A Coherent Hybrid SRAM and STT-RAM L1 Cache Architecture for Shared Memory Multicores

Jianxing Wang, Yenni Tim
Weng-Fai Wong, Zhong-Liang Ong

School of Computing
National University of Singapore
Singapore, 117417
wangjx, timyn, wongwf, zoo@nus.edu.sg

Zhenyu Sun, Hai (Helen) Li

Swanson School of Engineering
University of Pittsburgh
Pittsburgh, PA 15261
zhs25, hal66@pitt.edu

Abstract— STT-RAM is an emerging NVRAM technology that promises high density, low energy and a comparable access speed to conventional SRAM. This paper proposes a hybrid L1 cache architecture that incorporates both SRAM and STT-RAM. The key novelty of the proposal is the exploition of the MESI cache coherence protocol to perform dynamic block reallocation between different cache partitions. Compared to the pure SRAM-based design, our hybrid scheme achieves 38% of energy saving with a mere 0.8% IPC degradation while extending the lifespan of STT-RAM partition at the same time.

I. Introduction

The concern about energy consumption in the current technology nodes has driven researchers to examine alternative solutions. To reduce the increasing leakage energy in conventional SRAM-based caches, several non-volatile random-access memory (NVRAM) technologies [1] are being investigated as candidates for future on-chip processor caches, and off-chip primary memory. Among these NVRAM solutions, *spin-torque transfer RAM* (STT-RAM) [2] has emerged as one of the promising candidates. Besides its non-volatility, STT-RAM offers an access (read) speed comparable to SRAM, a high integration density comparable to DRAM, and is compatible with the CMOS process.

However, there are three main challenges in deploying STT-RAM as on-chip caches. The first two are its long write latency and high dynamic write energy. Many architectural solutions to mitigate these issues [3, 4, 5, 6, 7, 8, 9] have been investigated, mostly with the STT-RAM used in last level caches. Sun et. al. [10] has recently proposed architectures for pure STT-RAM L1 caches. Still, there is a third challenge that few of the earlier work had considered: reliability. In this paper, we would like to address all three challenges in using STT-RAM in the highest level of the cache hierarchy.

The key insights of the paper are that under a shared memory multicore environment, most of the private L1 cache blocks have predictable behaviors during an application's execution, and the transition between different cache coherency states provides a good approximation of a block's future behavior. Based on these observations, we propose a novel hybrid private L1 cache design scheme that operates with a small SRAM partition, and a larger STT-RAM partition. Our approach not only brings energy benefits to processors, but also significantly reduces the performance impact as well as the risk of STT-RAM cell failure. In particular, this paper makes the following contributions:

- We present a hybrid L1 cache architecture design that attempts to mitigate the performance and energy penalty induced by the slow and energy-consuming write operation of STT-RAM.
- We propose three different cache block migration schemes that are based on the MESI protocol. Simulation shows that while compared to a conventional pure SRAM-based cache design, our approach achieved significant energy savings with minor IPC degradation.
- We investigate the write endurance issue of STT-RAM and evaluate the improvement achieved by our hybrid cache architecture. We found that the write endurance of STT-RAM cache in our hybrid design is significantly improved compared to a pure STT-RAM-based design.

II. STT-RAM Basics

The data storage unit inside a STT-RAM cell is the *magnetic tunnel junction* (MTJ). Fig. 1 shows a typical one transistor, one MTJ (1T-1MTJ) cell structure. The MTJ has two ferromagnetic layers that are separated by an oxide tunnel barrier. The ferromagnetic layer at the bottom (the reference layer) comes with a fixed magnetization direction while that for the top layer (the free layer) can be changed. When the magnetization direction of the free layer is parallel (anti-parallel) to the reference

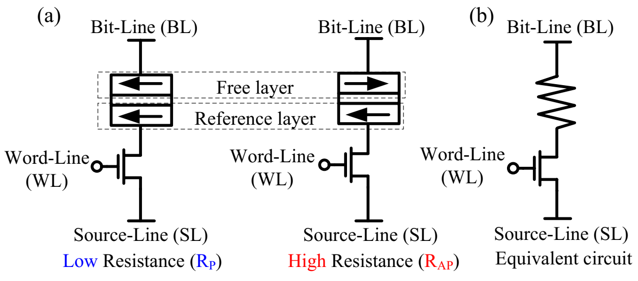

Fig. 1. STT-RAM cell structure

978-1-4799-2816-3/14/$31.00 ©2014 IEEE

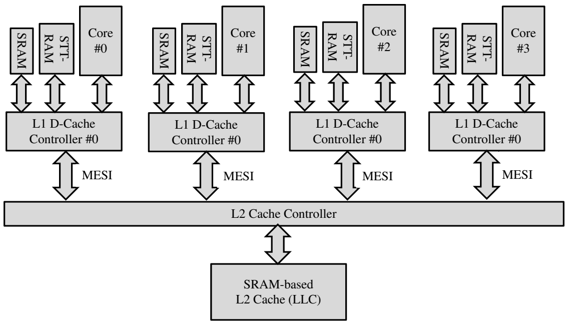

Fig. 2. The hybrid cache hierarchy.

layer, the overall resistance of the device is in low (high) state, which can be read out by injecting a sensing current. Conceptually, logical '1' is represented by low conductivity while logical '0' corresponds to the high conductive state. To write the device, a strong polarized current is applied from source-line to bit-line, which switches the magnetization direction of the free layer and turns the cell into parallel or anti-parallel state.

The *data retention time* (the expected time until a bit flip appears) of the STT-RAM cell can vary from 10+ years to micron-second level, depending on the MTJ designs [10].

III. Hybrid L1 Cache Architecture

A. The Proposed Hybrid Cache Hierarchy

Fig. 2 illustrates our proposed hybrid cache hierarchy for a quad-core machine. Each of the processor core has a private L1 cache comprising of a SRAM and a STT-RAM partition while sharing a unified L2 cache. The L1 cache controller is responsible for handling cache access operations, and transferring cache blocks between the two partitions when necessary.

B. A Naïve Solution

Due the high write latency and write energy of STT-RAM, it is desirable to move writes to the SRAM partition, and only read from the STT-RAM partition. However, the difficulty lies in predicting a cache block's access pattern. A naïve solution would allocate all read-miss blocks to STT-RAM, and write-miss blocks to SRAM when a cache miss happens. This method assumes for every cache block, the first operation on it would be the dominant one for this block until it is evicted. However, such an assumption does not guarantee to hold all the time. For example, during an application's warming up phase, cache behavior is generally unstable, and would easily invalidate this assumption.

Fig. 3 illustrates a read operation under the naïve implementation. Both cache partitions are accessed simultaneously, and at most one partition would produce a cache hit. On a cache hit, the corresponding data would be sent via a 2-to-1 MUX to the processor. The MUX is selected by the hit signal. In any case, the additional delay caused by the MUX is negligible when compared to a pure SRAM design as the two accesses are done in parallel.

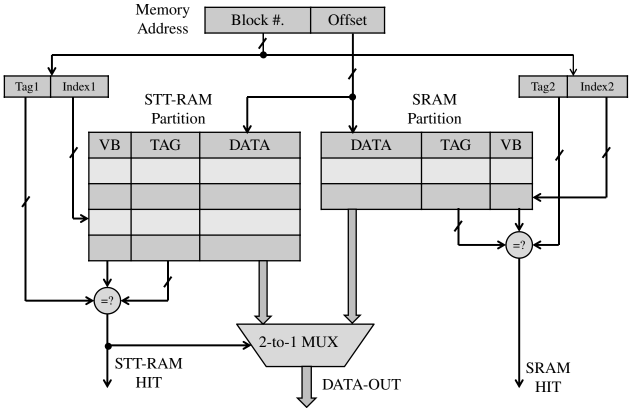

Fig. 3. Read from the hybrid cache.

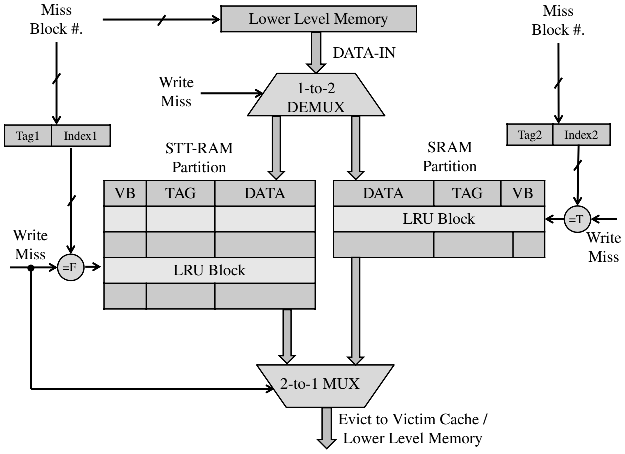

Fig. 4. Miss handling for the hybrid cache.

Under this hybrid model, cache miss is defined by none of the partitions produces a hit. When it happens, a prediction is made and the data block is loaded into the partition that potentially can maximize the performance and energy benefits. Fig. 4 illustrates this process. A DEMUX is added at the top to direct the input data from lower level memory to corresponding cache partition based on the prediction. Since STT-RAM has a longer write latency than SRAM, read-misses require more time to handle than write-misses. In addition, the *hit ratio* (HR) of each partition can be calculated separately by

$$HR_{SRAM} = \frac{\#Hits_{SRAM}}{\#Hits_{SRAM} + \#Misses_{write}}$$

$$HR_{STT-RAM} = \frac{\#Hits_{STT-RAM}}{\#Hits_{STT-RAM} + \#Misses_{read}}$$

C. Analysis of Cache Block Behavior under MESI

Although the naïve prediction method does reduce the energy consumption by a certain amount when compared to the pure SRAM baseline, it incurs a relatively large impact on performance (Section IV.B). The critical point is to mitigate the performance penalty induced by those ill-placed blocks which reside in a less beneficial cache partition.

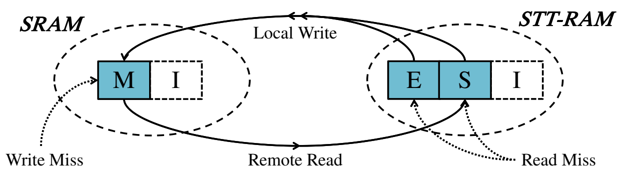

Fig. 5. Immediate transfer (IT) diagram.

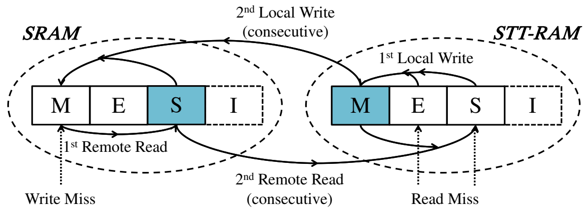

Fig. 6. Delayed transfer (DT) diagram.

Under the MESI protocol, if at one point in time a cache block is in the MODIFIED (M) state, it must have been written to before, and more importantly, it probably will be written to *again* exclusively by the current owner in the near future. In other words, such blocks are likely to be overwritten many times, and might benefit more by staying in the SRAM partition rather than in STT-RAM partition. On the other hand, EXCLUSIVE (E) and SHARED (S) states indicate that the block is a clean copy for now, and is likely to remain clean as a read-only exclusive/shared copy. Hence, it is better to keep such blocks in the STT-RAM partition of the cache. In addition, during a program's runtime, it can happen that a cache block may have to be moved from one partition to the other in order to take the advantages of both caches. Thus, some mechanisms are required to take care of the block transfer decision.

D. Transfer Mechanism 1 (Immediate Transfer)

Our first proposed block transfer mechanism is called the *immediate transfer policy* (IT). Fig. 5 shows a block's transition state and movement under this policy. While the naïve solution is based entirely on temporal locality, IT takes things a step further. A remote read operation hit on SRAM partition would cause the corresponding block to be evicted, and transferred to the STT-RAM partition in the new S state, while a local write hit on a STT-RAM partition makes the block dirty, and necessitate its transfer to the SRAM partition of the cache under its new M state. Note that here we only consider those operations that can change the coherent state, so there is no extra information required. Besides, the way of handling cache misses is the same as in the naïve solution.

Note that the order of handling the actual memory request and transferring the affected cache blocks can potentially cause differences in performance and energy saving, but we found that the variations is generally small, and can be safely ignored. Throughout this paper, we assume a block transfer completes before serving the memory request.

E. Transfer Mechanism 2 (Delayed Transfer)

Although IT does result in a better overall performance than the naïve approach, it fails to consider certain cases. In a typical scenario of a single producer with multiple consumers, some cache blocks may keep changing their states between M and S. IT may therefore be too aggressive under this circumstance. The *delayed transfer policy* (DT) differs from IT by delaying the transfer triggered by the first operation. Fig. 6 illustrates the transfer decision making process. Block migration only happens when two consecutive operations satisfying the transfer condition are encountered. In particular, to move a cache block from SRAM partition to STT-RAM partition, we need to have two remote reads with no intervening writes, while in order to move a block from STT-RAM partition to SRAM partition, two consecutive local writes must have be encountered.

The flowchart of DT is shown in Fig. 7. In the implementation, an extra *transfer determine* bit (TD bit) is added to each cache blocks. All newly-allocated blocks will have the TD bit reset to '0', but the runtime set and reset policy is different between SRAM partition and STT-RAM partition. In the SRAM partition, a remote read will set the TD bit to '1' while any write will have it reset to '0'. On the other hand, in the STT-RAM partition, only local writes set the TD bit to '1' and any other reads will reset it to '0'. Hence, when a potential transfer-trigger operation is encountered in a cache partition, TD bit can be used to determine whether a transfer is actually needed. Unlike IT, this transfer policy does not impose any MESI state restriction of a cache block in the caches, hence we may encounter blocks with E or S state in the SRAM partition, as well as M blocks in STT-RAM partition. The main objective here is to eliminate the thrashing effect where a block keep moving between the two partitions without receiving enough references.

IV. Evaluations

A. Experiment Setup

To evaluate our proposed design, we use the cycle-accurate simulator MARSSx86 [11] to model a conventional multicore x86-64 processor with a 2-level cache hierarchy as shown in Fig.2. Ten diverse multi-threaded workloads from PARSEC 2.0 [12] benchmark suite were chosen for the experiment. The complete list of simulator parameters is given in Table I. Both

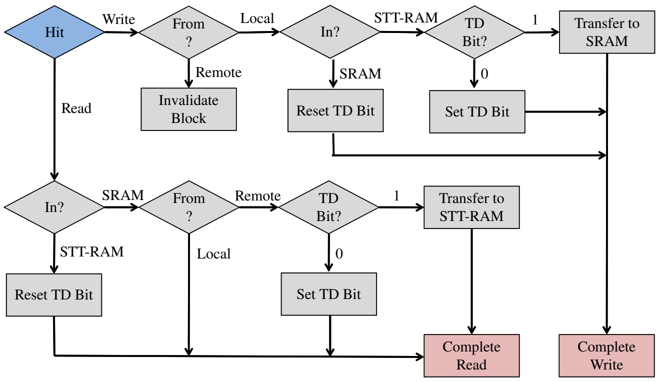

Fig. 7. Flowchart of DT under cache hit.

TABLE I
SIMULATION PLATFORM.

Processor Core	3GHz frequency, 4 cores
Pipeline Width	4
Functional Units	2 ALUs, 2 FPUs, 2 LUs, 1 SU
ROB / LSQ Entries	128 / 80
In-flight Branches	24
ITLB/DTLB Entries	32 / 32
Cache Block Size	64-byte
L1 I-Cache (SRAM)	64KB, 8-way, 3-cycle latency
L1 D-Cache (SRAM)	4-64KB, 8-way, 3-cycle latency
L1 D-Cache (STT-RAM)	64/128KB, 8-way, 3-cycle read latency, 9-cycle write latency
Coherent Protocol	Illinois MESI
Shared L2 Cache (SRAM)	2MB, 16-way, 15-cycle latency

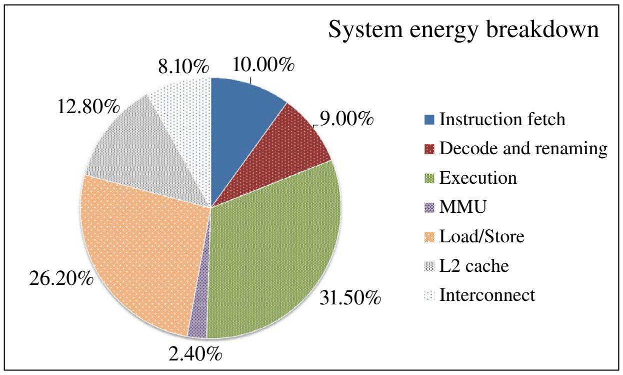

Fig. 8. Energy breakdown for each pipeline stage.

of the SRAM and STT-RAM cache energy and latency numbers were generated using NVSim [13] with a 32nm technology node assumed. To improve the write performance of STT-RAM, its non-volatility is relaxed with a retention time of $26.5\mu s$ [10]. This assumes that a conventional DRAM-style refresh scheme is used to maintain data integrity. The baseline reference is a pure 64KB SRAM cache and we took care not to exceed the silicon area of this when exploring possible hybrid cache size configurations.

Before evaluating the energy consumption of our hybrid cache architecture, we shall first show the energy distribution for each component within the processor. Fig. 8 presents a typical breakdown of energy for each pipeline stage during runtime using the McPAT tool [14] with our processor configuration (Table I) as input. The statistics were generated according to the event log of MARSSx86's cycle-accurate simulation. In particular, the load/store unit, and the L1 data cache in particular (which is part of the load/store unit) consumes more than $1/4$ and $1/5$ of the total energy, respectively. This result justify our effort in reducing the energy consumption of L1 cache. To simplify the discussion, we will only focus on the energy consumption of L1 data cache for the rest of the paper.

B. Performance

Fig. 9 shows the normalized *instructions-per-cycle* (IPC) for the different hybrid cache configurations running under the IT policy. All the hybrid cache configurations maintain a comparable performance to the pure SRAM baseline. In the worst case, only an average 1.5% performance degradation was recorded for the smallest size configuration. This is even better than the best case in the naïve solution (Fig. 11).

However, a particularly bad degradation of nearly 8% loss in IPC was recorded in `swaptions`. Unlike the others, `swaptions` requires more exclusive blocks for writing during its execution. This is why there is a continuous performance improvement as the size of the SRAM partition increases. Without a sufficiently large SRAM partition, doubling the size of STT-RAM partition will not compensate for the performance loss.

Fig. 11 compares the performance for the different transfer scenarios. Across all the cache size combinations, both IT and DT perform better than the naïve solution. On the other hand, the IPC differences between IT and DT are marginal. The largest one recorded is in the smallest cache size configuration (4KB SRAM + 64KB STT-RAM), where DT outperforms IT by around 1%. Notice that a serious IPC drop of 5.7% is recorded when the baseline is directly replaced with pure STT-RAM. This proves the importance of our hybrid design.

C. Energy

The total energy consumption is modeled by the sum of three components: *leakage energy*, *refresh energy* and *dynamic energy*. Besides the usual runtime read/write energy, dynamic energy in our context also includes the energy for probing both cache partitions, and the *transfer energy* when one of the block transfer policies is applied.

Fig. 10 shows the normalized energy consumption for the hybrid cache with IT policy. In general, our design saves more energy in those read-intensive benchmarks such as `vips` and `swaptions`. Even for the most write-intensive benchmark, `raytrace`, around 30% of energy saving is achieved in the best case. When taking IPC into consideration, a medium cache size combination (8KB SRAM + 64KB STT-RAM) provides the best trade-off among the four configurations tested.

Within the three transfer scenarios (Fig. 11), the largest energy saving occurs in the smallest cache configuration, though

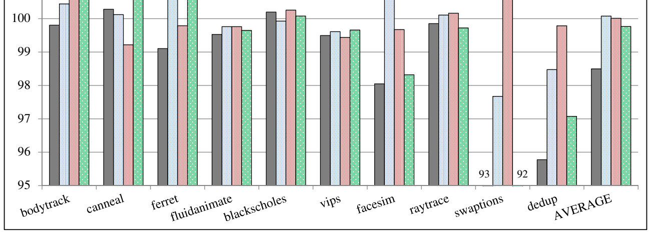

Fig. 9. Normalized IPC for hybrid cache with IT policy.

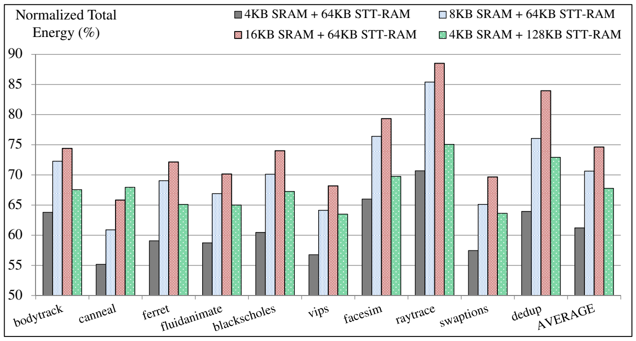

Fig. 10. Normalized total energy for hybrid cache with IT policy.

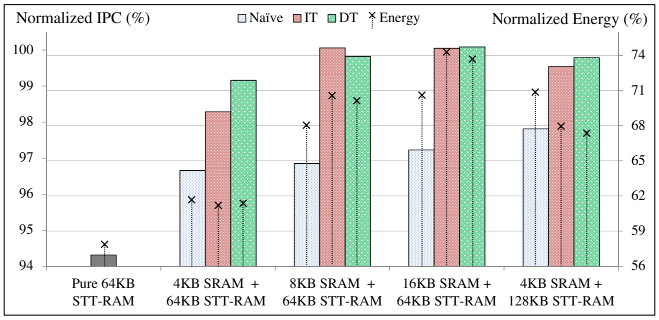

Fig. 11. Normalized IPC and total energy averaged from all workloads.

the lowest IPC rates were recorded for all of them. This is because the leakage and refresh energy consumed in a larger cache is more significant than the energy spent in extra dynamic operations that are caused by either a higher miss ratio (smaller cache) or ill-placed blocks (naïve solution). Although the pure STT-RAM solution consumes the least energy, the difference between it and the best hybrid scheme is merely 3%, while the IPC drop is significant. Overall, we consider DT(4KB SRAM + 64KB STT-RAM) to be the best hybrid cache configuration since it provides the best energy-IPC trade-off (saves 38% energy under a mere 0.8% IPC degradation).

D. Impact of Retention Time

The retention time of STT-RAM cells is related to the thermal barrier Δ of an MTJ, which can be expressed as $t = C \times e^{k\Delta}$ where t is the retention time, and C and k are fitting constants [9]. Any variations in the planar area and the thickness of the MTJ affect the thermal barrier, and thus impact the retention time. However, given a particular set of read/write latency and the cache size, the lowest possible retention time under a DRAM-style refresh scheme is bounded by $\#cache\,blocks \times (read\,latency + write\,latency) \times cycle\,time$.

Fig.12 shows the IPC and energy consumption under various retention time values for a 4KB SRAM + 64KB STT-RAM hybrid cache with IT policy. The data are normalized to the perfect case when the refresh is completely eliminated (inf). In general, a lower retention time causes more refresh conflicts and thus increases the latency for memory requests. It also consumes more energy due to a more frequent refresh schedule, but the impact become marginal while the refresh period is larger than $32\mu s$. We demonstrate these results is because STT-RAM are not yet in commercial deployment and the manufacturing parameters are not fully known. An extreme short retention period can seriously downgrade performance and lead to a higher energy consumption.

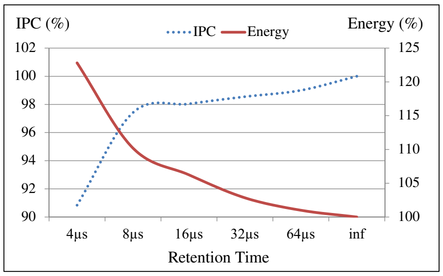

Fig. 12. IPC and energy consumption for various retention time values of a 4KB SRAM + 64KB STT-RAM hybrid cache with IT policy.

E. STT-RAM Endurance Study

The *write endurance* issue for STT-RAM has so far been only considered in the context of last level caches [15]. However, the issue is even more pressing when STT-RAM technology is employed in L1 caches as they have a much higher write activity than last level caches. Although a prediction of 10^{15} programming cycles [16] is often cited as the write endurance for STT-RAM, real experiments thus far have showed that the achieved endurance is only about 10^{13} write cycles [2]. This would be a severe constraint on any pure STT-RAM L1 solution, and is one of the key motivations for our hybrid design.

Fig. 13 demonstrates the average number of writes to STT-RAM partition happened within each CPU cycle. Among all the benchmarks, `facesim` has the highest average writes per cycle. Consider the case when the writes are perfectly distributed to all cache blocks in `facesim` for a pure STT-RAM design, each block would need to stand 2.4×10^5 writes per second. Under a conservative estimate, each block has a lifespan of about 1.3 years.

However, further analysis shows that writes are not at all evenly distributed. This is true within a single cache as well as across different private caches. We observed that some blocks are seldom used while several other blocks suffer from

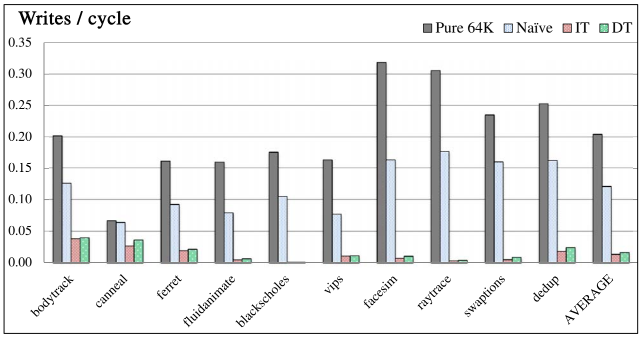

Fig. 13. Average number of STT-RAM writes occurs in each CPU cycle. All hybrid caches use 4K SRAM + 64K STT-RAM with IT policy.

TABLE II
STT-RAM CACHE LIFESPAN ESTIMATION BASED ON THE WRITE FREQUENCY OF `FACESIM`.

	Average partition	**Worst partition**	**Worst block**
Pure STT-RAM	1.3 years	0.3 years	$<$ 22 mins
Hybrid Naïve	3.5 years	1.0 year	0.9 hr
Hybrid IT	41.2 years	6.9 years	51.6 hrs
Hybrid DT	32.9 years	7.0 years	54.3 hrs

All hybrid caches use 4K SRAM + 64K STT-RAM.

a huge amount of writes. For example, about 15% writes target at a particular block in `facesim`, while the corresponding cache partition receives nearly 50% writes among the four private caches. In another words, assuming a 3 GHz clock and a write endurance of 10^{13} write cycles, that STT-RAM block may fail within half an hour if no proper measure is taken. Due to the low-latency requirement and high access frequency of L1 caches, existing wear-levelling methods for last level caches [15] are not feasible.

Table II lists the average and worst case lifespan of the STT-RAM cache for `facesim` under the write endurance assumption of 10^{13} programming cycles. The "average partition" column assumes that the writes are perfectly distributed to all of the private caches while the "worst partition" only assumes the writes are evenly distributed within each private cache partition. In the "worst block" case, the actual number of writes per cache block is computed. Compared to a pure 64KB STT-RAM solution, the lifespan of the most write-intensive cache block in a hybrid configuration with the DT policy is increased by up to 150×, while the worst cache partition lifespan is increased by 2333%. Although the DT policy performed slightly better than IT policy in the worst case endurance, it suffered from a lower average lifespan because the blocks in STT-RAM recieve more writes by the delayed migration. Note that the naïve solution for hybrid cache is still insufficient for actual deployment.

In practice, caches would come with redundancy or error correction code to improve reliability. Also, it is very unlikely to see such a sustained high write frequency on a particular block especially for personal workload. Thus the worst case block lifespan should be much longer. Nonetheless, the risk exists, especially for the current state-of-the-art in STT-RAM technology. Our experiments showed clearly that the proposed hybrid architecture can significantly reduce this risk.

V. CONCLUSION

This paper proposed a hybrid L1 cache architecture that uses both conventional SRAM as well as the new STT-RAM technology. By exploiting the MESI cache coherence protocol, our design mitigates the impact of STT-RAM's high write latency on IPC performance, while reducing the overall energy consumption compared to a pure SRAM-based design. In addition, our proposal enhances the reliability of STT-RAM, making it more suitable for actual deployment. As the CMOS technology continues to scale, increasing energy consumption and design complexity demands for more power-efficient designs. We believe that our hybrid cache is an attractive architecture for the next-generation non-volatile computing.

ACKNOWLEDGMENTS

This research was supported in part by the Singapore Ministry of Education Research Grant MOE2010-T2-1-075.

REFERENCES

[1] R. Bez and A. Pirovano, "Non-volatile memory technologies: emerging concepts and new materials," *Materials Science in Semiconductor Processing*, vol. 7, no. 4, pp. 349–355, 2004.

[2] Z. Diao, Z. Li, S. Wang, Y. Ding, A. Panchula, E. Chen, L. Wang, and Y. Huai, "Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory," *Journal of Physics: Condensed Matter*, vol. 19, no. 16, 2007.

[3] W. Xu, H. Sun, X. Wang, Y. Chen, and T. Zhang, "Design of Last-Level On-Chip cache using Spin-Torque transfer RAM (STT RAM)," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 19, no. 3, pp. 483–493, 2011.

[4] C. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. Stan, "Relaxing non-volatility for fast and energy-efficient STT-RAM caches," in *IEEE 17th International Symposium on High Performance Computer Architecture*, pp. 50–61, 2011.

[5] A. Jadidi, M. Arjomand, and H. Sarbazi-Azad, "High-endurance and performance-efficient design of hybrid cache architectures through adaptive line replacement," in *International Symposium on Low Power Electronics and Design*, pp. 79–84, 2011.

[6] M. Rasquinha, D. Choudhary, S. Chatterjee, S. Mukhopadhyay, and S. Yalamanchili, "An energy efficient cache design using spin torque transfer (STT) RAM," in *ACM/IEEE International Symposium on Low-Power Electronics and Design*, pp. 389–394, 2010.

[7] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM l2 cache for CMPs," in *IEEE 15th International Symposium on High Performance Computer Architecture*, pp. 239–249, 2009.

[8] J. Li, C. Xue, and Y. Xu, "STT-RAM based energy-efficiency hybrid cache for CMPs," in *IEEE/IFIP 19th International Conference on VLSI and System-on-Chip*, pp. 31–36, 2011.

[9] A. Jog, A. K. Mishra, C. Xu, Y. Xie, V. Narayanan, R. Iyer, and C. R. Das, "Cache revive: Architecting volatile stt-ram caches for enhanced performance in cmps," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pp. 243–252, 2012.

[10] Z. Sun, X. Bi, H. H. Li, W. Wong, Z. Ong, X. Zhu, and W. Wu, "Multi retention level STT-RAM cache designs with a dynamic refresh scheme," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 329–338, 2011.

[11] A. Patel, F. Afram, S. Chen, and K. Ghose, "MARSS: a full system simulator for multicore x86 CPUs," in *Proceedings of the 48th Design Automation Conference*, DAC '11, pp. 1050–1055, 2011.

[12] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," in *International Conference on Parallel Architectures and Compilation Techniques*, pp. 72–81, 2008.

[13] X. Dong, C. Xu, Y. Xie, and N. Jouppi, "NVSim: a Circuit-Level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, 2012.

[14] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture.*, pp. 469–480, 2009.

[15] Y. Chen, W. Wong, L. Hai, and C. Koh, "Processor caches built using multi-level spin-transfer torque ram cells," in *Proceedings of the 2011 ACM/IEEE international symposium on Low power electronics and design*, ISLPED '11, pp. 73–78, 2011.

[16] F. Tabrizi, "The future of scalable stt-ram as a universal embedded memory," *Embedded.com*, 2007.