Cache Revive: Architecting Volatile STT-RAM Caches for Enhanced Performance in CMPs

Adwait Jog[†] Asit K. Mishra[§] Cong Xu[†] Yuan Xie[†]
Vijaykrishnan Narayanan[†] Ravishankar Iyer[§] Chita R. Das[†]
[†]The Pennsylvania State University
University Park, PA 16802, USA
{adwait,czx102,yuanxie,vijay,das}@cse.psu.edu
Intel Corporation[§]
Hillsboro, OR 97124, USA
{asit.k.mishra,ravishankar.iyer}@intel.com

ABSTRACT

High density, low leakage and non-volatility are the attractive features of Spin-Transfer-Torque-RAM (STT-RAM), which has made it a strong competitor against SRAM as a universal memory replacement in multi-core systems. However, STT-RAM suffers from high write latency and energy which has impeded its widespread adoption. To this end, we look at trading-off STT-RAM's non-volatility property (data-retention-time) to overcome these problems. We formulate the relationship between retention-time and write-latency, and find optimal retention-time for architecting an efficient cache hierarchy using STT-RAM. Our results show that, compared to SRAM-based design, our proposal can improve performance and energy consumption by 18% and 60%, respectively.

Categories and Subject Descriptors

B.3.2 [**Hardware**]: Memory Structures—*Cache memories*; B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Advanced technologies, Memory technologies*

General Terms

Design, Experimentation, Measurement, Performance

Keywords

STT-RAM, Heterogeneous (hybrid) systems

1. INTRODUCTION

The emergence of multicore architectures in both embedded processors as well as general purpose computing domain has started to increasingly stress the demand for the on-chip cache memories. As the number of cores on a chip continues to increase with technology scaling, the demand for the on-chip memory would continue to increase significantly, further worsening the memory wall problem [3]. This memory wall problem is critical both from the performance and power perspectives. Thus, it is imperative to look for novel technology; circuit, and architectural techniques to address the memory wall problem.

DAC 2012, June 3-7, 2012, San Francisco, California, USA.

Spin-Transfer Torque RAM (STT-RAM) is a promising memory technology for future multi-core general purpose and embedded systems that delivers on many aspects desirable of a universal memory. STT-RAM has the potential to replace the conventional on-chip SRAM caches because of its higher density, competitive read times, and lower leakage power consumption compared to SRAM. Its high-density property (3x-4x denser than conventional SRAM) can provide denser memories at lower area footprint with near-zero leakage energy. However, the latency and energy overhead associated with the write operations are the 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 both at the architectural [16, 19] level and circuit level [18]. In contrast to these approaches, a recent work explored the feasibility of relaxing STT-RAM data retention times to reduce both write latencies and write energy [15]. 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 volatile STT-RAM.

The non-volatile nature and non-destructive read ability of STT-RAM are the key differences with regard to traditional on-chip cache design with SRAM technology. However, as our analysis will show, for many emerging applications, it is sufficient to store the valid data in the LLC for a few tens of ms in contrast to μs for the L1 cache [10]. Consequently, the duration of data retention in STT-RAM is an obvious candidate for device optimization in the cache design. We, therefore, conduct an application-driven study to analyze the inter-write times (refresh times) of the L2 cache blocks to determine a suitable data retention time. Although lifetime analysis of cache lines has been conducted earlier to improve performance and reduce power consumption [9, 10], we revisit this topic with a different intention - correlating STT-RAM data-retention time to cache lifetime. An extensive analysis of emerging workloads using the M5 simulator [2] 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.

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

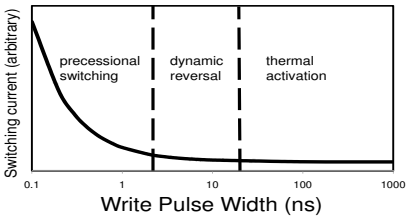


Figure 1: Demonstration of three different switching phases

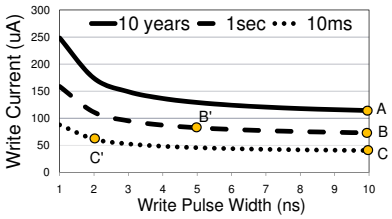


Figure 2: Write current vs. write pulse width for different MTJ retention time

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; and (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 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.

The primary contributions of this paper can be summarized as:
(1) Detailed characterization of STT-RAM volatility property: We present a detailed device characterization of data-retention tunability in STT-RAM cells, providing insight into the underlying principles enabling these tradeoffs.
(2) An application-driven study to determine retention time: With the aid of application level characterization, we propose the design of STT-RAM with the retention time in the range of $10ms$. Also, such a design makes the LLC homogeneous (all same type of STT-RAM cells) leading to lower die cost and ease in fabrication.
(3) 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. This scheme is simple, yet very energy and performance efficient.

2. STT-RAM DESIGN

In this section, we present a detailed discussion of STT-RAM models which we have developed to guide us in our exploration of suitable STT-RAM device for LLC. Preliminaries for STT-RAM can be reviewed in supplemental Sec. S.1 as well as in [4, 5, 12].

2.1 Write Current vs. Write Pulse Width

In this section, we will establish the relationship between write current and write pulse width. In [4], three distinct switching modes were identified based on the operating range of switching pulse width (τ): thermal activation (TA) ($\tau > 20ns$), precessional switching (PS) ($\tau < 3ns$) and dynamic reversal (DR) ($3ns < \tau < 20ns$). The relationship between switching current density (J_c), and write pulse width (τ) in these three operating ranges are characterized by the following analytical model [14, 18]:

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

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

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

where, $J_{c,TA}$, $J_{c,PS}$, $J_{c,DR}$ are the switching current densities for TA, PS and DR 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 the inverse of attempt frequency. C, γ, k, and τ_c are fitting constants.

Figure 1 shows write current vs. write pulse width characteristics for PS, DR and TA modes. In TA mode, the switching current increases very slowly with decrease in write pulse width. This suggests that, in TA mode, shorter pulse width is beneficial in reducing write latency and energy without hampering the read latency and energy. On the contrary, in PS mode, write current goes up rapidly as we reduce the write pulse width and minimum write energy of the MTJ can only be obtained at some particular write pulse width in this region, as shorter write pulse width with optimal write current can still provide necessary write energy to switch. DR mode has intermediate characteristics compared to PS and TA. Based on these trade-offs, this paper focuses on PS and DR modes for achieving our overall goal of minimizing write latency and energy.

2.2 Impact of Retention Time on MTJ Characteristics

Retention time of an MTJ is primarily impacted by the thermal stability of its free layer. The relationship between retention time and thermal barrier can be modeled as $t = C \times e^{k\Delta}$ [15], where t is the retention time and Δ is the thermal barrier, while C and k are fitting constants. This relationship suggests that retention time of an MTJ reduces exponentially with reduction in the thermal barrier. As described in [7], 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 Sec. 2.1, to conclude that faster write speed or/and small write current/energy can be obtained by lowering the thermal barrier of a MTJ, at the cost of lower retention time.

Thermal barrier of an MTJ can be lowered by reducing the MTJ planar area [15] and thickness of the MTJ [13]. In our study, we use $2F^2$ state-of-the-art in-plane MTJ [7] as our baseline with 10+years of retention time and thermal barrier of $72k_BT$, where k_B is the Boltzmann constant and T is the temperature. Since we use optimized $2F^2$ cell, there is not much leeway to reduce the planar area. Instead, we decrease the thickness of the free layer and lower the saturation magnetization to obtain lower thermal barrier. The minimum thickness of the free layer for the MTJs in our work is $2nm$ and we do not reduce it further to avoid reliability and process variation issues. Our volatile MTJs have thermal barriers of $46k_BT$and $40k_BT$, which correspond to retention times of $1sec$ and $10ms$ at $125\,^{\circ}$C, respectively.

We obtained raw experimental data from our device collaborator and further did curve fitting using the in-plane MTJ device equations (1)-(3). The curve-fitted results for three different MTJs (10years, $1sec$, $10ms$) are shown in Figure 2. Operating point A($10ns$, $114\mu A$) serves as our baseline [5]. By fixing the write pulse width at $10ns$, we reduce the write current to obtain volatile MTJs operating at B($10ns$, $73\mu A$) and C($10ns$, $40\mu A$). Then we apply the write current versus write time trade-off on these operating points to reduce the write latency. Specifically, we operate the MTJ with $1sec$ retention time at point B'($5ns$, $82\mu A$) which corresponds to 28% lower write current than that of baseline (point A). Also, we operate $10ms$-retention time MTJ at point C'($2ns$, $61\mu A$) which further cuts down the write current by 25%. Hence, we observe that relaxation of retention time of STT-RAM can reduce both write latency and energy. Based on this analysis, we integrate the cache-level SRAM and STT-RAM models in

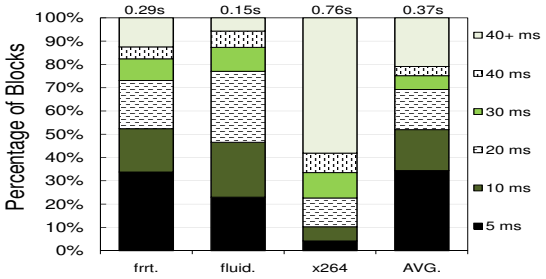


Figure 3: **Distribution of blocks showing different revival times (value on the top of a bar show the maximum revival time for that distribution)**

NVsim [6] and simulation results are tabulated in Table 3 (supplemental Sec. S.3).

3. AN APPLICATION-DRIVEN APPROACH TO DETERMINE RETENTION TIME

In order to leverage the volatile STT-RAM properties, 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 multithreaded PARSEC 2.1 [1] and multiprogrammed SPEC 2006 environment. These suites include emerging workloads from desktop to server domains. `dedup` and `bzip2` are compression schemes and `facesim` simulates face (important in facial recognition and virtual-reality systems). `vips` and `x264` are image processing and video encoding workloads, respectively. All these workloads can be hosted on high end mobile/tablet processors. The main idea is to understand the distribution of the inter-write intervals to an LLC and use the average of these intervals as the STT-RAM 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 as the $revival\ time$. 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 [10] has done similar type of revival time analysis, but it was for the L1 cache. Figure 3 shows the distribution of L2 cache blocks having different revival time intervals. These results are obtained by running emerging workloads on the M5 Simulator [2]. Table 2 contains additional details of the system configuration. Figure 3 shows the results of three applications along with the averages across the entire PARSEC 2.1 suite. We observe similar distribution for SPEC 2006 applications and more details are given in supplemental Sec. S.7. We observe from the figures that, on average, approximately 50% of the cache blocks get refreshed within $10ms$, which is in contrast to the μs reuse for L1 cache studied in [10]. About 20% of blocks remain in the cache for more than $40ms$ and rest of the blocks have intermediate revival times. Blocks that stay longer than the retention time in the cache without being refreshed

Table 1: **Retention and Write Latencies for STT-RAM L2 Cache**

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

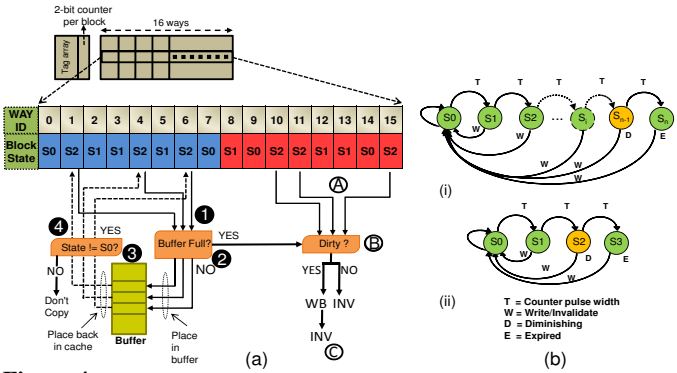


Figure 4: **A modified 16-way L2 cache architecture with a 2-bit counter and a small buffer**

are assumed to be expired. 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 1 summarizes the retention time and write latency tradeoffs based on the analysis of Sec. 2. The results indicate a significant reduction in write latency with reduction 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 which in turn diminish the benefits of performance/energy, since the number of blocks to be refreshed increases (Figure 3).

From above discussions we conclude that, from the application perspective, it is best to choose a retention time which minimizes the number of unrefreshed blocks and from the technology side, it is ideal to use STT-RAM with minimum write latency and energy.

4. ARCHITECTING VOLATILE STT-RAM

In this section, we propose 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 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 4(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 larger 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 at the cost of maintaining a counter with high number of bits.

Our experimental results show that a 2-bit counter, similar to the one used in [9], 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 4 (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.

4.2 Revived STT-RAM Scheme

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 as the *revived* STT-RAM scheme. These dirty blocks are, thus, not written back to the main memory. They are written to the temporary buffer and written back to the cache to start another fresh cycle. Figure 4(a) shows the schematic diagram of the proposed scheme. The main components of this design are:

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.

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 4 (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 associated with a buffer and the buffer controller. Let us consider that the buffering scheme incorporates eight MRU slots (in supplemental Sec. S.8 we show that eight MRU slots give the best benefits). In Figure 4(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 diminishing (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 IDs. Way and set IDs 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 (WB) those blocks as shown in Ⓒ and then invalidate (INV). If they are not dirty, they are just invalidated. During this whole refresh process, if a read request for the cache block arrives, it will be successfully completed as the data is still valid during that time. If a write/invalidate request arrives, the cache block goes back to its original state. To make sure we don't copy the stale data from the buffer, we perform a state check before copying back as shown in ❹. Moreover, our implementation assumes the worst-case temperature of 125 °C, and hence, there is no possibility of early expiration of block because of sudden reduction in STT-RAM retention time.

Table 2: Baseline system configuration

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

5. EXPERIMENTAL EVALUATION

We evaluate our designs using M5 Simulator [2] with PARSEC 2.1 and SPEC 2006 applications. We model a 2GHz processor with four cores. We modified the M5 simulator to model L2 cache banks composed of tunable retention time STT-RAM cells. Table 2 details our experimental system configuration. More details on methodology of collection of results are described in supplemental Sec. S.7.

The design scenarios we evaluate are:

• **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 a hypothetical case, where 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 hypothetical design has the capacity and area of an 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 Sec. 4, where all L2 cache banks are composed of 1sec retention time STT-RAM cells.

• **Volatile M-4MB(10ms):** This design is similar to Volatile M-4MB(1sec) but with 10ms STT-RAM retention time.

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

6. ANALYSIS OF RESULTS

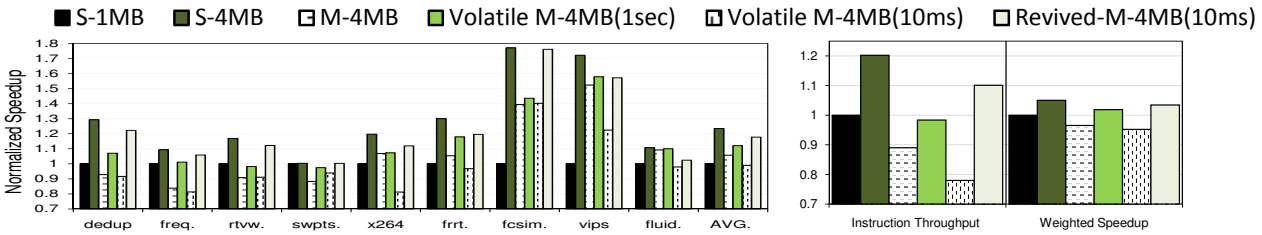
In this section, we provide a comparative analysis of the performance and energy results of the proposed six designs.

6.1 Performance Comparison

Figure 5(a) shows speedup results with multithreaded applications along with the average improvements. Only 9 applications are shown to reduce clutter in the plots, however, the average (arithmetic mean) numbers are computed across the entire suite.

(A) Speedup when replacing an SRAM with hypothetical cache: 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.

(B) Speedup when replacing an SRAM with 4MB, $10years$ **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 (see Table 4) and thus, they benefit from not only the 4x capacity increase of STT-RAM, but also from the presence of a L2 cache write buffer. On average, we find 6% improvement in speedup over S-1MB 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 M-4MB. This is because, the write request to L2 cache banks in



(a) PARSEC 2.1 (b) SPEC 2006

Figure 5: Performance of applications normalized to that of S-1MB

these two applications are staggered in time and an 10-entry write buffer proves to be sufficient to hide the long write latencies.

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 average 11% degradation) because of the high write latency of STT-RAM. Overall, when averaged across the entire suite, a traditional $10years$ STT-RAM gives a minimal 5% speedup improvement over S-1MB. However, a $10years$ STT-RAM cache organization has 14% lower performance when compared to the hypothetical design S-4MB and with write-intensive applications. This is the gap that our proposal seeks to bridge by tuning the retention time.

(C) 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 $10years$ 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 $10years$ retention time STT-RAM cache organization. On 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 hypothetical case.

(D) 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 $10ms$. After $10ms$, this STT-RAM device will lose its data and hence to keep the integrity of the data a large number of write-backs are forced from the LLC to the main-memory before the actual expiration happens.

(E) Speedup when replacing a SRAM with 4MB, $10ms$ retention time STT-RAM with refreshing (Revived-M-4MB(10ms)): This is our proposed scheme, which incorporates refreshing of dirty blocks beyond the $10ms$ retention time. Our scheme significantly improves performance when compared to all realistic design scenarios evaluated. On average, the proposed revived scheme is better than the conventional SRAM design (S-1MB) by 18%, traditional $10years$ 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. The Revived-M-4MB(10ms) scheme is closest to the hypothetical S-4MB case and is within 5% of it, showing the benefits of our scheme in making the STT-RAM device a choice for universal memory.

(F) Analysis with SPEC 2006: In general, the observations with SPEC applications are consistent with those made with PARSEC 2.1 applications. Figure 5(b) shows Instruction Throughput (IT) and Weighted Speedup (WS) with the multiprogrammed mixes (supplemental Table 5). Simply replacing a SRAM bank with 4MB STT-RAM would lead to 11% degradation in IT, and 4% degradation in WS. 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 write intensive mix (`bzip2`, `gcc`, `lbm` and `hmmer`) this improvement can be as high as 35% over the baseline SRAM design. WS also follows a similar trend as WS 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 hypothetical device (S-4MB) in terms of IT (WS).

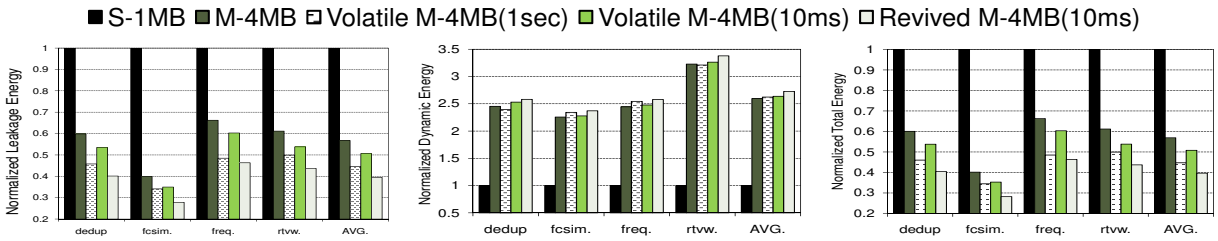
6.2 Energy Usage Comparison

Figure 6 shows normalized leakage, dynamic energy (reads + writes), and total energy for a subset of applications. While computing the energy numbers, we take into account the overheads of our proposed cache block refreshing schemes. We observe that on 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. This is mainly attributed to the fact that fraction of dynamic energy to the total energy is not significant because of very high leakage energy and fast switching times of cores. On average, we find 11% energy benefits of using revived M-4MB design over Volatile M-4MB(1sec) and 18% improvement over the baseline STT-RAM design. We observe similar benefits with SPEC 2006 applications, but for the sake of brevity we are not showing them.

7. PRIOR WORK

The work that is most closely related to ours is [15]. In this, the authors relax the retention time of STT-RAM from $10years$ to $56\mu s$ by reducing the planar area of MTJ. Our application driven analysis shows that the ideal retention time of LLC should be in the range of ms. Recently published work related to STT-RAM [7, 11] uses state-of-the-art MTJ designs in the range of 2-3F^2 which do not give the leeway of reducing the retention time by aggressively reducing the MTJ planar area, we reduce the retention time by lowering the saturation magnetization and the thickness of the free layer. Also, our proposed refresh scheme is simple, yet very performance and energy efficient compared to the DRAM-style refreshing proposed in [15] (also see Sec. S.9)

A very recent effort [17] explores the possibility of designing



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

Figure 6: Energy of applications normalized to that of S-1MB

LLC with STT-RAM banks of varying retention times and moving dying blocks from lowest retention time (μs) bank to the higher ones [17]. In contrast to this work that populates STT-RAM cells of different retention times, we tune our design for single retention times across the memory hierarchy. Our approach eases the challenge of higher die costs and yield issues associated with heterogeneous retention cells and irregular structures. Further, the μs retention times used in the prior effort are challenging to achieve in newer, scaled MTJ dimensions as indicated earlier.

8. CONCLUSIONS

Spin-Transfer Torque RAM (STT-RAM) is a promising candidate for future multi-core general purpose and embedded systems, 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 lifetime of LLC with the intention of using this time as the optimal retention time for the STT-RAM. 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. The results not only indicate that it is possible to get up to 18% improvement in speedup for PARSEC applications and 60% reduction in total energy consumption over S-1MB design, but also show that our proposed design can be within 5% of the hypothetical case (S-4MB) 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 to be a better solution for designing a performance and power efficient memory hierarchy for multi-core systems.

9. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their reviews and comments towards improving this paper. This work is supported in part by NSF grants CNS-0721479, CNS-1152449, CCF-1147388, CCF-0903432 and DoE grant DE-SC0005026.

10. REFERENCES

[1] C. Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, 2011.

[2] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt. The M5 Simulator: Modeling Networked Systems. *Micro, IEEE*, 26(4):52 –60, 2006.

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

[4] 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 2007*.

[5] X. Dong, X. Wu, G. Sun, Y. Xie, H. H. Li, and Y. Chen. Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement. In *DAC*, 2008.

[6] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Non-Volatile Memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012*.

[7] A. Driskill-Smith. Latest Advances in STT-RAM. In *2nd Annual NVM Workshop*, 2011.

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

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

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

[11] 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 *IEDM*, 2009.

[12] 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.

[13] A. Nigam, C. Smullen, V. Mohan, E. Chen, S. Gurumurthi, and M. Stan. Delivering on the promise of universal memory for spin-transfer torque RAM (STT-RAM). In *ISLPED*, 2011.

[14] 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 *IEDM*, 2009.

[15] C. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. Stan. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *HPCA*, 2011.

[16] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen. A novel architecture of the 3D stacked MRAM L2 cache for CMPs. In *HPCA*, 2009.

[17] Z. Sun, X. Bi, H. H. Li, W.-F. Wong, Z.-L. Ong, X. Zhu, and W. Wu. Multi retention level STT-RAM cache designs with a dynamic refresh scheme. In *MICRO*, 2011.

[18] C. Xu, D. Niu, X. Zhu, S. Kang, M. Nowak, and Y. Xie. Device-architecture co-optimization of STT-RAM based memory for low power embedded systems. In *ICCAD*, 2011.

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

SUPPLEMENTAL

In this supplementary section, we present additional details on STT-RAM modeling, application properties and refresh schemes. This section is organized as follows: Sec. S.1 presents the preliminaries on STT-RAM. Sec. S.2 and Sec. S.3 discusses STT-RAM modeling details and simulation results, respectively. Latency and energy numbers of our STT-RAM and SRAM designs are tabulated in Table 3. Sec. S.4 extends the discussion of revival time distribution, started in Sec. 3, for SPEC 2006 applications. Sec. S.5 describes the importance of performing a state check before copying the data back to the buffer. The read and write percentages of our applications taken from PARSEC 2.1 and SPEC 2006 suites are detailed in Sec. S.6 and Table 4. Sec. S.7 describes the evaluation metrics and methodology of collection of results. The sensitivity results to find the optimal number of MRU slots and buffer size are shown in Sec. S.8. In Sec. S.9 we compare our refreshing scheme (Revive) with DRAM-style refresh presented in [15]. Sec. S.10 describes how our refreshing scheme saves number of write backs leading to performance improvements. Finally, we conclude our supplementary section with comparisons to additional prior work (Sec S.11) and concluding remarks (Sec S.12).

S.1 Preliminaries on STT-RAM

STT-RAM uses an 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 7, an MTJ contains two ferromagnetic layers. One ferromagnetic layer has a fixed magnetization direction and called the reference layer. The second layer's magnetic direction can be changed by passing write current, and, thus it is called the free layer. The relative magnetization direction of two ferromagnetic layers determines the resistance of the 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. The current amplitude required to reverse the direction of the free ferromagnetic layer is determined by the size, the aspect ratio of MTJ, and the write pulse duration [4, 11].

S.2 STT-RAM Modeling

In this section, we first introduce models to determine the area of an STT-RAM cell. Typically, the 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 7, 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 [R3] is simulated. The driving current is obtained by assuming typical TMR (120%) and Low Resistance State (LRS) ($3k\Omega$) value [11] and wordline voltage to be 1.5V (the optimal value is extracted from [19]). Further, we oversize the access transistor width to guarantee enough write current is provided to MTJ using the methodology discussed in [R2]. To achieve high cell density, we model the STT-RAM cell area by referring to the DRAM design rules [8]. 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.

Thermal Barrier vs. Retention Time As described in Sec. 2.2, the retention time of an MTJ is largely determined by the *thermal stability* of the MTJ. The relationship between retention time (log scale) and thermal barrier is shown in Figure 8. We observe that MTJ with higher retention time has higher thermal barrier. For higher temperature, thermal barrier decreases at a faster rate, leading to faster reduction in retention time. Since there is a strong dependency of retention time of MTJ on the operating temperature, our implementation assumes the worst-case temperature of 125 °C to avoid any possibility of early expiration of block because of sudden reduction in STT-RAM retention time.

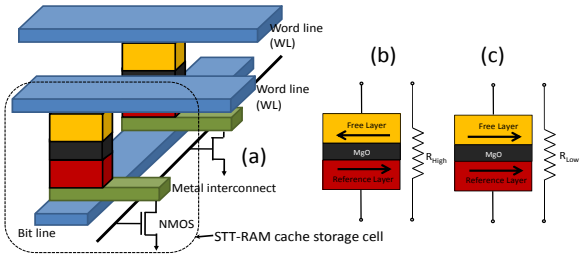


Figure 7: **(a) Structural view of an STT-RAM Cache Cell (b) Anti-Parallel High Resistance, Indicating "0" state (c) Parallel Low Resistance, Indicating "1" state**

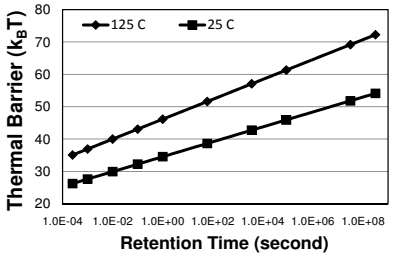


Figure 8: **MTJ thermal barrier for different retention times**

S.3 STT-RAM Simulation Results

Table 3 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' (shown in Figure 2) 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 [5]. For the leakage simulation, we didn't apply any power gating techniques for the cache banks and hence our leakage numbers are higher than the previously published numbers in recent STT-RAM papers. We observe that STT-RAM consumes almost half of the leakage power than that of SRAM. That is basically because of the fact that, half of STT-RAM die area is occupied by peripheral circuitry, which in turn means that half of the chip is leaky. The switching energy per STT-RAM cell is < 1pJ [7] and only half of the total write energy is consumed on switching the cells. For SRAM numbers, we use balance L2 design for leakage, density and performance (already implemented in CACTI).

S.4 Distribution for SPEC 2006 applications

As described in Sec. 3, the revival time is defined as the time interval between two successive writes. Figure 9 shows the distribution of L2 cache blocks having different revival time intervals. These results are obtained by running multiprogrammed applications on the M5 Simulator [2]. We observe that these results are similar to that of PARSEC 2.1 applications, and we draw similar conclusions as mentioned in Sec. 3.

Table 3: 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

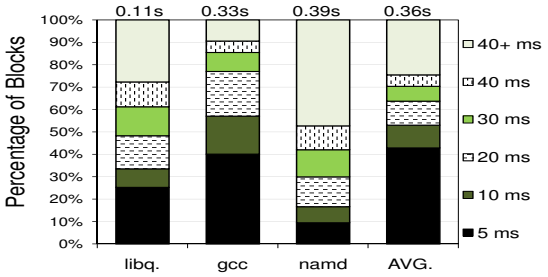


Figure 9: Distribution of blocks showing different revival times (value on the top of a bar show the maximum revival time for that distribution)

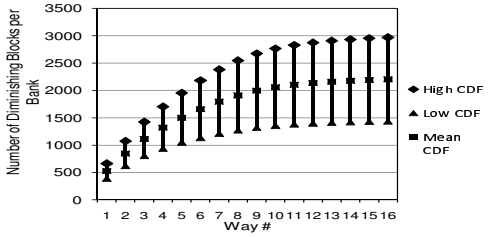


Figure 10: 95% Confidence Intervals of diminishing blocks for each way

S.5 Buffer Architecture

As mentioned in Sec. 4, key idea of our proposal is to copy $important\ diminishing$ blocks to a temporary buffer and immediately copy it back to the cache. Since buffering process for a block starts off in the penultimate state of the counter and not in the final state as usually would have happened in a typical counter [9], the data is still valid for incoming reads in the caches. This scheme cuts the overhead of searching in the buffer. Moreover, it is important to note that incoming writes during buffering process will put back the state of the cache block to S_0 making the corresponding buffer data stale, and hence it is necessary to perform a state check before copying the buffer data back.

S.6 Application Properties

Table 4 shows the properties of various emerging applications. *Read%* denotes the percentage of reads to the L2 cache out of the total L2 accesses and *Write%* denotes the percentage of writes to the L2 cache out of the total L2 accesses. *Intensity* denotes Read/Write intensity based on read%/write%.

S.7 Evaluation Metrics

In this section, we describe our evaluation metrics and methodology for collection of results. For the multithreaded 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 chip (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 other cores (Eq.(6)). For analyzing the energy behavior, we measure the leakage energy, dynamic energy and total energy for all designs.

(5) $Instruction\ throughput = \sum_i IPC_i$

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

Collection of Results:

We report results of 12 multithreaded applications(Table 4) and 14 multiprogrammed mixes (Table 5). We selected these multiprogrammed mixes from read and write intensive categories (Table 4) to get varying percentages of reads and writes. We use $simsmall$ input for mulithreaded applications and report the results of only Region of Interest, (ROI) (except for 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 multiprogrammed mixes, we fast forward 1B instructions, warm up caches for 500M instructions and then report results for 1B instructions.

S.8 Sensitivity Analysis

(A) 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 lead to more buffer overflows. Increasing the buffer size, leads to fewer buffer overflows, but this 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 Sec. 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 10. 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 11(a) shows the normalized speedup to 1300 blocks, with a subset of applications by varying the number of buffer entries. on 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 (resulting in a 3% area overhead) with the best possible performance per area-overhead.

(B) Sensitivity to number of MRU slots:

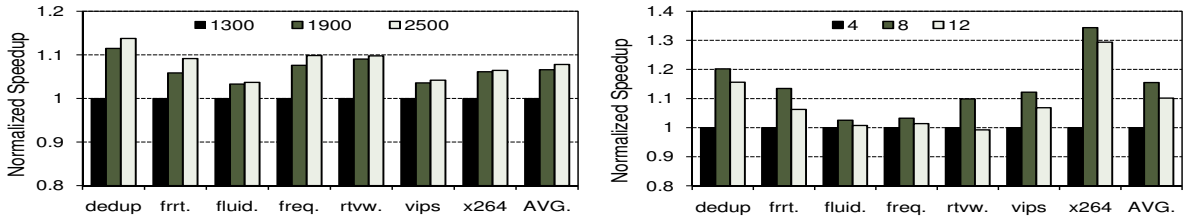
In order to calculate the optimal MRU slots for buffering, we collected statistics of MRU positions of diminishing blocks. Figure 10 shows the average cumulative distribution (mean CDF) of diminishing blocks per bank as a function of the number of ways in a set for applications. We observe that the number 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

Table 4: Application characteristics of PARSEC 2.1 and SPEC 2006 applications

#	PARSEC 2.1	Read%	Write%	Intensity	#	SPEC 2006	Read%	Write%	Intensity
1	blackscholes	91.9	8.1	Read	13	bzip2	86.2	13.8	Read
2	bodytrack (btrack.)	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 5: SPEC 2006 multiprogrammed mixes

Mixes	Applications
mix-1	mcf, leslie3d, bzip2, gcc
mix-2	mcf, gcc, bzip2, namd
mix-3	hmmer, sjeng, gcc, bzip2
mix-4	bzip2, lbm, hmmer, gcc
mix-5	gcc, GemsFDTD, omnetpp, bzip2
mix-6	bzip2, gcc, omnetpp, h264ref
mix-7	h264ref, bzip2, leslie3d, gcc
mix-8	gcc, sjeng, mcf, bzip2
mix-9	leslie3d, gcc, bzip2, hmmer
mix-10	sjeng, omnetpp, gcc, bzip2
mix-11	bzip2, gcc, mcf, hmmer
mix-12	bzip2, leslie3d, gcc, sjeng
mix-13	gcc, hmmer, GemsFDTD, bzip2
mix-14	namd, bzip2, h264ref, gcc



(a) Buffer Entries (b) MRU Slots.

Figure 11: Speedup as a function of number of buffer entries and MRU Slots

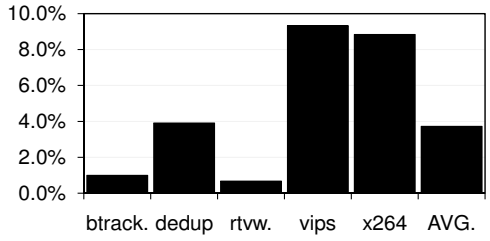


Figure 12: Energy impacts of Revive refresh scheme as a percentage of DRAM-style refresh

for the buffer size.

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 11(b) shows speedup of a subset of applications along with the average across all mulithreaded applications, with varying number of MRU slots. Buffer size is kept constant at 1900 per bank. We see degradation in performance when we decrease the number of 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.

S.9 Comparison to DRAM-style refresh

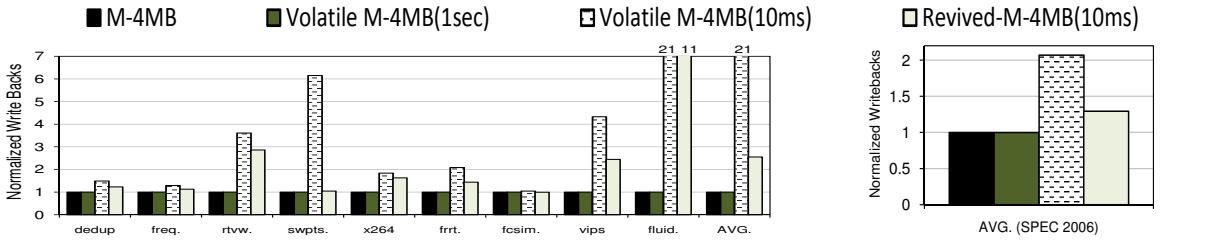
In this section, we compare our scheme, Revive, with the DRAM style in-place refresh scheme proposed in [15]. DRAM style in-place refresh unnecessarily refreshes every block in the LLC after every 10ms, leading to high refresh overheads. In our selective buffering scheme we try to minimize the number of refreshes required. Figure 12 shows the energy comparisons of our revive refresh scheme as a percentage of DRAM-style refresh scheme proposed in [15]. Since, our scheme only refreshes a small number of required blocks as opposed to all the LLC blocks, on average, it consumes only 4% of energy compared to the DRAM-style refresh. For the same reason, our scheme results in less than 90% of performance penalty taking into our observation that about 10% of the total diminishing blocks need to be written back to the main memory. Although our scheme has 4% buffer area overhead, the significant refreshing benefits obtained justifies our design. Moreover, refreshing benefits will diminish radically if it is done at μs level [15], and hence our ms proposal helps in sustaining the energy benefits.

S.10 Boosting Performance by saving number of write backs

In this section, we describe how reducing the number of write backs can boost the performance. Since our scheme proposes to retain the useful blocks (not writing back) in the cache after their expiration time by copying them to a temporary buffer and copying it back, it is important to see how many write backs our scheme is able to save. Figure 13 shows the number of write backs of all the designs normalized to M-4MB. We observe that the 4MB, $10ms$ retention time STT-RAM design, on average, has 21x more write backs than the traditional STT-RAM design. This leads to significant performance degradation across most applications when compared to simply using a $10years$ 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.

S.11 Additional Prior Work

Sun et al. [16] 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 la-



(a) PARSEC 2.1 applications (b) Average across multiprogrammed mixes

Figure 13: **Number of write backs normalized to M-4MB**

tency 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 [16]) write-buffer at every STT-RAM bank and our results would be significantly better without the presence of write-buffers.

In a recent work [12], 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 [12] 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. [16]. Contrasting this to our work, our scheme provides about 15%/4%(PARSEC IPC/SPEC weighted-speedup) improvement over $10years$ traditional STT-RAM, on top of the write buffering scheme, thereby making it more attractive compared to [12]. Overall, we believe that no prior work makes a case for tuning the retention time of STT-RAM banks based on profiling retention duration of LLC blocks of applications, which our proposal does.

The 3T1D designs proposed in [R1] has typical worst-case retention time in μs region, which makes it incompatible for our LLC design that needs ms retention time. Increasing the retention time of 3T1D cell to ms region will enlarge the size of the gated-diode or will increase the threshold voltage of the access transistor. These choices will incur significant area overhead or performance degradation for our cache design, respectively.

In regard to eDRAM, it also has similar density advantages but has at least 2x higher leakage than STT-RAM. Moreover, eDRAM is considered to have scalability challenges in sub-45nm process nodes due to the difficulty of precise charge placement and data sensing. STT-RAM is believed to at least scale beyond 10nm technology [7]. The retention time of eDRAM is in microseconds making it unsuitable for our design. (We claim the retention time to be in milliseconds). Also, the refresh energy overheads of eDRAM is much higher.

A 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 [19], hybrid SRAM/STT-RAM architecture [16] and read-preemptive write-buffer designs have been shown to mitigate write latency/energy. The circuit level techniques such as eliminating redundant bit-writes [19] and data inverting technique [16] 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 [19] 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.

Unlike recent STT-RAM papers that assume that the non-active STT-RAM banks are power gated, we conservatively assume that these banks leak. Consequently, our absolute leakage numbers are larger.

S.12 Concluding Remarks

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, the latency and energy overhead associated with write operations are major obstacles for being competitive with the SRAM. Our study showed 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. Thus, we proposed to trade-off the non-volatility (data-retention time) of STT-RAM for better write performance and energy for designing STT-RAM-based L2 cache. The paper addressed several critical design issues such as how we decide on 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 volatile STT-RAM. We studied 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 and energy gains, but required a data refreshing mechanism. For the aggressive $10ms$ retention time design, we proposed a selective block refreshing scheme for the cache blocks that have a higher refresh time than the STT-RAM retention time 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 indicated that not only we can eliminate the long write latency overhead of the NVM STT-RAM, but also can provide on an average 18% improvement in performance compared to the traditional SRAM-based design, while reducing the energy consumption by 60%.

S. REFERENCES

[R1] L. Xiaoyao, C. Ramon, W. Gu-Yeon, and B. David. Replacing 6T SRAMs with 3T1D DRAMs in the L1 Data Cache to Combat Process Variability. *IEEE Micro 2008*.

[R2] 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 TVLSI*, 2011.

[R3] W. Zhao and Y. Cao. New generation of predictive technology model for sub-45 nm early design exploration. *IEEE TED*, 2006.