IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON EMERGING TRENDS, ISSUES, AND CHALLENGES IN ENERGY-EFFICIENT CLOUD COMPUTING

Received January 27, 2018, accepted February 26, 2018, date of publication March 8, 2018, date of current version April 4, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2813668

Read-Tuned STT-RAM and eDRAM Cache Hierarchies for Throughput and Energy Optimization

NAVID KHOSHAVI[1], (Member, IEEE), AND RONALD F. DEMARA[2], (Senior Member, IEEE)
[1]Department of Computer Science, Florida Polytechnic University, Lakeland, FL 33805, USA
[2]Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816, USA

Corresponding author: Navid Khoshavi (nkhoshavinajafabadi@floridapoly.edu)

ABSTRACT As capacity and complexity of on-chip cache memory hierarchy increases, the service cost to the critical loads from last level cache (LLC), which are frequently repeated, has become a major concern. The processor may stall for a considerable interval while waiting to access the data stored in the cache blocks in LLC, if there are no independent instructions to execute. To provide accelerated service to the critical loads requests from LLC, this paper concentrates on leveraging the additional capacity offered by replacing SRAM-based L2 with spin-transfer torque random access memory (STT-MRAM) to accommodate frequently accessed cache blocks in exclusive read mode in favor of reducing the overall read service time. Our proposed technique improves the temporal locality while preventing cache thrashing via sufficient accommodation of the frequently read reused fraction of working set that may exhibit distant re-reference interval in L2. Our experimental results show that the proposed technique can reduce the L2 read miss ratio by 51.7% on average compared to conventional STT-MRAM L2 design across PARSEC and SPEC2006 workloads while significantly decreasing the L2 dynamic energy consumption.

INDEX TERMS Non-volatile memory, STT-MRAM retention relaxation, last level cache, energy overhead reduction, read service time, critical loads.

I. INTRODUCTION

The increasing bandwidth demand of current memory-intensive applications incurs significant data movement that negatively impacts off-chip bandwidth, on-chip memory access latency, and energy consumption [1]–[4]. To reduce data transfer between on-chip and off-chip memory components, commercial multi-core systems utilize multi-level cache methodology [5]–[8] whereby fast, low-capacity, and high leakage power SRAM arrays are employed in the upper-levels of cache, i.e. L1 and L2, while large, low leakage power and high refresh demand eDRAM is placed in LLC. The employment of relatively spacious SRAM arrays as L2 cache design in the middle of cache hierarchy results in two major challenges: 1) *leakage*: the high leakage power characteristic of SRAM cells results in excessive power budget [9], and 2) *area*: capacity constraints induced by the SRAM cell footprint prevent favorable residency of the working set to reside close to the active core [10], [11]. The negative effect of aforementioned issues is exacerbated with the high dynamic power dissipation incurred to drive on-chip interconnects while exchanging data [12], [13].

While other recent works have made significant advancement to optimize eDRAM LLC (L3 in this work) [14], in this paper we concentrate on applying new insights regarding working set behavior to optimize L2 cache using a heterogeneous STT-MRAM. STT-MRAM offers a promising alternative solution to take the place of area-inefficient and high leakage power SRAM technology. STT-MRAM is well-known for its non-volatility and near-zero standby power while offering at least 3x to 4x area savings compared to SRAM [15].

This work is an initiative study to answer how much performance gain and energy reduction can be achieved by replacing the conventional SRAM-based L2 with STT-MRAM. To maximize the benefit of extra capacity realized by this replacement, we have proposed a novel technique called *Read Reference Activity Persistent (RRAP)* strategy that accelerates the service to critical requests while also efficiently manages regular L2 cache requests. Essentially, the loads and stores exhibit different levels of criticality in the processor. While the stores requests can be postponed through buffering before cache or memory commitment, the loads requests demand

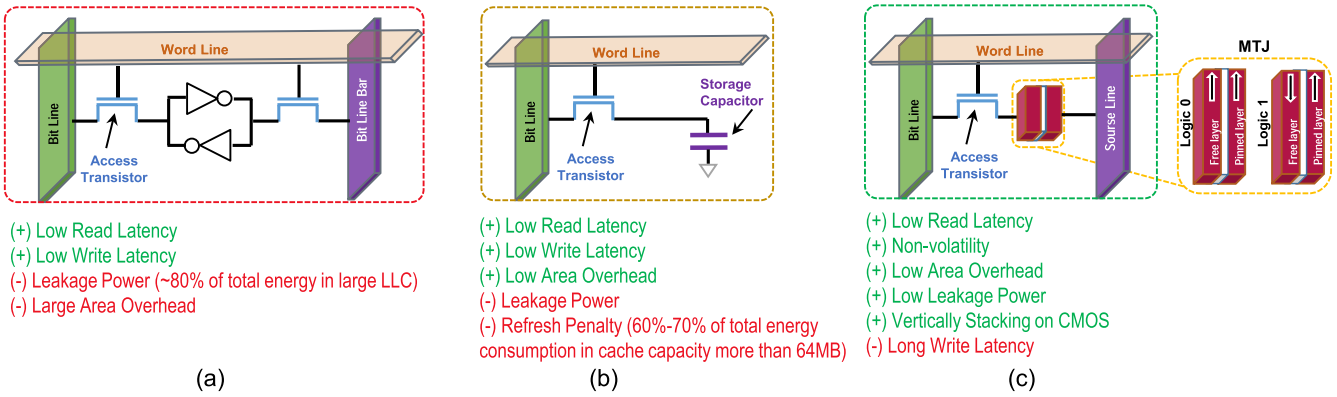

FIGURE 1. LLC organization based on (a) SRAM, (b) eDRAM, (c) STT-MRAM.

immediate treatment to maintain the stalls within an acceptable delay period in pipelined processors [16].

However, the cache memory shortage to maintain continuously expanding working sets close to the core causes the performance degradation or increased miss ratio. This degradation is exacerbated by introducing more cache levels with larger LLC capacity. For instance, the transfer latency for a cache block from L2 to L1 in hyper-threaded Pentium IV is 18 cycles, while this latency goes up to 360 cycles for data movement between main memory and L2 [17]. On the other hand, the existing cache replacement policies are ineffective to deal with continuously growing large memory-intensive workloads that typically are greater than available cache size [18]. In particular, a fraction of working set that exhibit distant re-reference interval may repeatedly be evicted and be brought back from/to L2 prior to contributing sufficient hits to L2.

Thus, the processor may stall for a long time to access the data stored in the cache blocks in LLC if there are no independent instructions to execute.

To provide accelerated service to the critical loads requests from LLC, RRAP technique targets Extensive Read Reused Access (ERRA) blocks that remain unchanged to be brought from LLC to hybrid L2 cache design in favor of reducing read miss ratio in L2, which in turn causes the reduction of overall read service time. In addition, RRAP improves the temporal locality of frequently reused blocks which may exhibit distant re-reference interval. In particular, it accommodates them by providing a sufficient residency interval in L2 to enable them for maximizing the hit contribution to L2 while the required energy for preserving them is trivial.

To accomplish this, RRAP initially employs a strategy to exploit the non-volatility of STT-MRAM for storing reused lines by read operations. Second, it takes advantage of retention-relaxed STT-MRAM having improved write performance to manage regular cache requests while maintaining the high cache utilization offered by SRAM. Compared to the existing works, this paper provides the following contributions:

- new insights regarding the distribution of read reused cache blocks within cache for PARSEC ans SPEC2006 suite benchmarks,
- a low-conflict in-situ monitoring mechanism to track LLC traffic and autonomously copying the exclusive read reused lines to upper-level High Retention STT-MRAM Cache (HRSC) of each core,
- improve the overall read service time by accelerating the service to the critical loads versus stores,
- identify preferred designs for Low Retention STT-MRAM Cache (LRSC) configuration by comprehensively evaluating the candidates in terms of energy consumption and performance delivery,
- optimization strategies for balancing workload lifetime performance versus energy reduction from diminished write latency, and
- propose *Reclusive*, an efficient cache configuration scheme to maximize RRAP throughput.

The remainder of the paper is organized as follows: The technology trends for cache organization are introduced in Section II. Section III identifies the motivation behind the RRAP strategy. Section IV presents the technical approach and the details of the proposed method. We summarize RRAP experimental results in Section V. The related works are discussed in Section VI. Section VII concludes the paper.

II. BACKGROUND ON TECHNOLOGY TRENDS FOR CACHE ORGANIZATION

The deployment of large SRAM in favor of the performance improvement comes with the cost of increased energy consumption and high area overhead. To address this issue, the researchers started to look into the alternative solutions which offer the higher energy-efficient, cost-effective, large-capacity, and competitive read/write operation speed compared to the traditional SRAM arrays as illustrated in Fig. 1.

The embedded DRAM (eDRAM) is one of the promising solutions pioneered in the design of on-chip memory hierarchy. As depicted in Fig. 1 (b), the eDRAM cells are arranged in a two-dimensional array whereby each storage capacitor

is connected to the *bitline* wire through the access transistor. Even though the employment of capacitor for maintaining the logic value has significantly saved the area compared to the SRAM utilizing 6 transistors per bit cell, the eDRAM technology suffers from high dynamic energy consumption due to mandatory periodic refresh required to keep the stored value in the valid state [19]. It has been reported in [20] that refresh scheme contributes around 70% to the overall energy consumption in LLC. Furthermore, the periodic refresh operation limits the CPU accesses to the eDRAM arrays which leads to typically deployment of the eDRAM as LLC by the memory designers.

Another promising alternative solution to replace with SRAM is STT-MRAM. The non-volatility and near-zero leakage power are two prominent characteristics of STT-MRAM cell. Furthermore, the STT-MRAM offers at least 3x to 4x area savings compared to the SRAM [21]. As illustrated in Fig. 1 (c), the STT-MRAM cache leverages an extra device to drive large current for a certain period when the write operation is issued. The high write current and slow write operation of STT-MRAM has motivated the researchers to devise the novel architectures to overcome these new challenges while developing STT-MRAM based L2 between L1 and LLC [15], [22]. Herein, we address all of the above issues through adjustment of the intrinsic non-volatility characteristics of STT-MRAM to achieve SRAM-competitive performance while incurring low energy consumption.

III. MOTIVATION

We classified the motivation behind our technique into two subsections. Each subsection discusses orthogonal mechanisms which cooperate in the proposed technique. These strategies exhibit a cause-and-effect relationship which can be exploited from two perspectives of *memory reference to shared LLC* and *clean victim blocks eviction in private L2*.

A. ANALYSIS OF MEMORY ACCESS PATTERN TO SHARED LLC

The cache lines, which are brought into a shared LLC, can be classified into two categories, namely, *non-reused* and *reused* cache lines. A non-reused cache line in LLC does not experience more than one read access, while a reused cache line is accessed from multiple cores during its residency in the LLC. Fig. 2 illustrates the distribution of read accesses to the cache line, in which the majority of the LLC lines are non-reused cache lines, as indicated by the leftmost blue column for each benchmark which averages 50.1% across the benchmark suite.[1]

Although the majority of the LLC line read accesses are non-reused cache lines, this portion of cache space is accessed only once for read operation purpose during program execution which results in spending a small fraction of time to read from the non-reused cache lines. On the other hand, reused cache lines with high read access rate, e.g. read more

[1]The experimental setup is explained in Section V

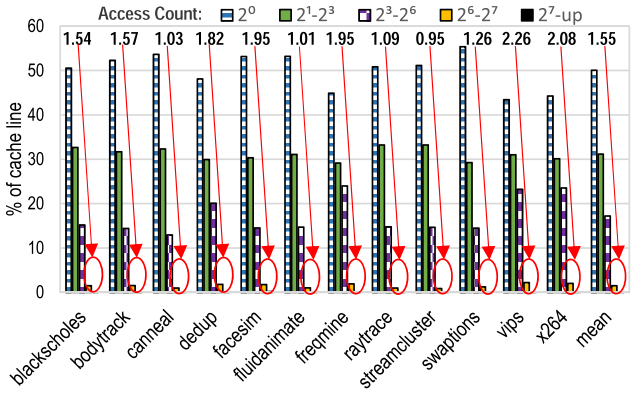

FIGURE 2. Percentage of total cache lines experiencing read operation.

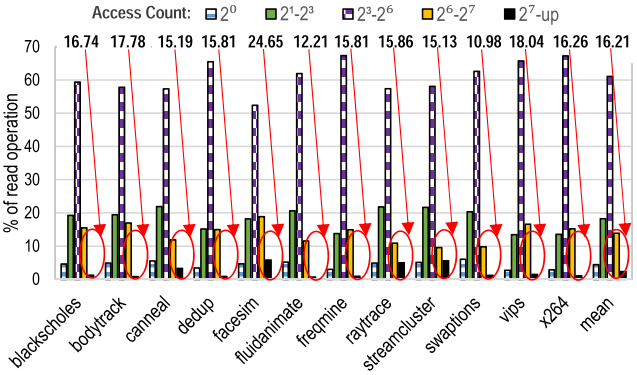

FIGURE 3. Proportion of read reference activity.

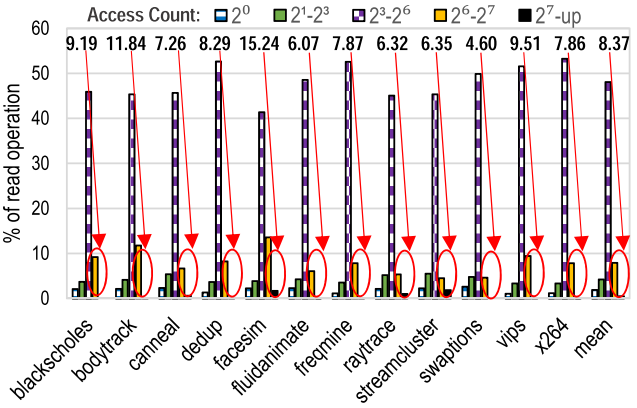

FIGURE 4. Proportion of exclusive-read reference activity.

than 64 times, have experienced a significant amount of program read operations, e.g. 16.2% on average, while they only occupy 1.55% of the entire cache space as depicted in Fig. 2 and 3.

These reused cache lines can experience either *read-only access* or *multifaceted access,* e.g. write, insert, evict. The exclusive-read reused cache lines, whose read accesses are more than 64, account for 8.37% of the total read accesses as shown in Fig. 4. Herein, we refer to this group of cache lines as Extensive Read Reused Access (ERRA) blocks that remain unchanged during program runtime. This observation

motivated us to re-design the conventional cache design in favor of ERRA cache blocks to reduce the response time to read requests from LLC which in turn cause overall IPC improvement. Moreover, our goal align with the recent works that attempt to prioritize the service to performance critical read reused cache blocks [16], [23] for achieving a better performance.

The ERRA blocks are excellent choices to be stored into high retention STT-MRAM arrays because 1) these lines are written once while experiencing read-only operations for the remaining of program execution, 2) the energy and performance overheads for accessing ERRA blocks are small because once the ERRA block is brought to the high retention STT-MRAM array, it only will be accessed by the read operations prior to its eviction.

The characteristics of high versus low retention memory arrays are identified in the following Section.

B. ANALYSIS OF CLEAN VICTIM EVICTION IN PRIVATE L2

Each LLC access (either hit or miss) is followed by finding the victim block in L2 to be replaced by the accessed LLC block. Based on non-inclusive property [24], the clean victim blocks are silently dropped from L2 during replacement process while the dirty victims are inserted into their replica blocks in LLC. Yet, the replica block may not exist in LLC for non-inclusive design which necessitates either: 1) eviction of a victim block from LLC, and 2) insertion of a dirty victim from L2 into LLC.

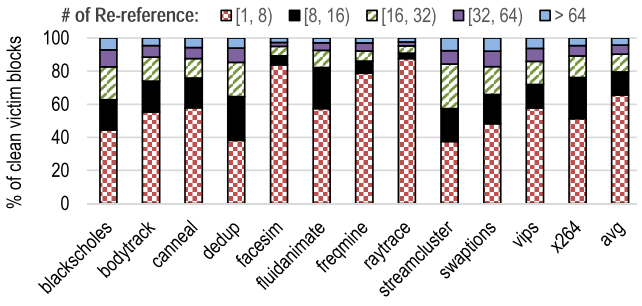

FIGURE 5. Proportion of the frequently re-referenced clean victim blocks after their eviction from L2.

In our study, we focus on improving the temporal locality of a fraction of the working set which contributes significant read hits to L2. To be specific, we investigate the reference pattern to the blocks whose residency in L2 reduces the L2 read miss ratio over time. However, these blocks are evicted from L2 before contributing acceptable L2 read hits due to weak temporal locality, ineffective replacement policy, or insufficient capacity to retain the referenced fraction of working set [18]. The consequence of this effect is that whenever there is a read miss on aforementioned blocks in L2, a significant delay and energy consumption would be imposed to the system to bring the referenced block from LLC or memory to L2 while this block may not be re-referenced again during its residency in L2. Fig. 5 illustrates the proportion of the frequently re-referenced clean victim blocks after their eviction from L2 for the baseline 512KB 8-way L2 cache. For example, around 20.4% of the evicted clean victim blocks from L2 are re-referenced more than 16 times on average during the program execution. This means that whenever the processor calls for one of the evicted clean victim blocks from L2, it must stall until the requested block be transferred from either LLC or memory to L2. If we add up the processor stalls for the frequently re-referenced clean victim blocks, it incurs significant energy consumption and delay overhead which increasingly sabotage the throughput of the system and aggravate the considered budget for the energy.

RRAP tackles this problem through the sufficient accommodation of frequently read accessed fraction of working set in L2 to accelerate the service to these critical loads and improve the read hits to L2.

IV. TECHNICAL APPROACH

The intuition behind RRAP is to benefit from the extra capacity provided by replacing SRAM-based L2 with STT-MRAM to accommodate the replica of ERRA cache lines from LLC. The proposed schematic view is illustrated in Fig. 6 and referred as Read Reference Activity Persistent (RRAP) cache design, in which the service to frequently-read cache lines of LLC is accelerated while energy consumption is significantly amortized due to near-zero standby power property of STT-MRAM arrays. To achieve this goal, the Reclusive cache allocation policy, inspired by FLEXclusive [25], is adopted in RRAP which will be presented in Section 4.2. During the program execution, the replica cache block in upper-level of cache is frequently accessed, while the duplicate copy of block in LLC remains unused. By reducing the access to the duplicate block in LLC, this cache block will be eventually selected as a candidate for eviction due to the replacement policy. However, the eviction of this cache block must not back-invalidate the replica cache block in the upper-level of cache. Accordingly, the read service time is not impacted by cache replacement policy in LLC.

In order to effectively manage regular L2 request, the non-volatility of STT-MRAM needs to be relaxed through modulating write pulse width and write current. By replacing SRAM with LRSC, the reduced die area provides additional capacity which can accommodate LLC frequent read reference lines with no extra energy consumption for data preservation. We have allocated this extra space to HRSC which retains the data for *10 years*. The combination of LRSC and HRSC in an L2 structure not only provides energy benefit, but also improves performance as will be shown herein.

A. READ REFERENCE ACTIVITY PERSISTENT (RRAP) CACHE HIERARCHY

Since the non-inclusive policy is adopted in RRAP, the cache blocks are brought into both the LRSC and LLC upon an LLC miss.

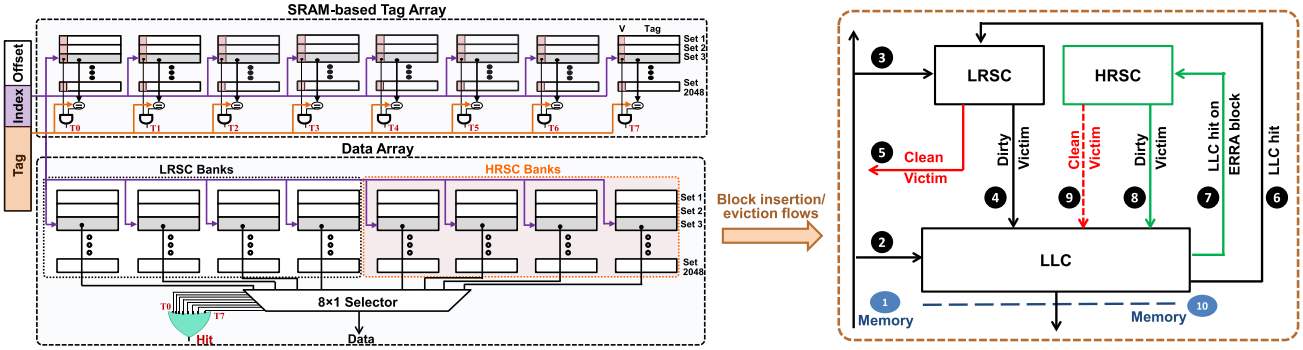

FIGURE 6. RRAP scheme: (a) Heterogeneous split cache architecture,(b) Block insertion/eviction flows.

on a read miss on L1, the associated tag arrays in LRSC and HRSC are simultaneously accessed to look for the missed data block in L2. Even though the parallel search for the missed data block incurs more energy consumption due to enabling two exclusive resources, the overall system performance remains unchanged. These include decoder, tag arrays and wordlines. The energy consumption overhead for these resources has been considered in our simulation results.

To achieve this, the ratio of read and write accesses to LLC cache blocks are monitored through read and write counters. We assign a 6-bit Read Counter (RC) and a 1-bit Write Counter (WC) to each LLC cache block, as justified below. The RC of a cache block in LLC records the read access history to that particular block. This history is utilized to determine if the cache block is a proper candidate to be copied into HRSC. We conducted an extensive exploration to evaluate the preferred value for the read threshold level, NR_{th}, within our design. We found that when NR_{th} is small, the ratio of blocks that must be copied into HRSC significantly increases. However, note that the limited capacity of HRSC constrains the number of read-intensive cache blocks that can be maintained. This results in a high ratio of cache blocks to be frequently replaced without providing adequate read services which in turn incurs significant write overhead and undermines the read-friendly property of HRSC. On the other hand, if NR_{th} is too large, then the HRSC utilization significantly decreases because only a few read-intensive cache blocks are selected to be brought into HRSC. Based on our experimental results, NR_{th} equal to *64* maximizes the HRSC utilization while incurring an acceptable cache block replacement rate in HRSC.

The WC determines whether this block has been accessed by a write operation prior to the time that the cache block has reached to NR_{th} or not. If an LLC line experiences any write operation while resident, it is copied into LRSC upon a read hit on LLC, even if the RC is saturated. Considering 7-bit per line of LLC imposes an acceptable 7-bit/64 Bytes = 1.3% area overhead.

Assuming the HRSC is full, one of the lines is evicted based on Least Recently Used (LRU) cache replacement strategy. Nonetheless, it has been reported in Section III that 1.55% of the entire LLC space are ERRA blocks, from which approximately half of them exclusively experience read operation. This means HRSC size is required to be in order of around 0.77% of the entire LLC size. For example, the HRSC capacity can be less than 800KB to fit all frequently read lines from an LLC with size of 96MB. Thus, we utilized the saved area as a result of replacing SRAM by LRSC in L2 to add facilities for HRSC-based operation. This data arrangement scheme enhances the cache service time because frequently exclusive-read blocks are maintained in the upper-level of cache with the lowest likelihood to be replaced by the new cache blocks.

The retention time of STT-MRAM cells integrated into LRSC design play a paramount role in determining the energy consumption and performance of the overall L2 design. Thus, we have conducted a study to evaluate the preferred retention time candidates which offer an optimum refresh scheme for LRSC arrangement.

B. RECLUSIVE: EFFICIENT CACHE CONFIGURATION TO MAXIMIZE RRAP THROUGHPUT

Besides considering the energy-efficient emerging devices in upper-level cache to accelerate service to the critical loads through increased cache capacity and reduced energy consumption, we also propose an efficient block insertion/eviction policy called *Reclusive*, a cache policy redesign of exclusive and non-inclusive designs, to maximize the throughput of hybrid cache designs entailing low and high retention STT-MRAM array.

The block insertion/eviction flow in Reclusive is illustrated in Figure 6 (b) and its corresponding algorithm is presented in Algorithm 1. On an LLC miss, the requested block is brought into LRSC and LLC. Transferring the requested block from memory to cache incurs one off-chip (①) and one on-chip (②,③) traffic. The off-chip traffic is the major delay contributor to the load operation issued by the processor because of: 1) the delay overhead for exchanging data over long wire between off-chip memory and on-chip LLC [12], 2) the restricted parallel access to multiple banks due to

Algorithm 1 Reclusive Block Insertion/Eviction Policy

Assumptions:
- *RC*: *Read Counter*, *WC*: *Write Counter*
- NR_{th}: read threshold level
- LRSC and HRSC are private non-LLC in *core i* where $i = \{1, 2, ..., number\ of\ cores\}$, L3 is shared LLC

Function insertion() /*algorithm for inserting requested block*/
begin
1 **if** *LLC miss from core i* **then**
2 eviction() /*evict *victim block* $\in LRSC_{core\ i}$ & *LLC*/
3 copy *block* $\in$ *memory* into $\text{LRSC}_{core\ i}$ & LLC
else
4 **if** *LLC write hit from core i* **then**
5 eviction() /*evict *victim block* $\in LRSC_{core\ i}$*/
6 copy *block* $\in$ *LLC* into $\text{LRSC}_{core\ i}$
7 $WC_{LLC\ block} = 1$
8 **else if** *LLC read hit from core i* **then**
9 **if** $RC == NR_{th}$ *and* $WC == 0$ **then**
10 **if** $HRSC_{core\ i}$ *is full* **then**
11 eviction() /*evict *victim block* $\in$ $HRSC_{core\ i}$*/
12 copy *block* $\in$ *LLC* into $\text{HRSC}_{core\ i}$
else
13 eviction() /*evict *victim block* $\in$ $LRSC_{core\ i}$*/
14 copy *block* $\in$ *LLC* into $\text{LRSC}_{core\ i}$
15 $++\ RC_{LLC\ block}$

Function eviction()/*algorithm for evicting victim block*/
16 **if** ($\exists$ *dirty victim block* $\in LRSC_{core\ i}/HRSC_{core\ i}$) ***OR*** ($\exists$ *clean victim block* $\in HRSC_{core\ i}$) **then**
17 **if** $\exists$ *replica block* $\in$ *LLC* **then**
18 update *replica block* $\in$ *LLC* /*insert dirty victim into same position in LLC, if $\exists$ clean victim block $\in \text{HRSC}_{core\ i}$ => no LLC update is needed*/
19 **else if** ($\nexists$ *replica block* $\in$ *LLC*) ***AND*** ($\exists$ *dirty victim block* $\in LRSC_{core\ i}/HRSC_{core\ i}$) **then**
20 evict *victim block* $\in$ *LLC* into memory
21 insert (*dirty victim block* $\in$ $LRSC_{core\ i}/HRSC_{core\ i}$) into *LLC*
22 set $WC_{inserted\ block\ \in\ LLC} = 1$
23 set $RC_{inserted\ block\ \in\ LLC} = 0$
24 **else if** ($\nexists$ *replica block* $\in$ *LLC*) ***AND*** ($\exists$ *clean victim block* $\in HRSC_{core\ i}$) **then**
25 evict *victim block* $\in$ *LLC* into memory
26 insert (*clean victim block* $\in HRSC_{core\ i}$) into *LLC*
27 set $RC_{inserted\ block\ \in\ LLC} = NR_{th}$
28 **else if** $\exists$ *clean victim block* $\in LRSC_{core\ i}/LLC$ **then**
29 silently drop clean victim block
30 **else if** $\exists$ *dirty victim block* $\in$ *LLC* **then**
31 evict victim block $\in$ LLC into main memory

shared data/address/command buses [26], and 3) the delay overhead to disconnect the row line and bitlines of the currently active row buffer and to activate the target row buffer in the case of *row buffer conflict*[2] [26].

Upon an LLC write hit, the requested block is brought into LRSC while the replica remains in the LLC (⑥). Likewise, the cache blocks are brought into LRSC on an LLC read hit while the LLC still keeps a duplicate copy of block (⑥), unless the cache block belongs to the ERRA group experiencing zero update memory operation, whereby the HRSC design is chosen for copying the requested block (⑦).

Upon an LLC access, the victim block in L2 must be selected for replacement by the accessed LLC block. Based on the history of access pattern to a LLC block, Reclusive selects the victim blocks either from LRSC or HRSC. The block eviction for LRSC is similar to the inclusive and non-inclusive caches, whereby the clean victim blocks are silently dropped (⑤) while the dirty victims are inserted into their replica blocks in LLC (④). Yet, the replica block may not exist in LLC for non-inclusive designs which necessitate the eviction of a victim block from LLC. If the evicted victim block from LLC is dirty, the block must be written back into the main memory which results in an off-chip transfer (⑩).

On the other hand, the employment of non-inclusive approach to deal with the victim blocks of HRSC incurs significant amount of energy and delay cost to retrieve the re-referenced block. The main reason is that while the blocks in HRSC are frequently referenced, their replicas in LLC become unreferenced. Eventually, these replica blocks in LLC become the potential candidates for the replacement. If the non-inclusive approach is considered to deal with the victim blocks of HRSC, all of the clean victim blocks must be silently dropped while their replicas may not be present in LLC. Since the chance of re-referencing the victim blocks which are evicted from HRSC is high, the access cost to the main memory will become a major concern if LLC cannot maintain the replica blocks.

To overcome this challenge, we propose the Reclusive strategy to exploit the exclusive feature to deal with the victim blocks from HRSC, whereby both clean and dirty blocks must always be inserted into LLC (⑧ or ⑨) regardless of their status in HRSC. In addition, we redesign Reclusive in such way to immediately promote the evicted clean victim blocks from HRSC as ERRA blocks during the insertion into LLC. This consideration aids to place the recently evicted clean blocks into HRSC cache again, if they are reused. Hence, it amortizes the corresponding service time to the critical loads and mitigates cache thrashing.

C. REFRESH SCHEME FOR LRSC

The retention time of STT-MRAM cell design utilized in LRSC architecture needs to be considered properly to meet the following key design issues:

[2] *row buffer conflict* occurs when the read/write request cannot be responded by the currently active row buffer.

1) DATA STABILITY DURING READ OPERATION

The data retention time should be sufficient to retain the stability of data while cache lines are accessed during read operations, otherwise the unstable data is sensed via sense amplifier, which in turn may cause the corrupted data to be provided to the CPU. Even though the sensing resolution and reliability of sense amplifiers employed in STT-MRAM cache designs influence the accuracy of the sensed data [27], [28], other characteristics such as resiliency to process variation [29], performance and power consumption [30], also play paramount roles for determining the preferred sense amplifier candidate.

2) COMPETITIVE PERFORMANCE DELIVERY COMPARED TO SRAM-BASED L2 DESIGN

The employment of high retention STT-MRAM cells in the cache design requires a long write pulse for write operation which results in performance degradation. This phenomenon becomes common in write-intensive applications. The retention relaxation of STT-MRAM can significantly improve switching performance by reducing the data retention time, which in turn causes reduced write latency.

3) CACHE BLOCK ACCESSIBILITY

To stabilize the data stored in a retention-relaxed cache line, the refresh operation re-writes the cache line which has reached the end of its lifespan. However, if the interval between refreshes is considered to be short, the performance of the system may significantly decrease because the cache block for normal read and write operations is not available during its refresh cycle. In addition, increasing the refresh ratio would undermine the STT-MRAM cell's endurance, which is on the order of 10^{12} [31], [32]. Thus, the refresh interval should be optimized to reduce the conflict ratio to these cache blocks while incurring insignificant refresh overhead.

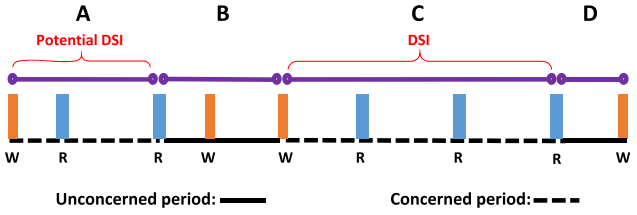

FIGURE 7. DSI equals to the sequence C which is largest interval between a write and the final subsequent read operation.

To find the optimum data retention time which addresses all aforementioned challenges, a new metric called *Data Stability Interval (DSI)* is defined for each cache block. DSI is the maximum interval between a write and the final subsequent read operation before the cache block can be accessed by a write or eviction operation. For instance, Fig. 7 shows the entire lifetime of a cache block which has been accessed by multiple read and write memory operations over time. Since the *interval A* begins with a write operation and followed by two read operations, it has the potential to be considered as DSI. The data unstability in the *interval B* does not impact on the processing data in CPU because the data will be overwritten during this period. In addition, accessing the cache line by a write operation is similar to refreshing that cache line which guarantees the stability of data up to the end of its lifespan. In *interval C*, the write operation is followed by three read accesses. The *interval C* exhibits longer period compared to *interval A*. Thus, the *interval C* is considered as DSI for this cache block. The ideal data retention time for LRSC design can be defined as follows:

$$Ideal \quad Retention \quad Time_{LRSC} = Max(DSI_0, DSI_1, \ldots, DSI_n) \tag{1}$$

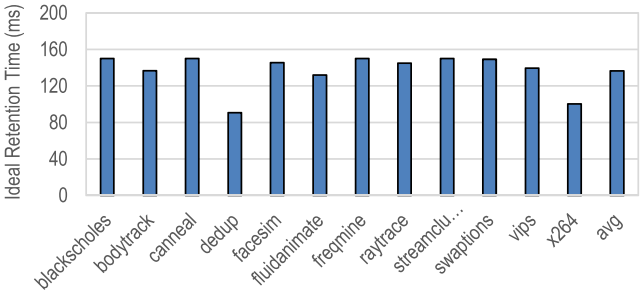

FIGURE 8. Ideal data retention time to avoid refresh scheme.

where n is the number of cache blocks. The refresh operation can be completely eliminated for LRSC design with ideal data retention time because the data is stable over the entire concerned period of the cache blocks. Fig. 8 shows the ideal data retention time obtained from Eq. 1 for both emerging read-intensive and write-intensive workloads selected from PARSEC benchmark suite. Typically, the ideal data retention time for read-intensive workloads is protracted to satisfy the exhaustive read accesses while lengthy read-read sequences are repeatedly taken place. On the other hand, the write-intensive workloads dictate extensive write operations to the cache lines, which intrinsically leads to refreshing accessed cache lines and reduced DSI. Thus, not all cache lines need to be designed ideally for read-intensive and write-intensive workloads because of the non-uniform access behavior to the cache lines. Furthermore, the program runtime behavior is often non-deterministic when running a set of multi-threaded workloads on a multiprocessor architecture which utilizes different branch prediction techniques for performance improvement and avoids unnecessary instruction execution. To clarify this issue, we have conducted an extensive application-driven study to classify DSI distribution over time and to find the optimum data retention time by taking the energy consumption and IPC of different LRSC designs into account.

Fig. 9 illustrates the DSI distribution for four selected workloads. The overall simulation period is divided into five periods and the cache lines are partitioned based on their DSI dispersion. Each bar in the plot depicts what percentage of

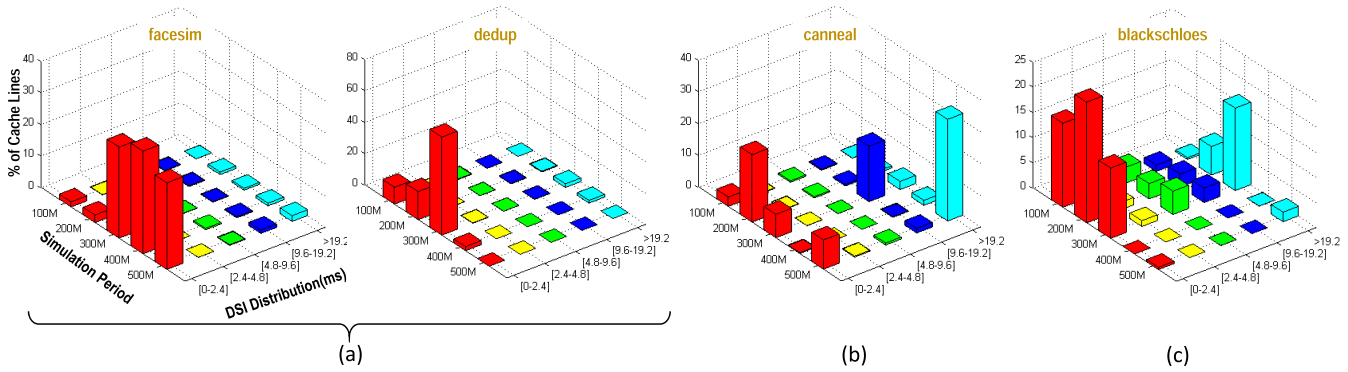

FIGURE 9. Three classes of DSI distribution: (a) unimodal, (b) bimodal, and (c) symmetric.

cache lines with calculated DSI are fall into which simulation period. For instance, less than 5% of all cache lines in *facesim* need to retain data longer than *19.2ms*, while this quantity goes up to 24.28% for *blackscholes* workload. Based on the non-uniformity of DSI distribution in different benchmarks, we identify three classes:

1) UNIMODAL DSI DISTRIBUTION

A considerable portion of cache lines exhibit short DSI distribution for *facesim* and *dedup* benchmarks, which result in their DSI often falling into the smallest expected period that cache line must preserve data. These benchmarks require relatively narrow window retention time whereby cache lines are accessed by regular write operations.

2) BIMODAL DSI DISTRIBUTION

In this distribution, cache lines are often partitioned into two groups. The first group has short DSI while the DSI of the second group often resides in long-lasting DSI category. The *canneal* demonstrates a bimodal DSI distribution among different cache lines in which around 41% of cache lines require data retention time less than *2.4ms* while this ratio goes up to 57% for cache lines with DSI more than *9.6ms*.

3) SYMMETRIC DSI DISTRIBUTION

The cache lines' DSI are uniformly distributed in each DSI category over simulation periods. The benchmark *blackscholes* has approximately symmetric DSI distribution unlike the behavior of aforementioned benchmarks.

Fig. 10 shows the total cache lines distribution with different DSI, where the majority of cache lines (on average 70.3%) require data retention time less than *2.4ms*. On the other hand, around 18% of cache lines need long-lasting retention time more than *19.2ms* to meet data stability purpose during read operation.

To find the sufficient data retention time that accommodates provision for the above three classes while miniaturizing conflict with normal memory accesses and delivering high performance, we selected three different STT-MRAM cell's retention time for comparison in terms of incurred energy consumption and offered performance as

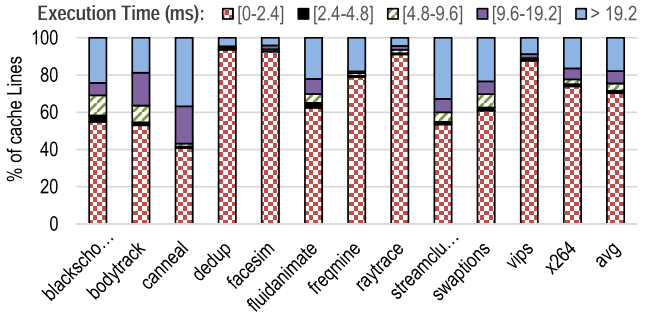

FIGURE 10. DSI distribution for all benchmarks.

TABLE 1. STT-MRAM cell retention time configurations for LRSC design.

Configuration	Design 1	Design 2	Design 3
Retention Time	140ms	10ms	1ms
Write Latency @3GHz	12 cycles	7 cycles	6 cycles

listed in Table 1. *Design 1* complies with the demands of both *bimodal* and *symmetric DSI distributions* to provide data retention time as high as ideal approach for cache lines with DSI more than *19.2ms*. *Design 3* offers a reduced retention time in favor of workloads with *unimodal DSI distribution* to promote write performance. *Design 2* has been considered as an intermediary to satisfy both groups of cache lines claiming either long or short DSI while also targeting cache lines having middle retention time. The reduced data′s lifespan stored in *Design 2* and *Design 3* require a refresh mechanism to prevent data loss. Therefore, the proposed refresh scheme in [22] is considered to sequentially refresh all cache blocks. We have included the energy contributions from peripheral circuits and energy consumption due to refresh mechanism in our simulation results. The detailed scheme for adjusting the retention time of the STT-MRAM cell is elaborated in Section IV-D.

The energy consumption and IPC comparison for the above three designs are shown in Fig. 11 and Fig. 12, respectively. The relatively high retention time of *Design 1* comes at the expense of high write current and slow write speed while completely eliminating the energy dissipation and memory access conflict induced by periodic refresh operation.

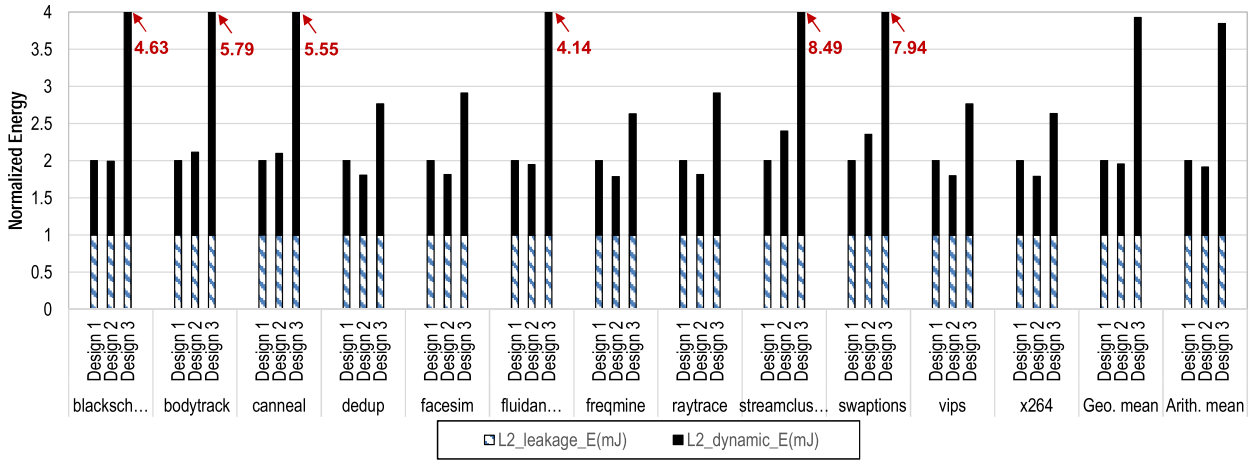

FIGURE 11. Energy breakdown comparison among three LRSC configurations normalized to Design 1.

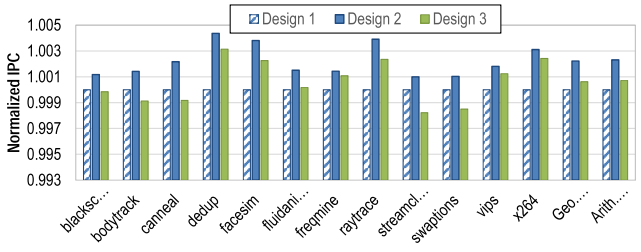

FIGURE 12. IPC comparison among three LRSC configurations normalized to Design 1.

Design 2 leverages the small write current realized by lowering retention time to diminish dynamic energy consumption and to improve write performance. Although it is expected to observe decent improvement in both criteria compared to previous design, the results show slight gain. The main reason is that the extra write operations associated with periodically refreshing cache lines incur additional energy dissipation while the conflict between refresh operations and normal memory accesses is trivial. The augmented number of refreshes for *Design 3* would undermine the benefits of volatile STT-MRAM with *1ms* retention time whereby the conflict ratio among refresh operations and normal read/write accesses significantly increases besides advancing the write speed. Thus, since the *Design 2* offers lowest energy consumption and highest IPC among three designs, we selected it as the basis for our LRSC design. *Design 2* benefits from reduced retention time to improve energy consumption and delivers competitive write performance, while the negative impact of its refresh operations is amortized. *Design 2* reduces energy consumption by 57.09% on average compared to *Design 3* and enhances the overall IPC slightly in comparison with *Design 1*. These experimental results confirm the conducted study in [9].

D. RETENTION RELAXATION IN LRSC DESIGN

STT-MRAM write performance improvement relates to the optimization of retention time [9], [22]. STT-MRAM is usually considered as non-volatile technology, while its non-volatile characteristics can be relaxed to obtain better write performance. The retention time of STT-MRAM, which is the period that STT-MRAM can retain data until a bit-flip occurs, can be modeled as [15]:

$$t = t_1 \times e^{\Delta} \tag{2}$$

where t is the retention time, t_1 is is fitting constant, and Δ is thermal barrier that determines the stability of STT-MRAM. The retention time of STT-MRAM can be reduced exponentially by using the thermal barrier reduction whereby Δ can be characterized using [33]:

$$\Delta \approx \frac{M_s H_k V}{T} \tag{3}$$

where M_s is the saturation magnetization, H_k is the in-plane anisotropy field, V is the volume of free layer, and T is the absolute temperature in Kelvin.

It has been demonstrated in [33] that there are two STT-MRAM switching modes: 1) *Thermal Activation (TA) region*, where I_{write} is less than critical write current (I_c) and 2) *Precessional Switching (PS) region*, where I_{write} is greater than I_c. In PS mode, the write pulse width reduction incurs the rapid increase of write current. Accordingly, some particular write pulse width can deliver optimal STT-MRAM write energy. The focus of this paper is on overall write latency and energy optimization in L2 structure design. Thus, herein we adjust Δ in Eq. 4 to 1) calibrate the STT-MRAM write speed such that it is comparable with SRAM, 2) find the optimal write current for shorter write pulse width which can still provide required write energy needed for switching in the PS region. The switching duration can be calculated as follows [34]:

$$\frac{1}{\tau_1} = \left[\frac{2}{(C + ln(\pi^2\Delta))}\right]\frac{\mu_B P}{em(1+P^2)}(I_{write} - I_c) \tag{4}$$

where τ_1 is the switching mean duration, $C = 0.577$ is Euler′s constant, μ_B is the Bohr magneton constant, P is the tunneling spin polarization of the ferromagnetic layers, e is

is the magnitude of the electron charge, m is the free layer magnetic moment, and I_c is critical write current computed as below:

$$I_c = 2\alpha \frac{\gamma e}{\mu_B g} E \tag{5}$$

where α is Gilbert damping coefficient, γ is the Gyromagnetic constant, g is the spin polarization efficiency factor and E is the barrier energy [33].

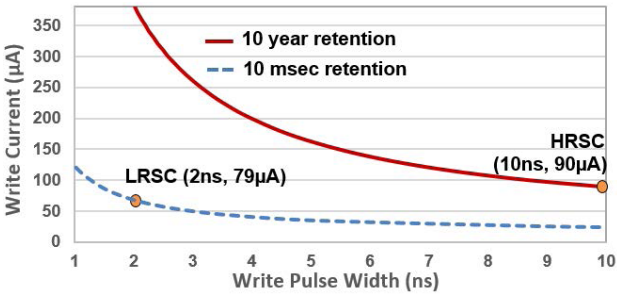

FIGURE 13. Write current vs. write pulse width for LRSC and HRSC.

RRAP utilizes two distinct memory arrays to provide categories of retention times matching the required memory reference characteristics as depicted in Fig. 13. In our study, STT-MRAM with a ten year retention time and high Δ is considered as the baseline. The volatile STT-MRAM with lower Δ offers retention time of *10ms*. As shown in Fig. 13, two operating points HRSC (10ns, 90μA) and LRSC (2ns, 79μA) are selected. Based on the RRAP methodology, LRSC requires to be operated with low retention time while high retention time STT-MRAM is needed for HRSC design. In order to reduce thermal barrier to achieve a low retention STT-MRAM cell, we utilized the previous methodology proposed in previous works [9], [22] in which the planar area and thickness of STT-MRAM is reduced resulting in exponentially reducing the retention time and required write current for switching. The write current, write pulse width, and the cell size associated to each point are integrated into NVSim to obtain energy consumption and latency factors. The detailed device characterization of L2 cache structure are listed in Table 2 whereby STT-MRAM offers at least 3x to 4x area savings compared to the SRAM.

V. EXPERIMENTAL EVALUATION

A. SIMULATOR CONFIGURATION

We evaluate our design using MARSSx86 with PARSEC 2.1 and SPEC2006 workloads. We model a Chip Multi-Processor (CMP) with eight single-threaded x86 cores. Each core consists of private L1, LHRSC L2 (LRSC and HRSC) and the LLC shared among all the cores. The detail of our model can be found in Table 3. Twelve workloads from the PARSEC and three workloads from SPEC2006 are selected and executed 500 million instructions starting at the Region Of Interest (ROI) after warming up the cache for 5 million instructions. The `simsmall` and `ref` input sets are used for PARSEC and SPEC2006 workloads, respectively. The latency and energy usage associated with read and write operations for SRAM and STT-MRAM cache accesses are provided by NVSim while DESTINY is used to model eDRAM model. This is because NVSim model of eDRAM is incomplete and has not been validated. In addition, the energy contributions from peripheral circuits are also included in our simulation.

TABLE 2. Detailed characteristics of private L2 cache bank configuration (32nm, temperature = 350K, SRAM tag array for all designs).

L2 Cache Technology	Area (mm^2)	RL (ns)	WL (ns)	RE (nJ)	WE (nJ)	LP (mW)
512KB SRAM	1.410	1.277	1.277	0.293	0.293	1753.444
1MB eDRAM	0.745	1.072	1.022	0.289	0.424	337.329
1MB STT-MRAM	0.526	1.340	10.218	0.280	0.654	212.022
512KB LRSC	0.243	1.260	2.153	0.233	0.269	104.797
512KB HRSC	0.357	1.261	10.153	0.233	0.601	114.915

RL: Read Latency, WL: Write Latency, RE: Read Energy, WE: Write Energy, LP: Leakage Power

TABLE 3. Memory subsystem

Processor	3GHz processor Fetch/Exec/Commit width 4
Private L1-I/D	SRAM, 32 KB, 8-way set assoc., WB cache
Private L2 Conf.	8-way set assoc., WB cache
Shared L3	eDRAM, 96 MB, 16-way set assoc., 16 bank, WB cache
Main memory	8 GB, 1 channel, 4 ranks/channel, 8 bank/rank

B. ENERGY USAGE COMPARISON

To comprehensively evaluate RRAP efficacy in terms of energy consumption, we compare the energy breakdown of RRAP against: `(i)` conventional 512KB SRAM-based L2, `(ii)` 1MB eDRAM-based L2, `(iii)` 1MB high retention STT-MRAM based L2, `(iv)` state-of-the-art SBAC bypass method for NVM-based L2 with *`bypassing depth = 2`* [35], `(v)` state-of-the-art multi-retention hybrid cache design which is proposed in [22] and referred herein as HLR technique. The detailed characteristics of considered technologies to be integrated into L2 design are listed in Table 2. Note here RRAP consists of 512KB LRSC (Design 2 in Section IV-C) and 512KB HRSC.

The write operation in HRSC and regular STT-MRAM requires high energy to amplitude write pulse signal for retaining the data for a longer time compared to two other technologies. On the other hand, the data's lifespan stored in LRSC is *10ms*, which makes it necessary to employ a refresh mechanism to prevent data loss. Therefore, a low-overhead refreshing scheme introduced by [22] is deployed herein which all cache blocks are refreshed sequentially. Consequently, as shown in Fig. 14, the consumed dynamic energy of RRAP approach is slightly higher than the dynamic energy consumed by SRAM-based L2, but it is still significantly less than dynamic energy consumption of eDRAM which demands exhaustive refresh operation every 40*us* [15], [36]. Note that the high energy required for write operation in

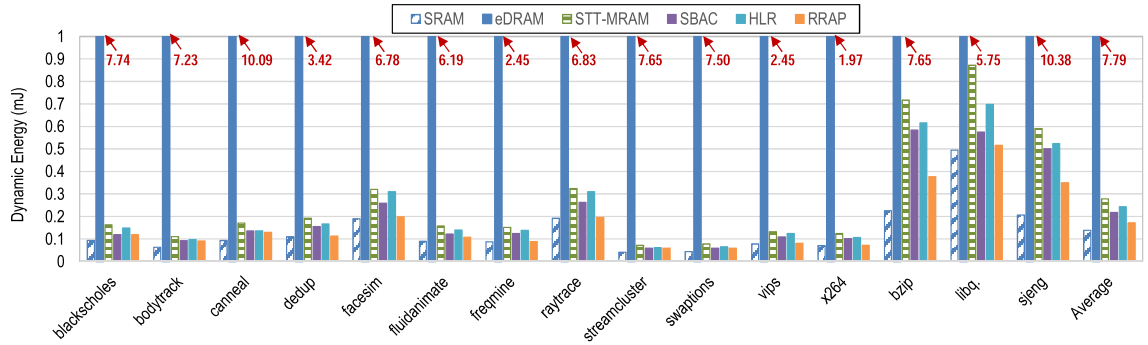

FIGURE 14. L2 dynamic energy breakdown for SRAM, eDRAM, regular STT-MRAM, SBAC, HLR, and RRAP.

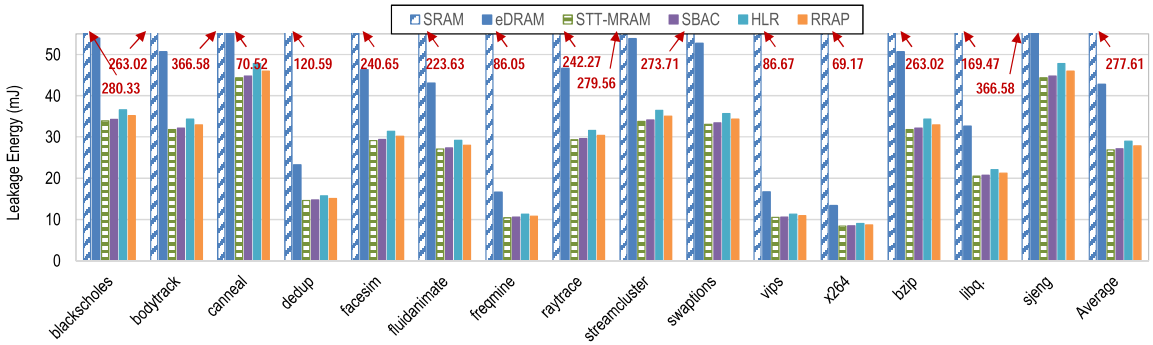

FIGURE 15. L2 leakage energy breakdown for SRAM, eDRAM, regular STT-MRAM, SBAC, HLR, and RRAP.

HRSC has negligible effect on the overall dynamic energy consumption of RRAP because the cache blocks are brought into HRSC once and are accessed by read operation for the remaining of execution time. Moreover, the high write energy and large memory cell size in regular STT-MRAM incur higher dynamic energy consumption in comparison with RRAP.

On the other hand, SBAC prevents the allocation of L2 to the blocks that occupy cache capacity without receiving sufficient cache hits. To aim this, SBAC bypasses L2 by directly providing the requested block from LLC to L1 when the data with reuse count less than *bypassing depth* is accessed. Thus, a fraction of write operations are eliminated which results in the dynamic energy consumed by SBAC becomes less than regular STT-MRAM. HLR inserts blocks into high retention region of L2 upon two incidents: 1) a read miss, 2) the lifespan of block resided in low retention region is close to the end. The large current associated with write operation in these incidents results in that HLR becomes less effective rather than SBAC and RRAP. In addition, unlike RRAP, HLR adopts typical LRU replacement policy to deal with the requested blocks. This incident may cause cache thrashing which means the useful blocks become potential candidates for replacement prior to contributing sufficient read hit to L2, resulting in increased energy consumption for data movement. RRAP reduces 38.27% of the dynamic energy consumed by regular STT-MRAM which is the lowest ratio among STT-MRAM based cache designs, demonstrating that our scheme outperforms SBAC and HLR.

Fig. 15 compares the leakage energy consumption for aforementioned designs. Although SRAM exhibits lower dynamic energy consumption, its high leakage power has exacerbated the overall consumed energy compared to other designs. While it is widely accepted that STT-MRAM is highly persistent against leakage, its peripherals consume significant leakage energy. Thus, the regular STT-MRAM and SBAC designs offer the least leakage energy compared to two other STT-MRAM based L2 designs due to leveraging smaller-sized peripherals.

C. READ MISS RATIO COMPARISON

In light of RRAP, the read miss ratio of L2 is significantly decreased as illustrated in Fig. 16. The main reason that RRAP can achieve the reduced miss ratio is the capability of RRAP approach for bringing frequently exclusive read accessed blocks from LLC to HRSC in upper-level of cache while guaranteeing the cache blocks to reside for a large window of execution time. This accommodation significantly hides the read miss latency required to bring the frequently re-referenced data from either main memory or LLC to L2. On the other hand, SBAC incurs greatest read miss ratio compared to other schemes because it prevents the allocation of L2 to the blocks with reuse count less than *bypassing depth*. In our experimental results, we consider the `bypassing`

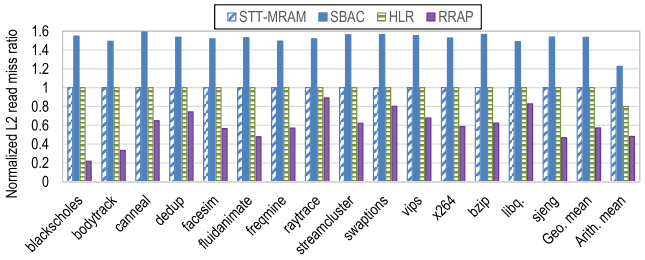

FIGURE 16. Comparison of L2 read miss ratio normalized to STT-MRAM.

depth = 2 based on [35] which means that if a block resided in LLC is referenced twice or more, it can be inserted into L2. Hence, SBAC policy imposes significant L2 miss ratio during initial placement and before that the frequently reused blocks surpass *bypassing depth*. Furthermore, the distant interval re-reference access pattern in SBAC cache configurations, which is typical in some workloads, can exacerbate the L2 read miss ratio because enforcing the eviction of blocks before contributing sufficient read hits to L2.

RRAP reduces the read miss ratio of the regular STT-MRAM based L2 design by 51.74% on average (arithmetic mean). The experimental results show that the read-intensive workloads, such as blackscholes, bodytrack and fluidanimate, leverage the full potential of RRAP to further diminish the read miss ratio compared to eDRAM-based and regular STT-MRAM L2 designs.

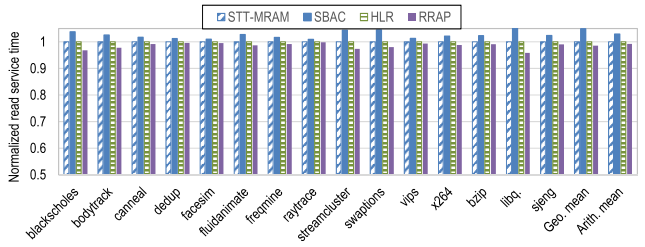

FIGURE 17. Comparison of read service time for the entire memory hierarchy normalized to STT-MRAM.

D. READ SERVICE TIME COMPARISON

Herein, the elapsed time to service a read request by the processor is referred to Read Service Time (RST). To estimate the RST for the entire system, we have considered the amount of read hits and misses in all three levels of cache while also taking the trip latency for transferring a cache block from lower-level to upper-level of cache into account. SBAC design offers the longest RST compared to other designs as shown in Fig. 17. The reason is primarily because of high L2 read miss ratio which results in each bypassed block additionally traverses the distance between LLC and L2 rather than other designs. On the other hand, the designs which utilize the regular STT-MRAM and HLR in L2 offer nearly identical RST, but still longer RST compared to RRAP. In particular, RRAP reduces the normalized RST around 1.47% compared to regular STT-MRAM. The main reason that this improvement may seem trivial is that the service time to bring a data block from main memory to LLC contributes the most to the overall RST. This negatively impacts on the efficiency of RRAP which decreases the read miss ratio in L2.

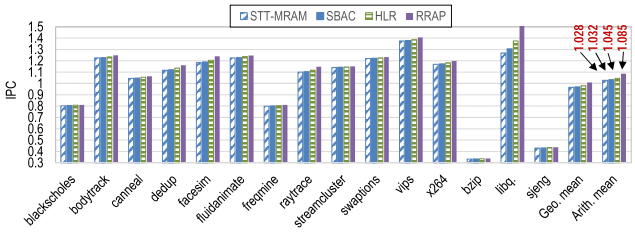

FIGURE 18. Comparison of RRAP's performance with other techniques.

E. PERFORMANCE COMPARISON

Besides the read miss ratio and RST comparison, we also consider IPC to compare the overall performance. Fig. 18 illustrates the IPC of the systems where SBAC, HLR, and RRAP exhibit greater performance compared to regular STT-MRAM design. The main reason for performance degradation in regular STT-MRAM is its high write latency while this latency has been amortized in other designs. SBAC reduces the latency associated with write operation via avoidance of L2 allocation to non-reused blocks which results in improved performance. HLR allocates STT-MRAM arrays with low retention feature to write intensive blocks for faster write response which in turn results in improved performance. RRAP adjusts HLR strategy to deal with typical access requests to L2 while also improving the temporal locality of frequently reused blocks which may exhibit distant re-reference interval. Thus, these useful blocks are maintained long-enough in L2 to contribute sufficient L2 read hits while the required energy for preserving them is trivial. To be specific, RRAP leverages an efficient retention-relaxed STT-MRAM as LRSC to amortize the incurred high latency associated with the write operation in STT-MRAM cells. At the same time, the cache blocks are inserted into HRSC only one time while the remained memory accesses to this region are read operations.

VI. RELATED WORK

A. EXPLORATION OF HYBRID CACHE DESIGN

Hybrid cache design approaches have received significant attention over past years. These techniques leverage the benefits of utilized technologies to maximize the performance and minimize the energy consumption. In [19], both SRAM and eDRAM technologies are employed in the second-level cache to offer area savings and reduced energy consumption compared to SRAM-based L2 design while the performance is increased by 5.9 percent on average. The proposed LLC design leverages the fast SRAM device to store the most likely referenced blocks in the future while the high-dense

and low-leakage eDRAM is utilized to reduce the overall energy consumption [19]. In other words, the larger SRAM banks can provide the higher performance while the deployment of larger area-efficient eDRAM requires lower energy for operation. Thus, the ratio of utilizing these technologies in the LLC has been optimized to provide the sufficient performance and reduce energy consumption. However, this design does not explicitly consider the critical load behavior.

The proposed work in [37] leverages the asymmetric write power associated with storing 'ones' and 'zeros' to place data blocks having majority 'zero' data into STT-MRAM while the remained data blocks are stored in the SRAM. To deal with the change of majority due to regular update operations, a two-bit counter has been considered for each cache line to track the status of the majority. To avoid write overhead associated with the unnecessary migration, the migration policy swap the data blocks according to the pace of majority-data changes. This design can limit the performance of multicore designs to the efficiency of the module which determines the majority of the data. Namely, each data block should be given to the dispatcher module before placement in the LLC which incurs significant overhead for memory-intensive applications and impose significant pressure on the placement policy.

The proposed hybrid LLC design proposed in [38] utilizes an intelligent adaptive data block placement by taking each cache line future access pattern into consideration. The write accesses are categorized into three main classes: prefetch-write, core-write and demand-write. Around 26% of all prefetch blocks are not accessed by the core after initial prefetch [38]. Furthermore, the data blocks moved to the LLC due to cache burst phenomenon are not accessed again until the eviction occurs [38]. The data blocks with the aforementioned characteristics are considered to be placed in the SRAM. On the other hand, the data blocks which experience long interval between consecutive reads are placed into the STT-MRAM to benefit from the low leakage power cost of long-residency offered by non-volatile devices [38]. This design may incur prediction design overhead which incurs extra energy consumption for tracking the access pattern to each cache line and making the decision for placement.

In [22], Low-Retention (LR) and High-Retention (HR) STT-MRAM arrays are utilized simultaneously to balance the performance versus the energy consumption. To accomplish this, the retention time of the STT-MRAM is relaxed for LR architecture to improve the write operation speed. On the other hand, the HR cache offers the long-term residency of data block while incurring very small leakage power. Regarding to the features that each cache design offers, the proposed method manages the write-intensive cache blocks to be placed into the LR cache in favor of performance, while the read-intensive cache blocks are kept in the HR arrays to meet the required power budget limits. Furthermore, the migration policy is devised to transfer the data blocks to the proper cache design by continuously monitoring the access pattern to each cache line. Since the lifespan of the cache lines in the L1 arrays is limited to the retention time which is deliberated at the design time, a refresh mechanism is designed to periodically refresh cache lines for preventing data loss. RRAP utilizes the same refresh approach to maintain the stability of data in LRSC design.

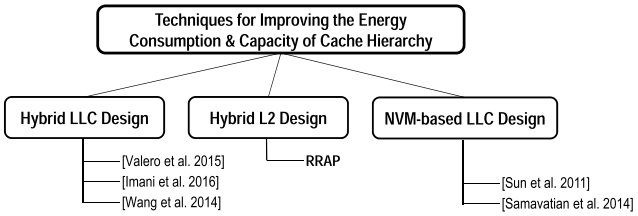

FIGURE 19. Taxonomy of techniques utilizing the emergent technologies for improving the performance and energy consumption. Approaches using eDRAM highlighted in bold face font.

In [21], the similar idea is utilized in the LLC design for GPU. The small-sized LR cache is employed to keep the write-intensive data blocks by considering the fact that the re-write interval time of blocks is typically lower than $100\mu s$. Again, the large HR arrays is considered to maintain the less-frequently written data blocks [21]. The proposed technique uses a write threshold on HR to determine whether to keep the data in HR or move it to the LR. To achieve this goal, the correlation between the threshold value and the performance has been accurately analyzed, and the optimum value has been selected as the threshold value. In addition, different degrees of associativity are considered for LR and HR designs to maximize the write utilization in LR and to improve the data migration policy. The taxonomy of the discussed techniques are presented in Fig. 19 and the contribution of our technique compared to other techniques is highlighted in table 4.

B. CACHE BYPASSING TECHNIQUES TO CONSERVE ENERGY CONSUMPTION

Besides proposing hybrid cache designs to enhance performance and energy consumption, the bypass methods [39]–[42] have received significant attention for their efficiency to improve the temporal locality. González *et al.* [39] proposed a data cache organization to improve both spatial and temporal locality through *locality prediction table*. Essentially, the history of recently referenced instructions is considered to bypass branches that can cause cache pollution. McFarling [40] proposed to dynamically exclude cache lines that contribute to the conflict misses in a direct-mapped cache. Rathi *et al.* [41] proposed to bypass STT-MRAM based LLC while directly connecting upper-level caches to the memory if the LLC is susceptible to Denial of Service (DoS) or any other attacks. Kim *et al.* [42] proposed a bypass method for inclusive NVM-based LLC to reduce the write operations overhead which in turn results in the improved performance and efficient energy consumption. However, both replacement policy and the available cache capacity determine the efficiency of the aforementioned techniques. For example, suppose the working set cannot fit into L2 due to the insufficient cache capacity. If the access

TABLE 4. Comparison of hybrid cache design techniques.

Approach	multi-retention STT-MRAM employment	Response time to write operation	Energy reduction method	Service to critical loads	Cache configuration	Contributions to overhead
Sun et al. [22]	yes	fast (volatile STT-MRAM)	low-leakage STT-MRAM & data migration	no	N/A	periodic refresh+ exhaustive data migration
Valero et al. [19]	no	fast (SRAM banks)	Hybrid SRAM & eDRAM banks design	no	N/A	periodic refresh+ swap between SRAM & eDRAM
samavatian et al. [21]	yes	fast (volatile STT-MRAM)	low-leakage STT-MRAM & data migration	no	N/A	periodic refresh+ data migration
Jog et al. [9]	yes	fast (volatile STT-MRAM)	low-leakage STT-MRAM & buffering-based refresh scheme	no	N/A	write back dirty blocks+ buffering
RRAP (approach herein)	yes	fast (volatile STT-MRAM)	low-leakage STT-MRAM & data movement reduction	yes	Reclusive	periodic refresh

sequences of the running application to the L2 blocks discard a portion of resided blocks whose will be accessed at the end of reference pattern, then the most of the newly inserted blocks will be superfluous before contributing sufficient hit to L2. This scenario may also occur in the applications that exhibit a burst reference pattern to non-temporal blocks [43]. Thus, even though bypass methods mitigate the poor locality in most of scenarios, in this case maintaining a frequently read accessed fraction of the working set in the upper level caches for a sufficient period improves the temporal locality.

VII. CONCLUSIONS

In conclusion, RRAP retains the on-chip cache utilization close to the requesting cores for data locality maximization and lower memory access latency while taking into account the energy consumption and performance speed-up. HRSC design leverages the non-volatility of STT-MRAM to store ERRA cache blocks, which in turn provides long-term residency of data without demanding extra energy to maintain the data stability. For LRSC design, we conducted an extensive exploration to find the preferred configuration in terms of energy consumption and performance. Accordingly, the write latency of LRSC is relaxed to maintain the cache utilization as high as conventional SRAM attains. Our experimental results show that RRAP can reduce the overall energy consumption and read miss ratio significantly, and read service time slightly within a comparable footprint to STT-MRAM based L2 designs.

REFERENCES

[1] G. Kurian, S. Devadas, and O. Khan, "Locality-aware data replication in the last-level cache," in *Proc. 20th Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2014, pp. 1–12.

[2] X. Chen *et al.*, "AOS: Adaptive overwrite scheme for energy-efficient MLC STT-RAM cache," in *Proc. 53nd Annu. Design Autom. Conf. (DAC)*, Jun. 2016, pp. 1–6.

[3] M. Ahmadian, A. Paya, and D. C. Marinescu, "Security of applications involving multiple organizations and order preserving encryption in hybrid cloud environments," in *Proc. 28th Int. Parallel Distrib. Process. Symp. Workshops*, 2014, pp. 894–903.

[4] M. Imani, D. Peroni, and T. Rosing, "Nvalt: Nonvolatile approximate lookup table for GPU acceleration," *IEEE Embedded Syst. Lett.*, vol. 10, no. 1, pp. 14–17, Mar. 2018.

[5] S. Borkar and A. A. Chien, "The future of microprocessors," *Commun. ACM*, vol. 54, no. 5, pp. 67–77, May 2011.

[6] D. Kim, S. Yoo, and S. Lee, "Hybrid main memory for high bandwidth multi-core system," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 1, no. 3, pp. 138–149, Jul. 2015.

[7] M. Imani, A. Rahimi, Y. Kim, and T. Rosing, "A low-power hybrid magnetic cache architecture exploiting narrow-width values," in *Proc. Non-Volatile Memory Syst. Appl. Symp.*, Aug. 2016, pp. 1–6.

[8] M. Ahmadian, J. Khodabandehloo, and D. C. Marinescu, "A security scheme for geographic information databases in location based systems," in *Proc. SoutheastCon*, 2015, pp. 1–7.

[9] A. Jog *et al.*, "Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs," in *Proc. 49th Annu. Design Autom. Conf. (DAC)*, 2012, pp. 243–252.

[10] A. Jaleel, M. Mattina, and B. Jacob, "Last level cache (LLC) performance of data mining workloads on a CMP–a case study of parallel bioinformatics workloads," in *Proc. 12th Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2006, pp. 88–98.

[11] M. Imani, S. Patil, and T. Rosing, "DCC: Double capacity cache for narrow-width data values," in *Proc. Great Lakes Symp. VLSI*, 2016, p. 1.

[12] M. N. Bojnordi and E. Ipek, "DESC: Energy-efficient data exchange using synchronized counters," in *Proc. 46th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2013, pp. 234–246.

[13] A. N. Udipi, N. Muralimanohar, and R. Balasubramonian, "Non-uniform power access in large caches with low-swing wires," in *Proc. Int. Conf. High Perform. Comput. (HiPC)*, 2009, pp. 59–68.

[14] R. Arroyo, R. Harrington, S. Hartman, and T. Nguyen, "IBM power7 systems," *IBM J. Res. Develop.*, vol. 55, no. 3, pp. 1–2, 2011.

[15] M.-T. Chang, P. Rosenfeld, S.-L. Lu, and B. Jacob, "Technology comparison for large last-level caches (L^3 Cs): Low-leakage SRAM, low write-energy STT-RAM, and refresh-optimized eDRAM," in *Proc. 19th Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2013, pp. 143–154.

[16] S. Khan, A. R. Alameldeen, C. Wilkerson, O. Mutlu, and D. A. Jimenezz, "Improving cache performance using read-write partitioning," in *Proc. 20th Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2014, pp. 452–463.

[17] N. Tuck and D. M. Tullsen, "Initial observations of the simultaneous multithreading Pentium 4 processor," in *Proc. 12th Int. Conf. Parallel Archit. Compilation Techn.*, 2003, pp. 26–34.

[18] M. K. Qureshi, A. Jaleel, Y. N. Patt, S. C. Steely, and J. Emer, "Adaptive insertion policies for high performance caching," *ACM SIGARCH Comput. Archit. News*, vol. 35, no. 2, pp. 381–391, 2007.

[19] A. Valero, J. Sahuquillo, P. Lopez, and J. Duato, "Design of hybrid second-level caches," *IEEE Trans. Comput.*, vol. 64, no. 7, pp. 1884–1897, Jul. 2015.

[20] A. Agrawal, P. Jain, A. Ansari, and J. Torrellas, "Refrint: Intelligent refresh to minimize power in on-chip multiprocessor cache hierarchies," in *Proc. 19th Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2013, pp. 400–411.

[21] M. H. Samavatian, H. Abbasitabar, M. Arjomand, and H. Sarbazi-Azad, "An efficient STT-RAM last level cache architecture for GPUs," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2014, pp. 1–6.

[22] Z. Sun *et al.*, "Multi retention level STT-RAM cache designs with a dynamic refresh scheme," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2011, pp. 329–338.
[23] V. Seshadri *et al.*, "Mitigating prefetcher-caused pollution using informed caching policies for prefetched blocks," *ACM Trans. Archit. Code Optim.*, vol. 11, no. 4, p. 51, 2015.
[24] H. Xu, Y. Alkabani, R. Melhem, and A. K. Jones, "FusedCache: A naturally inclusive, racetrack memory, dual-level private cache," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 2, pp. 69–82, Jun. 2016.
[25] J. Sim, J. Lee, M. K. Qureshi, and H. Kim, "FLEXclusion: Balancing cache capacity and on-chip bandwidth via flexible exclusion," in *Proc. 39th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2012, pp. 321–332.
[26] E. Kültürsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating STT-RAM as an energy-efficient main memory alternative," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Apr. 2013, pp. 256–267.
[27] W. S. Zhao *et al.*, "Failure and reliability analysis of STT-MRAM," *Microelectron. Rel.*, vol. 52, nos. 9–10, pp. 1848–1852, Sep./Oct. 2012.
[28] L. Jiang, W. Wen, D. Wang, and L. Duan, "Improving read performance of STT-MRAM based main memories through smash read and flexible read," in *Proc. 21st Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2016, pp. 31–36.
[29] E. Eken, Y. Zhang, W. Wen, R. Joshi, H. Li, and Y. Chen, "A novel self-reference technique for STT-RAM read and write reliability enhancement," *IEEE Trans. Magn.*, vol. 50, no. 11, pp. 1–4, Nov. 2014.
[30] J. Li, P. Ndai, A. Goel, S. Salahuddin, and K. Roy, "Design paradigm for robust spin-torque transfer magnetic RAM (STT MRAM) from circuit/architecture perspective," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 12, pp. 1710–1723, Dec. 2010.
[31] M. R. Jokar, M. Arjomand, and H. Sarbazi-Azad, "Sequoia: A high-endurance NVM-based cache architecture," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 3, pp. 954–967, Mar. 2016.
[32] C. Pan, S. Gu, M. Xie, Y. Liu, C. J. Xue, and J. Hu, "Wear-leveling aware page management for non-volatile main memory on embedded systems," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 2, pp. 129–142, Apr. 2016.
[33] W. Kang *et al.*, "An overview of spin-based integrated circuits," in *Proc. 19th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2014, pp. 676–683.
[34] D. Worledge *et al.*, "Sullivan, and R. Robertazzi, "Spin torque switching of perpendicular ta form," *Appl. Phys. Lett.*, vol. 98, no. 2, p. 2501, 2011.
[35] C. Zhang, G. Sun, P. Li, T. Wang, D. Niu, and Y. Chen, "SBAC: A statistics based cache bypassing method for asymmetric-access caches," in *Proc. Int. Symp. Low Power Electron. Design*, 2014, pp. 345–350.
[36] N. Khoshavi, X. Chen, J. Wang, and R. F. DeMara, "Bit-upset vulnerability factor for eDRAM last level cache immunity analysis," in *Proc. 17th Int. Symp. Quality Electron. Design (ISQED)*, 2016, pp. 6–11.
[37] M. Imani, S. Patil, and T. Rosing, "Low power data-aware STT-RAM based hybrid cache architecture," in *Proc. 17th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2016, pp. 88–94.
[38] Z. Wang, D. A. Jiménez, C. Xu, G. Sun, and Y. Xie, "Adaptive placement and migration policy for an STT-RAM-based hybrid cache," in *Proc. 20th Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2014, pp. 13–24.
[39] A. González, C. Aliagas, and M. Valero, "A data cache with multiple caching strategies tuned to different types of locality," in *Proc. ACM Int. Conf. Supercomput. 25th Anniversary Vol.*, 2014, pp. 217–226.
[40] S. McFarling, "Cache replacement with dynamic exclusion," *ACM SIGARCH Comput. Archit. News*, vol. 20, no. 2, pp. 191–200, 1992.
[41] N. Rathi, A. De, H. Naeimi, and S. Ghosh. (2016). "Cache bypassing and checkpointing to circumvent data security attacks on STTRAM." [Online]. Available: https://arxiv.org/abs/1603.06227
[42] M. K. Kim, J. H. Choi, J. W. Kwak, S. T. Jhang, and C. S. Jhon, "Bypassing method for STT-RAM based inclusive last-level cache," in *Proc. Conf. Res. Adapt. Convergent Syst.*, 2015, pp. 424–429.
[43] A. Jaleel, K. B. Theobald, S. C. Steely, Jr., and J. Emer, "High performance cache replacement using re-reference interval prediction (RRIP)," *ACM SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 60–71, 2010.

NAVID KHOSHAVI received the Ph.D. degree in computer engineering with the University of Central Florida in 2017. He is currently a Faculty Member with Florida Polytechnic University. His research interests include online error detection and recovery in multicore processors, hardware reliability and security, energy-aware memory design, emerging technology utilization in memory hierarchy module including emerging spintronic-based and eDRAM devices.

RONALD F. DEMARA received the Ph.D. degree in computer engineering from the University of Southern California in 1992. Since 1993, he has been a full-time Faculty Member with the University of Central Florida, where he is currently a Professor of electrical and computer engineering and a Joint Faculty of computer science. His research interests are in computer architecture with emphasis on resilience and energy-aware applications of emerging devices, and he has published approximately 200 articles on topics related to the field. He was a recipient of the Joseph M. Bidenbach Outstanding Engineering Educator Award from the IEEE in 2008. He is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS.