Design of Last-Level On-Chip Cache Using Spin-Torque Transfer RAM (STT RAM)

Wei Xu, Hongbin Sun, Xiaobin Wang, Yiran Chen, *Member, IEEE*, and Tong Zhang, *Senior Member, IEEE*

***Abstract*—Because of its high storage density with superior scalability, low integration cost and reasonably high access speed, spin-torque transfer random access memory (STT RAM) appears to have a promising potential to replace SRAM as last-level on-chip cache (e.g., L2 or L3 cache) for microprocessors. Due to unique operational characteristics of its storage device magnetic tunneling junction (MTJ), STT RAM is inherently subject to a write latency versus read latency tradeoff that is determined by the memory cell size. This paper first quantitatively studies how different memory cell sizing may impact the overall computing system performance, and shows that different computing workloads may have conflicting expectations on memory cell sizing. Leveraging MTJ device switching characteristics, we further propose an STT RAM architecture design method that can make STT RAM cache with relatively small memory cell size perform well over a wide spectrum of computing benchmarks. This has been well demonstrated using CACTI-based memory modeling and computing system performance simulations using *SimpleScalar*. Moreover, we show that this design method can also reduce STT RAM cache energy consumption by up to 30% over a variety of benchmarks.**

***Index Terms*—Cache memories, magnetic tunneling junction, spin-torque transfer.**

I. Introduction

As CURRENT mainstream memory technologies such as SRAM, DRAM, and flash memories are all facing serious scaling problems, there have been a resurgence of interest in searching for highly scalable universal memory [1]. Magnetoresistive RAM (MRAM) is one of the most promising candidates that have attracted a lot of attentions [2]–[4]. The basic building block in MRAM is magnetic tunneling junction (MTJ), and the data storage is realized by configuring the resistance of MTJs into one of two possible states (i.e., high-resistance state and low-resistance state). Different from the first generation of MRAM that uses explicitly generated magnetic fields to switch the state of MTJs, a new technique called spin-torque transfer (STT) uses through-MTJ current of spin-aligned electrons to switch the state of MTJ, which has a much greater scalability potential and hence has received a growing interest [5]–[9].

Manuscript received May 22, 2009; revised August 17, 2009. First published December 11, 2009; current version published February 24, 2011.

W. Xu and T. Zhang are with the Electrical and Computer Science Engineering (ECSE) Department, Rensselaer Polytechnic Institute, Troy, NY 12180 USA.

H. Sun is with the Institute of AI&R, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China.

X. Wang and Y. Chen are with the Seagate Technology, Bloomington, MN 55435 USA.

Digital Object Identifier 10.1109/TVLSI.2009.2035509

This work is interested in using STT RAM to implement last-level on-chip cache memory, e.g., L2 or L3 cache, in microprocessors. STT RAM is a promising alternative to SRAM to realize last-level on-chip cache for three main reasons:

1) Last-level cache memory tends to occupy a large on-chip silicon area, hence it is highly desirable to maximize its storage density. With a simple cell structure consisting of only one nMOS transistor and one MTJ and great scalability, STT RAM could achieve much higher storage density than SRAM. Meanwhile, for large capacity last-level cache memory in which interconnect delay tends to play a bigger role, STT RAM can achieve comparable access speed as SRAM.
2) It only incurs three more mask layers to embed MTJ devices on logic dies [10], which makes embedded STT RAM an economically viable option. Moreover, such modest extra mask cost may be potentially offset by the die area reduction by using STT RAM to replace SRAM as last-level on-chip cache.
3) SRAM-based last-level cache memory tends to incur a significant amount of leakage power consumption. Being nonvolatile, STT RAM can dramatically reduce the leakage power consumption.

Design of STT RAM cache memory is nontrivial because STT RAM is subject to an inherent tradeoff between write latency and read latency, which will be elaborated later. Such write versus read latency tradeoff is essentially determined by the size of the nMOS transistor in each STT RAM cell. The contribution of this work is two-fold.

1) We first quantitatively study how such write vs. read latency trade-off may affect processor performance over a wide spectrum of benchmarks when using STT RAM as on-chip L2 cache. Results show that different types of benchmarks may have conflicting expectations on the STT RAM L2 cache write vs. read latency tradeoff, i.e., some benchmarks favor the use of relatively small nMOS transistors in memory cells while some other benchmarks favor the use of relatively large nMOS transistors in memory cells.
2) Leveraging MTJ device switching characteristics, we propose a *dual-write-speed* STT RAM cache design method that can improve the performance of STT RAM cache with relatively small nMOS transistors. As a result, this method enables the designers to use relatively small nMOS transistors in STT RAM cache memory cells, which not only reduces cache memory footprint but also delivers satisfactory computing performance over a wide spectrum of benchmarks. Moreover, this dual-write-speed

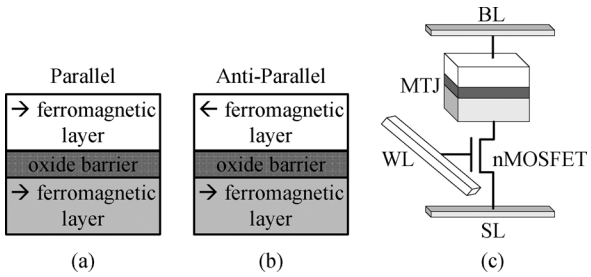


Fig. 1. (a) Parallel (low resistance) and (b) anti-parallel (high resistance) of one MTJ, and (c) structure of a 1T1MTJ STT RAM cell.

design method can also reduce the average STT RAM write energy consumption. We enhanced the CACTI 5 [11], a widely used cache memory modeling tool, to support the STT RAM cache memory modeling, and apply *SimpleScalar 3.0* [12] to carry out extensive computing system performance over a variety of benchmarks to study the involved design tradeoffs and demonstrate the effectiveness of this proposed dual-write-speed design method.

The remainder of this paper is organized as follows. Section II reviews the STT RAM basics and prior related work, and Section III discusses the CACTI-based STT RAM cache modeling. Section IV demonstrates the impact of STT RAM write vs. read latency tradeoff on computing system performance when using STT RAM to implement L2 cache, and Section V presents the proposed dual-write-speed STT RAM cache design method. Conclusions are drawn in Section VI.

II. Background and Prior Work

A. STT RAM Basics

As the basic storage element in STT RAM, each MTJ has two ferromagnetic layers separated by one oxide 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, as illustrated in Fig. 1(a) and (b), respectively. In STT RAM, parallel and anti-parallel magnetization are realized by steering a write current directly through MTJs along opposite directions. Let R_h and R_l denote the high and low MTJ resistance, respectively, $(R_h - R_l)/R_l$ is conventionally referred to as tunneling magneto-resistance ratio (TMR). One important MTJ device parameter in STT MRAM is its write current threshold: To successfully switch the resistance state, the current steered through MTJ must be higher than the write current threshold.

Fig. 1(c) shows the typical 1T1MTJ STT RAM cell structure. Each cell contains one MTJ as the storage element and one nMOS transistor as the access control device. 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 sustain a write current higher than the MTJ write current threshold, the size (or width) of the nMOS transistor within each cell must be sufficiently large. As a result, the nMOS transistor is relatively large and tends to dominate the overall memory cell size [5], [6], [13], [14].

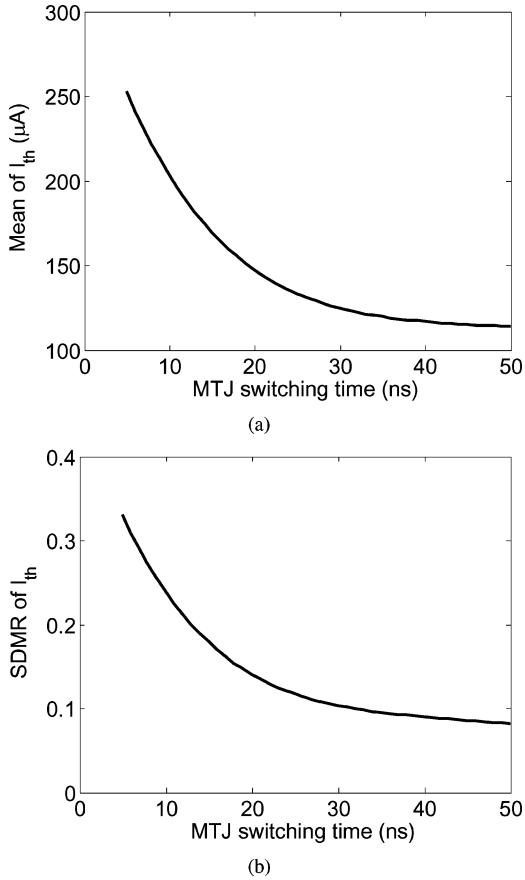


Fig. 2. Illustration of the relationship between (a) MTJ switching time and mean of write current threshold at 45 nm node and (b) MTJ switching time and standard deviation versus mean ratio (SDMR) at 45 nm node based on the analysis and data presented in [15].

To successfully switch the MTJ resistance state, the through-MTJ write current not only has to be larger than the write current threshold but also should sustain for a certain amount of time, called MTJ switching time. As demonstrated in [15], for any given MTJ, its switching time and write current threshold are variable and correlated, i.e., as MTJ switching time reduces, the write current threshold will increase. Moreover, due to the inevitable process variation, particularly as the technology scales down, write current threshold may vary from one MTJ to the other under the same MTJ switching time. Such write current threshold variation can be characterized by mean and standard deviation. The write current threshold standard deviation versus mean ratio (SDMR) and MTJ switching time are also correlated, i.e., as MTJ switching time reduces, the SDMR will increase. Based on the analysis and data presented in [15], Fig. 2(a) shows the relationship between MTJ switching time and mean of write current threshold at 45 nm node, and Fig. 2(b) shows the relationship between MTJ switching time and SDMR at 45 nm node.

Due to such strong correlations between MTJ switching time and write current threshold characteristics, STT RAM design involves an inherent tradeoff between write latency and read latency, i.e., to reduce memory write latency, we may have to sacrifice memory read latency. This will be quantitatively demonstrated in Section III and can be intuitively explained as follows.

To reduce memory write latency, MTJ switching time has to be reduced, leading to larger MTJ write current threshold mean and SDMR values. As a result, the nMOS transistor within each memory cell should have a larger size, which will increase the bit-line parasitic capacitance and hence increase the memory sensing latency.

B. Prior Work

In conventional toggle mode MRAM design, magnetic fields are explicitly generated and used to switch the state of MTJs. A 4-Mb toggle MRAM with an asynchronous SRAM-like interface was demonstrated using 0.18-μm CMOS technology in [16], which can achieve a 25 ns cycle time. In [17], a 16-Mb toggle MRAM was demonstrated using 0.18-μm CMOS technology with a 30 ns cycle time. STT RAM was presented for the first time by SONY in 2005 [5]. A 2 Mb STT RAM chip with a 40 ns access time using 0.2 μm technology was demonstrated in [13]. In [9], an efficient simulation tool was developed to predict STT RAM yield and a corresponding statistical optimization methodology was proposed to enhance memory yield at an early stage of the design cycle.

As pointed out in the above, because of its high storage density with superior scalability, low integration cost and reasonably high access speed, STT RAM appears to have a promising potential to replace SRAM as on chip cache for microprocessors. Dong *et al.* [4] presented a cache model for STT RAM-based cache and evaluate the system performance of STT RAM-based cache under a 3-D integrated system framework. Li *et al.* [18] evaluated the system performance degradation of using STT RAM as both L1 and L2 cache based on their STT RAM yield and density model, and presented a stretched write cycle technique to accommodate the long MTJ switching time by holding the word-line high for multiple cycles during the write operations. We note that a direct use of such word-line hold-time stretching scheme makes all memory cells subject to the same worst-case write latency. This work aims to leverage the inherent MTJ device variation to reduce the average STT RAM cache write latency as elaborated later.

III. STT RAM Cache Memory Modeling

This work concerns the use of STT RAM as last-level on-chip cache memory (e.g., L2 or L3 cache) in microprocessors. Due to the inherent STT RAM write versus read latency tradeoff, it is indispensable to develop an STT RAM cache memory modeling tool that can quantitatively estimate the memory write and read latency. Therefore, in this work, we first developed an STT RAM cache memory modeling tool based upon the version 5.0 of the widely used cache memory modeling tool CACTI [11]. As shown in Fig. 1(c), STT RAM has a memory cell structure similar to DRAM, i.e., each memory cell simply contains one data storage element and one nMOS transistor for access control. Moreover, since both STT RAM and DRAM cells have a three-terminal cell interface, STT RAM cell array has a structure very similar to that of DRAM cell array. Therefore, we modified the CACTI DRAM model to obtain the STT RAM cache model, as described below.

First, we changed the memory cell area parameter according to our STT RAM cell layout as shown in Fig. 3. Speed and area

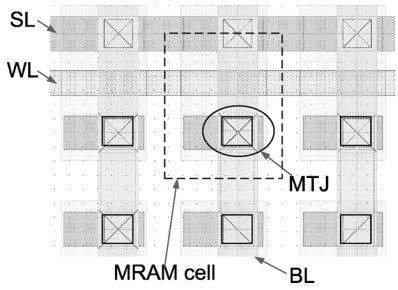


Fig. 3. STT RAM cell layout.

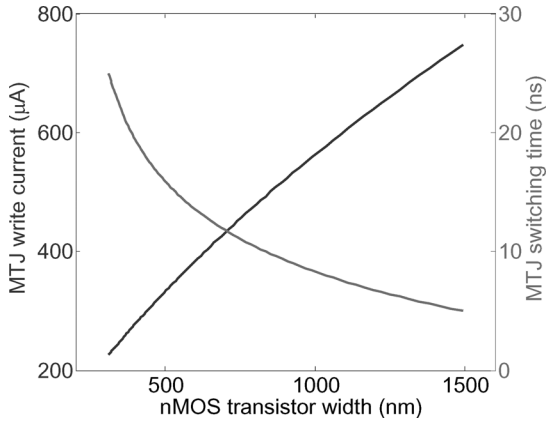


Fig. 4. Simulation results representing the relationship among MTJ switching time, nMOS transistor width, and write current at 45 nm node.

performance modeling in CACTI are decomposed into several parts, including global data/address bus H-tree routing, memory sub-array peripheral decoders, word-lines, bit-lines, and sense-amplifiers. Clearly, we can keep the modeling of H-tree routing intact. Furthermore, we keep the same modeling of sub-array peripheral decoders and word-lines. In the context of bit-lines and sense amplifiers, we carried out SPICE simulations to estimate their latency using the STT RAM sensing scheme proposed in [19] and PTM transistor model [20] at 45 nm node. Since the MTJ TMR affects the memory sensing latency, we set the high and low resistance of MTJ as 2 and 1 KΩ, respectively, at 45 nm node according to [14].

As discussed in Section II and illustrated in Fig. 2, MTJ switching time and write current threshold characteristics (including mean and standard deviation) are correlated. In this work, we size the nMOS transistor within each memory cell using the 6σ rule, i.e., the nMOS transistor should be able to sustain a write current that is 6σ larger than the mean of MTJ write current threshold. To enable the STT RAM CACTI modeling tool to explore a large spectrum of MTJ switching time, we use the data in Fig. 2 and carry out SPICE simulations at 45 nm node to quantitatively reveal the relationship among MTJ switching time, nMOS transistor width, and write current, as shown in Fig. 4. These results are embedded into the STT RAM cache memory modeling tool.

For SRAM cache, the actual delay to read or write a cell (i.e., the period during which the word-line is held high) is typically a single cycle, and the entire access latency is dominated by

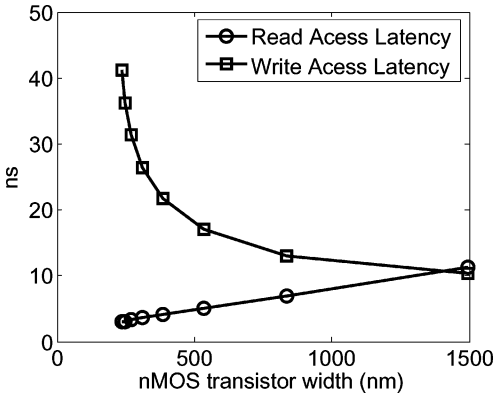


Fig. 5. STT RAM read and write access latency under different nMOS transistor width in a 4 MB STT RAM cache memory at 45 nm node.

global routing. In contrast, STT RAM cache has heavily asymmetric read and write latencies. A single cycle is used to read the memory cell and the read latency is still dominated by H-tree routing delay. But the write latency of STT RAM cache heavily depends on MTJ device switching time. As shown in Fig. 4, the MTJ device switching time is in the orders of 10 ns, which tends to be (much) longer than the routing delay. Therefore, the word-line must be held as high for multiple cycles during the write operations, as suggested in [18], which has been used in our modeling tool.

Using the developed CACTI-based STT RAM cache memory modeling tool, we studied a 4 MB STT RAM cache memory at 45 nm node. Fig. 5 shows the memory read and write access latency under different width of the nMOS transistor within each memory cell. The results clearly show a tradeoff between the STT RAM read and write latency. A larger nMOS transistor size directly increases the word-line/bit-line capacitance and memory footprint, leading to longer latencies associated with word-line/bit-line and H-tree routing. Hence, the STT RAM read latency increases as we increase the nMOS transistor size. On the other hand, since a larger nMOS transistor size can dramatically reduce MTJ switching time, which can well offset the increase of word-line/bit-line and H-tree routing, the overall STT RAM write latency tends to decrease as we increase the nMOS transistor size.

IV. Case Study: Using STT RAM as L2 Cache

Since computing system performance can be affected by both read and write speed of cache memories, we further study how the write versus read speed tradeoff inherent in STT RAM may impact the overall computing system performance when using STT RAM as L2 cache memory. In this work, we carry out simulation using the *SimpleScalar 3.0* simulator. However, it should be pointed out that *SimpleScalar 3.0* does not take into account of the cache write latency during the simulation. This is reasonable for conventional SRAM cache, since SRAM write latency is very small and hence we can use a small write buffer to ensure data are only written to cache memory during idle cycles. However, since STT RAM tends to have a much longer write latency, we must explicitly take into account of the cache write latency during simulations. Therefore, we modified the *SimpleScalar 3.0* simulator by explicitly adding a write buffer that can hold eight entries. We set that cache read operations have a higher priority than cache write operations, i.e., a cache write operation will be immediately terminated once we need to read from the cache unless the write buffer is full, and we always try to empty the write buffer during idle cycles.

TABLE I
SimpleScalar Simulation Parameters

Processor	
Functional Units	4 integer ALUs 1 integer multiplier/divider 4 FP ALUs 1 FP multiplier/divider
LSQ size	64 instructions
RUU size	64 instructions
Fetch width	4 instructions/cycle
Decode width	4 instructions/cycle
Issue width	64 instructions/cycle
Commit width	4 instructions/cycle
Fetch queue size	64 instructions
Cycle time	0.5ns
Cache and memory hierarchy	
L1 instruction cache	32KB, 2-way, 64-byte blocks 1 cycle latency
L1 data cache	32KB, 2-way, 64-byte blocks 1 cycle latency
L2 cache	4MB unified, 4-way, 64-byte blocks
Main memory	300 cycle latency
Branch logic	
Predictor Bimodal predictor size Level 1 predictor Level 2 predictor	comb. of bimodal and 2-level 16K 16K entries, 12 bit history 16K entries
BTB	16K entry, 2-way
Miss-prediction penalty	12 cycles

Table I lists the simulator configuration parameters used in this work, where STT RAM is used to realize the 4 MB L2 cache. Based upon the results shown in Fig. 5, we considered eight different instantiations of the 4 MB STT RAM L2 cache with different write versus read speed tradeoffs and hence nMOS transistor sizes and overall cache area, as listed in Table II, where the cycle time is 0.5 ns. By carrying out SPICE simulations, we estimate the STT RAM cell dynamic energy consumption of read and write operations for different STT RAM cell design cases as listed in Table II. Due to its nonvolatility feature, STT RAM cells do not consume any standby leakage power. The leakage power of STT RAM is incurred by peripheral circuits, which is also listed in Table II. For the purpose of comparison, we also list the corresponding parameters of a 4 MB SRAM L2 cache obtained from CACTI estimation.

Using the modified *SimpleScalar* simulator, we simulated all 26 SPEC2000 benchmarks, including 12 integer benchmarks and 14 floating point benchmarks. Since L2 cache access occurs only in case of L1 cache miss after the initialization phase, we recorded the occurrences of L1 instruction cache and L1 data cache misses, and accordingly calculated the miss rates as listed in Table III. Based upon the L1 cache miss rates, we categorize these 26 benchmarks into the following three types:

- Type-I benchmarks that have high L1 data cache miss rates;
- Type-II benchmarks that have low L1 data cache miss rates but high L1 instruction cache miss rates;

TABLE II
PARAMETERS OF EIGHT DIFFERENT 4 MB STT RAM L2 CACHES AND A 4 MB SRAM L2 CACHE AT 45 nm NODE

Case	Latency (# of cycles)		nMOSFET	Cache	Dynamic energy (pJ/cell)		Leakage
	Read	Write	size (W/L)	area (mm^2)	Read	Write	power (W)
R6W83	6	83	5.3	7.2	0.12	7.92	0.48
R7W63	7	63	6.0	7.6	0.13	6.67	0.48
R8W53	8	53	6.9	8.3	0.14	6.24	0.48
R9W44	9	44	8.6	9.4	0.16	5.98	0.48
R10W39	10	39	10.0	10.3	0.18	5.87	0.48
R11W35	11	35	11.9	11.6	0.21	5.79	0.51
R12W32	12	32	14.1	15.0	0.24	5.45	0.52
R14W27	14	27	18.6	16.2	0.29	5.43	0.52
SRAM	8	8	N/A	16.7	0.10	0.10	7.65

TABLE III
SPEC2000 BENCHMARKS USED IN THIS WORK

	SPEC2000	L1 cache miss rate	
		data cache	instruction cache
Type-I	ammp	5.31%	~0
	applu	4.92%	~0
	apsi	13.79%	0.14%
	art	32.49%	~0
	equake	10.88%	~0
	facerec	2.12%	~0
	fma3d	1.18%	0.08%
	galgel	6.67%	~0
	gzip	2.71%	~0
	lucas	10.16%	~0
	mcf	21.20%	~0
	mgrid	3.06%	~0
	parser	2.87%	~0
	swim	8.57%	~0
	twolf	6.63%	0.11%
	vpr	6.25%	~0
	wupwise	1.63%	~0
Type-II	eon	0.48%	0.32%
	gcc	1.39%	0.55%
	crafty	2.61%	0.27%
	vortex	0.65%	0.52%
Type-III	bzip2	0.11%	~0
	gap	0.37%	0.01%
	mesa	0.90%	~0
	perlbmk	0.18%	~0
	sixtrack	0.24%	0.01%

- Type-III benchmarks that have low miss rates for both L1 data cache and L1 instruction cache.

Fig. 6 shows the simulation results in terms of normalized instruction per cycle (IPC) when using the above eight different STT RAM L2 caches and SRAM-based L2 cache. To improve the IPC estimation accuracy, we follow the *SimPoint* simulation strategy [21] in all our simulations. The results clearly show that, with different L1 miss behaviors, these three types of benchmarks respond to various STT RAM design configurations in different manners, which can be intuitively explained as follows.

1) With high L1 data cache miss rates, Type-I benchmarks tend to incur a large number of L2 cache read and write operations. If the size of nMOS transistors within STT MRAM cells is too small (such as the cases *R6W83* and *R7W63*), L2 cache write latency will be so large that it tends to become the overall computing system performance bottleneck. There is a significant performance degradation when replacing SRAM L2 cache with STT RAM L2 cache. When we increase the size of nMOS transistors, L2 cache write latency dramatically decreases and the read access latency only slowly increases, as shown in Fig. 5. Therefore, the overall system performance tends to improve if we reduce L2 cache write latency at the cost of longer L2 cache read latency, as shown in Fig. 6. However, due to the relatively long access latency compared with SRAM L2 cache, the overall computing system performance is still worse than that of using SRAM L2 cache.
2) Type-II benchmarks have low L1 data cache miss rates and high L1 instruction cache miss rates. Since L1 instruction cache misses most likely incur L2 cache read operations, Type-II benchmarks tend to have more L2 cache read operations than L2 cache write operations. As a result, the overall computing system performance tends to improve if we use a smaller nMOS transistor within STT RAM cells (i.e., reduce L2 cache read latency at the cost of longer L2 cache write latency), as shown in Fig. 6. It is also noticed that the STT RAM cache with a smaller nMOS transistor (such as the case *R6W83*) has the similar overall computing performance with SRAM cache for Type-II benchmarks, because the smaller read latency can well offset the performance degradation due to the longer write latency, when L2 write operation number is small.
3) For Type-III benchmarks, miss rates of both L1 data cache and L1 instruction cache are low, i.e., this type of benchmarks tends to incur relatively small number of L2 cache access operations. As a result, various L2 cache write versus read speed tradeoffs tend to have little impact on the computing system performance in this context, as clearly shown in Fig. 6. Therefore, compared with SRAM cache, the performance degradations for all STT RAM cache cases are very small.

The above simulation results and discussions suggest that Type-II and Type-III benchmarks favor the use of relatively small nMOS transistors in STT RAM cache memory cells, which could not only maintain good computing system performance but also reduce the cache memory footprint. As pointed out earlier, last-level cache usually occupies a large percentage of on-chip silicon area, hence this could contribute to compensating the extra mask cost for embedding STT RAM and even reducing the overall chip fabrication cost. However, the use of small nMOS transistors tend to largely hurt the computing performance of Type-I benchmarks with high L1 data cache miss rates. To address this dilemma, in the remainder of this paper, we present an STT RAM architecture design method that can mitigate the performance degradation for Type-I benchmarks when small nMOS transistors are being used. This makes it practically feasible to use relatively small nMOS transistors in

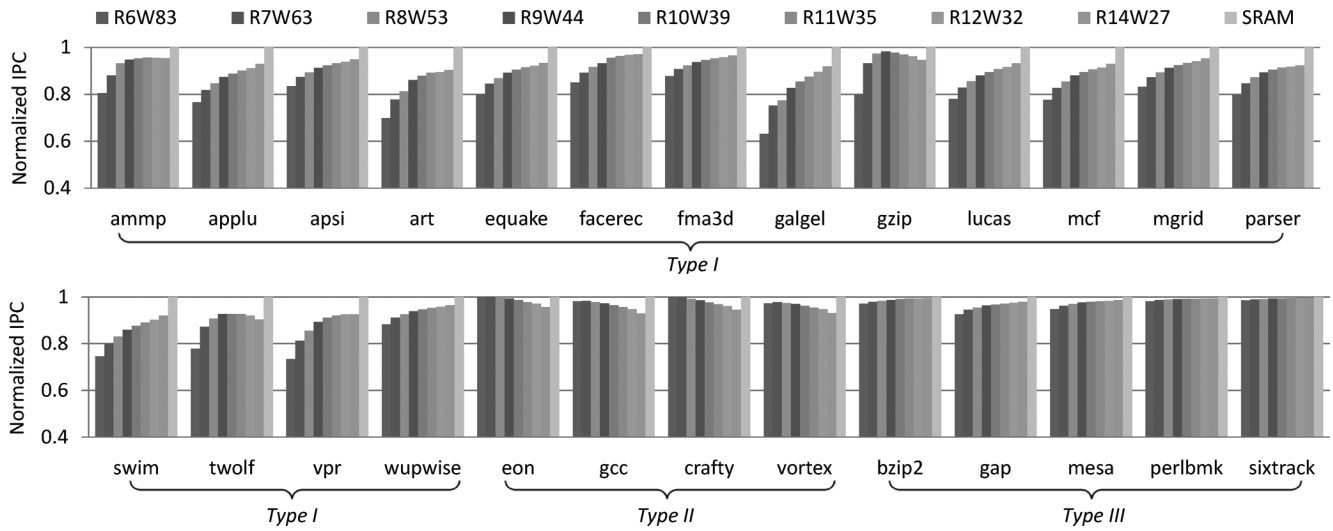


Fig. 6. Normalized IPC under different STT RAM L2 cache cell design and comparison with using SRAM L2 cache.

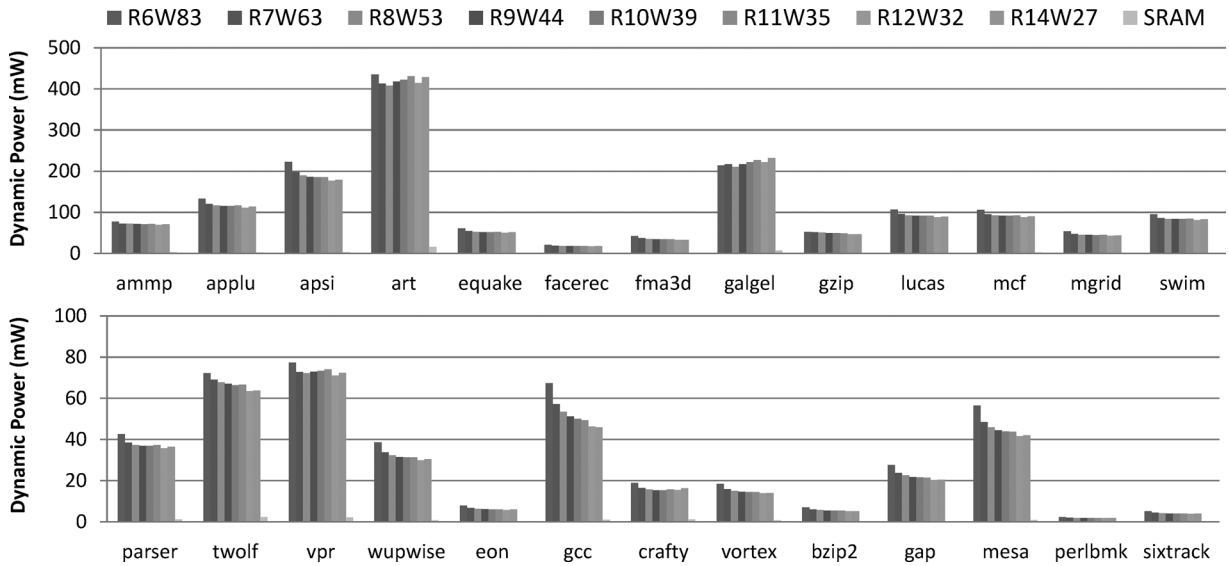


Fig. 7. Estimated dynamic power of different STT RAM L2 cache cell designs and comparison with using SRAM L2 cache.

STT RAM cache design, which can deliver good computing system performance over a wide spectrum of applications and meanwhile ensure a small cache memory footprint.

Moreover, during SimpleScalar simulations, we trace the cache access statistics of each benchmark. By combining the cache access statistics and the energy consumption results listed in Table II, we estimate the dynamic power consumption of all the 26 benchmarks for different STT RAM cache designs and SRAM cache as shown in Fig. 7. Due to its large write energy consumption, the dynamic power consumption of STT RAM L2 cache is much higher than SRAM L2 cache. Table IV lists the average dynamic power consumption of different STT RAM cache designs. Using the developed CACTI-based STT RAM cache memory modeling tool, we further estimate the leakage power consumption as listed in Table IV. Although the dynamic power consumption of STT RAM L2 cache is much higher than SRAM L2 cache due to its large write energy consumption, STT RAM L2 cache can still save a large amount of power consumption because of its nonvolatile nature with much lower leakage power consumption. Finally, we note that such significant power saving can readily offset the modest IPC degradation and hence ensure significant savings in terms of overall energy consumption.

TABLE IV
POWER ESTIMATION OF DIFFERENT STT RAM CACHE DESIGNS

Case	Average dynamic power (mW)	Leakage power (W)	Total (W)
R6W83	79.27	0.48	0.56
R7W63	73.09	0.48	0.55
R8W53	71.06	0.48	0.55
R9W44	71.02	0.48	0.55
R10W39	71.22	0.48	0.55
R11W35	71.91	0.51	0.58
R12W32	68.99	0.52	0.59
R14W27	70.75	0.52	0.59
SRAM	1.96	7.65	7.65

V. Proposed Dual-Write-Speed Method

To reduce the performance degradation for applications with high L1 data cache miss rates when using small nMOS transistors in STT RAM cache memory cells, we have to reduce the cache memory write latency. Recall that, for STT RAM circuit design, we first choose the target MTJ switching time $T^{(t)}_{\text{switch}}$ based on which we can derive the MTJ write current threshold mean μ_{th} and standard deviation σ_{th}. Then we use the conventional 6σ design rule to set the size of nMOS transistor sizes so that they could sustain a worst-case MTJ write current $I_{6\sigma} = \mu_{th} + 6 \cdot \sigma_{th}$. Since a larger MTJ write current enables a shorter MTJ switching time, under such conservative design scenario, most STT RAM cells could actually enable a switching time (much) shorter than $T^{(t)}_{\text{switch}}$, particularly as the technology scales down. Intuitively, this provides a great potential to reduce the *average* STT RAM cache write latency. In a straightforward manner, we could first offline determine the shortest MTJ switching time allowed by all the cells within each cache block, and then follow such *just enough* MTJ switching time for each cache block during the run time. Therefore, by tracking the average-case MTJ switching time other than sticking to the worst-case MTJ switching time, we could largely reduce the average memory write latency and hence improve the computing system performance, particularly for those Type-I benchmarks with high L1 data cache miss rates.

Although the above straightforward design approach ensures the minimal average cache write latency, there are many different possible values of just-enough MTJ switching time among all the cache blocks, which will make this ideal design strategy subject to several drawbacks from practical implementation perspectives, including: 1) it will incur significant memory overhead for the on-chip storage of write latency setup for each cache block and 2) it will make the memory peripheral circuits very complicated in order to support many different timing configurations. Therefore, we simplify such ideal design strategy to a so-call dual-write-latency design approach. Let $T^{(t)}_{\text{switch}}$ denote the target worst-case MTJ switching time obtained by the 6σ design rule, and given one design parameter $T^{(s)}_{\text{switch}} < T^{(t)}_{\text{switch}}$, we partition all the cache blocks into the following two categories.

- *Fast Cache Block*: If the shortest switching time allowed by one cache block is not larger than $T^{(s)}_{\text{switch}}$, this cache block is called a fast cache block and we use $T^{(s)}_{\text{switch}}$ as the MTJ switching time for this cache block.
- *Slow Cache Block*: If the shortest switching time allowed by one cache block is larger than $T^{(t)}_{\text{switch}}$, this cache block is called a slow cache block and we use $T^{(t)}_{\text{switch}}$ as the MTJ switching time for this cache block.

As a result, STT RAM cache memory only needs to support two different write modes, i.e., a fast write mode and a slow write mode, corresponding to fast cache blocks and slow cache blocks. Accordingly, we need to embed a write mode flag memory to store the 1-bit write mode configuration information associated with each cache block. Hence, as illustrated in Fig. 8, to carry out each cache write operation, we first fetch the corresponding write mode flag bit, based on which we execute either a fast cache write with the MTJ switching time of $T^{(s)}_{\text{switch}}$ or a slow cache write with the MTJ switching time of $T^{(t)}_{\text{switch}}$.

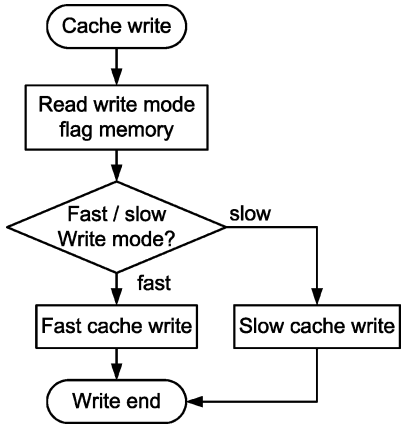


Fig. 8. Cache write operation flow chart of the proposed dual-write-speed method.

To implement this dual-write design strategy, the key design issue is how to choose the appropriate threshold $T^{(s)}_{\text{switch}}$ that can achieve the minimal average cache write latency. Let $\mathcal{P}^{(s)}$ and $\mathcal{P}^{(l)}$ represent the percentages of fast cache blocks and slow cache blocks within the entire cache memory, respectively. Clearly, we have $\mathcal{P}^{(s)} + \mathcal{P}^{(l)} = 1$. The objective is to select a $T^{(s)}_{\text{switch}}$ so that it could minimize the average cache write latency, i.e.,

$$\underset{T^{(s)}_{\text{switch}}}{\operatorname{argmin}} \left(\mathcal{P}^{(s)} \cdot T^{(s)}_{\text{switch}} + \mathcal{P}^{(l)} \cdot T^{(t)}_{\text{switch}} \right). \tag{1}$$

Let $\mu^{(s)}_{th}$ and $\sigma^{(s)}_{th}$ denote the mean and standard deviation of the MTJ write current threshold, respectively, when the MTJ switching time is $T^{(s)}_{\text{switch}}$. Let $I_{6\sigma}$ denote the worst-case MTJ write current using the 6σ design rule when MTJ switching time is $T^{(t)}_{\text{switch}}$. Assuming the MTJ write current threshold has a Gaussian distribution, the probability that one MTJ can be correctly written with $T^{(s)}_{\text{switch}}$ under the write current $I_{6\sigma}$ is

$$Pr^{(s)}_{\text{MTJ}} = \Phi\left(\frac{I_{6\sigma} - \mu^{(s)}_{th}}{\sigma^{(s)}_{th}}\right) \tag{2}$$

where $\Phi(x)$ is the cumulative distribution function for the standard normal distribution. Assuming each L2 cache block contains N bits, the probability that one cache block is a fast cache block is

$$\mathcal{P}^{(s)} = (Pr^{(s)}_{\text{MTJ}})^N. \tag{3}$$

Accordingly, we can search for the optimal $T^{(s)}_{\text{switch}}$ that can minimize the average MTJ switching time based on the above equations. Based upon the data presented in [15] and shown in Fig. 2, Fig. 9(a) shows the average MTJ switching time under

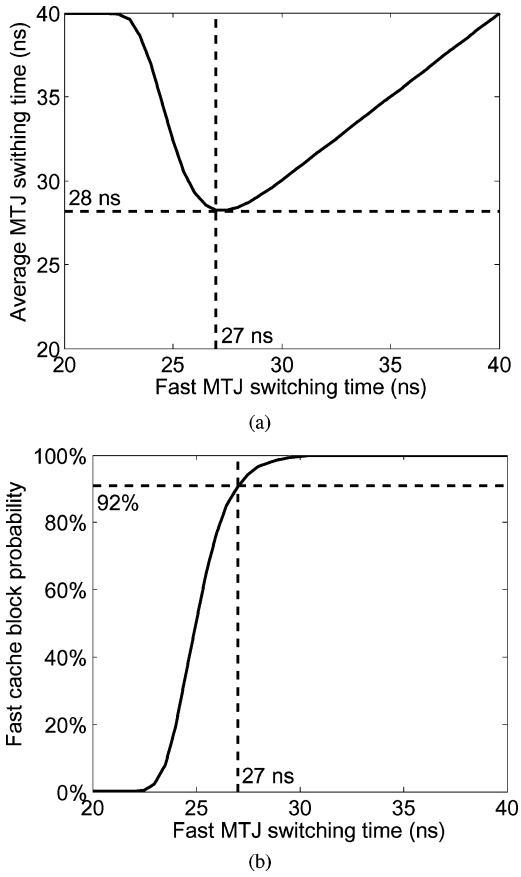


Fig. 9. (a) Average MTJ switching time versus the fast MTJ write switching time $T_{\rm switch}^{(s)}$ and (b) the probability of the fast cache blocks versus $T_{\rm switch}^{(s)}$ for the STT RAM design case *R6W83*.

TABLE V
OPTIMAL $T_{\rm switch}^{(s)}$ AND CORRESPONDING RESULTS

Case	$T_{switch}^{(s)}$ (ns)	$T_{switch}^{(t)}$ (ns)	$T_{switch}^{(avg)}$ (ns)	$\tau_w^{(s)}$ (cycle)	$\tau_w^{(l)}$ (cycle)
R6W83	27	40	28	58	84
R7W63	22.5	30	23.1	49	64
R8W53	19	25	19.5	42	54
R9W44	15.5	20	15.9	36	45

different $T_{\rm switch}^{(s)}$ for the STT RAM cell design case *R6W83* discussed in the above, and the corresponding probability of fast cache blocks is shown in Fig. 9(b), where we set the cache block size as 64 bytes and the target worst-case MTJ switching time $T_{\rm switch}^{(t)}$ in case of *R6W83* is 40 ns. As shown in Fig. 9, when $T_{\rm switch}^{(s)} = 27$ ns under which 92% cache blocks are fast cache blocks, we can achieve the minimum average MTJ switching time as 28 ns, which is reduced by 30% compared with 40 ns in the conventional design practice. Similarly, we obtain the optimal $T_{\rm switch}^{(s)}$ for the other four design cases and list the results in Table V. Using the CACTI-based STT RAM cache modeling tool presented in Section III, we obtain the fast cache block write latency $\tau_w^{(s)}$ and the slow cache block write latency $\tau_w^{(l)}$ as listed in Table V, where the cache access cycle time is 0.5 ns as used in the above *SimpleScalar 3.0* simulation. We note that we also use the CACTI tool to estimate the read latency of the write mode flag memory, which has been taken into account for the estimation of overall cache memory write latency.

To further demonstrate the effectiveness of this proposed dual-write-speed STT RAM cache design method, we carry out simulations using *SimpleScalar 3.0* for Type-I benchmarks, for which the computing system performance is significantly affected by the cache write latency. The simulator configuration parameters remain the same as listed Table I. Fig. 10 shows the simulation results of the normalized IPC, where *conv.* and *prop.* denote the conventional single-write-speed cache design approach and our proposed dual-write-speed design method, respectively. The simulation results clearly show that this proposed dual-write-speed STT RAM cache design method can largely improve the performance for Type-I benchmarks. As a result, we could use STT RAM cache with relatively small cell size in practical computing system design, which can not only reduce the cache memory footprint but also perform well over a wide spectrum of applications.

Fig. 11 further shows the average IPC degradation when using STT RAM L2 cache to replace SRAM L2 cache for all the SPEC2000 benchmarks. For the case *R9W44* with relatively small cell size, our proposed dual-write-speed STT RAM L2 cache design only has 5% degradation, while the conventional single-write-speed cache design needs a much larger memory cell size (such as the case *R14W27*) to achieve the same performance degradation. Meanwhile, we note that the STT RAM cache area of the case *R9W44* is only 56% of SRAM cache area as shown in Table II. Moreover, we expect that such IPC performance degradation may be even smaller in practical computing systems, which is explained as follows. It is well known that SRAM is subject to soft errors, and as a result, SRAM cache always uses error correcting code (ECC) to ensure its soft error tolerance. This may degrade the system IPC performance from two perspectives: 1) the storage of ECC coding redundancy increases the SRAM cache footprint and increases interconnect latency and 2) ECC decoder may induce non-negligible latency overhead. In contrast, STT RAM cache is not subject to soft errors and hence does not need to use ECC. Nevertheless, use of ECC is not incorporated in the CACTI cache modeling. Therefore, it is reasonable to expect that STT RAM cache may have less IPC degradation or even perform as well as SRAM cache in practical computing systems.

Furthermore, as the industry is moving towards the multicore era, the last-level cache is typically shared and must support coherent traffic in multicore parallel processing. This may increase the intensity of write operations and make the write latency of STT RAM cache more critical from the computing performance perspective. As a result, it will make the relatively long write latency of STT RAM a more severe issue. Hence, it is reasonable to expect that, aiming to reduce the average STT RAM write latency, this dual-write-speed scheme can be more indispensable if STT RAM is to be used as last-level cache in future multicore microprocessors.

Finally, we note that this proposed dual-write-mode design approach can also reduce STT RAM cache write energy consumption. This is because MTJ switching energy consumption is linearly proportional to the width of the write current pulse (i.e., the MTJ switching time). Since this proposed dual-write-mode design approach reduces the average MTJ switching time, it can directly reduce the average MTJ switching energy consumption and hence the average STT

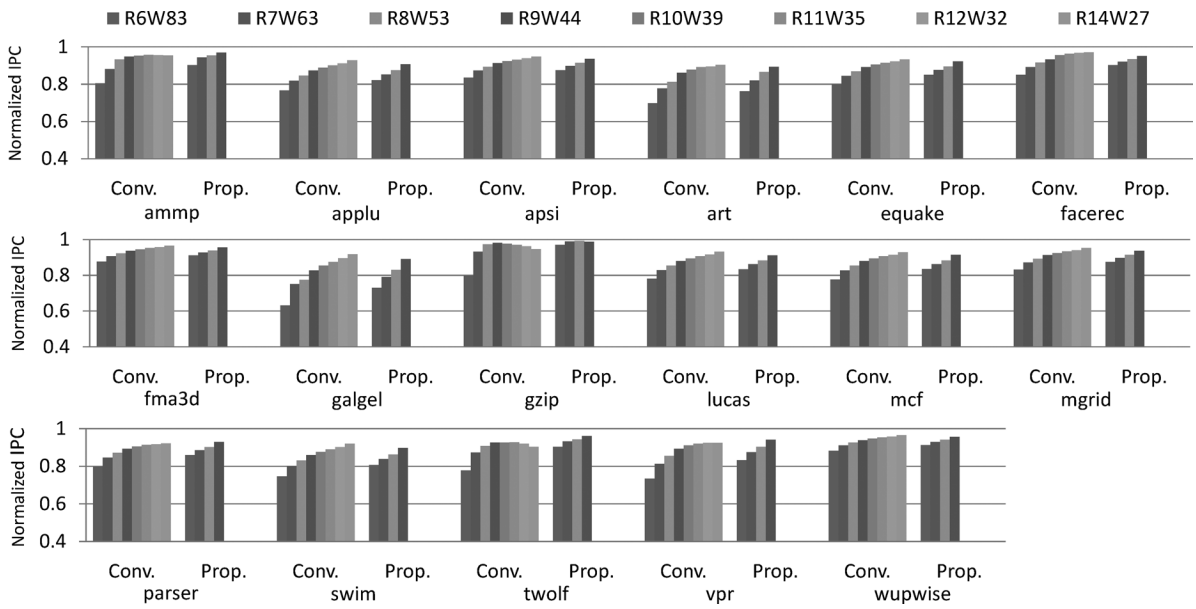


Fig. 10. Normalized IPC result for Type-I benchmarks.

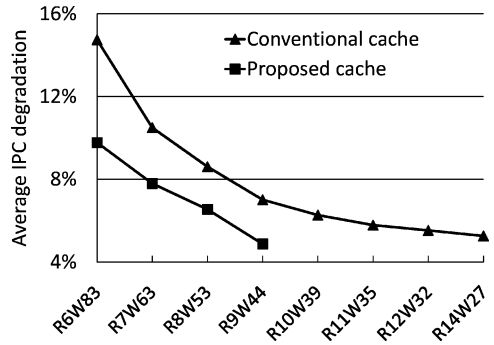


Fig. 11. Average IPC degradation when using STT RAM L2 cache to replace SRAM L2 cache for SPEC2000 benchmarks.

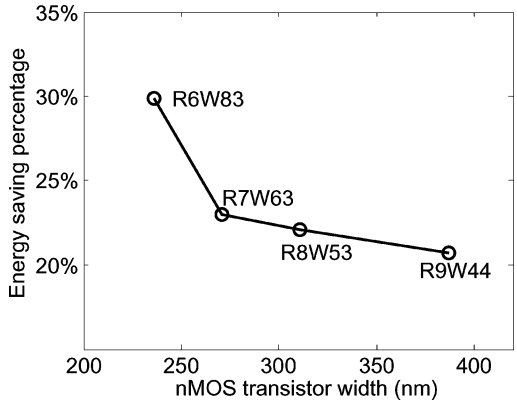


Fig. 12. Estimation result of MTJ switching energy saving for each STT RAM cell design case.

RAM cache write energy consumption. For example, let us consider the design case *R6W83*, the fast MTJ switching time $T_{\text{switch}}^{(s)}$ and the target worst-case MTJ switching time $T_{\text{switch}}^{(t)}$ are 40 and 27 ns, respectively. Since 92% of cache blocks are fast cache blocks as pointed out in the above, the average MTJ switching time is only 28 ns, leading to an average 30% MTJ switching energy saving. Accordingly, Fig. 12 shows the average MTJ switching energy savings of four different STT RAM cell design cases.

Since the overall STT RAM cache write energy consumption depends on the actual cache data access characteristics, we further carry out computing system simulations to more realistically evaluate the energy saving potential. In particular, we trace the cache access statistics of each benchmark during *SimpleScalar* simulations. Table VI lists the trace results, where N_{read}, N_{write}, $N_{\text{write}}^{(s)}$, and $N_{\text{write}}^{(l)}$ are the access numbers of L2 cache read, write, fast cache block write, and slow cache block write operations, respectively. Meanwhile, by carrying out SPICE simulations, we estimate the STT RAM cell energy consumption of read and write operations for different STT RAM cell design cases, which are listed in Table VII. Based on the above simulation results, Fig. 13 shows the overall STT RAM L2 cache energy saving when using the proposed dual-write-speed design approach for a variety of benchmarks. The results clearly demonstrate a consistently good energy saving potential, e.g., when using STT RAM cell design case *R6W83*, the saving can be up to 30% for all the benchmarks.

VI. Conclusion

This paper exploits the potential of using embedded STT RAM instead of SRAM to implement last-level on-chip cache memories such as L2 or L3 cache in microprocessors. First, this paper shows that the inherent write latency versus read latency tradeoff in STT RAM makes the memory cell sizing a nontrivial task, and different benchmarks may have conflicting expectations on memory cell sizing. Taking advantage of correlation between MTJ device switching time and write current, we further propose a dual-write-speed STT RAM architecture design method that can largely improve the average write latency of STT RAM cache with relatively small memory cell size, and hence make it perform well over a wide spectrum of computing benchmarks. Besides performance improvement, this design method can also contribute to reducing STT RAM cache energy consumption. CACTI-based STT RAM modeling tool and *SimpleScalar* have been used to evaluate the involved

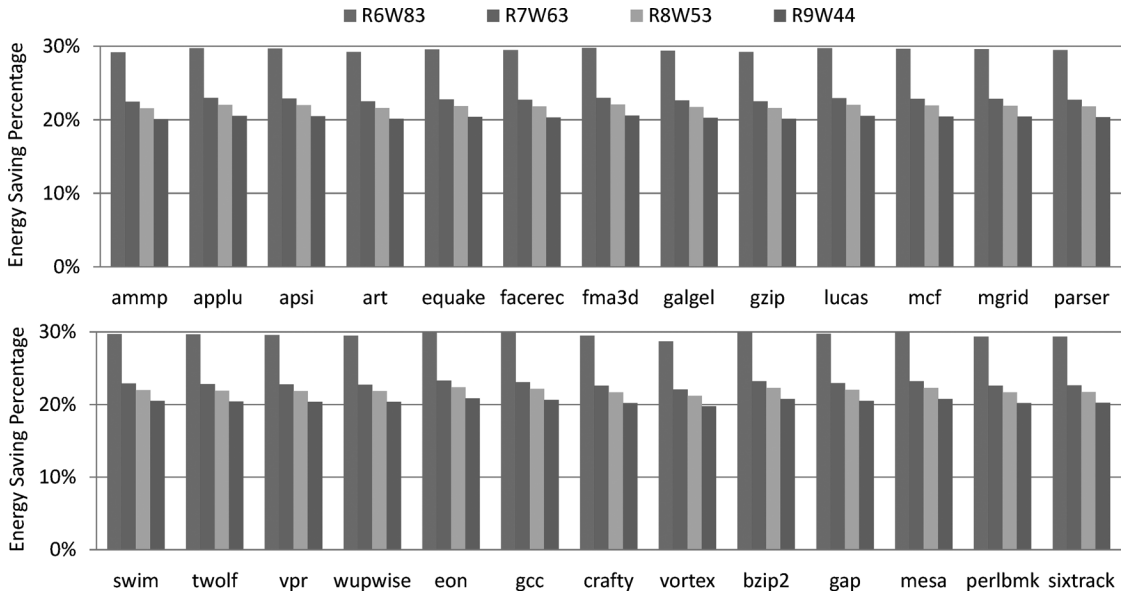


Fig. 13. Energy saving results for STT RAM L2 cache when using the proposed dual-write-speed design method.

TABLE VI
STT RAM L2 CACHE ACCESS RECORD

Benchmark	Conventional method		Proposed dual-write-speed method		
	N_{read}	N_{write}	N_{read}	$N_{write}^{(s)}$	$N_{write}^{(l)}$
ammp	1.94×10^6	5.24×10^5	1.94×10^6	4.83×10^5	4.14×10^4
applu	1.98×10^6	2.88×10^6	1.98×10^6	2.65×10^6	2.28×10^5
apsi	5.38×10^6	6.34×10^6	5.38×10^6	5.83×10^6	5.09×10^5
art	1.04×10^7	3.46×10^6	1.04×10^7	3.18×10^6	2.84×10^5
equake	4.74×10^6	2.94×10^6	4.74×10^6	2.71×10^6	2.36×10^5
facerec	5.27×10^5	2.68×10^5	5.27×10^5	2.46×10^5	2.16×10^4
fma3d	4.70×10^5	4.57×10^5	4.70×10^5	4.23×10^5	3.48×10^4
galgel	2.97×10^6	1.27×10^6	2.97×10^6	1.16×10^6	1.02×10^5
gzip	9.82×10^5	3.81×10^5	9.82×10^5	3.48×10^5	3.22×10^4
lucas	2.14×10^6	3.20×10^6	2.14×10^6	2.94×10^6	2.57×10^5
mcf	1.27×10^7	1.19×10^7	1.27×10^7	1.09×10^7	9.55×10^5
mgrid	1.07×10^6	8.44×10^5	1.07×10^6	7.76×10^5	6.78×10^4
parser	1.06×10^6	5.01×10^5	1.06×10^6	4.61×10^5	3.96×10^4
swim	2.94×10^6	3.38×10^6	2.94×10^6	3.11×10^6	2.71×10^5
twolf	2.28×10^6	7.70×10^5	2.28×10^6	7.17×10^5	5.35×10^4
vpr	2.54×10^6	1.18×10^6	2.54×10^6	1.09×10^6	9.05×10^4
wupwise	4.58×10^5	3.51×10^5	4.58×10^5	3.21×10^5	2.93×10^4
eon	2.10×10^5	4.85×10^4	2.10×10^5	4.65×10^4	1.94×10^3
gcc	5.70×10^5	4.40×10^5	5.70×10^5	4.08×10^5	3.11×10^4
crafty	9.76×10^5	9.74×10^4	9.76×10^5	9.42×10^4	3.16×10^3
vortex	4.28×10^5	9.44×10^4	4.28×10^5	8.59×10^4	8.45×10^3
bzip2	3.07×10^4	2.73×10^4	3.07×10^4	2.55×10^4	1.80×10^3
gap	1.47×10^5	2.60×10^5	1.47×10^5	2.39×10^5	2.10×10^4
mesa	3.14×10^5	2.79×10^5	3.14×10^5	2.61×10^5	1.84×10^4
perlbmk	7.47×10^4	2.13×10^4	7.47×10^4	1.97×10^4	1.61×10^3
sixtrack	5.93×10^4	2.61×10^4	5.93×10^4	2.39×10^4	2.14×10^3

TABLE VII
STT RAM CELL ENERGY CONSUMPTION IN A 4 MB STT RAM CACHE AT 45 nm NODE

Case	Operation energy (pJ)		
	read	fast write mode	slow write mode
R6W83	0.12	5.35	7.92
R7W63	0.13	5.00	6.67
R8W53	0.14	4.74	6.24
R9W44	0.16	4.64	5.98

design tradeoffs and well demonstrate the effectiveness of this proposed design method.

REFERENCES

[1] K. Kim and G. Jeong, "Memory technologies for sub-40 nm node," in *Proc. IEEE Int. Electron Devices Meet. (IEDM)*, Dec. 2007, pp. 27–30.
[2] E. J. Prinz, "The zen of nonvolatile memories," in *Proc. ACM/IEEE Des. Autom. Conf. (DAC)*, Jul. 2006, pp. 815–820.
[3] J. M. Slaughter, "Recent advances in MRAM technology," in *Proc. Device Res. Conf. (DRC)*, Jun. 2007, pp. 245–246.
[4] X. Dong, X. Wu, G. Sun, Y. Xie, and Y. Chen, "Circuit and microarchitecture evaluation of 3D stacking magnetic ram (MRAM) as a universal memory replacement," in *Proc. ACM/IEEE Des. Autom. Conf. (DAC)*, Jun. 2008, pp. 554–559.
[5] M. Hosomi, H. Yamagishi, T. Yamamoto, K. Bessho, Y. Higo, K. Yamane, H. Yamada, M. Shoji, H. Hachino, C. Fukumoto, H. Nagao, and H. Kano, "A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-RAM," in *Proc. IEEE Int. Electron Devices Meet. (IEDM)*, 2005, pp. 459–462.

[6] K.-T. Nam, S. C. Oh, J. E. Lee, J. H. Jeong, I. G. Baek, E. K. Yim, J. S. Zhao, S. O. Park, H. S. Kim, U. Chung, and J. T. Moon, "Switching properties in spin transper torque MRAM with sub-50 nm MTJ size," in *Proc. 7th Ann. Non-Volatile Memory Technol. Symp. (NVMTS)*, Nov. 2006, pp. 49–51.

[7] Z. Diao, Z. Li, S. Wang, Y. Ding, A. Panchula, E. Chen, L. Wang, and Y. Huai, "Spin-transfer torque switching in magnetic tunnel junction and spin-tansfer torque random access memory," *J. Phys.: Condensed Matter*, vol. 19, pp. 165209–165209, Apr. 2007.

[8] Y. Chen, X. Wang, H. Li, H. Liu, and D. Dimitrov, "Design margin exploration of spin-torque transfer RAM (SPRAM)," in *Proc. IEEE Int. Symp. Quality Electron. Des. (ISQED)*, 2008, pp. 684–690.

[9] J. Li, C. Augustine, S. Salahuddin, and K. Roy, "Modeling of failure probability and statistical design of spin-torque transfer magnetic random access memory (STT MRAM) array for yield enhancement," in *Proc. ACM/IEEE Des. Autom. Conf. (DAC)*, Jun. 2008, pp. 278–283.

[10] E. J. Marinissen, B. Prince, D. Keltel-Schulz, and Y. Zorian, "Challenges in embedded memory design and test," in *Proc. Des., Autom. Test Eur. (DATE)*, Mar. 2005, pp. 722–727.

[11] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, CACTI 5 "Advanced architecture laboratory HP Laboratories,", HPL-2007-167, 2007. [Online]. Available: http://www.hpl.hp.com/techreports/2008/HPL-2008–20.html

[12] SimpleScalar 3.0 2003. [Online]. Available: http://www.simplescalar.com

[13] T. Kawahara, R. Takemura, K. Miura, J. Hayakawa, S. Ikeda, Y. Lee, R. Sasaki, Y. Goto, K. Ito, I. Meguro, F. Matsukura, H. Takahashi, H. Matsuoka, and H. Ohno, "2 Mb spin-transfer torque ram (SPRAM) with bit-by-bit bidirectional current write and parallelizing-direction current read," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2007, pp. 480–481.

[14] X. Wang, Y. Chen, H. Li, D. Dimitrov, and H. Liu, "Spin torque random access memory down to 22 nm technology," *IEEE Trans. Magn.*, vol. 44, no. 11, pp. 2479–2482, Nov. 2008.

[15] X. Wang, Y. Zheng, H. Xi, and D. Dimitrov, "Thermal fluctuation effects on spin torque induced switching: Mean and variations," *J. Appl. Phys.*, vol. 103, pp. 034507–034507, Feb. 2008.

[16] T. W. Andre, J. J. Nahas, C. K. Subramanian, B. J. Garni, H. S. Lin, A. Omair, and W. L. Martino, "A 4-Mb 0.18-μm 1T1MTJ toggle MRAM with balanced three input sensing scheme and locally mirrored unidirectional write drivers," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 301–309, Jan. 2005.

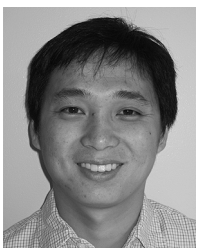
[17] T. Sugibayashi, N. Sakimura, T. Honda, K. Nagahara, K. Tsuji, H. Numata, S. Miura, K. Shimura, Y. Kato, S. Saito, Y. Fukumoto, H. Honjo, T. Suzuki, K. Suemitsu, T. Mukai, K. Mori, R. Nebashi, S. Fukami, N. Ohshima, H. Hada, N. Ishiwata, N. Kasai, and S. Tahara, "A 16-Mb toggle MRAM with burst modes," *IEEE J. Solid-State Circuits*, vol. 42, no. 11, pp. 2378–2385, Nov. 2007.

[18] J. Li, P. Ndai, A. Goel, H. Liu, and K. Roy, "An alternate design paradigm for robust spin-torque transfer magnetic RAM (STT MRAM) from circuit/architecture perspective," in *Proc. Asia South Pacific Des. Autom. Conf.*, Jan. 2009, pp. 841–846.

[19] W. Xu, T. Zhang, and Y. Chen, "Design of spin-torque transfer magnetoresistive RAM and CAM/TCAM with high sensing and search speed," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, to be published.

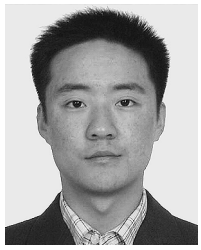
[20] Nanoscale Integration and Modeling (NIMO) Group, ASU, Tempe, AZ, Predictive Technology Model (PTM) 2004. [Online]. Available: http://www.eas.asu.edu/~ptm/

[21] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," in *Proc. Int. Conf. Arch. Support for Program. Lang. Operat. Syst.*, Oct. 2002, pp. 45–57.



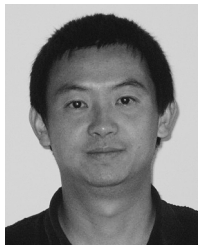
Wei Xu received the B.S. and M.S. degrees from Fudan University, Shanghai, China, in 2003 and 2006, respectively. He is currently working toward the Ph.D. degree at the Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY.

His current research interests include circuit and system design for memory and data storage systems. He is engaged in magnetoresistive memory and phase change memory design.



Hongbin Sun received the B.S. degree in electrical engineering from Xian Jiaotong University, Xian, China, in 2003. He is currently pursuing the Ph.D. degree from the School of Electronic and Information Engineering, Xian Jiaotong University, Xi'an, China.

From November 2007 to October 2008, he was a visiting scholar with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY. His current research interests include fault tolerant computer architecture, 3-D memory-processor integration, and VLSI architecture for digital video processing.



Xiaobin Wang received the Ph.D. degree in physics from the University of California, San Diego (UCSD), in 2003.

At UCSD, he was a Graduate Research Assistant under Prof. H. N. Bertram, where his study was focused on dynamic thermal magnetization switching and magnetization noise in magnetic devices. He has been with Seagate Technology, Bloomington, MN, since 2003, where his work includes prediction of recording system performance through bottom up (from physics to system performance) and top down (from system performance to component requirements) approaches, company product platform and basic technology roadmap modeling, recording system and head design. He began to work on alternative technologies in 2007. He is interested in magnetization switching, magnetization variability and their applications in nano-scale device design. He has published over 50 papers and has over 30 U.S. patents pending. He was invited to contribute at various conferences and journals.

Yiran Chen (M'05) received the B.S. and M.S. degrees in electronics engineering from Tsinghua University, Beijing, China, and the Ph.D. degree from the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN.

He was with PrimeTime Group, Synopsys, Inc., Sunnyvale, CA, where he developed the award-winning statistical static timing analysis tool PrimeTimeVX. In 2007, he joined the Alterative Technology Group, Seagate Technology LLC, Bloomington, MN, where he was engaged in the next generation nonvolatile memory. His current research interests include VLSI design/computer-aided design (CAD) for nanoscale Silicon and nonsilicon technologies, low-power circuit design and computer architecture, and emerging memory technologies. He has authored or coauthored more than 25 technical papers in refereed journals and conferences. He has more than 50 pending U.S. patents.

Tong Zhang (M'02–SM'08) received the B.S. and M.S. degrees in electrical engineering from Xian Jiaotong University, Xian, China, in 1995 and 1998, respectively, and the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 2002.

He is currently an Associate Professor with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY. His current research interests include algorithm and architecture codesign for communication and data storage systems, variation-tolerant signal processing IC design, fault-tolerant system design for digital memory, and computer architecture.