Architectural Exploration to Enable Sufficient MTJ Device Write Margin for STT-RAM Based Cache

Hongbin Sun[1], Chuanyin Liu[1], Tai Min[2], Nanning Zheng[1], *Fellow, IEEE*, and Tong Zhang[3]

[1]Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China
[2]MagIC Technologies, Inc., Milpitas, CA 95035 USA
[3]Rensselaer Polytechnic Institute, Troy, NY 12180 USA

As a promising nonvolatile memory technology, magnetic tunnel junction (MTJ) based spin-torque transfer RAM (STT-RAM) has recently attracted much attention. However, recent device research suggested that, in order to maintain sufficient MTJ write margin to prevent device breakdown, MTJs may be subject to unconventionally high random write error rates (e.g., 10^{-3} and above) as memory cell size is being scaled down. In this paper, we aim to develop a STT-RAM cache design solutions that can effectively tolerate high MTJ write error rates at small performance and implementation cost, which makes it much easier to maintain sufficient MTJ write margin and hence push the STT-RAM scalability envelope. Using the full system simulator PTLsim and a variety of benchmarks, we show that the proposed architecture design can readily accommodate MTJ write error rate upto 2% at the penalty of less than 3% processor performance degradation, less than 10% silicon area overhead, and negligible energy consumption overhead.

Index Terms—**Cache memory, error correction code, fault tolerance, STT-RAM, write margin.**

I. Introduction

STT-RAM has recently attracted much attention, and it is expected to offer suitable scalability toward the 22-nm node and below [1]–[5]. Since STT-RAM can potentially enable high-capacity on-chip nonvolatile data storage with relatively high access speed and very low leakage power consumption, many recent computer architecture research efforts have explored the potential and demonstrated appealing possible advantages of using STT-RAM to implement low-level on-chip caches in microprocessors (e.g., see [6]–[8]).

The basic storage element of STT-RAM is magnetic tunnel junction (MTJ), and the nonvolatile data storage is realized by modulating the resistance of MTJs. To program MTJs in STT-RAM, the write voltage across MTJs should be high enough to ensure desired MTJ state switching, and meanwhile cannot be too high in order to avoid junction barrier breakdown. Therefore, it is crucial to realize sufficient MTJ device write margin between the programming voltage and device dielectric breakdown voltage. Unfortunately, recent device research revealed two new phenomena, including backhopping and low probability bifurcated switching (LPBS) [9]–[12], that can substantially reduce MTJ write margin and hence seriously threaten the scalability of STT-RAM. In order to maintain sufficient MTJ device write margin as memory cell size is being scaled down, these two physics phenomena make MTJs subject to unconventionally high random write error rates (e.g., 10^{-2} and above) [9]. Complementary to on-going material/device research that aims to tackle this grand challenge by significantly reducing MTJ write error rate through material/device engineering and innovations, architecture level techniques that can effectively tolerate relatively high MTJ write error rates may greatly contribute to solving this grand challenge and ensuring future scalability of STT-RAM, which motivates this work.

This paper concerns the design of STT-RAM based cache memory that can tolerate relatively high random MTJ write error rates. We propose an architectural fault tolerance strategy, which tolerates random write errors by explicitly storing the location of each write error in a separate storage memory. Compared with error correction code (ECC) that uniformly protects all the data block, this approach demands much less amount of extra storage space for realizing memory fault tolerance. As a result, when being combined with a simple recursive write-read-verify scheme, it can very effectively reduce the overall performance degradation under high random write error rates. The effectiveness of applying this architecture design technique in modern processor cache memory has been demonstrated by carrying out extensive simulations. The full-system simulator PTLsim [13] and Cacti memory modeling tool [14] are used for processor simulation and cache memory modeling at the 65nm technology node. Simulation results show that, by tolerating random MTJ write error rate as high as 2%, these design techniques only induce less than 3% computing system performance degradation on average over a wide spectrum of SPEC2000 benchmarks, while only incurring less than 10% area overhead and negligible dynamic energy consumption overhead. The results show that architecture level techniques can indeed play an important role in ensuring sufficient MTJ write margin and hence the future scalability of STT-RAM.

II. Device Write Margin of STT-MRAM

As discussed in the previous publication [6]–[8], [15], each typical 1T1MTJ STT-RAM cell contains one MTJ as the storage element and one nMOS transistor as the access control device. As the basic storage element, each MTJ has two ferromagnetic layers separated by one Mg-oxide (MgO) barrier layer. The resistance of each MTJ depends on the relative magnetization directions of the two ferromagnetic layers, i.e., when the magnetization is parallel (or anti-parallel), MTJ is in a low (or high) resistance state. In STT-RAM, parallel or anti-parallel magnetiza-

Manuscript received September 17, 2011; revised January 13, 2012 and February 24, 2012; accepted February 27, 2012. Date of publication April 05, 2012; date of current version July 20, 2012. Corresponding author: H. Sun (e-mail: sunsir@mail.xjtu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMAG.2012.2193589

tion is realized by steering a write current directly through MTJs along opposite directions. During write or read operations, the bit-line (BL) and source-line (SL) establish appropriate voltage drop across the cell, and the word-line (WL) turns on/off the nMOS transistor to realize memory cell access control.

To successfully switch the MTJ resistance state, the across-MTJ voltage and hence through-MTJ write current should be sufficiently large. However, a too high across-MTJ voltage may result in dielectric breakdown of the MgO barrier. Currently, MTJs with 1.2 nm or less MgO layer are necessary in order to realize a low resistance-area product (RA) and hence reduce the required through-MTJ write current. A low write current can directly translate into smaller access nMOS transistor and the overall memory cells size. The dielectric breakdown of such thin MgO barriers under voltage/current stress therefore becomes critical to the success of STT-RAM technology. As a result, it is crucial to maintain sufficient write margin between the MTJ write voltage and the device breakdown voltage.

Recent device research for the MTJ with longitudinal magnetic anisotropy (LMA) discovered new phenomena that can significantly erode the MTJ device write margin for highly scaled technology nodes [9]–[12]. One of them is the so-called backhopping phenomenon in which the magnetization of free layer can switch back to its original state after a successful write operation. Another one is called low probability bifurcated switching (LPBS), which can further largely reduce the slope of the write error rate vs. write voltage curve, as illustrated in [9]. We note that these two phenomena occur randomly on all the MTJ devices and are not associated with device defects. Clearly, the newly discovered backhopping and LPBS phenomena for highly scaled technology nodes can largely narrow the write margin, which seriously threatens the scalability of STT-RAM. More recent studies on the write error rate (WER) of LMA type MTJ suggests that even though at relative low writing voltage, up to 80% of Vc0, the WER behaves as expected since both thermal heating and field-like term are negligible [16]. But for the required reliable writing of a device, the applied write voltage needs to be higher than Vc0, especially at fast switching region less than 50ns. Then the existence of the “stagnation point” or “zero-torque point” can significantly increase the required switching voltage, could be several times larger than Vc0 to obtain reliable low WER [17]. This increase is mainly due to thermal fluctuation other than the magnitude of the onset spin torque. The situation is somewhat better for the MTJ with perpendicular magnetic anisotropy (PMA) which reduces the Vc0 significantly, hence less required switching write voltage, lowering both field-like term and thermal fluctuation. This potential advantage has been extensively investigated to reduce the WER of PMA type MTJs [18]–[20]. However, the WER problem of LMA-type MTJs is still untouched.

In parallel to intensive on-going material/device research that aim to tackle this grand challenge by significantly reducing MTJ write error rate while still maintaining sufficient margin, intuitively it can be highly desirable if the architecture and application can effectively tolerate high random memory write error rates. Since backhopping and LPBS induced random errors only

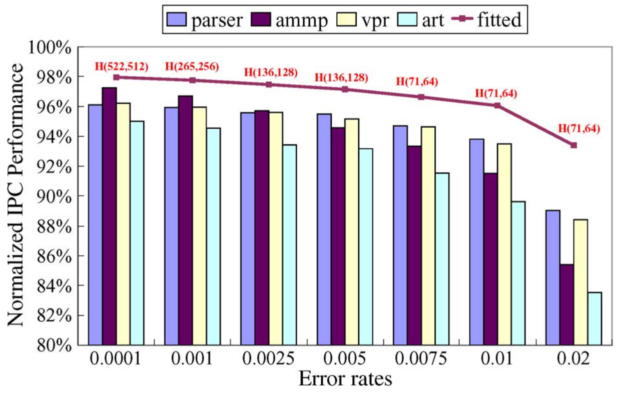

Fig. 1. Normalized IPC performance vs. write error rate when choosing one Hamming code for each error rate.

occur when MTJs are being written, it is intuitive that we can use a *recursive write-read-verify* scheme to mitigate the errors. Its basic idea is simple: Right after memory write, we read the data out and verify whether every bit has been correctly written, and if write errors are identified, subsequent write-read-verify operations will be re-issued until all the bits are correctly written. This scheme can be directly combined with conventional fault tolerance techniques, such as error correction codes, to more effectively handle relatively high random MTJ write error rates. The key is to leverage extra architectural fault tolerance capability to largely reduce the average number of write-read-verify recursions, hence directly reduce the overall performance degradation. Combining recursive write-read-verify with the conventional ECC has been extensively studied in the recent work [15].

Nevertheless, as the random write error rate increases, we have to use a stronger ECC that demands more coding redundancy and hence increases the size of each data block. Clearly, a larger data block is subject to more write errors and accordingly demands even stronger ECC. This confliction tends to largely degrade the effectiveness of such ECC-based solution under very high write error rates (e.g., 1% and above). Fig. 1 shows the normalized IPC (instruction per cycle) performance under different write error rate using Hamming code. We can see that, under error rate of 2%, the average IPC reduces to 93.8%. In particular, for benchmark “art”, the IPC performance can be even as low as 84%. We note that, as the location of backhopping and LPBS induced write error can be identified right after each write-read-verify operation, it is very inefficient to protect STT-RAM cache with conventional fault tolerant solutions, which are originally designed to compensate unpredictable defects. This directly motivates the work in this paper, which aims to find more efficient fault tolerance technique to fully utilize the location-aware advantage of random MTJ write error.

III. A Design Solution for Tolerating High Random MTJ Write Error Rates

As we discussed in the above section, using the conventional ECC technique in STT-RAM cache to tolerate write errors becomes inefficient when the error rate is high. In this section, we present a new design solution that can more effectively tolerate higher write error rates at smaller computing performance penalty.

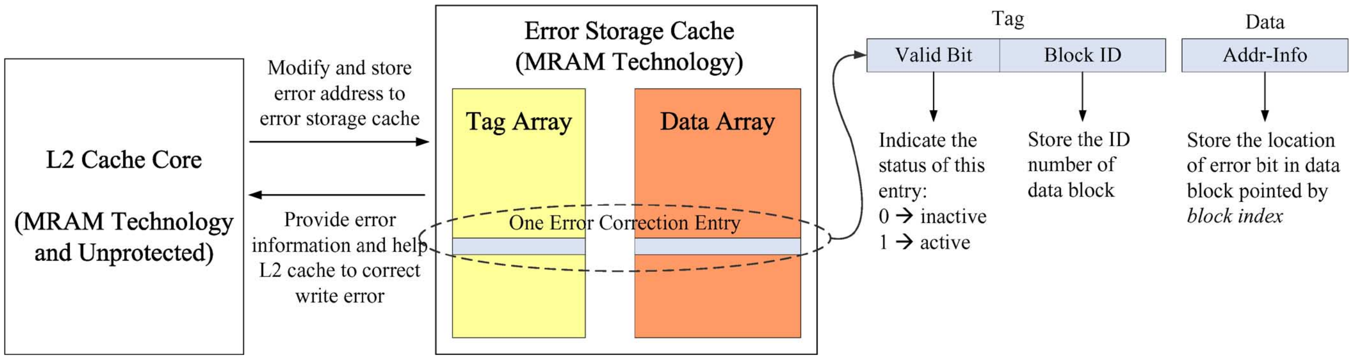
L2 Cache Core
(MRAM Technology and Unprotected)
Modify and store error address to error storage cache
Provide error information and help L2 cache to correct write error
Error Storage Cache
(MRAM Technology)
Tag Array
Data Array
One Error Correction Entry
Tag
Data
Valid Bit
Block ID
Addr-Info
Indicate the status of this entry:
0 → inactive
1 → active
Store the ID number of data block
Store the location of error bit in data block pointed by *block index*

Fig. 2. Overview of proposed STT-RAM L2 cache architecture.

A. Architecture Design

The basic idea is to explicitly store the location of write error using a separate small cache called *error storage cache*. In contrast to ECC, this approach can utilize the location-aware advantage of MTJ write error to dynamically provide just-enough error correction capability for different cache sub-blocks using just-enough extra redundant storage space. As a result, it can more effectively reduce the required number of write-read-verify recursions. Assuming we target at STT-RAM based L2 cache, Fig. 2 shows the overview of this proposed cache architecture, which consists of two main blocks:

1) A conventional STT-RAM L2 cache core that follows the current L2 cache design practice (e.g., see [21]) and does not have any protection against the write errors. Hence, the cache core has to use the recursive write-read-verify operation to identify and further correct the random MTJ write errors;
2) A small error storage cache that stores the location of all the write errors being identified by the write-read-verify operation. By providing the addresses of error bits when necessary, this error storage cache can help to correct the error bits and hence maintain the data integrity of the overall L2 cache.

As shown in Fig. 2, the error storage cache contains both tag and data arrays just as the conventional cache does. The locations of write errors are stored in the unit of error correction entry. Each error correction entry consists of three components: *valid bit*, *block index*, and *addr-info*. The valid bit and block index locate in the tag array, while the addr-info locates in the data array. The valid bit indicates the status of the corresponding entry: setting the valid bit to ‘0’ (false) means that the entry is inactive and data in block index and addr-info are invalid; while setting the valid bit to ‘1’ means that data in this entry should be used to correct the write error. The block index stores the ID of cache block that contains write error(s) and addr-info stores the location of one error bit within this cache block.

The error storage cache aims to reduce the number of main cache core write-read-verify recursions and hence reduce the computing system performance degradation. Fig. 3 and Fig. 4 illustrate the corresponding overall cache write and read operation flow, respectively. During the cache write, L2 cache core first checks its tag array to determine whether it is a hit or miss.

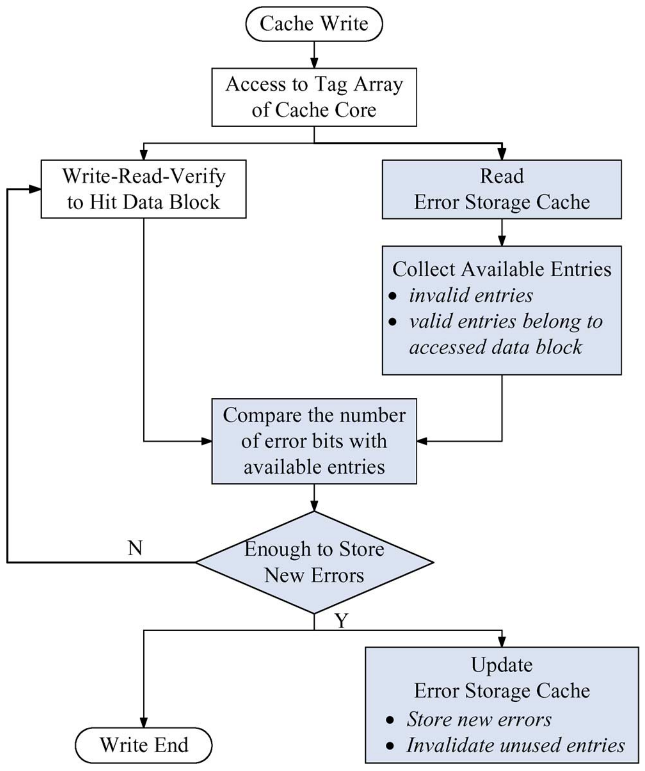
Cache Write
Access to Tag Array of Cache Core
Write-Read-Verify to Hit Data Block
Read Error Storage Cache
Collect Available Entries
• *invalid entries*
• *valid entries belong to accessed data block*
Compare the number of error bits with available entries
Enough to Store New Errors
N
Y
Write End
Update Error Storage Cache
• *Store new errors*
• *Invalidate unused entries*

Fig. 3. Cache write operation flow in the proposed cache architecture.

In case of cache hit, the data array and error storage cache are accessed in parallel. The cache core writes to the data array, reads the data out and verifies whether any random write errors occur. In the meantime, the error storage cache identifies its available error correction entries, which include both empty entries of which the valid bit is false and active entries that contain the error location information associated with the cache block being accessed. If the available entries are sufficient to store the error information for current cache write, we update the error storage cache with the new locations and finish the cache write. Otherwise, we have to issue another write-read-verify operation to reduce or eliminate the residual write errors in the corresponding cache block.

Regarding cache memory read as illustrated in Fig. 4, data array and error storage cache are only accessed in case of L2 cache hit. If the error storage cache misses, it means the cache

TABLE I
CACTI REPORT FOR L2 CACHE, ESC(4K,128), ESC(2K,128), ESC(1K,128), ESC(512,128) AND ESC(256,128)

Memory	Size	Assoc.	Access time (*ns*)	Cycle time (*ns*)	Tag access time (*ns*)	Data access time (*ns*)	Dyn. read energy (*nJ*)	Area (mm^2)
L2 cache	4MB	16	2.85	0.29	0.86	1.99	0.64	10.18
ESC(4K,128)	576KB	128	1.85	0.21	-	-	0.19	1.88
ESC(2K,128)	288KB	128	1.61	0.21	-	-	0.13	0.95
ESC(1K,128)	144KB	128	1.40	0.21	-	-	0.11	0.50
ESC(512,128)	72KB	128	1.24	0.21	-	-	0.09	0.27
ESC(256,128)	36KB	128	1.14	0.21	-	-	0.08	0.15

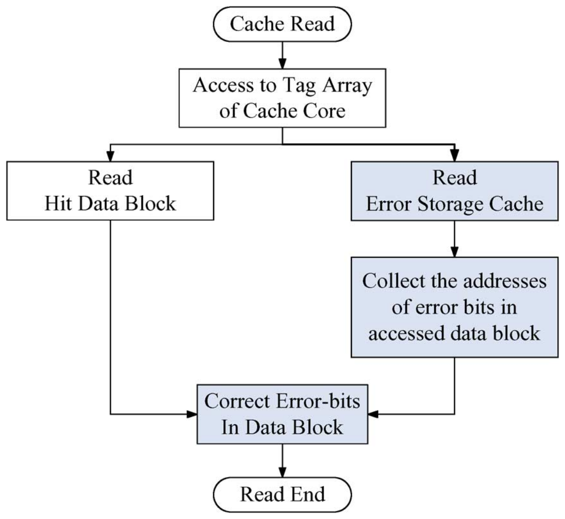
Cache Read
Access to Tag Array of Cache Core
Read Hit Data Block
Read Error Storage Cache
Collect the addresses of error bits in accessed data block
Correct Error-bits In Data Block
Read End

Fig. 4. Cache read operation flow in the proposed cache architecture.

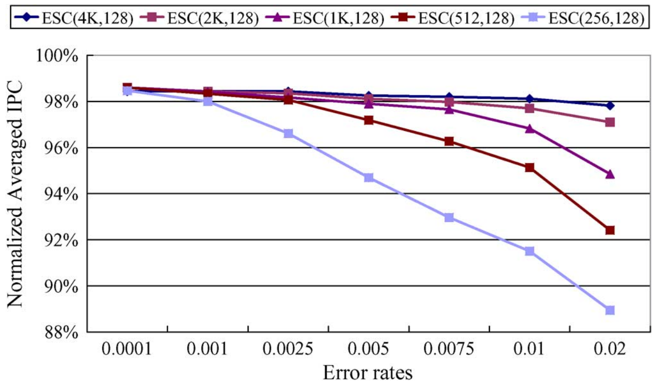

Fig. 5. Normalized IPC performance for proposed STT-RAM L2 cache that is protected by error storage cache (ESC).

block read from L2 cache core is error free and the operation is finished; Otherwise, the information of error locations is read from the error storage cache and used to correct the write errors in the cache block read from L2 cache core. Since we only need to flip the value of error bit according to the error location information, the error correction latency can be considerably shorter compared with ECC techniques. As a result, the error correction operation of proposed error storage cache has little impact on the read latency of L2 cache.

Another important design issue of error storage cache is how to avoid/mitigate the potential performance degradation due to error storage cache update. As error storage cache has to be updated after L2 cache write, an immediate write operation to error storage cache may block the subsequent L2 cache read. To address this issue, we propose two techniques, including (1) we use a small SRAM buffer in the error storage cache to temporarily hold the data, and update the error storage cache during the next L2 cache write. (2) To reduce the number of write-read-verify recursions in error storage cache update and hence reduce write latency of error storage cache, we use Hamming code to protect the error storage cache following the same principle as the conventional ECC-based design solution. Since the error storage cache is considerably smaller than the L2 cache core, the ECC-induced overhead tends to be very small in the entire L2 cache. If more than one errors occur during error storage cache update, we have to repeat the write-read-verify operations to write the corresponding data again until no more than one write errors are left.

B. An Experimental Case Study

We carry out simulations to evaluate the effectiveness of the above proposed cache architecture and further compare with the approaches previously presented in [15] in terms of performance, power consumption, and area overhead. During the simulation, we use the same experimental methodology to [15], which includes a cycle-accurate full system simulator PTLsim [13], modified cache modeling tool Cacti [14], and the selected 14 benchmarks from the SPEC2000 suite [22].

In this case study, we choose the set size of error storage cache to be 128, which means that the error storage cache is 128-way set associative. By setting the associativity as 128, we vary the set number of error storage cache to evaluate the performance vs. cost overhead tradeoffs, where a larger set number means more redundancy resource is available in error storage cache. We use ESC (set number, associativity) to represent the configuration of error storage cache, where ESC stands for error storage cache. Five representative ESC configurations are selected in our experiment, including ESC(4K,128), ESC(2K,128), ESC(1K,128), ESC(512,128) and ESC(256,128). By using the modified STT-RAM Cacti tool, we estimate the access time, dynamic energy and area overhead, and list the results in Table I. We note that, to reduce the number of write-read-verify recursions in error storage cache, Hamming code (522,512) is used to protect the data in each set.

Fig. 5 shows the performance comparison among the above five different configurations of error storage cache. As we can intuitively expect, the configuration with larger set number has a relatively better performance. With the increase of random write error rate, average IPC performance of all the five configuration

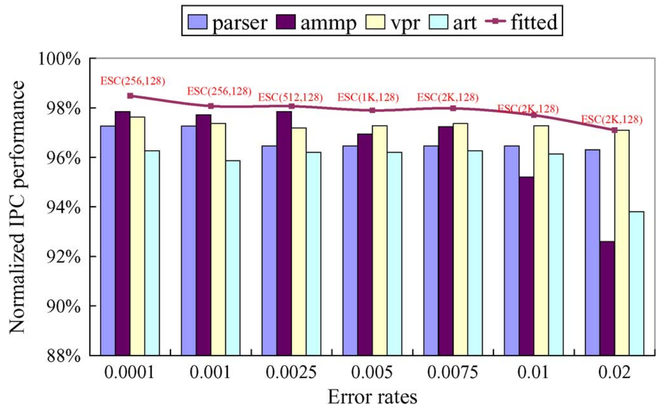

Fig. 6. Normalized IPC performance for proposed STT-RAM L2 cache that is protected by error storage cache (ESC).

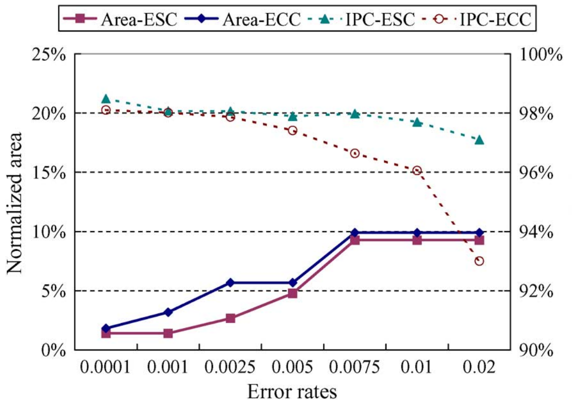

Fig. 7. Normalized area overhead comparison between proposed error storage cache (ESC) and error correction code (ECC) technique.

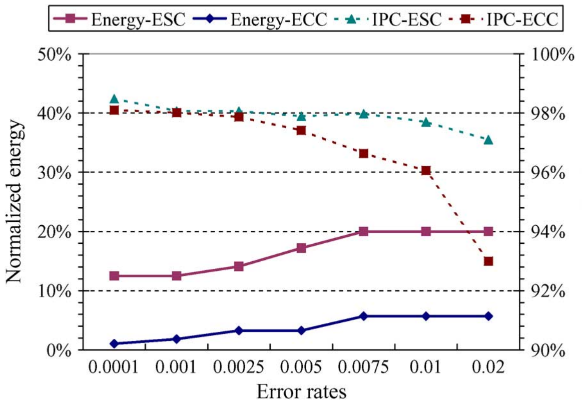

Fig. 8. Normalized dynamic energy overhead comparison between proposed error storage cache (ESC) and error correction code (ECC) technique.

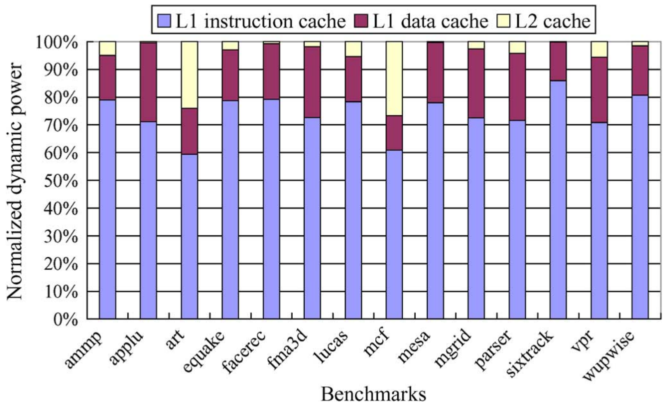

Fig. 9. Normalized dynamic power consumption comparison among L1 instruction cache, L1 data cache and L2 cache.

reduces. However, two of five configurations, i.e. ESC(4K,128) and ESC(2K,128) have a much better performance that even at the error rate of 2% the performance degradation is only less than 1.1% with respect to that at the error rate of 0.01%. This means that these two configurations are considerably effective on reducing the performance degradation induced by write-read-verify recursions at very high random write error rates. We then manipulate the simulated data in Fig. 5 and draw a fitted curve by varying the set number according to the error rate. The fitted curve and four worst case benchmarks are shown in Fig. 6. As illustrated in Fig. 6, as the error rate increases from 0.01% to 2%, the IPC performance remains almost constant. At the error rate of 2%, ESC(2K,128) improves the performance from about 93% to 97.1% on average compared with the case when H(71,64) code is being used as shown in Fig. 1. For those benchmarks with heavy L2 cache access workload, the performance improvement can be even higher. Take "art" for example, its normalized IPC performance is improved from 83.8% to 93.9% when using error storage cache instead of ECC to realize memory fault tolerance, representing almost 10% improvement.

We then compare the area and energy overhead when using either error storage cache or ECC to realize memory fault tolerance. We consider the normalized area overhead and dynamic read energy overhead per block regarding to the ideal L2 cache for both error storage cache and Hamming code. The modeling results of error storage cache is listed in Table I and the comparison results are illustrated in Fig. 7 and Fig. 8. As shown in Fig. 7, the area overhead of using error storage cache is much smaller than using ECC when the error rate is below 0.5%. And their area overhead becomes similar when the error rate further increases. However, as shown in Fig. 8, the dynamic energy overhead of using error storage cache is much higher than using ECC, e.g., at the error rate of 2%, its energy overhead reaches up to 20%, while that of using ECC is only 6%.

The above comparison between using error storage cache or ECC to realize memory fault tolerance may suggest that, in spite of the advantage of much better performance, using error storage cache tends to incur a considerable energy dissipation penalty. However, the dynamic energy consumption of L2 cache is actually not as significant as we may intuitively expect. As shown in Fig. 9, for the great majority of benchmarks, the energy consumption of L2 cache only occupies less than 5% with respect to the whole cache hierarchy. Hence the energy consumption of using error storage cache is relatively insignificant, especially when taking into consideration that using STT-RAM cache will save a considerable percentage of leakage power.

In addition, we conduct experiments to examine the sensitivity of the proposed architecture to different configuration parameters. Table II shows the normalized IPC performance on average with the associativity size of L2 cache varying from 8-way to 32-way and error storage cache varying from 64-way

TABLE II
THE ESC'S SENSITIVITY TO SET ASSOCIATIVITY

	L2 cache			**Error storage cache**		
Associativity	8	16	32	64	128	256
Nor. IPC	97.05%	97.05%	97.06%	96.59%	97.05%	97.12%

to 256-way, respectively. As the cache set associativity size increases, the average performance almost stays constantly. The above results indicate that the proposed approach of using error storage cache is robust to different cache system parameters.

IV. CONCLUSION

The newly discovered backhopping and low probability bifurcated switching phenomena make it a grand challenge to maintain sufficient MTJ device write margin and meanwhile ensure very low random MTJ write errors for future highly scaled STT-RAM. This seriously threatens the scalability and the practical promise of STT-RAM. In parallel to on-going material/device research that investigates to reduce MTJ write error rate while still maintaining sufficient margin, this paper aims to address this grand challenge from the architecture and application perspective. Focusing on using STT-RAM to implement low level on-chip cache, we present a fault tolerance cache architecture design that can tolerate relatively high random MTJ write error rates. As the simulation results demonstrate, these design approaches can tolerate up to 2% of random write error rate in STT-RAM based L2 cache and meanwhile maintain very good processor speed performance at small implementation overhead.

ACKNOWLEDGMENT

This research was funded in part by grants from the National Natural Science Foundation of China (No. 61103048) and the National Science and Technology Major Project of China (No. 2010ZX01032-001-001-5).

REFERENCES

[1] Semiconductor Industry Association, The International Technology Roadmap for Semiconductors (ITRS) [Online]. Available: http://www.itrs.net/reports.html

[2] K. Kim and G. Jeong, "Memory technologies for sub-40nm node," in *Proceedings of IEEE International Electron Devices Meeting (IEDM)*, Dec. 2007, pp. 27–30.

[3] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of candidate device technologies for storage-class memory," *IBM J. Res. Develop.*, vol. 52, no. 4/5, pp. 449–464, 2008.

[4] E. Chen and D. Apalkov *et al.*, "Advances and future prospects of spin-transfer torque random access memory," *IEEE Trans. Magn.*, vol. 46, no. 6, pp. 1873–1878, Jun. 2010.

[5] W. Zhu, H. Li, Y. Chen, and X. Wang, "Current switching in MgO-based magnetic tunneling junctions," *IEEE Trans. Magn.*, vol. 47, no. 1, pp. 156–160, Jan. 2011.

[6] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *Proc. IEEE Int. Symp. High Performance Computer Architecture*, Jun. 2009, pp. 239–249.

[7] 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 Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 3, pp. 483–493, 2011.

[8] K. Lee and S. H. Kang, "Development of embedded STT-MRAM for mobile system-on-chips," *IEEE Trans. Magn.*, vol. 47, no. 1, pp. 131–136, Jan. 2011.

[9] T. Min *et al.*, "A study of write margin of spin torque transfer magnetic random access memory technology," *IEEE Trans. Magn.*, vol. 46, no. 6, pp. 2322–2327, Jun. 2010.

[10] T. Min, J. Z. Sun, R. Beach, D. Tang, and P. Wang, "Back-hopping after spin torque transfer induced magnetization switching in magnetic tunneling junction cells," *J. Appl. Phys.*, vol. 105, no. 07D126, 2009.

[11] S.-C. Oh *et al.*, "Bias-voltage dependence of perpendicular spin-transfer torque in asymmetric MgO-based magnetic tunnel junctions," *Nature Phys.*, vol. 5, pp. 898–902, 2009.

[12] J. Z. Sun and M. C. Gaidis *et al.*, "High-bias backhopping in nanosecond time-domain spin-torque switches of MgO-based magnetic tunnel junctions," *J. Appl. Phys.*, vol. 109, no. 07D109, Apr. 2009.

[13] M. T. Yourst, "PTLsim: A cycle accurate full system x86-64 microarchitecture simulator," in *Proc. IEEE Int. Symp. Performance Analysis of System and Software*, Apr. 2007.

[14] CACTI: An integrated cache and memory access time, cycle time, area, leakage, and dynamic power model [Online]. Available: http://www.hpl.hp.com/research/cacti/.

[15] H. Sun, C. Liu, N. Zheng, T. Min, and T. Zhang, "Design techniques to improve the device write margin for MRAM-based cache memory," in *Proc. ACM Great Lakes Symp. VLSI*, May 2011, pp. 97–102.

[16] R. Heindl, W. H. Rippard, S. E. Russek, M. R. Pufall, and A. B. Kos, "Validity of the thermal activation model for spin-transfer torque switching in magnetic tunnel junctions," *J. Appl. Phys.*, vol. 109, no. 073910, 2011.

[17] R. Heindl, W. H. Rippard, S. E. Russek, and A. B. Kos, "Physical limitations to efficient high-speed spin-torque switching in magnetic tunnel junctions," *Phys. Rev. B*, vol. 83, no. 054430, pp. 1–4, 2011.

[18] S. Ikeda, K. Miura, and H. Yamamoto *et al.*, "A perpendicular-anisotropy CoFeB-MgO magnetic tunnel junction," *Nat. Mater.*, vol. 9, pp. 721–724, 2010.

[19] D. C. Worledge, G. Hu, D. W. Abraham, and J. Z. Sun *et al.*, "Spin torque switching of perpendicular TaCoFeBMgO-based magnetic tunnel junctions," *Appl. Phys. Lett.*, vol. 98, no. 022501, 2011.

[20] J. J. Nowak, R. R. Robertazzi, and J. J. Sun *et al.*, "Demonstration of ultralow bit error rates for spin-torque magnetic random-access memory with perpendicular magnetic anisotropy," *IEEE Magn. Lett.*, vol. 2, no. 3000204, 2011.

[21] J. L. Hennessy and D. A. Patterson, *Computer Architecture a Quantitative Approach*, 4th ed. San Mateo, CA: Morgan Kaufmann, 2006.

[22] Standard Performance Evaluation Corporation [Online]. Available: http://www.spec.org, 2000/2006