over L2 cache bank). and lower corresponds to 1300 blocks (2% area overhead).

Figure 11 shows speedup of subset of PARSEC applications by varying the number of buffer entries. Going from 1900 to 2500 entries is giving only less than 0.5% speedup improvement. We assumed our buffer size to be 1900 entries for minimum area overhead giving best possible results.

Sensitivity to number of MRU slots: Choosing optimal number of MRU slots to buffer was discussed in 4. Figure **??** shows that mean cumulative distribution of diminishing blocks per way across all PARSEC benchmarks. We see that after 8 MRU slots, the number of diminishing blocks becomes negligible, which suggests that optimal number of MRU slots is 8. Figure 11 shows speedup of subset of PARSEC

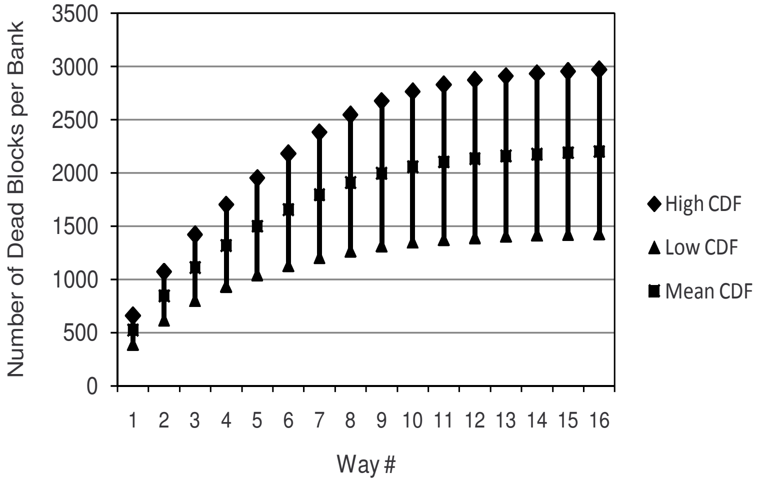


Figure 12. **95% Confidence Intervals of Diminished Blocks for each Way**

applications, along with the average across 12 PARSEC applications, with varying number of MRU slots. Buffer size is kept constant at 1800 per bank. We see degradation in performance when we decrease slots from 8 to 4 because 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 slots from 8 to 12. Since buffer size is kept constant, considering 12 MRU slots for buffering instead of 8, increases 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 4 increasing the number of bits of the counter decreases the left over time at the cost of incrementing counters at finer granularity. Our experiments showed that there is no observable difference in performance and energy by increase/decrease in 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 $10 years$ 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 has much smaller baseline planar area that is around $2F^2$. 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 millisecond and optimize STT-RAM cache correspondingly. Further, analysis of retention times of last level cache blocks of actual applications, show that the retention times are in the order of milliseconds, thus, making our proposal much more amenable to implement. Hence, compared to [19], our scheme will provide significant better performance.

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 [?] 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 11 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, 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. In this context, we conduct an application-driven study to characterize the life time of a second level cache 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 leat 50% of the cache blocks are updated in 10ms and thus, choose 10 ms 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 write backs all the dirty blocks in the cache at the end of the retention time and the second approach uses a limited 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 1 second retention time with write back, second with 10ms retention time with write back and the third with 10ms retention time with buffering, called revived-STT-RAM. 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 XX% improvement in instruction throughput and YY% reduction in total energy consumption, 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/. 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. 22

[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. 3, 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. 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. 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. 8

[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. 22

[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, 12

[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. 7, 22

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

[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. 7, 22

[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. 23

[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. 22

[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, 21, 22

[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, 22, 23

[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, 22