A Novel Architecture of the 3D Stacked MRAM L2 Cache for CMPs

Guangyu Sun[†], Xiangyu Dong[†], Yuan Xie[†], Jian Li[‡], Yiran Chen[§]
[†]Pennsylvania State University, [‡]IBM Austin Research Lab, [§]Seagate Technology
[†]{gsun, xydong, yuanxie}@cse.psu.edu, [‡]jianli@us.ibm.com, [§]yiran.chen@seagate.com

Abstract

Magnetic random access memory (MRAM) is a promising memory technology, which has fast read access, high density, and non-volatility. Using 3D heterogeneous integrations, it becomes feasible and cost-efficient to stack MRAM atop conventional chip multiprocessors (CMPs). However, one disadvantage of MRAM is its long write latency and its high write energy. In this paper, we first stack MRAM-based L2 caches directly atop CMPs and compare it against SRAM counterparts in terms of performance and energy. We observe that the direct MRAM stacking might harm the chip performance due to the aforementioned long write latency and high write energy. To solve this problem, we then propose two architectural techniques: read-preemptive write buffer *and* SRAM-MRAM hybrid L2 cache*. The simulation result shows that our optimized MRAM L2 cache improves performance by 4.91% and reduces power by 73.5% compared to the conventional SRAM L2 cache with the similar area.*[1]

1 Introduction

The diminishing return of endeavors to increase clock frequencies and exploit instruction level parallelism in a single processor have led to the advent of chip multiprocessors (CMPs) [8]. The integration of multiple cores on a single chip is expected to accentuate the already daunting "memory wall" problem [6] and it becomes a major challenge of supplying massive multi-core chips with sufficient memories.

The introduction of the three-dimensional (3D) integration technology [9, 26] provides the opportunity of stacking memories atop compute cores and therefore alleviates the memory bandwidth challenge of CMPs. Recently, active research [4, 13, 22] has targeted SRAM caches or DRAM memories stacking.

Magnetic Random Access Memory (MRAM) is a promising memory technology with attractive features such as fast read access, high density, and non-volatility [14, 27]. However, previous research on leveraging MRAM as on-chip memories is very limited. How to integrate MRAM into compute cores on planular chips is the key obstacle since the MRAM fabrication involves hybrid magnetic-CMOS processes. Fortunately, 3D integrations enable the cost-efficient integration of heterogeneous technologies, which is ideal for MRAM stacking atop compute cores. Some recent work [10, 12] has evaluated the benefits of MRAM as a universal memory replacement for L2 caches and main memories in single-core chips.

In this paper, we further evaluate the benefits of stacking MRAM L2 caches atop CMPs. We first develop a cache model for stacking MRAM and then compare the MRAM-based L2 cache against its SRAM counterpart with the similar area in terms of performance and energy. The comparison shows that: (1) For applications that have moderate write intensities to L2 caches, the MRAM-based cache can reduce the total cache power significantly because of its zero standby leakage and achieve considerable performance improvement because of its relatively larger cache capacity; (2) For applications that have high write intensities to L2 caches, the MRAM-based cache can cause performance and power degradations due to the long latency and the high energy of MRAM write operations.

These two observations imply that MRAM-based caches might not work efficiently if we directly introduce them into the traditional CMP architecture because of their disadvantages on write latency and write energy. In light of this concern, we propose two architectural techniques, *read-preemptive write buffer* and *SRAM-MRAM hybrid L2 cache*, to mitigate the MRAM write-associated issues. The simulation result shows that performance improvement and power reduction can be achieved effectively with our proposed techniques even under the write-intensive workloads.

2 Background

This section briefly introduces the background of MRAM and 3D integration technologies.

2.1 MRAM Background

The basic difference between the MRAM and the conventional RAM technologies (such as SRAM/DRAM) is that the information carrier of MRAM is Magnetic Tunnel Junctions (MTJs) instead of electric charges [27]. As shown in Fig. 1, each MTJ contains *a pinned layer* and *a free layer*. The *pinned layer* has fixed magnetic direction

[1]This work was supported in part by NSF grants (CAREER 0643902, CCF 0702617, CSR 0720659), a gift grant from Qualcomm, and IBM Faculty Award.

978-1-4244-2932-5/08/$25.00 ©2008 IEEE

while the *free layer* can change its magnetic direction by spin torque transfers [14]. If the free layer has the same direction as the pinned layer, the MTJ resistance is low and indicates state "0"; otherwise, the MTJ resistance is high and indicates state "1".

The latest MRAM technology (spin torque transfer ram, STT-RAM) changes the magnetic direction of the free layer by directly passing spin-polarized currents through MTJs. Comparing to the previous generation of MRAM using external magnetic fields to reverse the MTJ status, STT-RAM has the advantage of scalability, which means the *threshold current* to make the status reversal will decrease as the size of the MTJ becomes smaller. In this paper, we use the terms "MRAM" and "STT-RAM" equivalently.

The most popular structure of MRAM cells is composed of one NMOS transistor as the access device and one MTJ as the storage element ("1T1J"structure) [14]. As illustrated in Fig. 1, the storage element, MTJ, is connected in series with the NMOS transistor. The NMOS transistor is controlled by the the word line (WL) signal. The detailed read and write operations for each MRAM cell is described as follows:

- **Read Operation**: When a *read operation* happens, the NMOS is turned on and a small voltage difference (-0.1V as demonstrated in [14]) is applied between the bit line (BL) and the source line (SL). This voltage difference causes a current through the MTJ whose value is determined by the status of MTJs. A sense amplifier compares this current to a reference current and then decides whether a "0" or a "1" is stored in the selected MRAM cell.
- **Write Operation**: When a *write operation* happens, a large positive voltage difference is established between SLs and BLs for writing for "0"s or a large negative one for writing "1"s. The current amplitude required to ensure a successful status reversal is called threshold current. The current is related to the material of the tunnel barrier layer, the writing pulse duration, and the MTJ geometry [11].

In this work, we use the writing pulse duration of $10ns$ [27], below which the writing threshold current will increase exponential. In addition, we scale the MRAM size of previous work [14] down to $65nm$ technology node. Assuming the size of MTJs is $65nm \times 90nm$, the derived threshold current for magnetic reversal is about $195\mu A$.

2.2 3D Integration Overview

The 3D integration technology has recently emerged as a promising means to mitigate interconnect-related problems. By using the vertical through silicon via (TSV), multiple active device layers can be stacked together (through wafer stacking or die stacking) in the third dimension [26].

3D integrations offer a number of advantages over traditional two-dimensional (2D) designs [9]: (1) shorter global interconnects because the vertical distance (or the length of TSVs) between two layers is usually in the range of 10 μm to 100 μm [26] depending on manufacturing processes; (2) higher performance because of reducing the average interconnect length; (3) lower interconnect power consumption due to the wire length reduction; (4) denser form factor and smaller footprint; (5) support for the cost-efficient integration of heterogenous technologies.

In this paper, we rely on the 3D integration technology to stack a massive amount of L2 caches (2MB for SRAM caches and 8MB for MRAM caches) on top of CMPs. Furthermore, the heterogenous technology integration enabled by 3D makes it feasible to fabricate MRAM caches and CMP logics as two separate dies and then stack them together in a vertical way. Therefore, the magnetic-related fabrication process of MRAM will not affect the normal CMOS logic fabrication and keep the integration cost-efficient.

3 MRAM and Non-Uniform Cache Access (NUCA) Models

In this section, we describe an MRAM circuit model and a NUCA model which is implemented with Network-on-Chip (NoC).

3.1 MRAM Modeling

To model MRAM, we first estimate the area of MRAM cells. As shown in Fig. 1, each MRAM cell is composed of one NMOS transistor and one MTJ. The size of MTJs is only limited by manufacturing techniques, but the NMOS transistor has to be sized properly so that it can drive sufficiently large current to change the MTJ status. The current driving ability of NMOS transistor is proportional to its W/L ratio. Using HSPICE simulation, we find that the minimum W/L ratio for the NMOS transistor under $65nm$ technology node is around 10 to drive the threshold writing current of $195\mu A$. We further assume the width of the source or drain regions of an NMOS transistor is $1.5F$, where F is the feature size. Therefore, we estimate the MRAM cell size is about $10F \times 4F = 40F^2$. The parameters of our targeted MRAM cell are tabulated in Table .

Table 1. MRAM Cell Specifications

Technology	$65nm$
Write Pulse Duration	$10ns$
Threshold Current	$195\mu A$
Cell Size	$40F^2$
Aspect Ratio	2.5

Despite the difference in storage mechanisms, MRAM and SRAM have the similar peripheral interfaces from the circuit designers' points of view. By simulating with a modified version of CACTI [2], our result shows that the area of a 512KB MRAM cache is similar to a 128KB SRAM cache

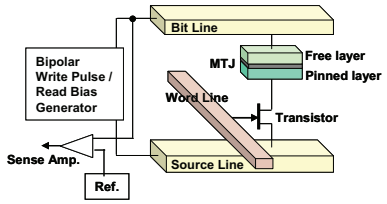

Figure 1. An illustration of an MRAM cell

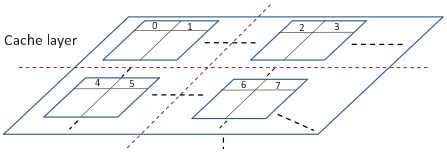

Figure 2. Eight caches ways are distributed in four banks. Assume four cores and accordingly four zones each layer.

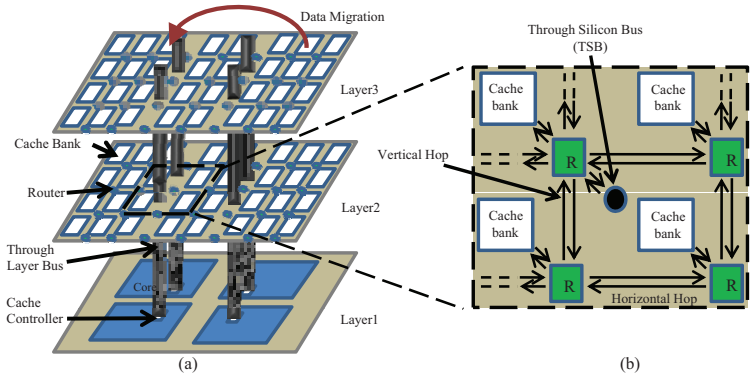

Figure 3. (a) An illustration of the proposed 3D NUCA structure, which includes 1 core layer, 2 cache layers. There are 4 processing cores per core layer, 32 cache banks per cache layer, and 4 through-layer-bus across layers; (b) Connections amongst routers, caches banks and through-layer-buses.

whose cell is about $146F^2$ (this value is extracted from CACTI). Table 2 lists the comparison between a *512KB MRAM cache bank* and a *128KB SRAM cache bank*, which are used later in this paper, in terms of area, access time, and access energy.

Table 2. Comparison of area, access time, and energy comparison(65nm technology)

Cache size	128KB SRAM	512KB MRAM
Area	$3.62mm^2$	$3.30mm^2$
Read Latency	$2.252ns$	$2.318ns$
Write Latency	$2.264ns$	$11.024ns$
Read Energy	$0.895nJ$	$0.858nJ$
Write Energy	$0.797nJ$	$4.997nJ$

3.2 Modeling 3D NUCA Cache

As the caches capacity and area increase, the wire delay has made the Non-Uniform Cache Access (NUCA) architecture [18] more attractive than the conventional Uniform Cache Access (UCA) one. In NUCA, the cache is divided into multiple banks with different access latencies according to their locations relative to cores and these banks can be connected through a mesh-based Network-on-Chip (NoC).

Extending the work of CACTI [2], we develop our NoC-based 3D NUCA model. The key concept is to use NoC routers for communications within planular layers, while using a specific through silicon bus (TSB) for communications among different layers. Figure 3(a) illustrates an example of the 3D NUCA structure. There are four cores located in the *core layer* and 32 cache banks in each *cache layer* and all layers are connected by through silicon bus (TSB) which is implemented with TSVs. This interconnect style has the advantage of short connections provided by 3D integrations. It has been reported the vertical latency of traversing a 20-layer stack is only $12ps$ [23], thus the latency of TSB negligible compared to the latency of 2D NoC routers. Consequently, it is feasible to have single-hop vertical communications by utilizing TSBs. In addition, hybridization of 2D NoC routers with TSBs require one (instead of two) additional link on each NoC router, because TSB can move data both upward and downward [20].

As shown in Figure3(a), cache layers are on top of core layers and they can either SRAM or MRAM caches. Figure3(b) shows a detailed 2D structure of cache layers. Every four cache banks are grouped together and routed to other layers via TSBs.

Similar to prior approaches [7, 20], the proposed model supports *data migration*, which moves data closer to their accessing core. For set-associative cache, the cache ways belonging to the set should be distributed into different banks so that data migration can be implemented. In our 3D NUCA model, each cache layer is equally divided into several zones. The number of zones is equal to the number of cores and each zone has a TSB located at its center. The cache ways of each set are uniformly distributed into these zone. This architecture promises that, within each cache set, there are several ways of cache lines close to the active core. Fig. 2 gives an illustration of distributing eight ways into four zones. Fig. 3(a) shows an example of data migration after which the core in the upper-left corner can access the data faster. In this paper, this kind of data migrations is called *inter-migration* to differentiate another kind of migration policy introduced later.

The advantages of this 3D NUCA cache are:(1) placing L2 caches in separate layers makes it possible to integrate MRAM with traditional CMOS process technology; (2) separating cores from caches simplifies the design of TSBs and routers because TSBs are now connected to cache controllers directly, and there is no direct connection be-

Table 3. Baseline configuration parameters

Processors	
# of cores	8
Frequency	3GHz
Power	6W/core
Issue Width	1 (in order)
Memory Parameters	
L1 cache	private, 16+16KB, 2-way, 64B line, 2-cycle, write-through, 1 read/write port
SRAM L2	shared, 2MB (16x128KB), 32-way, 64B line, read/write per bank : 7-cycle, write-back, 1 read/write port
MRAM L2	shared, 8MB (16x512KB), 32-way, 64B line, read penalty per bank : 7-cycle, write penalty per bank : 33-cycle, write-back, 1 read/write port
Write buffer	4 entry, retire-at-2
Main Memory	4GB, 500-cycle latency
Network Parameters	
# of Layers	2
# of TSB	8
Hop latency	TSB 1 cycle, V_hop 1 cycle H_hop 1 cycle
Router Latency	2-cycle

tween routers and cache controllers.

We provide one TSB for each core in the model. Considering that the TSV pitch size is reported to be only 4-10μm [23], thus even a 1024-bit bus (much wider than our proposed TSB) would only incur an area overhead of $0.32mm^2$. In our study, the die area of an 8-core CMP is estimated to be $60mm^2$ (discussed later). Therefore, it is feasible to assign one TSB for each core and the TSV area overhead is negligible.

3.3 Configurations and Assumptions

Our baseline configuration is an 8-core in-order processor using the Ultra SparcIII ISA. In order to predict the chip area, we investigate some die photos, such as Cell Processor [16], Sun UltraSPARC T1 [19], etc. and estimate the area of an 8-core CMP without caches to be $60mm^2$. By using our modified version of CACTI [2], we further learn that one cache layer fits to either a $2MB$ SRAM or an $8MB$ MRAM L2 cache assuming each cache layer has the similar area to that of core layer ($60mm^2$). The configurations are detailed in Table 3. Note that the power of processors is estimated based on the data sheet of real designs [16, 19].

We use the Simics toolset [24] for performance simulations. Our 3D NUCA architecture is implemented as an extended module in Simics. We use a few multi-threaded benchmarks from *OpenMP2001* [3] and *PARSEC* [1] suites. Since the performance and power of MRAM caches are closely related to transaction intensity, we select some simulation workloads as listed in Table 4 so that we have a wide range of transaction intensities to L2 caches. The average numbers of total transactions (TPKI)[2] and write transactions (WPKI) of L2 caches are listed in Table 4. For each simulation, we fast forward to warm up the caches and then run 3 billion cycles. We use the total IPC of all the cores as the performance metric.

Table 4. L2 transaction intensities

Name	TPKI	WPKI
galgel	1.01	0.31
apsi	4.15	1.85
equake	7.94	3.84
fma3d	8.43	4.00
swim	19.29	9.76
streamcluster	55.12	23.326

3.4 SNUCA and DNUCA

Static NUCA (SNUCA) and *Dymaic NUCA* (DNUCA) are two different implementations of the NUCA architecture proposed by Kim, *et al.* [18]. SNUCA statically partitions the address space across cache banks, which are connected via NoC; DNUCA dynamically migrates frequently accessed blocks to the closest banks. These two NUCA implementations result in different access patterns and variable write intensities. In our later simulations, we use both SNUCA-SRAM and DNUCA-SRAM L2 caches as our baselines when evaluating the performance and power benefits of MRAM caches.

4 Direct Replacing SRAM with MRAM as L2 Caches

In this section, we directly replace SRAM L2 caches with MRAM ones that have the comparable area, and show that without any optimization, a naive MRAM replacement will harm both performance and power when the workload write intensity is high.

4.1 Same Area Replacement

As shown in Table 2, a 128KB SRAM bank has the similar area as a 512KB MRAM bank does. Thereby, in order to keep the area of cache layers unchanged, it becomes reasonable to replace SRAM L2 caches with MRAM ones whose capacity is 3 times larger. We call this replacement strategy as "same area replacement".

Using this strategy, we integrate as many caches in the cache layers as possible. Considering our baseline SRAM

[2]TPKI is the number of total transactions per 1K instructions and WPKI is the number of write transactions per 1K instructions.

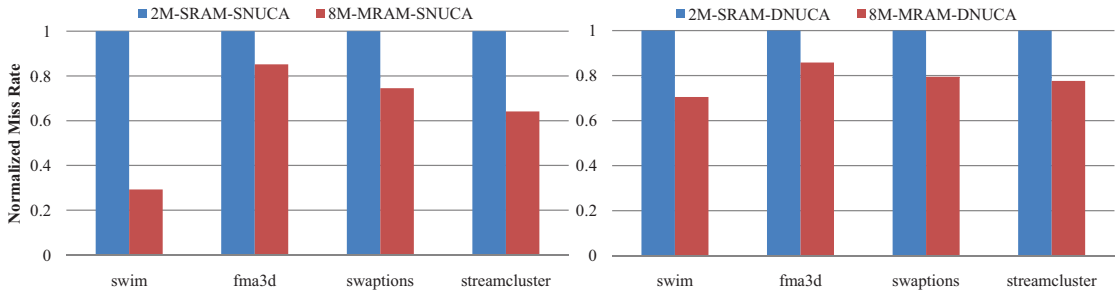

Figure 4. The comparison of L2 caches access miss rates for SRAM L2 cache and MRAM L2 cache that have similar area. Larger capacity of MRAM cache results in smaller cache miss rates.

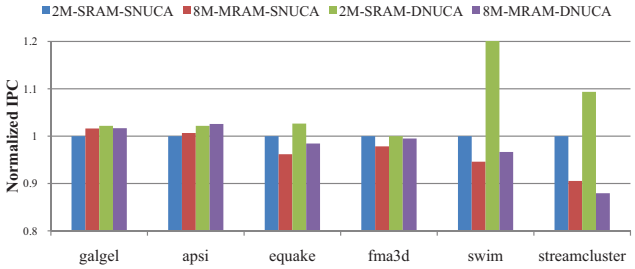

Figure 5. IPC comparison of SRAM and MRAM L2 caches(Normalized by 2M SNUCA SRAM cache).

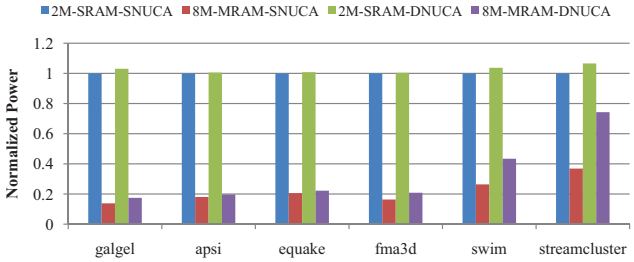

Figure 6. Power comparison of SRAM and MRAM L2 caches (Normalized by 2MB SNUCA SRAM cache).

L2 cache has 16 banks and each cache bank has the capacity of 128KB, we keep the number of banks unchanged but replace each 128KB SRAM L2 cache bank with a 512KB MRAM cache bank. The read/write access time and read/write energy consumption are tabulated in Table 2 for both SRAM and MRAM.

4.2 Performance Analysis

Because the number of banks remains the same and our modified CACTI shows 128KB SRAM bank and 512KB MRAM bank have similar read latencies (2.252ns versus 2.318ns in Table 2), the read latencies of the 2MB SRAM cache and the 8MB MRAM cache are similar as well. Since the MRAM cache capacity is 3 times larger, the access miss rate to the L2 cache decreases as shown in Fig. 4. On average, the miss rates are reduced by 19.0% and 12.5% for SNUCA MRAM cache and DNUCA MRAM cache, respectively.

The IPC comparison is illustrated in Fig. 5. Caused by the large MRAM cache capacity, the L2 cache miss rate decrease improves the performance of the first two workloads (*"galgel"* and *"apsi"*); however the performance of the rest four workloads is not improved as expected. On average, the performance *degradation* of *SNUCA MRAM* and *DNUCA MRAM* is 3.09% and 7.52% compared to their SRAM counterparts, respectively.

This performance degradation of direct MRAM replacement can be explained by Table 4, where we can observe the write operation intensity (presented by WPKI) of "euqake", "fma3d", "swim", and "streamcluster" is much higher than that of "galgel" and "apsi". Due to the long latency of MRAM write operations, the high write intensity is reflected by the performance loss. When the write intensity is sufficiently high, the resulting performance loss overwhelms the performance gain achieved by reduced L2 cache miss rate. This observation is further supported by comparison between SNUCA and DNUCA. From Fig. 5, one can observe that performance degradation is more significant when we use DNUCA MRAM caches because data migrations in DNUCA initiate more write operations than SNUCA does and thus cause high write intensities.

To summarize, we conclude our first observation of using MRAM caches as:

Observation 1 *Replacing SRAM L2 caches directly with MRAM, which has the similar area but with a large capacity, can reduce the access miss rate of the L2 cache. However, the long latency associated with the write operations to the MRAM cache has a negative impact on the performance. When the write intensity is high, the benefits caused by miss rate reductions could be offset by the long latency of MRAM write operations and eventually result in performance degradation.*

4.3 Power Analysis

The major contributors of the total power consumption in caches are leakage power and dynamic power:

- *Leakage Power:* When process technology scales down to sub-90nm, the leakage power in CMOS technology becomes dominant. Since MRAM is a non-volatile memory technology, there is no power supply to each MRAM cell and then MRAM cells do not consume any standby leak-

age power. Therefore, we only consider peripheral circuit leakage power for MRAM caches and the leakage power comparison of 2MB SRAM and 8MB MRAM is listed in Table 5.

Table 5. Leakage power of SRAM and MRAM caches at $80\,^{\circ}\mathrm{C}$

Cache configurations	Leakage power
2MB $16 \times 128KB$ SRAM cache	2.089W
8MB $16 \times 512KB$ MRAM cache	0.255W

- *Dynamic Power:* The dynamic power estimation for the NUCA cache is described as follows. For each transaction, the total dynamic power is composed of the memory cell access power, the router access power, and the power consumed by wire connections. In this paper, these values are either simulated by HSPICE or obtained from our modified version of CACTI. The access number of routers and the length of wire connections vary from the location of the requesting core and the requested cache lines.

Fig. 6 shows the power comparison of SRAM and MRAM L2 caches. One can observe that:

- For SRAM L2 caches, since the leakage power dominates, the total power for SNUCA SRAM and DNUCA SRAM are very close. On the contrary, the dynamic power dominates the MRAM cache power.
- For all the workloads, MRAM caches consume less power than SRAM caches do. The average power savings across all the workloads are about 78% and 68% for SNUCA and DNUCA, respectively. The power saving for DNUCA MRAM is smaller because of the high write intensity caused by data migrations. It is obvious that the "low leakage power" feature makes MRAM more attractive to be used as large on-chip memory, especially when SRAM leakage power becomes worse with technology scaling.
- The average power savings for the first four workloads are more than 80%. However, for the workload *"streamcluster"*, the total power saving is only 63% and 30% for SNUCA and DNUCA, respectively, due to its much higher L2 cache write intensity (see Table 4).

To summarize, our second conclusion of direct MRAM cache replacement is:

Observation 2 *Direct replacing the SRAM L2 cache with a MRAM cache, which has similar area but with larger capacity, can greatly reduce the leakage power. However, when the write intensity is high, the dynamic power increases significantly because of the high energy associated with the MRAM write operation and the amount of total power saving could be reduced.*

These two conclusions show that, if we directly replace SRAM caches with MRAM caches using "same area strategy", the long latency and high energy consumption of MRAM write operations can offset the performance and power benefit brought by MRAM cache when the cache write intensity is high.

5 Novel 3D-stacked cache architecture

In this section we propose two techniques to mitigate the write operation problem of using MRAM caches: *read-preemptive write buffer* is employed to reduce the stall time caused by the MRAM long write latency; *SRAM-MRAM hybrid L2 cache* is proposed to reduce the number of MRAM write operations and thereby improve both performance and power. Finally, we combine these two techniques together as an optimized MRAM cache architecture.

5.1 Read-preemptive Write Buffer

The first observation in Section shows that the long MRAM write latency has a serious impact on the performance. In the scenario where a write operation is followed by several read operations, the ongoing write operation may block the upcoming read operations and cause performance degradations. Although the write buffer design in modern processors works well for SRAM caches, our experiment result in Subsection 4.2 shows that this write buffer does not fit for MRAM caches due to the large variation between MRAM read latency and write latency. In order to make MRAM caches work efficiently, we explore the proper write buffer size and propose a *"read-preemptive"* management policy for it.

5.1.1 The Exploration of the Buffer Size

The choice of the buffer size is important. The larger the buffer size is, the more write operations can be hidden. Thereby, the number of stall cycles decreases. However, on the other hand, the larger the buffer size is, the longer time it takes to check whether there is a '"hit" in the buffer and then to access it. Furthermore, the design complexity and the area overhead also increase with the buffer size growth. Fig. 7 shows the relative IPC improvement by using different buffer sizes for workloads "streamcluster" and "swim". Observing the simulation result, we choose the size of 20 entries as the optimal MRAM write buffer size. Compared to the SRAM write buffer, which has only 4 entries (as listed in Table 3), the MRAM write buffer size is much larger and we use 20-entry write buffer for MRAM caches in the later simulations.

5.1.2 Read-preemptive Policy

Since the L2 cache can receive requests from from the upper level memory (L1 cache) and the write buffer, a priority policy is necessary to solve the conflict that a read request

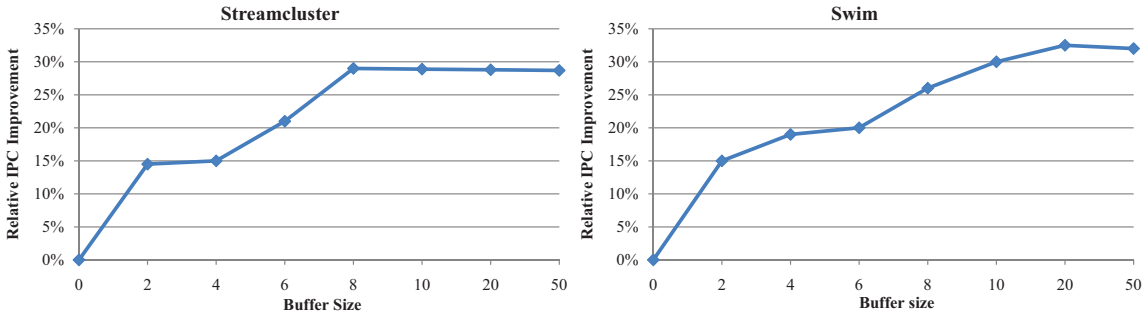

Figure 7. The impact of buffer size. The IPC improvement is normalized by that of 8M MRAM cache without write buffer

and a write request compete for the execution right. For MRAM caches, write operation latencies are much larger than read latencies, thus our objective is to prevent write operations from blocking read operations. As a result, we have our first rule:

Rule1: *The read operation always has the higher priority in a competition for the execution right.*

Additionally, consider there is a read request blocked by a write operation that is already in process, the MRAM write latency is so large that its retirement may block one or more read request for a long period and further causes performance degradations. In order to mitigate this problem, we propose another read-preemptive rule as follows:

Rule2: *When a read request is blocked by a write retirement and the write buffer is not full, the read request can trap and stall the write retirement if the preemption condition (discussed later) is satisfied. Then, the read operation obtains the right of the execution to the cache. The stalled write retirement will retry later.*

Our proposed read-preemptive policy tries to execute MRAM read requests as early as possible, but the drawback is that some write retirements need to be re-executed and the possibility of full buffer increases. The pivot is to find a proper *preemption condition*. One extreme method is to stall the write retirement as long as there is a read request, which means that read requests can always be executed immediately. Theoretically, if the write buffer size is large enough, no read request will be blocked. However, since the buffer size is limited, the increased possibility of full buffer could also harm the performance. In some other cases, stalling write retirements for read requests are not always good. For example, if a write retirement almost finishes, no read request should stall the retirement process. Consequently, we propose to use the *retirement accomplishment degree*, denoted as α, as the preemption condition. The *retirement accomplishment degree* is the accomplishment percentage of the ongoing write retirement, below which no preemption will occur.

Fig. 8 compares the IPC of using different α in our read-preemptive policy. Note that $\alpha = 100\%$ represents the non-conditional preemption policy and $\alpha = 0\%$ represents the traditional write buffer. We can find that, for the workloads with low write intensities, such as *"galgel"* and *"apsi"*, the performance improves as α increases and the non-conditional preemption policy works the best. However, for the benchmark with high write intensities, like *"streamcluster"*, the performance improves at the beginning but then degrades as α increases. Generally, in this paper, we set $\alpha = 50\%$ to make our read-preemptive policy effective for all the workloads.

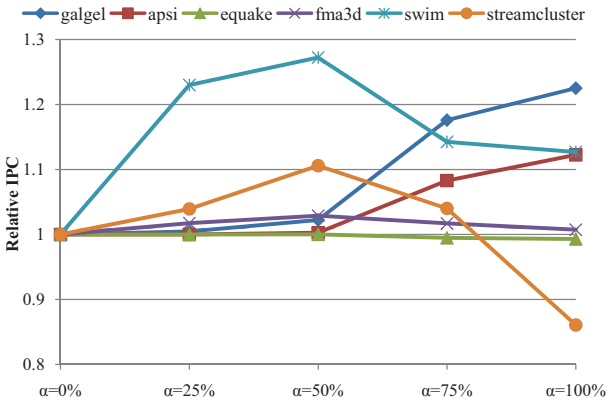

Figure 8. The impact of α on the performance. The IPC values are normalized by that of using the traditional policy.

A counter is required in order to make the accomplishment degree aware to the cache controller. The counter resets to zero and begins to count the number of cycles when a retirement begins. The cache controller check the counter and decides whether to stall the retirement for the read request. The area of 20 buffer entries can be evaluated as a cache whose size is $20 \times 64Byte$(less than 2KB). We use a $7 - bit$ counter to record the retirement accomplishment degree. Since the area of each 3D-stacked layer is around $60mm^2$, the area overhead of our proposed *read-preemptive write buffer* is less than 1%. Similarly, the leakage power increase caused by this buffer is also negligible.

Fig. 9 and Fig. 10 illustrates the performance and power improvement gained by our proposed *read-preemptive write buffer*. Compared to the IPC of SRAM baseline configurations, the average performance improvements are 9.93%

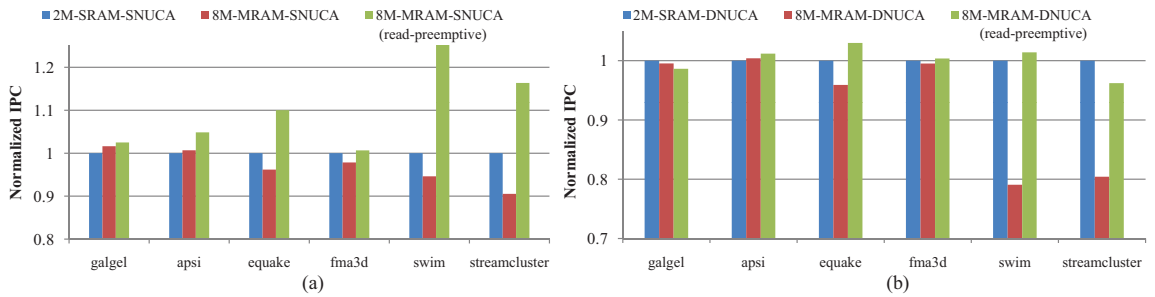

Figure 9. The comparison of IPC among 2M SRAM, 8M MRAM with traditional write buffer, and 8M MRAM with read-preemptive write buffer (Normalized by that of SRAM).

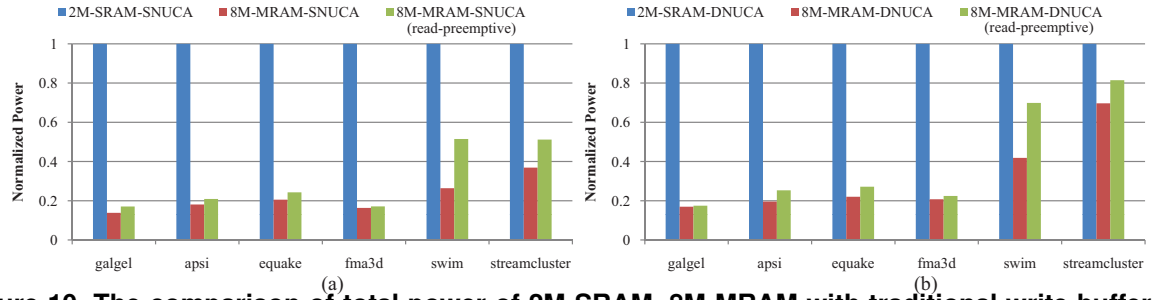

Figure 10. The comparison of total power of 2M SRAM, 8M MRAM with traditional write buffer, and 8M MRAM with read-preemptive write buffer (Normalized by that of SRAM).

and 0.41% for SNUCA and DNUCA, respectively. The average power reductions are 67.26% and 59.3% for SNUCA and DNUCA, respectively. Compared to the result of direct MRAM replacement shown in Fig. 5 and 6, the performance degradation is eliminated but the amount of power savings decreases. It is because using our read-preemptive write buffer causes some re-executions of write operations which consume more power.

5.2 SRAM-MRAM Hybrid L2 Cache

The aforementioned read-preemptive write buffer hides the MRAM long write latency, but the total number of write operations remains the same. In order to reduce the number of write operations to MRAM cells, we propose another technique called *SRAM-MRAM Hybrid Cache* and show how this technique can further reduce the dynamic power as well as improve the performance.

5.2.1 SRAM-MRAM Hybrid Cache Implementation

The proposed hybrid cache implementation is that, instead of building a pure MRAM cache, we compose the ways in each cache set with a majority of MRAM cache lines and a minority of SRAM ones. The main purpose is to keep as many write intensive data in the SRAM part as possible and hence reduce the number of write operations to the MRAM part. In this work, we design an SRAM-MRAM hybrid L2 cache with 31 ways of MRAM and 1 way of SRAM (*31M1S*).

After having these hybrid cache lines, the second step is to distribute MRAM cache lines and SRAM ones into separate cache banks. Considering the SRAM part is the minority in the proposed *31M1S* cache, one partitioning alternative is to distribute these SRAM cache lines into different banks so that there are several SRAM cache lines close to each processing core. However, this method requires each cache bank to be a heterogenous memory array with SRAM and MRAM cells and increases the complexity of the cache design. In addition, this distributed partitioning of SRAM cells implies that the SRAM and MRAM cells have to be fabricated together. Considering the specialization of the MRAM fabrication process, this method also eliminates the cost advantages of stacking MRAMs on top of processing cores.

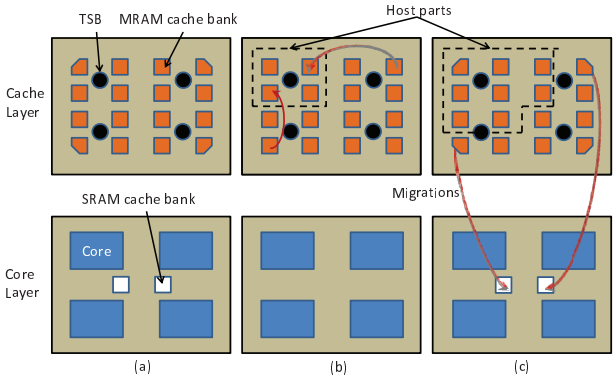

Figure 11. SRAM-MRAM hybrid cache implementation (a)one placement method of SRAM and MRAM cache banks,(b) data migrations in original MRAM caches, (c)data migrations in hybrid SRAM-MRAM caches.

Therefore, we use another alternative that, we reduce the number of cache lines in some MRAM cache banks

compared to the pure MRAM cache structure (as shown in Fig. 11(a) that the MRAM banks at four corners are smaller than other MRAM banks), compensate this cache line loss with SRAM ones, and collect all the SRAM cache lines together to build several entire SRAM banks on the core layer. As shown in Figure 11(a), SRAM cache banks are placed in the center of the core layer instead of being distributed. In this method, SRAM and MRAM cache banks have no difference from the architectural point of view.

Note that after placing one way of SRAM cache lines in the core layer, the area of the core layer will increase and the area of the cache layer will decrease. In this work, the total size of all the SRAM cache lines is $256KB$, the derived area overhead is about 12.5%.

5.2.2 Hybrid Cache Management Policy

Another important issue is how to manage the hybrid L2 cache to improve the performance and reduce the power. Because the key point is to reduce the number of write operations to MRAM cache cells, we need to move as many write intensive data in SRAM cache banks as possible. The management policy of the hybrid cache can be described as follows:

- The cache controller is aware of the locations of SRAM cache ways and MRAM cache ways. When there is a write miss, the cache controller first try to place the data in the SRAM cache ways.
- Considering the high probability that a core write data to a specific group of cache lines repeatedly, data in MRAM caches should be migrated to SRAM caches if the some cache lines are frequently written to. In this work, data in MRAM caches will be migrated to SRAM caches when they are accessed by two successive write operations. This kind of data migration is named *intra-migration* to differentiate *inter-migration* policy introduced in Section 3. Due to the existence of this intra-migration policy, the number of write accesses from cores to MRAM caches can be reduced.
- Note that read operations from cores are also possible to cause data migrations, the number of which could be even larger than that of direct write accesses from cores. Therefore, a new type inter-migration policy is introduced. Figure 11(b) and (c) compare the banks from which data can be migrated toward the core in upper-left corner. Figure 11(b) shows that, in original inter-migration policy, the cache layer is divided into 4 uniform groups and there is only one core associative with each part. In this work, banks in each group are named as the *host banks* of their corresponding core. Data can only be migrated from *non-host banks*. For the traditional management policy, the data will be migrated to *host bank*. For the management policy proposed for the hybrid cache, the data can only be migrated to SRAM banks.

Two data migrations are illustrated in Fig. 11(b) for the traditional inter-migration. When using the hybrid SRAM-MRAM cache, the *host banks* for a core is redefined as shown in Fig. 11(c). Two corresponding data migrations are also shown in Fig. 11(c). Using this policy, there is no data migration between two MRAM cache lines, which reduces the number of write operations greatly. The drawback is that SRAM banks are shared by all cores so that their limited sizes may increase L2 miss rates. Considering we have $8M$ of total cache size, which is considerably large for most applications, our simulation results show that the increase of L2 miss rates is very small.

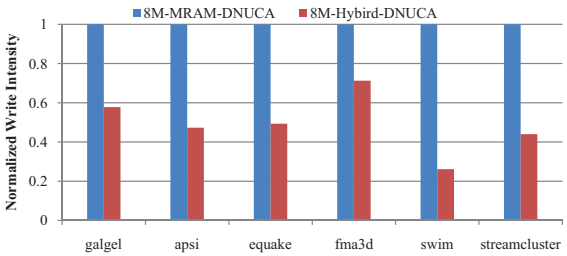

Figure 12. The MRAM write intensity to MRAM before and after using hybrid SRAM-MRAM caches.

Fig. 12 shows the number of MRAM write operations per 1K instructions is reduced dramatically by using our hybrid SRAM-MRAM approach. As a result, the dynamic power associated with write operations to MRAM cells is also reduced and the performance is improved. Fig. 13 shows the performance comparison. On average, the hybrid cache structure improves the performance by 5.65%, which means it mitigates the performance loss of MRAM caches from 8.48% to 2.61% compared to their SRAM counterparts.

In Fig. 14 shows the power comparison. We observe that the total power is reduced except for *"galgel"*. It is because both read and write intensities in *"galgel"* are so small that the dynamic power is very low. Consequently, the introduction of SRAM cache lines in the hybrid cache brings the leakage power back and eliminates the dynamic power reduction achieved by the hybrid structure. However, as the write intensity increases, the SRAM-MRAM hybrid cache starts to save total power consumptions. For example, the total power consumption is cut by more than half for workloads such as *"swim"* and *"streamcluster"*. On average, after the transition from SRAM caches to MRAM ones, our proposed hybrid cache further reduces the total power by 12.45%.

5.3 Combination of Read-preemptive Buffer and Hybrid Architecture

We combine the two techniques together as an optimized MRAM L2 cache architecture . In this architecture, we get more benefits from the advantages of the MRAM cache; at

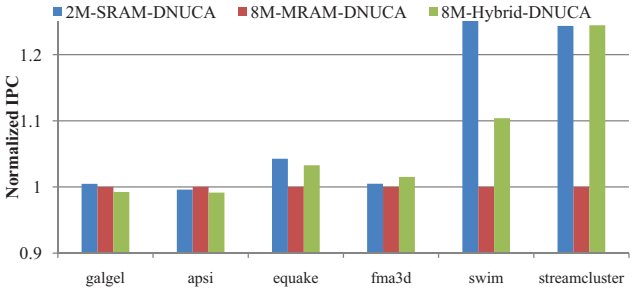

Figure 13. The comparison of IPC among 2M SRAM cache, 8M MRAM pure cache, and 8M SRAM-MRAM hybrid cache (Normalized by the IPC of 8M MRAM pure cache).

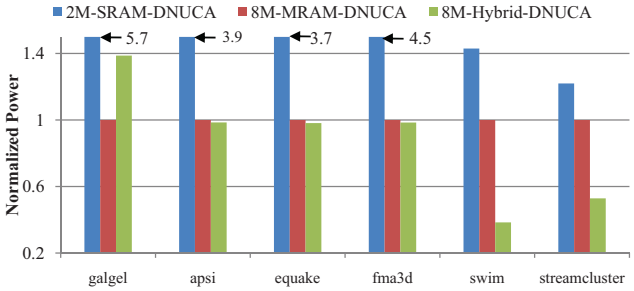

Figure 14. The comparison of total power consumption among 2M SRAM cache, 8M MRAM pure cache, and 8M SRAM-MRAM hybrid cache (Normalized by the total power consumption of 8M MRAM pure cache).

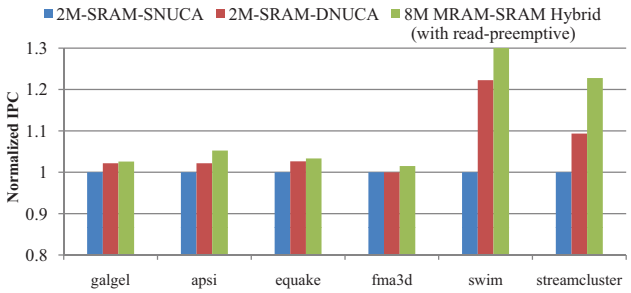

Figure 15. The comparison of IPC among 2MB SRAM SNUCA cache, 2MB SRAM DNUCA cache, and 8MB SRAM-MRAM hybrid cache with read-preemptive write buffer (Normalized by the IPC of 2MB SRAM SNUCA cache).

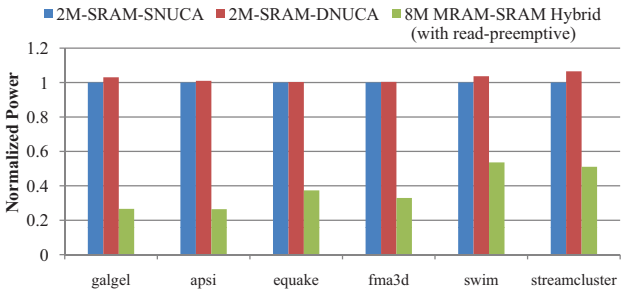

Figure 16. The comparison of total power consumption among 2MB SRAM cache, 2MB SRAM cache, and 8MB SRAM-MRAM hybrid cache with read-preemptive write buffer (Normalized by the total power consumption of 2MB SRAM cache).

the same time, mitigate the penalties caused by write operations. The performance and power comparisons are shown in Fig. 15 and Fig. 16, respectively. The average IPC is improved by 4.91% compared to the SRAM SNUCA baseline, while power saving is 73.5%. Table 6 gives an overview of the performance and power improvements.

Table 6. The performance and power improvement overview (Use 2MB SRAM L2 SNUCA cache as the baseline)

	Performance	Total Power
Read-preemptive buffer	9.93%	67.26%
Hybrid cache	-2.61%	85.45%
Combined	4.91%	73.5%

6 Related Work

Some previous research focused on the performance improvement by stacking DRAM main memories on top of processors. 3D cache model has been developed to facilitate architectural level analysis [15, 25]. Performance analysis of 3D stacking memory was studied by Loi *et al.* [21]. Li, *et al.* have also reported performance improvement by using stacked SRAM L2 caches for CMPs [20] . Black, *et al.* studied the benefits of stacking a large DRAM or SRAM cache on a Intel Core 2 Duo processor, and achieved considerable performance improvement [4]. Loh [22] presented an aggressive 3D DRAM integration as on-chip main memories. Kgil *et al.* [17] have implemented an aggressive CMP method by replacing all the L2 caches with in-order simple processor cores, and uses 3D stacking DRAM to satisfy the memory capacity and bandwidth requirements. Ghosh, *et al.* proposed a new method to reduce the power consumption in systems where the DRAM is stacked on top of the processor cores [13]. A prototype of the 80-core Teraflop processor with an SRAM layer stacked on top of the processor cores, which was designed and fabricated by Intel, also demonstrated the benefits of stacking SRAM memories on CMPs [5]. There is previous work that studied the benefits of replacing SRAM caches with MRAM caches for a single processor core [10, 12]. However, they only focused on the single-core architecture with UCA caches. For CMPs, especially when the entire cache capacity becomes much larger, the effect of NUCA has to be considered and the data migration in NUCA can also change the cache read/write behavior.

7 Conclusion

MRAM is a promising candidate of on-chip memories and the emerging 3D heterogeneous integration makes it feasible to stack MRAM as L2 caches for CMPs. In this work, we present a cache model for MRAM L2 cache stacking and evaluate its performance and power benefit. Even though replacing SRAM L2 cache with MRAM can result in significant power savings, the drawback comes from MRAM's long write latency and high write energy. As a result, for applications with high L2 cache write intensities, the performance can be degraded and the power saving can be reduced. Therefore, we propose two techniques: read-preemptive write buffer to mitigate the performance penalty caused by the long write latency; SRAM-MRAM hybrid L2 cache to reduce the number of MRAM write operations. Our result shows these two techniques can make MRAM cache work effective for most workloads regardless of their write intensities.

References

[1] http://parsec.cs.princeton.edu/.

[2] http://www.hpl.hp.com/research/cacti/.

[3] http://www.spec.org/.

[4] B. Black, M. Annavaram, N. Brekelbaum, et al. Die Stacking (3D) Microarchitecture. In *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 469–479, 2006.

[5] S. Borkar. 3D Technology: A System Perspective. In *Technical Digest of the International 3D System Integration Conference*, pages 1–14, 2008.

[6] D. Burger, J. R. Goodman, and A. Kagi. Limited Bandwidth to Affect Processor Design. *Micro, IEEE*, 17(6):55–62, 1997.

[7] Z. Chishti, M. D. Powell, and T. N. Vijaykumar. Optimizing Replication, Communication, and Capacity Allocation in CMPs. *SIGARCH Comput. Archit. News*, 33(2):357–368, 2005.

[8] J. D. Davis, J. Laudon, and K. Olukotun. Maximizing CMP Throughput with Mediocre Cores. In *PACT '05: Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques*, pages 51–62, 2005.

[9] W. R. Davis, J. Wilson, S. Mick, et al. Demystifying 3D ICs: The Pros and Cons of Going Vertical. *IEEE Design and Test of Computers*, 22(6):498–510, 2005.

[10] R. Desikan, C. R. Lefurgy, S. W. Keckler, and D. Burger. On-chip MRAM as a High-Bandwidth Low-Latency Replacement for DRAM Physical Memories. Technical report, 2002.

[11] Z. Diao, Z. Li, S. Wang, et al. Spin-Transfer Torque Switching in Magnetic Tunnel Junctions and Spin-Transfer Torque Random Access Memory. *Journal of Physics: Condensed Matter*, 19(16):165209 (13pp), 2007.

[12] X. Dong, X. Wu, G. Sun, et al. Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) as a Universal Memory Replacement. In *DAC '08: Proceedings of the 45th annual conference on Design automation*, pages 554–559, 2008.

[13] M. Ghosh and H.-H. S. Lee. Smart Refresh: An Enhanced Memory Controller Design for Reducing Energy in Conventional and 3D Die-Stacked DRAMs. In *MICRO '07: Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 134–145, 2007.

[14] M. Hosomi, H. Yamagishi, T. Yamamoto, et al. A Novel Non-Volatile Memory With Spin Torque Transfer Magnetization Switching: Spin-RAM. In *International Electron Devices Meeting*, pages 459–462, 2005.

[15] P. Jacob, O. Erdogan, A. Zia, et al. Predicting the Performance of a 3D Processor-Memory Chip Stack. *IEEE Design and Test of Computers*, 22(6):540–547, 2005.

[16] J. A. Kahle, M. N. Day, H. P. Hofstee, et al. Introduction to the Cell Multiprocessor. *IBM Journal of Research and Development*, 49(4/5):589–604, 2005.

[17] T. Kgil, S. D'Souza, A. Saidi, et al. PicoServer: Using 3D Stacking Technology to Enable a Compact Energy Efficient Chip Multiprocessor. In *Proc. the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, number 11, pages 117–128, 2006.

[18] C. Kim, D. Burger, and S. Keckler. An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches. In *Proc. the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2002.

[19] P. Kongetira, K. Aingaran, and K. Olukotun. Niagara: A 32-Way Multithreaded SPARC Processor. *IEEE Micro*, 25(2):21–29, 2005.

[20] F. Li, C. Nicopoulos, T. Richardson, et al. Design and Management of 3D Chip Multiprocessors Using Network-in-Memory. In *ISCA '06: Proceedings of the 33rd Annual International Symposium on Computer Architecture*, pages 130–141, 2006.

[21] C. C. Liu, I. Ganusov, M. Burtscher, and S. Tiwari. Bridging the Processor-Memory Performance Gap with 3D IC Technology. *IEEE Design and Test of Computers*, 22(6):556–564, 2005.

[22] G. H. Loh. 3D-Stacked Memory Architectures for Multi-core Processors. In *ISCA '08: Proceedings of the 35th International Symposium on Computer Architecture*, pages 453–464, 2008.

[23] G. L. Loi, B. Agrawal, N. Srivastava, et al. A Thermally-Aware Performance Analysis of Vertically Integrated (3-D) Processor-Memory Hierarchy. In *DAC '06: Proceedings of the 43rd Annual Conference on Design automation*, pages 991–996, 2006.

[24] P. S. Magnusson, M. Christensson, J. Eskilson, et al. Simics: A Full System Simulation Platform. *Computer*, 35(2):50–58, 2002.

[25] Y.-F. Tsai, Y. Xie, N. Vijaykrishnan, and M. J. Irwin. Three-Dimensional Cache Design Exploration Using 3DCacti. In *ICCD '05: Proceedings of the 2005 International Conference on Computer Design*, pages 519–524, 2005.

[26] Y. Xie, G. H. Loh, B. Black, and K. Bernstein. Design Space Exploration for 3D Architectures. *ACM Journal on Emerging Technologies in Computing Systems*, 2(2):65–103, 2006.

[27] W. Zhao, E. Belhaire, Q. Mistral, et al. Macro-model of Spin-Transfer Torque based Magnetic Tunnel Junction Device for Hybrid Magnetic-CMOS Design. In *IEEE International Behavioral Modeling and Simulation Workshop*, pages 40–43, 2006.