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

Anonymous MICRO submission

Paper ID 761

Abstract

Spin-Transfer Torque RAM (STT-RAM) is a CMOS compatible emerging non-volatile memory (NVM) technology that has the potential to replace the conventional on-chip SRAM caches for designing a more efficient memory hierarchy for future multicore architectures. However, it's high write latency and dynamic write energy are major obstacles for being competitive with the SRAM-based cache hierarchy. On the other hand, STT-RAM technology has another adaptable feature that it is possible to reduce its write latency by reducing its retention time, thereby making it volatile. In this paper, we exploit this volatile property of the STT-RAM for designing an efficient L2 cache architecture. The paper addresses several critical design issues such as how do we decide a suitable retention time for last level cache, what is the relationship between retention time and write latency, and how do we architect the cache hierarchy with a volatile STT-RAM. Through an extensive execution driven analysis of the inter-write time of several PARSEC and SPEC 2006 banchmarks, we observe that retention time in the order of 10-40 ms is a good design point to handle most of the writes. Then 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 writeback 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

Spin-Transfer Torque RAM (STT-RAM) is a promising memory technology that delivers on many aspects desirable of an universal memory. They exhibit high density, fast read times and low static power consumption. However, the high write latencies and write energy are key drawbacks of this technology. Consequently, recent efforts have focused on masking the effects of high write latencies and write energy at the architectural level. In contrast to these architectural approaches, a recent technique considers relaxing STT-RAM data retention times to reduce both write latencies and write energy. The focus of this paper is to tune this data retention time to closely match the required lifetime of cache line blocks to achieve significant performance and energy gains.

The non-volatile nature and non-destructive read ability of STT-RAM provides a key difference with regard to a comparably high-density DRAM memory. However, for many applications it is sufficient if the data stored in a cache hierarchy remains valid for a few tens of milliseconds. Consequently, the duration of data retention in STT-RAM is an obvious candidate for device optimization for cache design. First, we analyze how changes to the STT-RAM retention times influence the performance, power and area characteristics. A key distinction from prior efforts to relax data retention times is our consideration of device variability in our analysis (NEED SOME THING FROM CONG ON THIS IN HIS TEXT). This analysis reveals the granularities at which a device designer can reliably tune the data retention times.

Analyzing the lifetime of cache lines has been the focus of prior efforts to improve performance and reduce power consumption. In this work, we revisit this topic with the aim of identifying the suitable data retention times for STT-RAM caches. A key challenge in determining a suitable data retention

times for the STT-RAM is to balance the reduced write latency of cells with lower retention time with the overhead for data refresh or writeback of cache lines with longer lifetimes. Our analysis of the cache lifetimes performed for a multi-threaded workload demonstrates that a significant fraction of L2 cache lines can operate correctly without any additional support when the STT-RAM retention times are of the order of 50ms. However, architectural support is required to ensure that correct program state is maintained for the rest of the cache lines that have lifetimes exceeding 50ms. While a simple DRAM-style refresh has been proposed in [Sudhanva-et.al] to ensure correctness, it is possible to avoid many of these refresh by pursuing a life-time aware refresh strategy.

This work makes the following contributions

- We present a detailed device characterization of data retention tunability in STT-RAM Cells providing insight to the underlying principles enabling these tradeoffs

- 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.

- We present a simple buffering mechanism to ensure integrity of programs given the volatile nature of our tuned STT-RAM cells.

- Finally, we show that our combined device-architecture life time tuning approach is better than recent efforts that attempt to address the long write latencies of STT-RAM.

The rest of this paper is as follows.

Next, we evaluate the lifetimes of cache lines when executing multi-threaded workloads. A cache line is no longer required, if the

for the duration of its lifetime in the program execution.

The cache hierarchy is a key component influencing both the

2. STT-RAM Design

2.1. STT-RAM Cell Basics

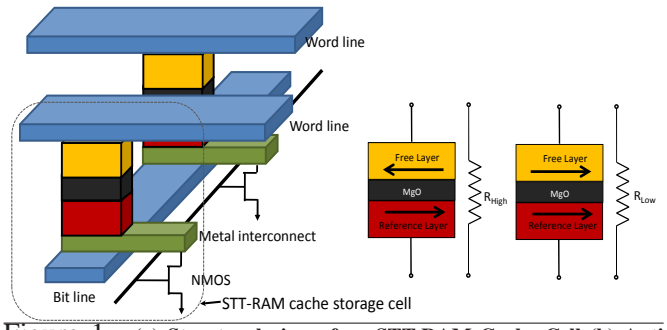


Figure 1. **(a) Structural view of an STT-RAM Cache Cell (b) Anti Space Parallel (High Resistance, Indicating "1" state (c) Parallel (Low Resistance, Indicating "0" state**

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

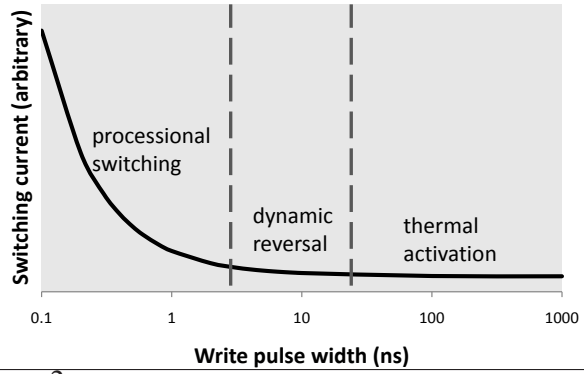


Figure 2. **(a) Structural view of of STT-RAM Cache Cell (b) Anti Space Parallel (High Resistance, Indicating "1" state (c) Parallel (Low Resistance, Indicating "0" state**

The current amplitude required to reverse the direction of the free ferromagnetic layer is determined by a lot of factors such as material property, device geometry and importantly the write pulse duration. Generally, the longer the write pulse is applied, the less the switching current is needed to switch the MTJ state. Three distinct switching modes were identified [**?**] 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 [**?**]. The equations are listed as follows,

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

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

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

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

2.3. STT-RAM Cell Area 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 [**?**] is simulated. The BL-to-SL current and SL-to-BL current are obtained by assuming typical TMR (120%) and LRS ($3k\Omega$) value [**?**] and bursting wordline voltage to be 1.5V (the optimal value is extracted from [**?**]). And we over size the access transistor width to guarantee enough write current provided to MTJ using the methodology discussed in [**?**]. To achieve high cell density, we model the STT-RAM cell area by referring to DRAM design rules [**?**]. 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

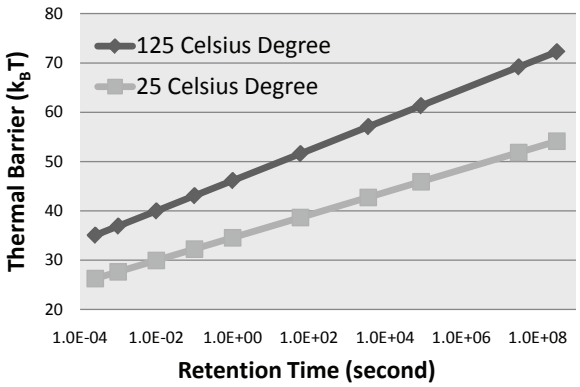


Figure 3. **MTJ thermal stability requirement for different retention time**

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 **??** 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

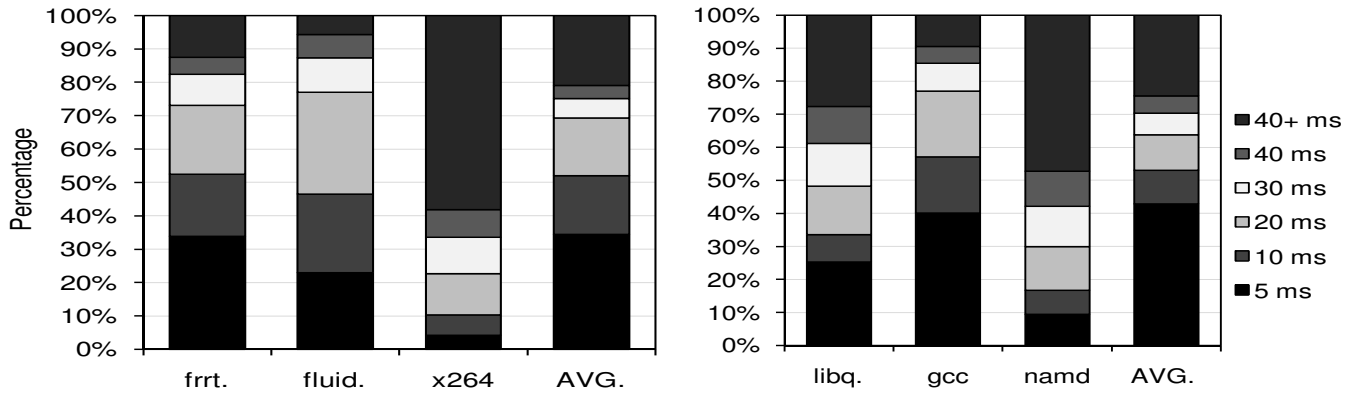
			Area (mm^2)	Read Latency (ns)	Write Latency (ns)	Leakage Power (mW)
$1MB$ SRAM			2.612	1.012	1.012	4542
$4MB$ STT-RAM	$t = 10yr$	Leakage Opt.	2.628	2.434	4.919	1399
		Latency Opt.	3.003	0.998	10.61	2524
	$t = 1s$	Leakage Opt.	2.203	2.044	3.552	1388
		Latency Opt.	2.904	0.973	5.571	2235
	$t = 10ms$	Leakage Opt.	2.167	1.956	3.390	1151
		Latency Opt.	2.901	0.959	2.598	2227

2.5. STT-RAM Cache Simulation Setup

We simulate SRAM-based caches and STT-RAM-based caches with a tool called NVsim [**?**], 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 [**?**]. 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, the read latency of leakage-optimized $4MB$ STT-RAM cache is significantly larger than $1MB$ SRAM cache because sensing the state of STT-RAM cell takes longer and fast SRAM sensing. Thus, we reduce the array size to improve the latency of STT-RAM cache. As can be seen in Table 1, the latency-optimized STT-RAM cache has noticeable better delay with $14\% - 34\%$ area overhead compared to leakage-optimized STT-RAM cache with the same retention time.

3. Understanding tradeoffs of Retention Time

In order to utilize the volatile STT-RAM as the last level cache in designing an effective cache hierarchy, we need to know what should be the ideal/feasible retention time. Ideally, the STT-RAM write latency should be competitive to SRAM latency and the cache retention time should be high. However, as discussed in the following 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 first



(a) PARSEC (b) SPEC 2006

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

attempt to decide an ideal retention time by analyzing the characteristics of a last level cache in a multi-programmed 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 don't consider the time in between the invalidation and the next write. Previous works [**?**] do similar type of revival time analysis, but for 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 [**?**]) and multi-programmed (SPEC 2006 [**?**]) applications on the M5 Simulator [**?**] that models a 2GHz processor consisting of 4 cores, with 4MB L2 cache. Table 3 contains additional details of the system configuration. Figures 4 (a) and (b) show the results of three PARSEC and SPEC benchmarks along with the averages across 12 PARSEC and 14 SPEC benchmarks, respectively. We observe from the figure that, on an average, approximately 50% of cache blocks get

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

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. We conclude that 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 too volatile leading to degraded performance, while increasing the retention time would affect the write latency. In the next subsection, we will see how increasing the retention time, has negative impacts on write latency.

3.2. Low Retention STT-RAM Characteristics

Table 2 shows that there is significant reduction in write latency with reduction in retention time. Section 4 explains how these numbers originate. 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, lets consider an utopian cache with 10 year retention time having minimum write latency and energy. To bridge the gap between current and utopian cache, we need to reap the benefits of both: application characteristics and emerging device technology. From 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. We choose 10 ms retention time as optimal retention time which balances both the sides. In Section 7, we do a sensitivity analysis by choosing retention times 100 ms,

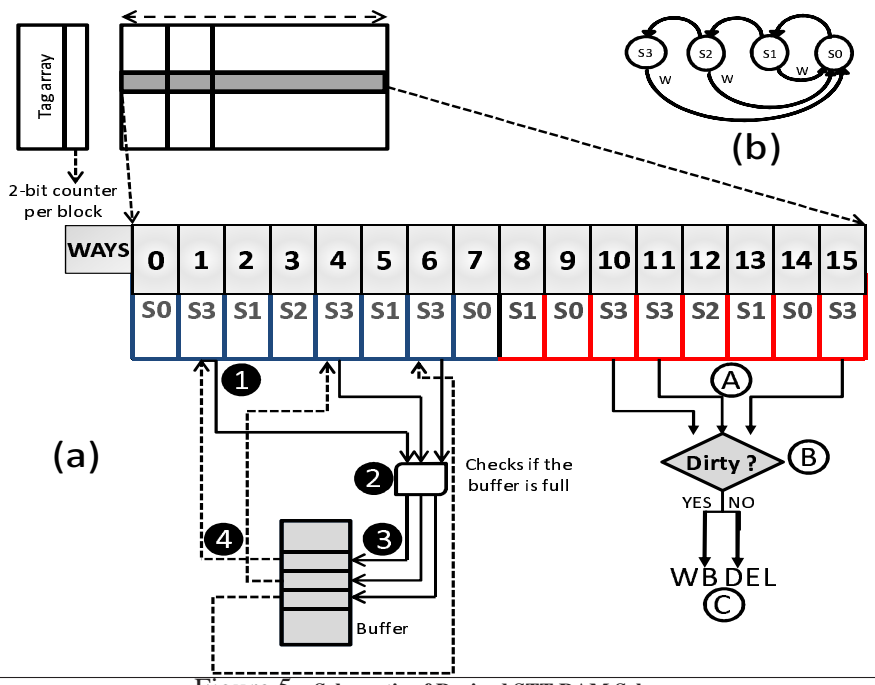


Figure 5. **Schematic of Revived STT-RAM Scheme**

500 ms and 1sec. In Section 5, we propose micro-architecture techniques to deal with blocks having revival time greater than 10ms.

4. Architecting Volatile STT-RAM

In Section 3, we argued that 10 ms is the ideal retention time for L2 cache blocks. We observe from figure 4 that on an average, approximately 50% of the cache blocks will expire after 10 ms, if no action is taken. It will not only result in additional cache misses but also the expiration of dirty blocks would result in data loss. In this section, we will first describe the counter design used for tracking revival times of the cache blocks. We will then propose our architectural solution starting with naive scheme of writing back all the dirty blocks, to more sophisticated schemes where we minimize the number of refreshes and write backs.

Counter Design: We maintain a 2 bit counter per cache block similar to the one used in [**?**]. These bits are kept as a part of the SRAM tag array. Figure 5 (b) shows the transition diagram of the counter. Each cache block can be in one of the four possible states. The block in *S3* state indicates that it has attained its retention time and is going to expire. After every 2.5 ms (= retention time / number of states), block

changes it state as shown in the figure. The block goes back to its initial *S0* state if the data is written or invalidated in between. We calculate the overhead of 2 bit counters to be 0.4% over one L2 cache bank.

4.1. Volatile STT-RAM

In this naive design, we write back all the dirty blocks which are going to expire. It has negative impact on the performance for two reasons: 1) There will be large number of write backs to the main memory. 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

Figure 5 shows the schematic diagram of overall architecture design of this scheme. The main components of this design are:

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

Buffer Controller Buffer controller consists of a 12 bit overflow detector for the buffer. We also maintain a 12 bit block identifier associated with every buffer entry. If an expired block is directed to the buffer, overflow detector is first checked to see the occupancy of the buffer. If the buffer can hold blocks, a block id is generated by concatenating its set and way id and stored along with actual block in one of the empty buffer entries. If the buffer is fully occupied, block is written back to the main memory if it is dirty, otherwise we will let it expire.

Implementation Details Revived STT-RAM Scheme is hereby proposed. In figure 5 (a) L2 cache is shown with associated tag array containing counter bits. One of the sets, is shown in detail to clarify the details of the scheme. All the blocks in the way are marked with their current state. We only refresh expiring blocks which are present in first eight MRU slots. We will shortly justify our decision of taking eight MRU Slots. ❶ shows a block which is in of the eight MRU slots and is expiring. This block is directed to the buffer. ❷ checks the occupancy of the buffer and if it is not full, it is copied to ❸ along with block id. This block id is again used to copy back the block in ❶ . Ⓐ shows the blocks which are

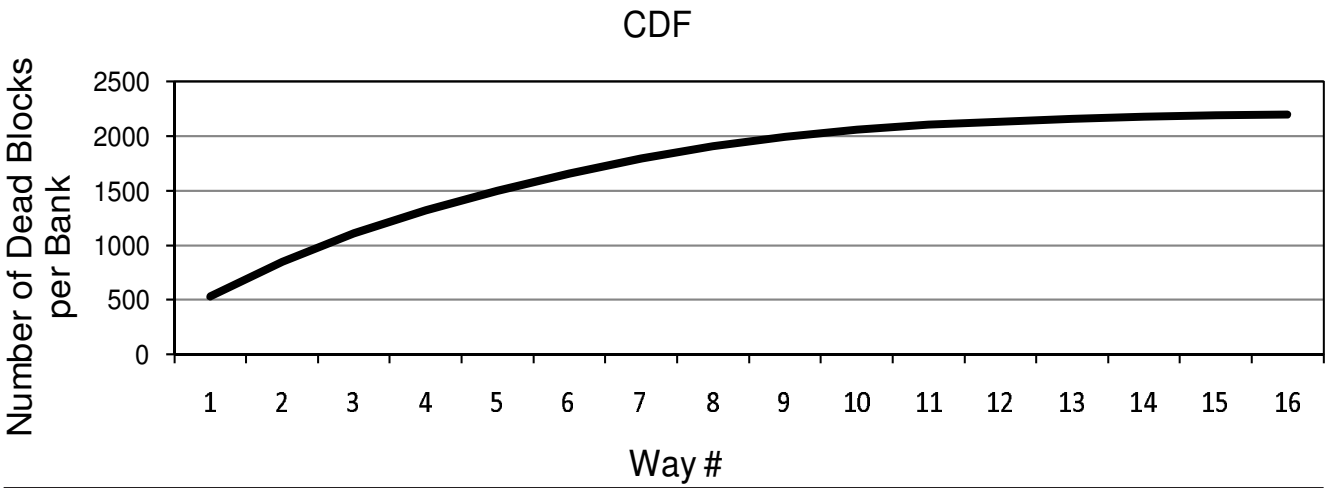


Figure 6. CDF

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

Processor Pipeline	2 GHz processor, 64-entry instruction window, Fetch/Exec/Commit width 8
L1 Caches	64 KB per-core (private), 4-way set associative, 64B block size, write-back, split I/D caches, 10 MSHRs
L2 Caches	1MB banks, shared, 16-way set associative, 64B block size, 4-cycle bank latency, 10 MSHRs
Main Memory	4GB DRAM, up to 16 outstanding requests for each processor, 400 cycle access

not in MRU slots, but are going to expire. We check these blocks in Ⓑ to see whether they are dirty or not. If they are dirty we first write back those blocks as shown in Ⓒ. If they are not dirty, we let them expire.

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

5. Experimental Evaluation

Experimental Setup We evaluate our design and schemes on the modified ALPHA M5 Simulator [] . We operate M5 Simulator in Full System (FS) mode for PARSEC applications and in System Emulation (SE) Mode for SPEC 2006 Multiprogammed mixes. We model a 2GHz processor with four out of order

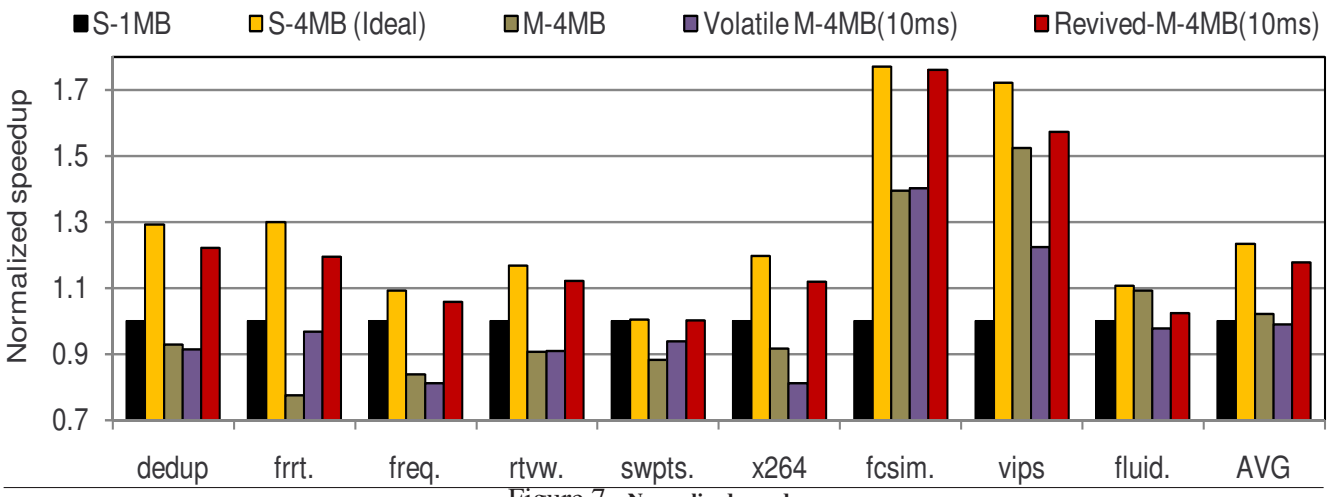


Figure 7. **Normalized speedup**

cores. We modified M5 simulator to model low retention time STT-RAM for L2 cache. The L2 cache is banked with different read and write latencies, with all the banks connected via a shared memory bus. We assume a fixed 400 cycles main memory latency for all our simulations. Table 3 details our experimental system configuration.

Collection of Results We report results of 12 multithreaded PARSEC applications and 14 SPEC 2006 multiprogrammed mixes. Table **??** shows the characterization of PARSEC applications and list of multiprogrammed mixes. We use sim-small input for PARSEC benchmarks and report the results of only Region of Interest (ROI) after skipping the initialization and termination phases (except facesim, where we report results for only 2B instructions of ROI) We also warm up caches for 100M Instructions in ROI. For SPEC multiprogrammed mixes, we fast forward 1B Instructions, warm up caches for 500M instructions and then report results for 1B instructions.

Performance Metrics For 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 decrease in execution time of the slowest thread. We randomly choose 14 multiprogram mixes, each mix with four different applications and assign each every application to a core. We report Instruction throughput and Weighted Speedup for SPEC multiprogrammed mixes, which are defined as:

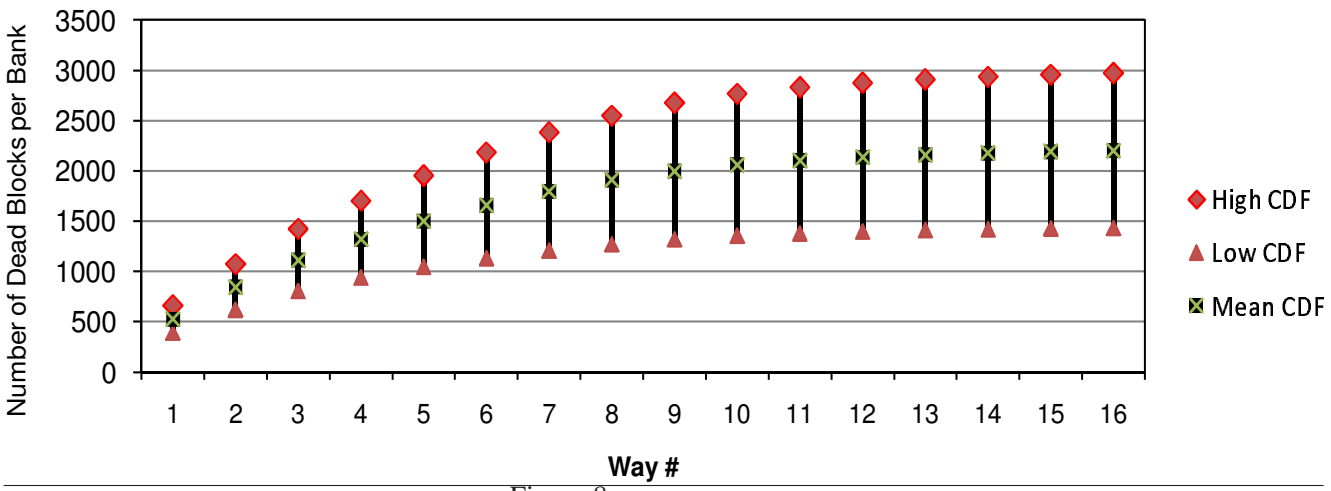


Figure 8. **Confidence Intervals**

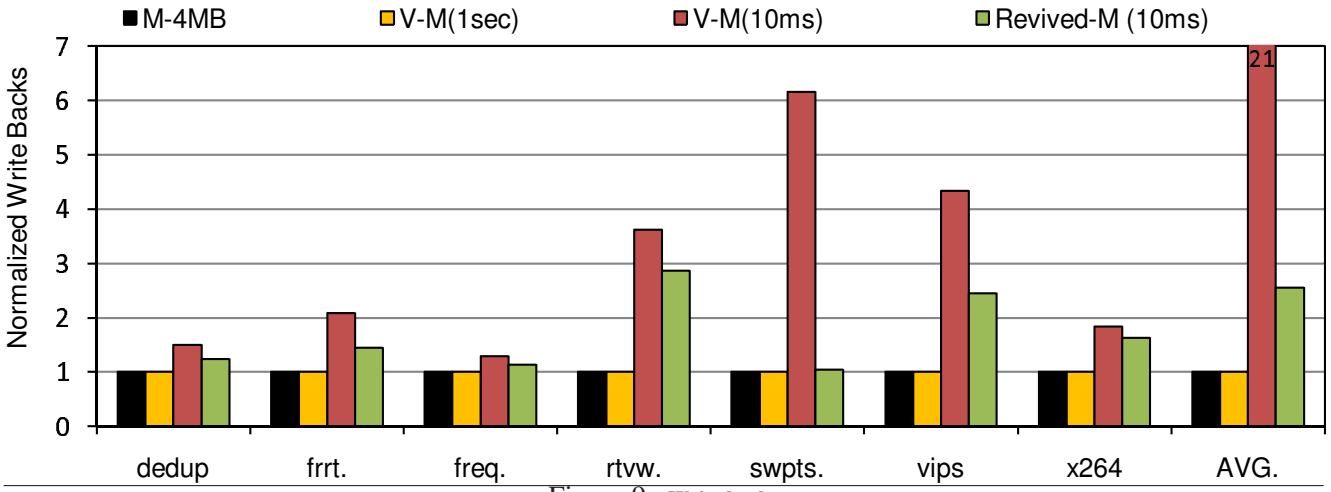


Figure 9. **Write backs**

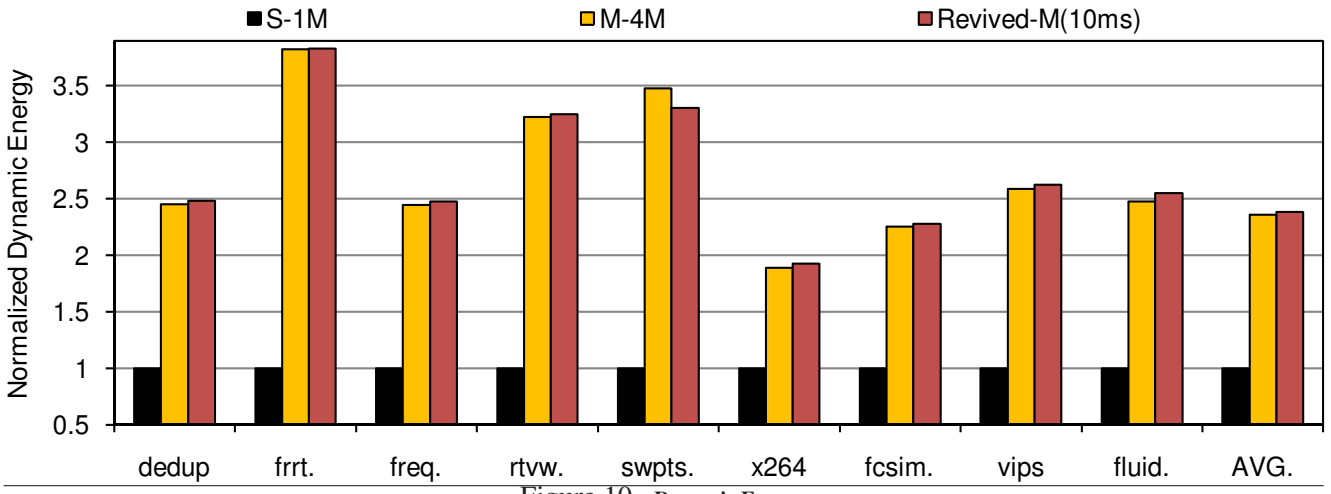


Figure 10. **Dynamic Energy**

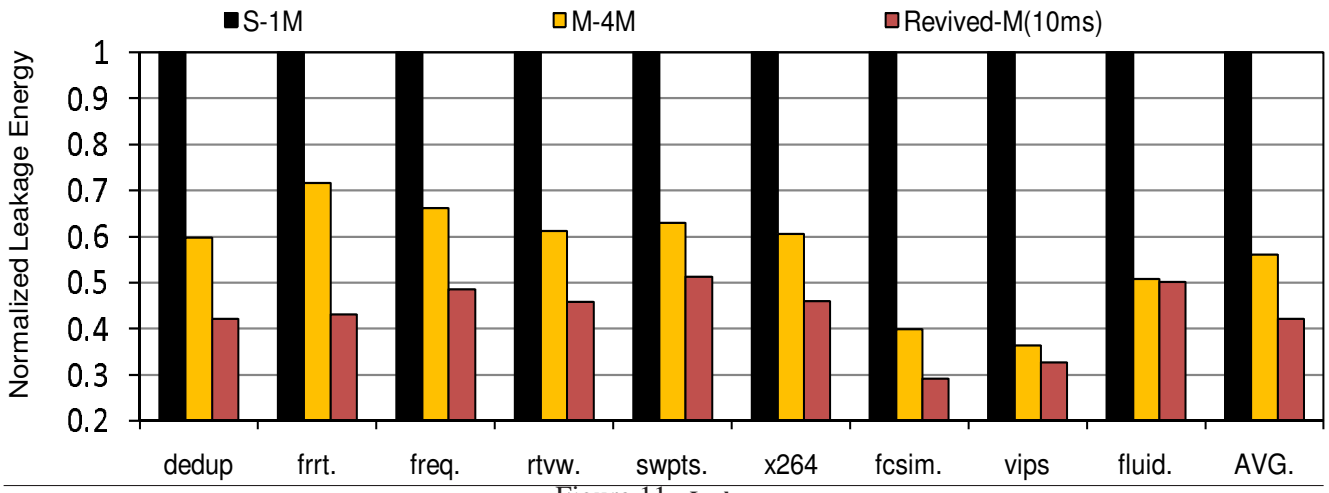


Figure 11. **Leakage**

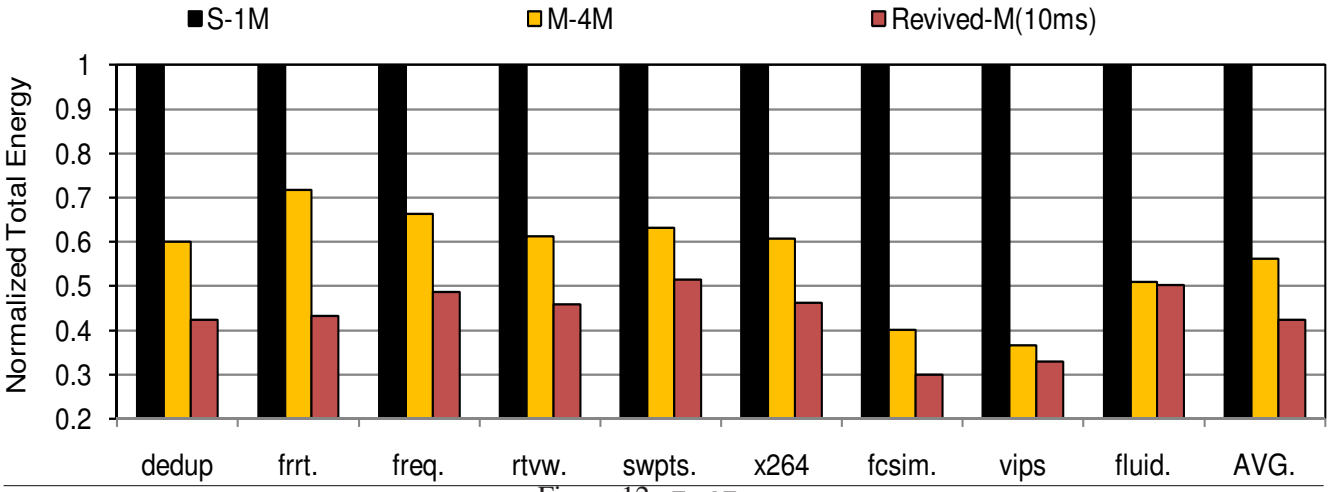


Figure 12. **Total Energy**

6. Results

Normalized speedup 7 shows normalized speedup of PARSEC multithreaded applications. S-1MB stands for 1MB SRAM Bank, M-4MB non-volatile STT-RAM Bank, Volatile M-4MB for 4MB STT-RAM Bank with no buffers, and Revived M-4MB with buffers. We observe that our scheme is giving 11% IPC improvement over traditional

Weighted Speedup and Instruction Throughput

Normalized Energy

6.1. Sensitivity Analysis

Effect of variation in retention times

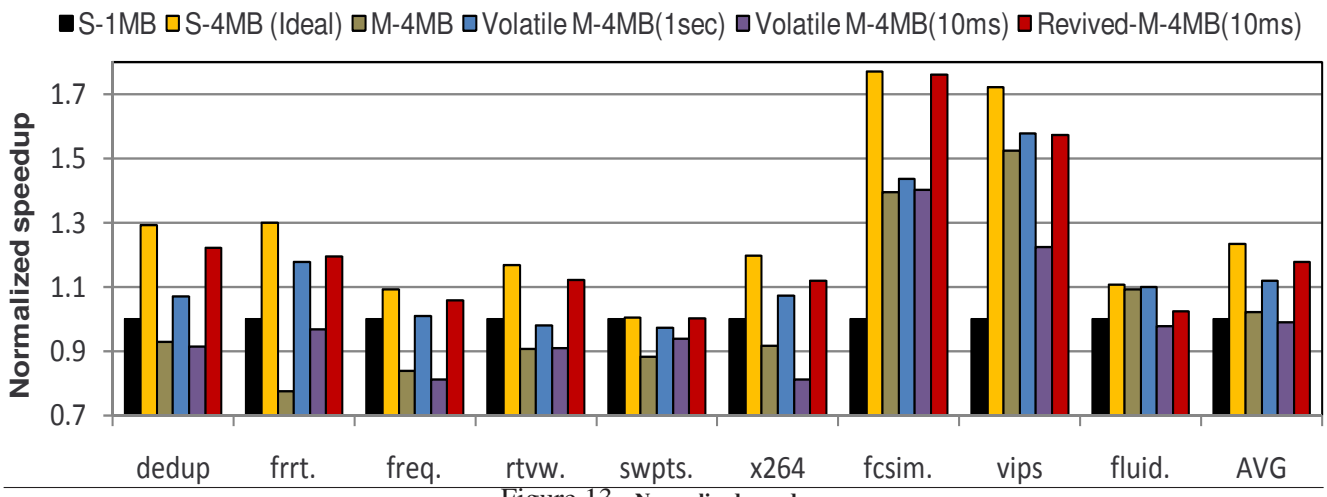


Figure 13. **Normalized speedup**

Choosing correct buffer slots

Number of bits for counter

6.2. Scaling Effects

7. Prior Work

8. Conclusions

Sample bibliography [1]".

References

[1] Authors. Frobnication tutorial, 2011. Supplied as additional material `tr.pdf`. 16