

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

## Abstract

*Magnetic Random Access Memory (MRAM) is considered to be a promising future memory technology due to its low leakage power, high density and fast read speed. The heterogeneous integration enabled by the 3D integration technology makes it cost-efficient to stack MRAM on top of conventional CMPs. However, one disadvantage of MRAM is its long latency and high energy consumption associated with write operations. In this paper, we first present a cache model of stacking MRAM-based L2 cache on top of Chip Multiprocessors (CMPs), and compare it against its SRAM counterpart in terms of area, performance, and energy consumption. Through simulation results, we observe that a naive implementation of MRAM stacking can harm the chip performance and fail to fully take the advantages of MRAM, due to the aforementioned long latency and high energy of writes. We then propose two architectural techniques: read-preemptive write buffer and SRAM-MRAM hybrid L2 cache, which can mitigate the penalty due to long latency and high energy associated with write operations in MRAM, and therefore help improve the performance and reduce the power of the MRAM-based L2 cache. Our simulation results show that considerable performance improvement and power reduction can be achieved with our proposed techniques.*

## 1 Introduction

The diminishing returns of endeavors to increase clock frequency and exploit instruction level parallelism in a single processor have led to the advent of Chip Multiprocessors (CMPs), which are processing cores forming a decentralized micro-architecture that scales more efficiently with increased integration densities [1, 2]. The integration of multiple cores on a single chip is expected to accentuate the already daunting “memory wall” problem [2, 3]. Supplying enough data to massive multi-core chips will become a major challenge for performance scalability.

The introduction of the three-dimensional (3D) integration technology [4,5] provides an opportunity to stack memory on top of processor cores; therefore, it helps alleviate the memory bandwidth challenges in CMPs. Recently, active research has targeted stacking SRAM caches or DRAM memories on top of processor cores [6, 7]. For example, Black *et al.* have studied the benefits of stacking a large SRAM or DRAM cache on an Intel Core 2 Duo processor [6]. Loh [7] has presented a 3D CMP architecture with stacked DRAM memories, and showed that 3D stacking can improve performance with reduced access latency and increased memory bandwidth. Intel has also demonstrated an 80-core Teraflop prototype chip with SRAM layer stacked on top of the processing cores [8].

Magnetic Random Access Memory (MRAM) is considered as a promising memory technology due to many attractive features such as low leakage, high density, fast read speed, non-volatility, and immunity to radiation-induced soft errors [9, 10]. However, previous research on integrating MRAM to the on-chip memory is very limited. One key

obstacle is the difficulty of integrating MRAM with conventional CMOS logic in 2D chips. The fabrication of MRAM involves hybrid magnetic-CMOS process, requiring the growth of magnetic stacks between metal layers. Consequently, it incurs extra cost and additional fabrication complexity of integrating MRAM with conventional CMOS logic into the same 2D chip. Fortunately, the emerging 3D integration technology enables low cost integration of mixed technologies, which is ideal for stacking MRAM memory on top of CMOS logic.

Some recent work has evaluated the benefits of MRAM as a universal memory replacement for L2 caches and main memories in a single processor [11, 12]. In this paper, we evaluate the benefits of stacking MRAM L2 cache on top of CMPs. We first develop a cache model for stacking MRAM, and compare the MRAM-based L2 cache against its SRAM counterpart in terms of area, performance and energy consumption. Our simulation results show that: (1) For applications that have moderate write intensity to L2, the MRAM-based L2 can reduce the total cache power significantly due to its low leakage power consumption, and achieve considerable performance improvement due to reduced L2 cache miss rate with larger cache capacity; (2) For applications that have high write intensity to the L2 cache, MRAM-based cache can cause performance degradation and increased dynamic power, due to long latency and high energy consumption associated with write operations.

These two observations imply that, although the advantageous features of MRAM are favorable, MRAM-based caches may not work efficiently if we directly combine it with a traditional CMP architecture. Its long latency and high energy of *write operations* may eliminate the benefits in some cases. In light of this problem, we then propose two architectural techniques: *read-preemptive write buffer* and *MRAM-SRAM hybrid L2 cache*, which can mitigate the penalties due to long latency and high energy associated with write operations in MRAM caches, and therefore help improve the performance and reduce the power of the MRAM-based L2 cache. Our simulation results show that performance improvement and power reduction can be achieved effectively with our proposed techniques.

The rest of the paper is organized as follows. Section 2 provides the background on the MRAM and 3D integration technology. Section 3 presents a traditional 3D NUCA cache model for CMPs; Section 4 discusses the benefits and limitations of replacing the SRAM L2 cache with the MRAM one; Section 5 proposes several techniques to improve the performance and reduce the power consumption of MRAM L2 cache associated with penalties of the write operation; Finally we introduce related work in Section 6 and conclude the paper in Section 7.

## 2 Background

This section provides the background of the MRAM technology and gives a brief overview of the 3D integration technology. The features of MRAM and its compatibility with 3D stacking are also introduced.

### 2.1 An Overview of the MRAM Technology

The basic difference between the MRAM and the conventional RAM technologies (such as SRAM/DRAM) is that the information carrier of MRAM is a Magnetic Tunnel Junction (MTJ) instead of electric charges [9, 10]. Each MTJ contains *two ferromagnetic layers* and *one tunnel barrier layer*. Figure. 1 shows a conceptual illustration of a MTJ.

One of the ferromagnetic layer (reference layer) has fixed magnetic direction while the other one (free layer) can change its magnetic direction by an external electromagnetic field or a spin-transfer torque. If the two ferromagnetic layers have different directions, the MTJ resistance is high, indicating a “1” state (the anti-parallel case in Figure. 1(a)); if the two layers have the same direction, the MTJ resistance is low, indicating a “0” state (the parallel case in Figure. 1(b)).

The MRAM technology to be discussed in this paper is called *Spin-Transfer Torque RAM* (STT-RAM), which is a new generation of MRAM technologies. STT-RAMs change the magnetic direction of the free layer by directly passing a spin-polarized current through the MTJ structure. Comparing to the previous generation of MRAMs that uses external magnetic fields to reverse the MTJ status, STT-RAMs 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 the STT-RAM memory cell design, the most popular structure is composed of one NMOS transistor as the access controller and one MTJ as the storage element (“1T1J” structure) [9]. As illustrated in Fig. 2, 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 voltage ( $V_{BL} - V_{SL}$ ) is applied between the bit-line (BL) and the source-line (SL). This voltage is negative and is usually very small (-0.1V as demonstrated in [9]). This voltage difference will cause a current passing through the MTJ, but it is not small enough to invoke a disturbed write operation. The value of the current is determined by the equivalent resistance of MTJs. A sense amplifier compares this current with 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 positive voltage difference is established between SLs and BLs for writing for a “0” or a negative voltage difference is established for writing a “1”. 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 [13].

In this work, we select the writing pulse duration to be a typical value of  $10ns$  [10]. In addition, we scale the STT-MRAM specification from previous work [9] to the  $65nm$  technology node. Assuming the size of MTJs to be  $65nm \times 90nm$ , the derived threshold current for magnetic reversal is about  $195\mu A$ .

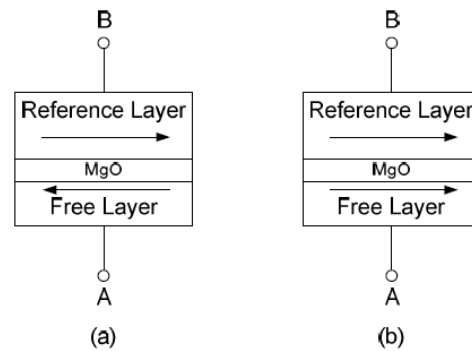


Figure 1: A conceptual view of MTJ structure. (a) Anti-parallel (high resistance), which indicates “1” state; (b) Parallel (low resistance), which indicates “0” state.

## 2.2 The 3D Integration Technology

The increasing number of devices and functionality on a single chip leads to increasing complexity in interconnecting devices with a large number of metal layers. With continued technology scaling, interconnect has emerged as a dominant source of circuit delay and power consumption. The reduction of interconnect delays and power consumption are of paramount importance for deep-submicron designs. 3D ICs have recently emerged as a promising means to mitigate these interconnect-related problems [4,5,14,15]. Several 3D integration technologies have been explored recently, including wire bonded, microbump, contactless (capacitive or inductive), and through-silicon-via (TSV) vertical interconnects [5]. TSV 3D integration has the potential to offer the greatest vertical interconnect density, and therefore is the most promising one among all the vertical interconnect technologies. In 3D ICs that are based on TSV technology, multiple active device layers are stacked together (through wafer stacking or die stacking) with direct vertical TSV interconnects [4].

3D ICs offer a number of advantages over traditional two-dimensional (2D) designs [5]: (1) shorter global interconnects because the vertical distance (or the length of TSVs) between two layers is usually in the range of  $10\ \mu m$  to  $100\ \mu m$  [4], depending on manufacturing processes; (2) higher performance because of the reduction of average interconnect length, as well as bandwidth improvement due to die stacking; (3) lower interconnect power consumption due to wire-length reduction; (4) higher packing density and smaller footprint; (5) support for the low cost integration of mixed-technology chips (for example, MRAM stacking on top of CMOS processor cores). More details about 3D integration can be found in a tutorial paper by Davis, *et al.* [5].

## 3 Technology and Architecture Models

In this section, we present our 3D NUCA model implemented with Network-on-Chip (NoC), which is suitable for both SRAM and MRAM caches.

### 3.1 Modeling an MRAM-based Cache

To model the MRAM-based cache, the first step is to estimate the area of a MRAM cell. As shown in Fig. 2, each MRAM cell is composed of one NMOS transistor and one MTJ. Note that the new generation of MRAM (STT-RAM) technology offers scalability for MTJ, the size of MTJ is only limited by manufacture techniques. However, the NMOS transistor has to be sized properly so that it can pass sufficient current for the MTJ to change the cell status. Since the current driving ability of NMOS transistor is proportional to its W/L ratio, the W/L ratio becomes the key point to determine the MRAM cell size.

By using HSPICE simulation, we find that the minimum W/L ratio for the NMOS transistor in  $65nm$  technology

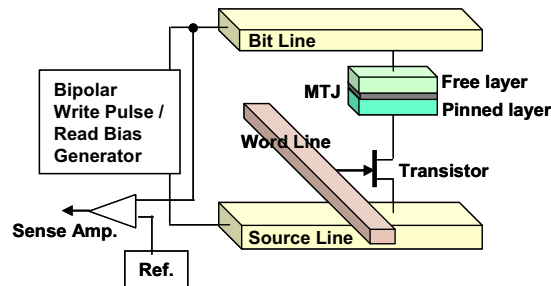


Figure 2: An illustration of an MRAM cell

should be 10 in order to drive the required threshold current of  $195\mu A$  for the status reversal. We assume the width of the source or drain regions of an NMOS transistor is  $1.5F$ , where  $F$  is the feature size. Combining these requirements together, the size of the minimum MRAM cell we can design is set to  $10F \times 4F$ , which equals to  $40F^2$ . The specification of our targeted MRAM cell is listed in Table .

Table 1: MRAM Cell Specifications

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

Despite the physical mechanism in the storage cell, MRAM and SRAM caches have almost the same peripheral interfaces from the circuit designer’s point of view. In 65nm technology, the cell area simulation results show that the area of one MRAM cell is about 25% of one SRAM cell( $146F^2$  extracted from CACTI). Table 2 shows the comparison of area, access time and access energy for a 512KB MRAM cache bank and a 128KB SRAM bank, which are used in this work.

Table 2: Area, access time and energy comparison(65nm technology)

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

### 3.2 The 3D NUCA Cache Model

As the caches size increases, the wire delay in deep sub-micron regions has made the Non-Uniform Cache Architecture (NUCA) [16] more attractive than the one with uniform access latencies. In NUCA, the cache is divided into multiple banks with different access latencies according to their locations relative to cores. These individual banks can be connected through a mesh-based interconnection network called Network-on-Chip (NoC).

Based on the latest 2D cache simulator CACTI [17], we developed our NoC-based 3D NUCA cache model, which can incorporate the features of 3D integrations. The key concept is to use NoC routers for communications within a 2D layer, while using a special through-silicon-bus (TSB) to communicate in the vertical direction between layers. We also elaborated low-level parameters such as the latency and the energy of interconnection wires and modified the technology parameters to be consistent with the MRAM technology.

Figure 3(a) illustrates an example of the 3D NUCA structure. There are four cores located in the layer 1. In each core, there is a cache controller connected to a through-silicon-bus (TSB) from which data are moved through layers between processing cores and caches. The TSBs are implemented with through-silicon-vias (TSVs). This bus structure has the advantage of short connections provided by 3D integrations. It has been reported that the vertical latency for traversing the height of a 20-layer stack is only 12ps [18]. The latency of traversing on a TSB through one layer is negligible compared to the one traversing between two NoC routers in 2D. Consequently, it is feasible to have single-hop communication with these buses because of the short distance between the layers. Using the vertical buses, the processing cores can have the same access latency to banks in the same location of different layers. In addition, hybridization of the NoC router with the bus requires only one additional link (instead of two) on the NoC router, because the bus is a single entity for communicating both up and down [19].

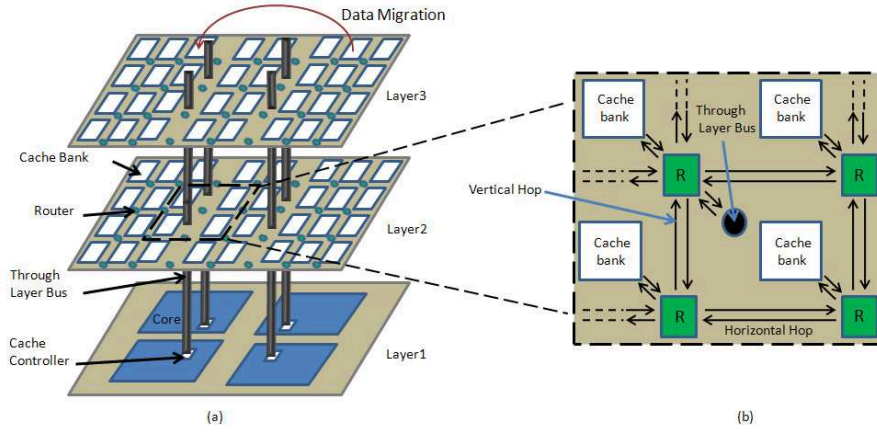


Figure 3: (a) An illustration of the proposed 3D NUCA structure, which includes 1 core layer, 2 cache layers. There are 4 processing cores per core layer, 32 cache banks per cache layer, and 4 through-layer-bus across layers; (b) Connections amongst routers, caches banks and through-layer-buses.

As shown in Figure3(a), L2 caches are placed on top of cores and they are located on layer 2 and layer 3. There are 32 cache banks in each layer which are connected with network on-chip routers. Each cache bank can be implemented with either SRAMs or MRAMs. Figure3(b) shows a detailed 2D structure for a cache layer. Three out of the four routers have five input/output ports connecting the four neighbor routers and the internal cache bank, while the fourth router needs an extra input/output port to communicate with the through-silicon-bus (TSB).

Similar to prior approaches [19–21], the proposed model supports the migration of moving data closer to the accessing core. For set associative cache, the cache ways belonging to the same index should be distributed into different banks in order to implement the data migration. In our model, each cache layer is equally divided into several zones. The number of zones is equal to the

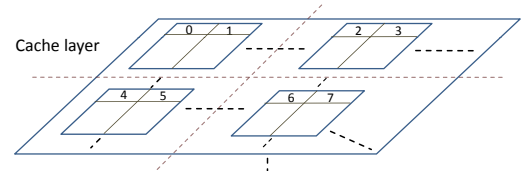


Figure 4: Eight caches ways are distributed in four banks. Assume four cores and accordingly four zones each layer

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. Figure 4 gives an illustration of distributing eight ways into four zones. Note that, in this architecture, one cache request should be sent to all zones so that all caches ways are accessed. It will increase the network congestion which are also modeled as a part of router latency. The migration policy and cache banks placements are tailored to the 3D architecture so that data migrations are limited to the same layer. This limitation help reduce the workloads of TSBs. Figure 3(a) shows an example of data migration after which the core in the upper-left corner can access the data faster. Note that this data migration policy is called *inter-migration* in this work, in order to differentiate another migration policy introduced later.

The  $memory(caches) + logic(cores)$  structure is employed in our model as shown in Figure3(a). The advantages of this structure 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 the TSB and routers because the TSB is now connected to cache controllers directly, and there is no direct connection between routers and cache controllers.

We provide one through-layer-bus for each core in the model. This aggressive structure gains more benefits from the high band-width advantages of 3D stacking. It is reported that TSVs have pitches of only 4-10 $\mu m$  [18]. Even at the high-end with a 10 $\mu m$  TSV-pitch, a 1024-bit bus (much wider than our TSB) would only require an area of 0.32 $mm^2$ . In our model, the die area of an 8-core CMP is assumed to be 60 $mm^2$  (introduced later). Therefore, it is feasible to assign one TSB for each core.

### 3.3 Configurations and Assumptions

Our baseline configuration is an 8-core in-order processor using the Ultra SparcIII ISA. We estimate the area of SRAM cache banks with CACTI6.0 [17]. For MRAM cache banks, we extract low-level parameters of wire connections and routers from CACTI, and input them to our MRAM cell model to estimate the area. Currently, there is no tool to predict the area of processing cores. We investigate real hardware, such as Cell Processor, Sun UltraSPARC T1, etc [22] [23], and estimate the area of eight processing cores to be 60 $mm^2$ . Furthermore, we assume that one cache layer fits to either a 2MB SRAM or an 8MB MRAM L2 cache. The configurations are detailed in Table 3. Note that the power of each core is also estimated based on data sheets from real products [22] [23].

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 the *OpenMP2001* [25] and *PARSEC* [26] suites, as shown in Table 4. These benchmarks are chosen so that we have a range of transaction intensities to the L2 caches, since the performance and power of MRAM caches are closely related to transaction intensity. The average numbers of total transactions and write transactions to L2 caches per 1K instructions are listed in Table 4. We pin one thread on one core during the simulation. For each simulation, we fast forward to warm up the caches, and then run three billion cycles in detailed mode. We use the total IPC of all the cores as the metric of the performance.

## 4 Replacing SRAM with MRAM as L2 caches

In this section, we briefly discuss two approaches of replacing SRAM L2 caches with MRAM caches, using the architecture of CMPs introduced in the prior section.

### 4.1 Approach I: Same Capacity Replacement

The first replacement strategy is to replace the SRAM L2 cache with the MRAM L2 cache that has the same capacity (For example, replacing a 2MB SRAM L2 cache with a 2MB MRAM L2 cache). By using this approach, the die area of the MRAM L2 cache is much smaller than that of SRAM L2 cache.

The area of a MRAM cache bank is only about 25% of the area for a SRAM cache bank with the same capacity (for example, a 128KB SRAM bank has the similar area as a 512KB MRAM bank does, as shown in Table 2). Even though

<b>Processors</b>	
Number of cores	8
Frequency	3GHz
Power	6W/core
Issue Width	1 (in order)
<b>Memory Parameters</b>	
L1 (private I/D)	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
<b>Network Parameters</b>	
Number of Layers	2
Number of TSB	8
Hop latency	TSB : 1 cycle V_hop : 1 cycle; H_hop : 1 cycle
Router Latency	2-cycle

Table 3: Baseline configuration parameters

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

Table 4: L2 transaction intensities. TPKE is the number of total transactions per 1K instructions and WPKE is the number of write transactions per 1K instructions.

the bit line read delay for MRAM and SRAM cell are different, it is only a small percentage of the total cache access delay, comparing to other delay components such as H-Tree input and output delay, decoder delay [12]. Consequently, the read latency of a 128KB SRAM bank is similar to its 512KB MRAM counterpart (as shown in Table 2). When we replace the SRAM cache with the MRAM one, the total number of MRAM cache banks should be about 25% of the bank number for SRAM caches. In our NUCA cache model, the smaller bank number also implies the smaller number of routers and the shorter length of interconnection. Consequently, the average access latencies from cores are reduced. However, due to the space limitation, we will not discuss the details of the evaluation of using this “same capacity” replacement strategy. As we mentioned before, we employ the features of MRAM caches, such as low leakage power and high density, to resolve the “memory wall” challenges for CMPs, and it is desirable to integrate as larger capacity of on-chip memory as possible. This leads to another approach called “Same Area” strategy, which is described in the next subsection.

## 4.2 Approach II: Same Area Replacement

An alternative approach is to replace a SRAM L2 cache with a MRAM L2 cache that has the similar area. In this strategy, we integrate as many caches in the cache layers as possible. In a 2M SRAM cache layer that has  $16 \times 128KB$  banks, the SRAM banks can be replaced by a MRAM cache with  $16 \times 512KB$  banks, with the similar average on-chip-

network access latency for each bank (as shown in Table 2).

#### 4.2.1 performance Analysis

Because the number of banks remains the same, the average latencies of read operations to the SRAM cache and the MRAM cache are similar. Since the capacity of the MRAM L2 cache is much larger, the access miss rate to the L2 cache should decrease. The normalized access miss rates of L2 caches (a comparison between using the 2M SRAM L2 cache and the 8M MRAM L2 cache) are listed in Figure 5. On average, the miss rates are reduced by 19.0% and 12.5% for SNUCA MRAM cache and DNUCA MRAM cache, respectively.

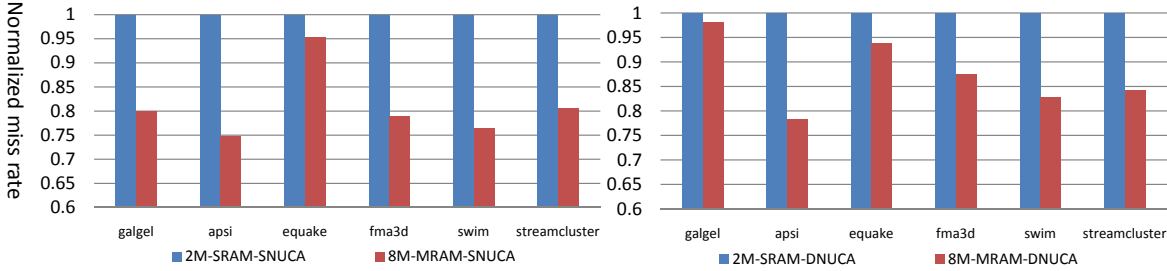


Figure 5: The comparison of L2 caches access miss rates for SRAM L2 cache and MRAM L2 cache that have similar area. Larger capacity of MRAM cache results in smaller cache miss rates.

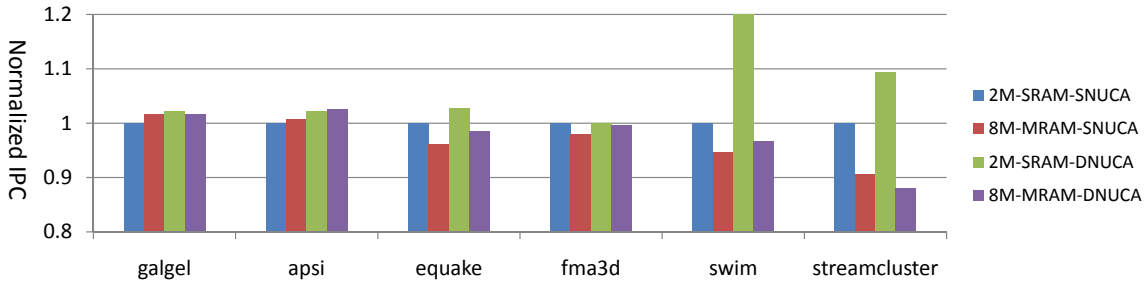


Figure 6: IPC using different configurations of MRAM and SRAM caches(Normalized by that of SRAM).

The IPC values of running different benchmarks are listed in Figure 6. Although the miss rates L2 cache decrease when we use the MRAM cache, resulting a small performance improvement for the first two benchmarks (“galgel” and “apsi”), but the performance for the last four benchmarks is not improved as expected. On average, the performance degradations are about 3.09% and 7.52% for SNUCA MRAM and DNUCA MRAM, respectively. From Table 4, one can observe that the intensities of write operations (WPKI) of the first two benchmarks are much lower than that of the last four benchmarks. It means that a smaller L2 miss rate of using a large MRAM cache can bring benefits for certain, especially when the intensity of write operations is low. However, when the intensity of write operations is high, the performance penalty due to long latency associated with write operations offsets the benefits from smaller miss rates due to the larger capacity of MRAM cache. This observation is further supported by comparing the results of using SNUCA and DNUCA. From Figure 6, one can observe that performance degradation is more significant when we use DNUCA MRAM cache. It is because data migrations in DNUCA incur more write operations, which cause more penalties when using the MRAM cache.

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

**Observation 1** *Replacing the SRAM L2 cache with a MRAM cache, 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 there is a high intensity of write operations to the MRAM cache, the benefits of smaller miss rate could be totally eliminated by the penalties due to write operations, eventually resulting in performance degradation.*

#### 4.2.2 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-90 $\mu m$ , the leakage power in CMOS technology becomes dominant. Since MRAM caches use non-volatile MJT to store data, it is feasible to turn off the power supply to the MRAM cells that are not being accessed for leakage power reduction. Figure 8 lists the total leakage power of SRAM and MRAM caches used in this work.
- *Dynamic Power:* The calculation of the dynamic power 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 work, these values are either simulated by HSPICE or extracted from the CACTI [17]. The access number of routers and the length of wire connections vary based on the location of the requesting core and the requested cache lines. Moreover, the write and the read operation also consumes different amount of dynamic power. Therefore, we need to know the exact number of read and write transactions in order to get the average dynamic power.

Figure 7 shows the comparison of the total power for SRAM and MRAM L2 caches. One can observe that:

- For all benchmarks, the total power of the MRAM caches is smaller than that of the SRAM cache. The average power savings across all benchmarks are about 78% and 68% for SNUCA and DNUCA, respectively. The power saving for DNUCA MRAM is smaller, because more write operations due to data migration cause extra dynamic power consumptions. It is obvious that the “low leakage power” feature makes the MRAM more attractive to be used as large on-chip memory, especially when leakage power dominates SRAM’s total power consumption when technology scales.
- The average power saving for the first four benchmarks is more than 80%, due to low intensity of write operations to L2 cache. However, for the last benchmark (“*streamcluster*”), the total power savings are about 63% and 30% for SNUCA and DNUCA, respectively. The saving is much smaller than the average across all benchmarks, because it has much higher L2 cache write intensity than other benchmarks (see Table 4). The high energy associated with the write operations in MRAM makes the total power savings not as significant as other benchmarks.

- For SRAM L2 cache, since leakage power dominates, the total power for SNUCA SRAM and DNUCA SRAM are very close. On the contrary, the dynamic power may dominate in the MRAM cache because of its low leakage power and high energy of the write operation. Consequently, for benchmarks with high intensity of write operations to L2 (such as “swim” and “streamcluster”), the total power of DNUCA MRAM cache could be much higher than that of SNUCA MRAM cache.

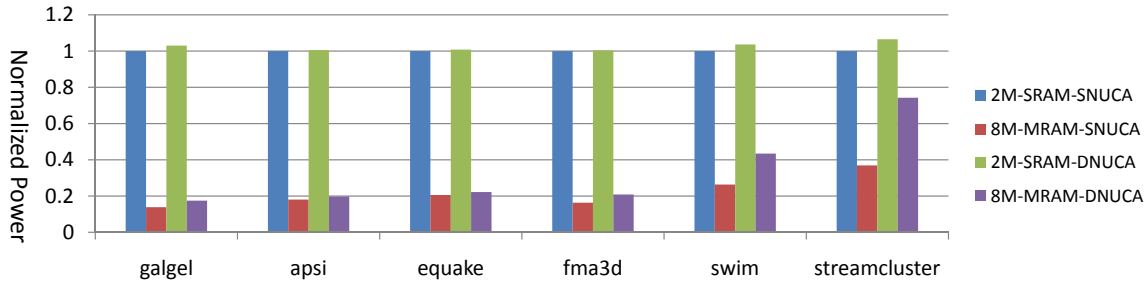


Figure 7: Total power of the SRAM caches and MRAM cache (Normalized by the power of 2MB SNUCA SRAM cache).

Note that thermal issues are one of the major concerns for the adoption of 3D integration technology. A large power saving introduced by replacing SRAM L2 cache with MRAM L2 cache implies that it is good for thermal management in 3D stacking chip, especially when multiple layers of memory are stacked on CMPs.

To summarize, our second conclusion of using the MRAM cache is:

**Observation 2** *Replacing the SRAM L2 cache with a MRAM cache, which has similar area but with larger capacity, can greatly reduce the leakage power, resulting a significant total power saving when leakage power dominates. However, when there are intensive write operations, the dynamic power increase significantly because of the high energy associated with write operations to the MRAM cache, and the total power saving could be reduced.*

The two conclusions show that, if we directly replace the SRAM cache with the MRAM cache in a straightforward way, the performance and power penalties caused by the long latency and high energy associated with the write operations in MRAM cache can offset the performance and power benefits of using the MRAM cache when there are high intensity of write operations to the L2 cache.

## 5 Novel 3D-stacked cache architecture

In this section we propose two techniques in order to mitigate the write operation problem of using MRAM caches. In 5.1, a preemptive write buffer is employed to reduce the stall time caused by the long latency of the write operation to the MRAM cache. In 5.2, an approach of MRAM-SRAM hybrid L2 cache is proposed to reduce the number of write operations to the MRAM cache so that we could improve both the performance and the power. We combine these two

Cache Configuration	Leakage Power
2M SRAM Cache	2.089W
8M MRAM Cache	0.255W

Figure 8: Leakage power of SRAM and MRAM caches at 80 °C.

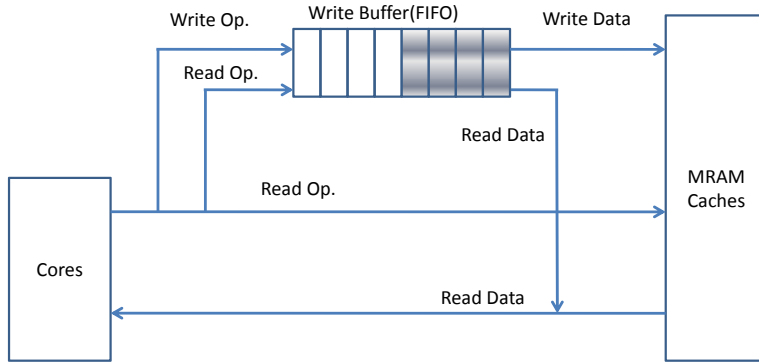


Figure 9: The process structure of cache accesses after adding a write operation buffer.

techniques together as an optimized architecture for the MRAM cache.

## 5.1 A Read-preemptive Write Buffer

The first observation in Section 4.1 shows that the long latency of a write operation to the MRAM has a serious impact on the performance. In the scenario that a write operation is followed by several read operations, the write operation may block the read operations, and the performance degrades. If these read operations do not access the same cache line as the write operation does, the read requests can be bypassed to the cache. In the modern processor, a write buffer is usually employed to bypass the read request. As listed in Table 3, when we use the SRAM cache, a write buffer with four entries works well enough. However, the experimental results in subsection 4.2.1 show that this write buffer is not fit for the MRAM cache. In order to make the MRAM cache work efficiently, we explore the proper size of the write buffer, and propose a read-preemptive management policy for it.

### 5.1.1 The Structure of the Write Buffer

The structure of the write buffer is shown in Figure 9, and it is operated as follows,

- When there is a write operation, the request is sent into the buffer. If there is a *hit* in the buffer (a valid entry of the same cache line is found in the buffer), the data in the entry are replaced, otherwise, the buffer locates an empty entry to the operation. Note that one entry contains one cache line.
- When there is a read operation, the request is sent to both the buffer and the cache. The buffer searches all valid entries. If there is a read *hit* in the buffer, the buffered data are returned. Otherwise data in the MRAM cache are returned.
- The buffer executes (retires) the buffered write operations to the MRAM cache following the retirement policy.
- If the buffer is full, requests from the upper level memory (L1 cache) are blocked till one entry of the buffer is retired.

### 5.1.2 The Exploration of the Buffer Size

The choice of the buffer size is important. The larger the buffer size, the more write operations could be buffered so that the stall cycles caused by full buffer decrease. However, the more buffered data, the longer time it takes the controller

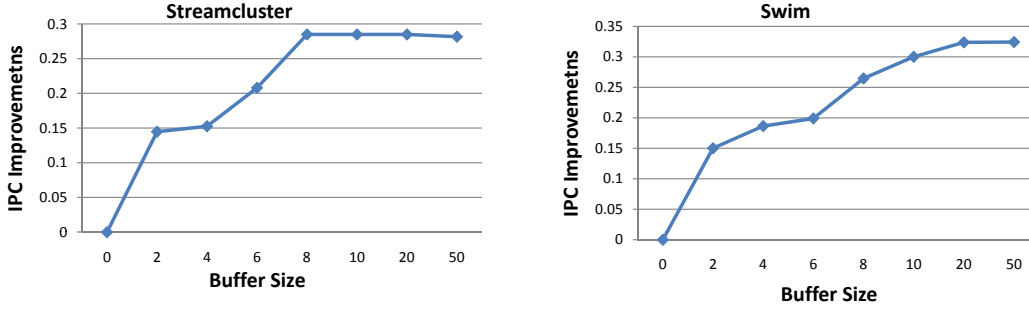


Figure 10: The impact of buffer size. The IPC improvement is normalized by that of 8M MRAM cache without write buffer

to search them all. The design complexity and the area of the buffer also increase with the buffer size. Figure 10 shows the IPCs improvements of using different size of buffers for two benchmarks. Note that the IPCs are normalized by that of using a MRAM cache without the write buffer (buffer size equals to 0 entry). Experimental results show that the size of 20 entries is an optimal choice because, for all benchmarks, the performance increase little with the buffer size larger than 20 entries. Compared to the write buffer for the SRAM cache which has four entries, the write buffer for the MRAM cache requires a much larger size. Note that write buffers in MRAM caches all have the size of 20 entries in the rest of the paper.

### 5.1.3 A Read-preemptive Policy

Since the L2 cache can receive requests both from the upper level memory (L1 cache) and the write buffer, a priority policy is necessary to solve the conflict that a read request and a write request compete for the right of the execution. For a MRAM cache, the latency of the write operation is much larger than that of the read one, and the object is to prevent the write operation from blocking the read one. Thus we have:

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

A read request can also be blocked by a write request that is already in the process of the retirement. The write latency to the MRAM is so large that a retirement of the write request may block one or more read request for a long period, and the performance degrades. In order to mitigate this problem, we propose a 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 retirement if the preemption condition is satisfied. Then, the read operation obtains the right of the execution to the cache. The stalled retirement will retry later.*

This read-preemptive policy tries to execute the read requests in the MRAM cache as early as possible. The drawback is that some retirements need to be re-executed, and the possibility of full buffer increases. The pivot is to find a proper preemption condition. One special method is that the retirement is always stalled when a read request is blocked. It means that there is no preemption condition and the read requests are executed to the cache immediately. Theoretically,

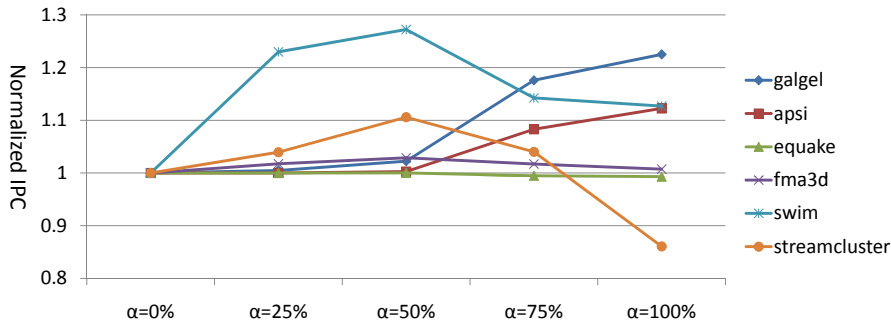


Figure 11: The impact of  $\alpha$  on the performance. The IPC values are normalized by that of using the traditional policy.

if the size of the write buffer is large enough, the read request will never be blocked. However, since the size of the buffer is limited, the increase of the full buffer possibility could also harm the performance. Especially, when the write intensity is high, the non-conditional preemption policy may degrade the performance.

In some cases, it is not a good choice for the read request to stall a write retirement. For example, if the retirement is almost finished, the read request should not stall the retirement process. Consequently, we propose to use the accomplishment degree of the retirement as the preemption condition. Let  $\alpha$  denote the accomplishment degree of the retirement, after which the read request will not stall the retirement process. The Figure 11 compares the IPC of using different  $\alpha$  in the policy. Note that “ $\alpha = 100\%$ ” represents the non-conditional preemption policy and “ $\alpha = 0\%$ ” represents the traditional policy. We can find that, for the benchmarks (*galgel* and *apsi*) with low write intensities, the performance increase as the  $\alpha$  increases, and the non-conditional preemption policy works best. However, for the benchmark with high intensities of write operations, the performance increase at first, and then decrease as the  $\alpha$  increases. Especially, for the benchmark *streamcluster* with the highest write intensity, the performance degrades much with the non-conditional preemption policy. In this work, we set  $\alpha = 50\%$ , and performance of all benchmarks increase with the read-preemptive policy compared to those with the traditional policy. A counter is requested 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. Note that the exact cycles of the write operation could be calculated (except for the case of miss) because the location of the cache bank being accessed is recorded in the write request. However, in order to simplify the design, the cache controller use the average access cycle number instead of calculating the exact number for each request.

The area of the 20 buffer entries can be evaluated as that of a cache whose size is  $20 \times 64\text{Byte}$  (less than 2KB). We use a 7-bit counter to estimate the accomplishment of the retirement. Since the layer area is fit for a 2MB SRAM cache. The area overhead is less than 1%, which is negligible. Similarly, the increase of the leakage power caused by the buffer is also negligible.

Figure 12 and 13 compares the performance and the power after using our read-preemptive write buffer to those of using the traditional architecture. Compared to the IPC of using the SRAM cache in the baseline configuration, the average performance improvements are 9.93% 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 results in Figure 6 and 7, the performance improve a lot, but the power reductions decrease. It is because using our read-preemptive write buffer causes re-executions of some write operations so that the dynamic power increases, and the buffer with larger size also incurs more power.

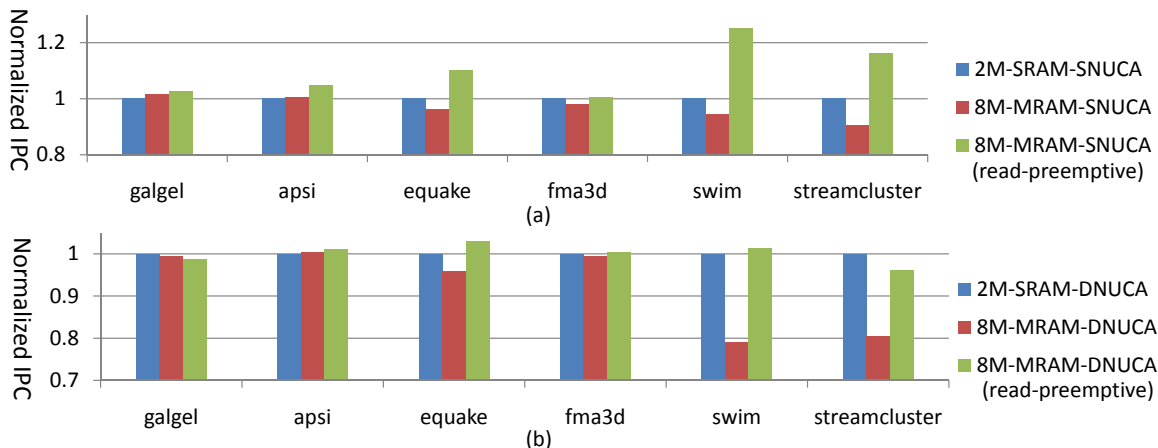


Figure 12: The comparison of IPC among 2M SRAM, 8M MRAM with traditional write buffer, and 8M MRAM with read-preemptive write buffer (Normalized by that of SRAM).

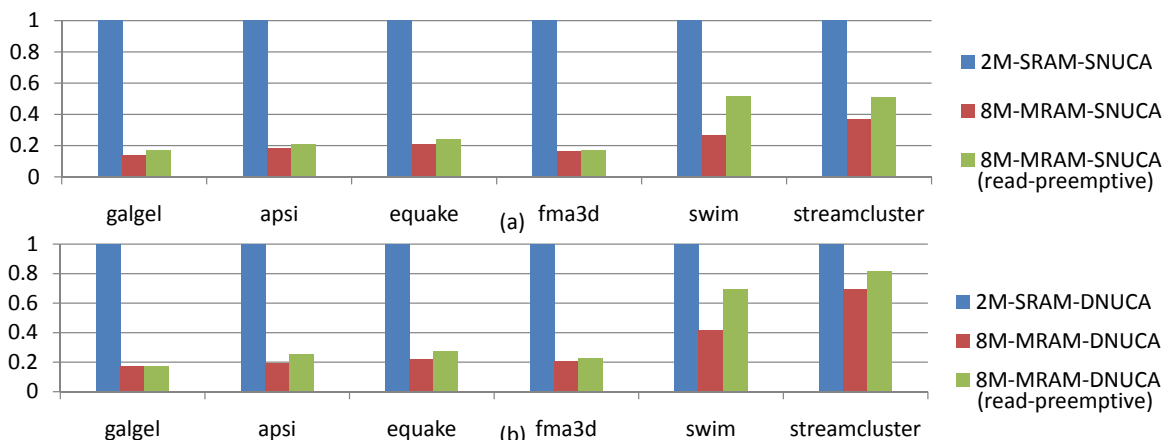


Figure 13: The comparison of total power of 2M SRAM, 8M MRAM with traditional write buffer, and 8M MRAM with read-preemptive write buffer (Normalized by that of SRAM).

## 5.2 MRAM-SRAM Hybrid L2 Caches

While the read-preemptive write buffer, which is introduced above, reduces the number of non-critical blocks and hence hides the long latency of MRAM write operations, the total number of write operations remains the same. In order to reduce the number of write operations to MRAM cells, in this section, we propose another technique called *MRAM-SRAM Hybrid Cache* and show how this novel technique can help to mitigate the MRAM write intensity and

further reduce the dynamic power as well as improve the performance.

### 5.2.1 The Implementation of the Hybrid Cache

The proposed implementation of the hybrid cache is that, instead of building a pure MRAM cache, we compose the ways in each cache set with a large portion of MRAM cache lines and a small portion of SRAM cache lines. The main purpose is to keep as many write-intensive data in the SRAM part of caches as possible and thus reduce the number of write operations to the MRAM part of caches. In this work, for the baseline 32-way set-associative L2 cache design (see Table 3), we design an MRAM-SRAM hybrid L2 cache with 31 ways of MRAM and 1 way of SRAM (*31MIS*).

After having these hybrid cache lines, the second step is how to distribute MRAM cache lines and SRAM ones into separate cache banks. Considering the SRAM part is the minority in the proposed *31MIS* cache, one partitioning alternative is to distribute these SRAM cache lines into different banks so that there are at least several SRAM cache lines close to each processing core. However, this method requires each cache bank as a heterogeneous 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 on one layer. Considering the specialization of the MRAM fabrication process, this method also eliminates the cost advantages of stacking MRAMs on top of processing cores.

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. 14(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 14(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 view. The central location of these SRAM cache banks provides moderate access latencies from all cores. The implementing of the MRAM-SRAM hybrid cache using this method only requires the re-design of the NoC topology and does not have any negative impact from the architectural viewpoint. 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 The Management Policy of the Hybrid Cache

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

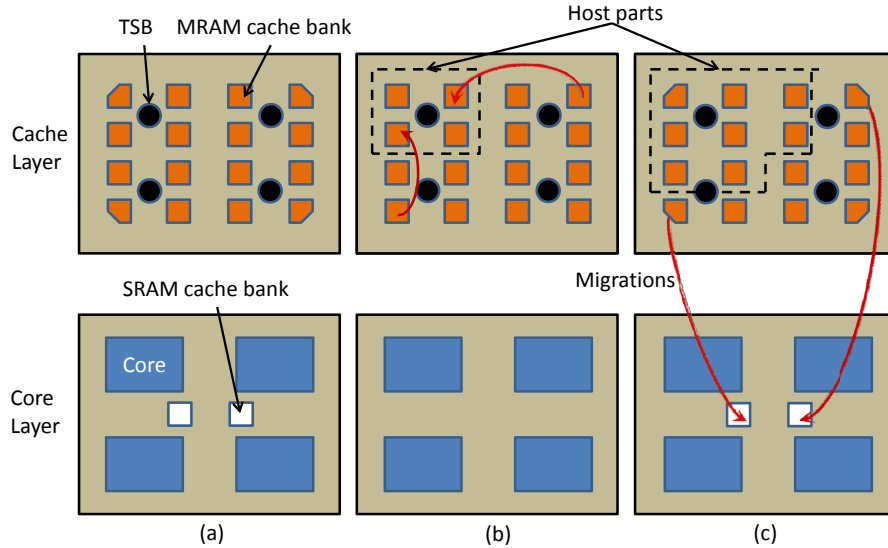


Figure 14: The cache layer from a four-core(TSB) processor used to demonstrate (a)one placement method of SRAM and MRAM cache banks,(b) data migrations in original MRAM caches, (c)data migrations in hybrid MRAM-SRAM caches.

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* in order to differentiate *inter-migration* policy introduced in Section 2. 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 14(b) and (c) compare the banks from which data can be migrated toward the core in upper-left corner. Figure 14(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 Figure 14(b) for the traditional inter-migration. When using the hybrid MRAM-SRAM caches, the *host banks* for a core is redefined as shown in Figure 14(c). Two corresponding data migrations are also shown in Figure 14(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 these SRAM banks are shared for all cores so that their limited sizes may increase L2 miss rates. Note that 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.

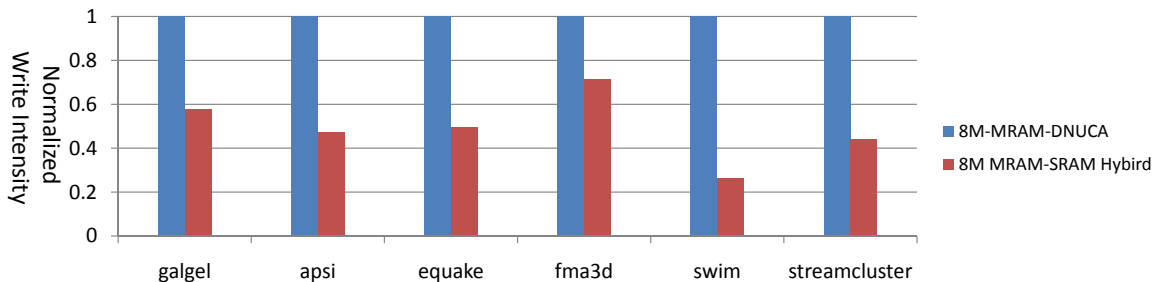


Figure 15: The write intensity to MRAM of using hybrid MRAM-SRAM caches compared to that of pure MRAM caches.

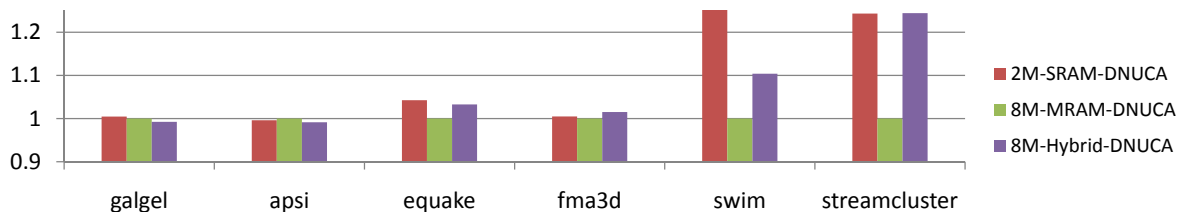


Figure 16: The comparison of IPC among 2M SRAM cache, 8M MRAM pure cache, and 8M MRAM-SRAM hybrid cache (Normalized by the IPC of 8M MRAM pure cache).

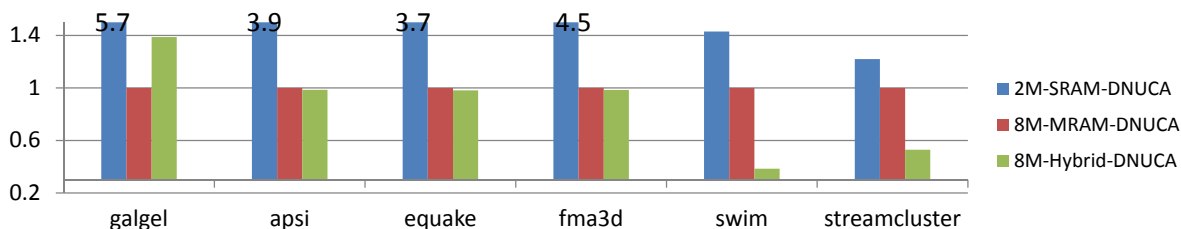


Figure 17: The comparison of total power consumption among 2M SRAM cache, 8M MRAM pure cache, and 8M MRAM-SRAM hybrid cache (Normalized by the total power consumption of 8M MRAM pure cache).

Fig. 15 shows the number of write operations to the MRAM cache per 1K instructions are reduced dramatically by using our hybrid MRAM-SRAM approach. As a result, the dynamic power associated with write operations to MRAM cells is reduced, and the performance is improved. Fig. 16 shows the performance comparison among the proposed MRAM-SRAM hybrid cache structure and its pure SRAM or pure MRAM counterparts. Note that the read-preemptive write buffer is not used here because we want to highlight the benefits of using the hybrid cache structure alone. 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. 17, we show the comparison in terms of power consumption. Only focusing on the total power consumption difference between the pure MRAM cache and the MRAM-SRAM hybrid cache, we find that the total power is reduced except for *galgel*. It is because that both the read and the write intensities in *galgel* are so small that the consumed dynamic power is very low. Consequently, the introduction of SRAM cache lines in the hybrid cache brings the leakage power back and this amount of leakage power overhead eliminates the dynamic power reduction achieved by the hybrid

structure. However, as the write intensity increases, the MRAM-SRAM hybrid cache can lower the total power consumption by reducing the dynamic power while paying a small amount of leakage power overhead. For example, the total power consumption is cut by more than half for the applications such as *swim* and *streamcluster*. On average, after the transition from SRAM caches to MRAM ones, this newly-proposed hybrid cache further reduces the total power by 12.45%.

### 5.3 The Combination of Two Techniques.

We combine the two techniques together as an optimized architecture for the MRAM cache. In this architecture, we get more benefits from the advantages of the MRAM cache, and mitigate the penalties caused by write operations. The performance and power comparisons are listed in Figure 18 and Figure 19, respectively. The average IPC is improved by 14.0% and 4.91% when we compare that of using the MRAM cache in the optimized architecture to that of using the DNUCA MRAM and DNUCA SRAM in a traditional architecture, respectively. The power consumption reductions are 5.8% and 73.5%, respectively.

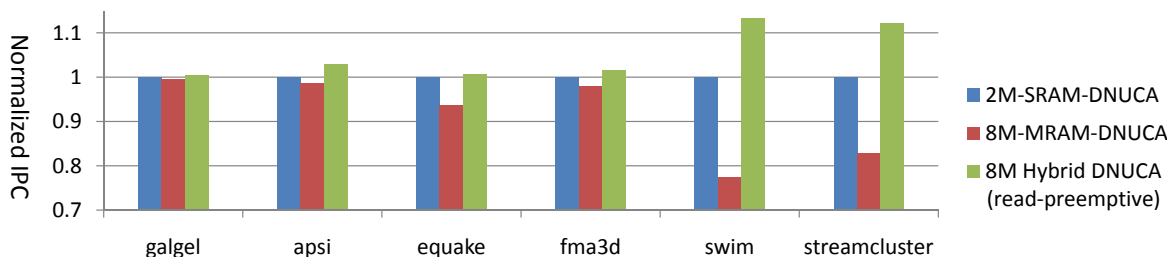


Figure 18: The comparison of IPC among 2M SRAM cache, 8M MRAM pure cache, and 8M MRAM-SRAM hybrid cache with read-preemptive write buffer (Normalized by the IPC of 2M SRAM cache).

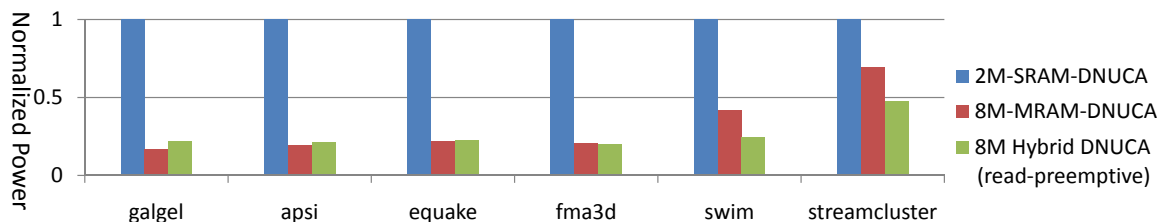


Figure 19: The comparison of total power consumption among 2M SRAM cache, 8M MRAM pure cache, and 8M MRAM-SRAM hybrid cache with read-preemptive write buffer (Normalized by the total power consumption of 2M SRAM cache).

## 6 Related Work

Many researchers have studied the design of microprocessors using 3D technology [4, 27–30]. For example, how to design microprocessor components in 3D were investigated [4, 27]. Automated design flow for 3D microarchitecture evaluation was proposed [31], and stacked processor architecture was studied [28]. To improve the reliability of chips when the process is being shrunk aggressively, 3D stacking can also offer redundant checker processors on the second die without significantly impacting the cost [29]. It is also demonstrated that a “snap-on” layer can be stacked on top

of processor cores to assist in debugging and testing (such as the Introspective 3D chips [30]), or serve as a special acceleration layer (such as the POD system [32]).

Some previous research focused on the performance improvement from using 3D integration to seek DRAM main memory on top of processors. 3D cache models were developed to facilitate architectural level analysis [33,34]. Performance analysis of 3D stacking memory was studied by Loi *et al.* [28]. PicoServer has 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 [35]. Li [19] *et al.* have also reported performance improvement by using stacked SRAM L2 caches for CMPs. Black *et al.* [6] have studied the benefits of stacking a large DRAM or SRAM cache on a Intel Core 2 Duo processor, and achieved considerable performance improvement. 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 [36]. A prototype of 80-core Teraflop processor [8] 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.

There is previous work that studied the benefits of replacing SRAM caches with MRAM caches for a single processor core. The first generation of MRAM, which is called Toggle-Mode MRAM, has been studied as an SRAM substitute for the on-chip L2 cache [11]. The second generation of MRAM, which is called Spin-Torque-Transfer MRAM, has also been studied as a replacement for SRAM L2 caches stacked on top of a single microprocessor [12]. However, both projects only focused on the single core architecture with caches that have uniform access latencies. For multi-core CMP processors, especially when the entire cache capacity becomes much larger, the effect of non-uniform access latency (or non-uniform cache access architecture, also named NUCA) has to be considered. In addition, in NUCA architectures, data migration can also change the cache read/write behavior so that it will be quite different from that of a single core processor.

## 7 Conclusion and Future Work

Magnetic Random Access Memory (MRAM) is a promising candidate to be used as on-chip memories, due to its low leakage power, high density, and fast read speed. The emerging 3D stacking technology enables the heterogeneous integration, making it feasible to stack MRAM as L2 caches for CMPs. In this work, we have presented a cache model for MRAM L2 cache stacking, and conducted the performance and power evaluation. Even though replacing SRAM L2 cache with MRAM can result in significant power savings, the drawback of using the MRAM comes from the long latency and high energy associated with the write operations. As a result, for applications with high intensity of write operations to L2 cache, performance degradation could happen and the power savings could be reduced. Consequently, we propose to use a read-preemptive write buffer with aggressive priority policy, to mitigate the performance penalty caused by the long write latency; we also propose a hybrid MRAM-SRAM L2 cache to reduce the number of write

operations to the MRAM so that the performance is improved and the dynamic power is reduced. Our future work include the evaluation of the impact of other MRAM characteristics on CMP design, such as the non-volatility and the immunity to radiation-induced soft errors.

## References

- [1] 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.
- [2] D. H. Albonesi and I. Koren. Improving the Memory Bandwidth of Highly-Integrated, Wide-Issue, Microprocessor-Based Systems. In *PACT '97: Proceedings of the 1997 International Conference on Parallel Architectures and Compilation Techniques*, pages 126–135, 1997.
- [3] D. Burger, J. R. Goodman, and A. Kagi. Limited Bandwidth to Affect Processor Design. *Micro, IEEE*, 17(6):55–62, 1997.
- [4] 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.
- [5] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon. Demystifying 3D ICs: The Pros and Cons of Going Vertical. *IEEE Design and Test of Computers*, 22(6):498–510, 2005.
- [6] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCaule, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb. Die Stacking (3D) Microarchitecture. In *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 469–479, 2006.
- [7] 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.
- [8] S. Borkar. 3D Technology: A System Perspective. In *Technical Digest of the International 3D System Integration Conference*, pages 1–14, 2008.
- [9] 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 Non-Volatile Memory With Spin Torque Transfer Magnetization Switching: Spin-RAM. In *International Electron Devices Meeting*, pages 459–462, 2005.
- [10] W. Zhao, E. Belhaire, Q. Mistral, C. Chappert, V. Javerliac, B. Dieny, and E. Nicolle. 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.
- [11] 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.
- [12] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen. 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] Z. Diao, Z. Li, S. Wang, Y. Ding, A. Panchula, E. Chen, L-C Wang, and Y. Huai. 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.
- [14] C. Ababei, Y. Feng, B. Goplen, H. Mogal, T. Zhang, K. Bazargan, and S. S. Sapatnekar. Placement and Routing in 3D Integrated Circuits. *IEEE Design and Test of Computers*, 22(6):520–531, 2005.
- [15] J. W. Joyner and J. D. Meindl. Opportunities for Reduced Power Dissipation Using Three-Dimensional Integration. In *Interconnect Technology Conference, 2002. Proceedings of the IEEE 2002 International*, pages 148–150, 2002.
- [16] C. Kim, D. Burger, and S.W. 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.
- [17] <http://www.hpl.hp.com/research/cacti/>.
- [18] G. L. Loi, B. Agrawal, N. Srivastava, S-C Lin, T. Sherwood, and K. Banerjee. 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.

- [19] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. Kandemir. 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.
- [20] Z. Chishti, M. D. Powell, and T. N. Vijaykumar. Distance Associativity for High-Performance Energy-Efficient Non-Uniform Cache Architectures. In *MICRO 36: Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, page 55, 2003.
- [21] 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.
- [22] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy. Introduction to the Cell Multiprocessor. *IBM Journal of Research and Development*, 49(4/5):589–604, 2005.
- [23] P. Kongetira, K. Aingaran, and K. Olukotun. Niagara: A 32-Way Multithreaded SPARC Processor. *IEEE Micro*, 25(2):21–29, 2005.
- [24] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. Högberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A Full System Simulation Platform. *Computer*, 35(2):50–58, 2002.
- [25] <http://www.spec.org/>.
- [26] <http://parsec.cs.princeton.edu/>.
- [27] G. H. Loh, Y. Xie, and B. Black. Processor Design in 3D Die-Stacking Technologies. *IEEE Micro*, 27(3):31–48, 2007.
- [28] 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.
- [29] N. Madan and R. Balasubramonian. Leveraging 3D Technology for Improved Reliability. In *MICRO '07: Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 223–235, 2007.
- [30] S. Mysore, B. Agrawal, N. Srivastava, S-C Lin, K. Banerjee, and T. Sherwood. Introspective 3D Chips. In *Proceedings of the 2006 ASPLOS Conference*, volume 40, pages 264–273, 2006.
- [31] J. Cong, A. Jagannathan, Y. Ma, G. Reinman, J. Wei, and Y. Zhang. An Automated Design Flow for 3D Microarchitecture Evaluation. In *ASP-DAC '06: Proceedings of the 2006 Conference on Asia South Pacific Design Automation*, pages 384–389, 2006.
- [32] D. Woo, J. Fryman, A. Knies, M. Eng, and H-H S. Lee. POD: A 3D-Integrated Broad-Purpose Acceleration Layer. In *IEEE MICRO Special Issue on Accelerator Architectures*, 2008.
- [33] 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.
- [34] P. Jacob, O. Erdogan, A. Zia, P.M. Belemjian, R.P Kraft, and J.F. McDonald. Predicting the Performance of a 3D Processor-Memory Chip Stack. *IEEE Design and Test of Computers*, 22(6):540–547, 2005.
- [35] T. Kgil, S. D'Souza, A. Saidi, N. Binkert, R. Dreslinski, T. Mudge, S. Reinhardt, and K. Flautner. PicoServer: Using 3D Stacking Technology to Enable a Compact Energy Efficient Chip Multiprocessor. *Proceedings of the 2006 ASPLOS Conference*, 41(11):117–128, 2006.
- [36] 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.