

# Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance

Sangyeun Cho      Hyunjin Lee  
Computer Science Department  
University of Pittsburgh  
{cho,abraham}@cs.pitt.edu

## ABSTRACT

The phase-change random access memory (PRAM) technology is fast maturing to production levels. Main advantages of PRAM are non-volatility, byte addressability, in-place programmability, low-power operation, and higher write endurance than that of current flash memories. However, the relatively low write bandwidth and the less-than-desirable write endurance of PRAM remain room for improvement. This paper proposes and evaluates *Flip-N-Write*, a simple microarchitectural technique to replace a PRAM write operation with a more efficient read-modify-write operation. On a write, after quick bit-by-bit inspection of the original data word and the new data word, Flip-N-Write writes either the new data word or the “flipped” value of it. Flip-N-Write introduces a single bit associated with each PRAM word to indicate whether the PRAM word has been flipped or not. We analytically and experimentally show that the proposed technique reduces the PRAM write time by half, more than doubles the write endurance, and achieves commensurate savings in write energy under the same instantaneous write power constraint. Due to its simplicity, Flip-N-Write is straightforward to implement within a PRAM device.

## Categories and Subject Descriptors

B.3 [Memory Structures]: Semiconductor Memories; B.7 [Integrated Circuits]: Types and Design Styles—*Memory technologies*; C.4 [Performance of Systems]: Design studies

## General Terms

Design, Performance, Reliability

## Keywords

Phase-change memory, memory write performance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MICRO’09, December 12–16, 2009, New York, NY, USA.  
Copyright 2009 ACM 978-1-60558-798-1/09/12...\$10.00.

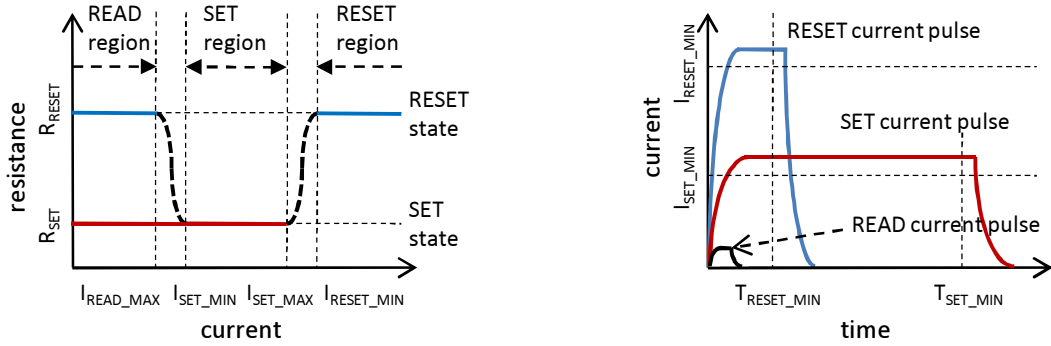
## 1. INTRODUCTION

The phase-change random access memory (PRAM) technology holds great promise to become the next generation non-volatile memory technology thanks to its excellent operating characteristics and scalability [20]. First of all, PRAM is a high-performance byte-addressable device that can directly replace memory with the standard address-data interface such as SRAM and NOR flash. PRAM is a non-volatile device with an adequate retention property (over 10 years) and has higher write endurance than the flash memory. Moreover, unlike popular flash memory devices with the block erase and program requirements, PRAM supports in-place programmability without the need for the erase operation. Lastly, the scalability of the PRAM technology potentially surpasses that of the DRAM and the NAND flash technologies [9, 20]. The PRAM technology is poised to enable a “unified memory” that is tightly integrated in the memory hierarchy of various processor-based systems [10, 18].

This paper proposes and evaluates *Flip-N-Write*, a simple microarchitectural technique to improve a PRAM’s write bandwidth, write energy, and write endurance under an instantaneous write power constraint. The main idea of Flip-N-Write is (i) to replace a write operation with a read-modify-write operation in order to skip bit programming action if not needed (e.g., writing a “0” on “0”) and (ii) to limit the maximum number of bits to program by introducing a “flip bit.” The flip bit indicates whether the associated PRAM word has been flipped or not. By opportunistically re-encoding the data to write to PRAM, Flip-N-Write folds the actual number of bits to program (as well as the number of write steps in an iterative write sequence) to half, potentially doubling the write bandwidth of a PRAM device and improving its write endurance and write energy.

Previously, Yang et al. [24] proposed *data-comparison write* (DCW), an idea essentially identical to above (i). However, DCW does not reduce the data write time of PRAM devices with a standard address-data interface, since the maximum number of bits to update still remains the same. This paper presents a rigorous analysis of the conventional PRAM write operation, the DCW write operation, and the Flip-N-Write’s write operation using statistical and system-level simulation methods.

Due to its simplicity, Flip-N-Write is straightforward to implement within a PRAM chip (or a PRAM macro). Our inspection of the basic algorithm of Flip-N-Write demonstrates that it will not increase the critical read path of PRAM. The major overhead of Flip-N-Write is the additional flip bit we associate with each PRAM word. How-



**Figure 1: PRAM operations.** Typical R-I curve of GST (left). State transitions occur when specific levels of currents are applied to the memory cell. Current pulses during the read, SET, and RESET operations (right). Plots are not to scale.

ever, with 16 bits or more in a PRAM word, the memory array area overhead is kept within 7% while the gain in write energy consumption, write endurance, and write time can be 50% or more. When Flip-N-Write is implemented within a PRAM chip, it becomes transparent to the rest of the system and still offers benefits in the presence of other architecture-level PRAM optimizations [11, 19, 25].

The rest of this paper is organized as follows. Section 2 presents the basic operations of PRAM to form proper background. Section 3 describes Flip-N-Write in detail and compares it with a closely related previous proposal. Section 4 evaluates the proposed technique using a system-level simulation methodology. Related work is summarized in Section 5 and finally, conclusions are drawn in Section 6.

## 2. BASIC PRAM OPERATIONS

### 2.1 Operating principles

PRAM operation is based on the memory cell material having at least two phases with remarkably different properties. The “amorphous phase” shows high resistivity. However, in the “crystalline phase,” the resistivity becomes significantly low—the change can be as large as five orders of magnitude [20]. PRAM stores information in terms of the resistivity of such a phase-change material. The phase-change materials considered for use in PRAM are  $\text{Ge}_2\text{Sb}_2\text{Te}_5$  (GST) or Ag- and In-doped  $\text{Sb}_2\text{Te}$  (AIST). Figure 1(a) shows a typical R-I curve of the GST material.

In PRAM, the phase-change material is crystallized by heating it above its crystallization temperature (SET operation), and it is melt-quenched to make the material amorphous (RESET operation). These operations are controlled by electrical current (see Figure 1(b)): High-power pulses for the RESET operation that places the memory cell into the high-resistance RESET state and moderate power but longer duration pulses for the SET operation returning the cell to the low-resistance SET state. Finally, retrieving data is done with very low power by sensing the device resistivity.

Thanks to its resistive and non-volatile cell properties, PRAM has very low-power read capability. The resistivity of PRAM devices ranges from  $1\text{k}\Omega$  (SET state) to  $1\text{M}\Omega$  (RESET state) or more, making the read current extremely small [2, 5, 7, 12]. When data in PRAM are seldom accessed, the PRAM device can be opportunistically powered off (to save static energy) without losing the data.

### 2.2 PRAM’s write-related limitations

PRAM’s write performance depends on how quickly the device material can be crystallized (SET) or quenched into the amorphous state (RESET). Due to the phase-change material’s physical property, the SET pulse is usually longer than the RESET pulse (Figure 1(b)) and dictates the write speed of PRAM. On the other hand, the SET pulse level is typically 40–80% of the RESET pulse level [20].

As the voltage levels needed for read, SET, and RESET operations are higher than the nominal supply voltage, charge pump circuits are employed. For example, Lee et al. [12] use voltages ranging between  $V_{\text{DD}} + 1$  and  $V_{\text{DD}} + 3$ . Applying a higher voltage than the phase-change threshold voltage [20], a phase-changing write operation draws significantly more current and power than a read operation. Because of the potentially large current flowing during the write operation, especially for a mobile system, many PRAM prototypes support “iterative writing” of smaller data units than memory word to limit the instantaneous current level. For instance, the prototypes described in [7, 12] support  $\times 2$ ,  $\times 4$ , and  $\times 8$  write modes in addition to the fastest  $\times 16$  mode, and the design in [5] performs “serial writing” of only one bit at a time. Depending on the application area, the low write bandwidth of PRAM (given a power constraint) may pose difficulty during system optimization.

When a PRAM device is used as a non-volatile program storage (e.g., to replace a NOR flash chip), the most frequent operation is read (for instruction fetching). In this case, writing to PRAM (i.e., firmware update) is a rare event, and the low write bandwidth issue is relatively minor. However, when PRAM is used for data storage (e.g., to replace a NAND flash chip or even a DRAM chip), writing actions will occur much more frequently and increasing the write bandwidth of PRAM and improving the write-related energy and endurance become critical design issues. Existing chip prototypes have a write endurance ranging between  $10^5$  and  $10^9$  [1, 2, 12], which is sufficient for traditional non-volatile applications, but questionable for fully replacing DRAM [10, 11, 19, 25].

## 3. FLIP-N-WRITE

### 3.1 Basic idea

We improve the write bandwidth, write energy and write endurance of PRAM with Flip-N-Write. The key idea is to

suppress unnecessary bit programming actions by inspecting the old data word before writing the new data word and to opportunistically re-encode the new data word to further minimize bit programming. The following pseudo-code captures the write operation under the Flip-N-Write scheme:

```
// N: word width
// F and F': new and old flip bit values
0: Write(A: address, D: data)
1: D' := Read(A); // old data
2: F' := Read_Flip_Bit(A); // old flip bit
3: if hamming_dist({D,0}, {D',F'}) > N/2
4:   D := ~D; // flip data
5:   F := 1; // set flip bit
6: else
7:   F := 0; // don't flip
8: for each bit in {D',F'} and {D,F}
9:   if they differ, update memory bit
```

The first step of Flip-N-Write is to read existing “old data” from the target address (lines 1 and 2). Next, the old data and the new data are compared bit by bit to determine how many bits differ (line 3). Notice that the flip bit is taken into consideration during this process. Flip-N-Write flips the data (line 4) and the flip bit value (line 5) to write to PRAM if the number of bits to program (computed in line 3) is over  $N/2$  where  $N$  is the memory word width of PRAM. As will be discussed shortly, this “flipping” of data helps strictly bound the actual number of bits to program. Finally, only the bits (in the buffer  $\{D, F\}$ ) that are different than the data in PRAM ( $\{D', F'\}$ ) are actually updated (lines 8 and 9). Again, the flip bit is included in this step.

With Flip-N-Write the maximum number of bits to update never exceeds  $N/2$ ; when the difference (hamming distance) between the old data and the new data of  $N$  bits is greater than  $N/2$ , Flip-N-Write utilizes the flip bit to re-encode the new data such that the actual number of bits to update now falls below  $N/2$ . Due to the addition of the flip bit, the maximum number of bits to update is equal to  $N/2$ . The bottom line is, that the actual update step in line 9 is done for at most  $N/2$  bits.

Given the above property of Flip-N-Write, its benefits are clear. First, write energy and write endurance can be improved; the number of bits to program is decreased since all redundant bit updates (e.g., writing a “0” on “0”) are eliminated. We will study this aspect in the following subsection. Second, write bandwidth can be improved. This is because Flip-N-Write strictly (thus predictably) bounds the actual number of bits to update. With simple look-ahead identification of write bit positions, the write logic in a PRAM device can easily cut the number of write steps by half, by doubling the write unit size without increasing the instantaneous write current. We will discuss this aspect in Section 3.3.

### 3.2 Comparing Flip-N-Write with DCW

To better illustrate the benefit of Flip-N-Write in terms of write energy and write endurance, we compare in this subsection Flip-N-Write and the data-comparison write (DCW) scheme—a closely related previous work by Yang et al. [24]—against the conventional PRAM write scheme. The main metric we use is the number of actual bit updates per memory word write. Note that with the conventional PRAM

write scheme, the number of bits to update equals the memory word width. Further note that for PRAM write energy is much higher than read energy (6–10 times reported in [11]). Both Flip-N-Write and DCW save write energy by introducing small read overhead and cutting down more expensive bit updates.

The pseudo-code of DCW is given in the following:

```
// N: word width
0: Write(A: address, D: data)
1: D' := Read(A);
2: for (i := 0, i < N, i++)
3:   if (D(i) <> D'(i))
4:     update memory bit
5: end for
```

Like Flip-N-Write, DCW replaces a write operation with a read-modify-write operation (line 1). Starting from the first bit to the last bit, each bit in the read buffer ( $D'(i)$ ) is compared with the new data bit ( $D(i)$ ) and the corresponding memory bit is updated if the new data bit differs from the bit in the memory (lines 2–4). It is important to observe that the maximum number of bits to update using DCW is still  $N$ , the word width. The actual bit programming (lines 2–4) can be done sequentially [24] or in parallel [25].

On average, DCW reduces the number of bits to update significantly. Given  $N$  bits in a word, the average number of bits to update (denoted  $\bar{K}$ ) using the DCW scheme is:

$$\bar{K} = \sum_{i=0}^N i \cdot \frac{1}{2^N} \binom{N}{i} = \frac{N}{2} \quad (1)$$

In the above equation (1), we assumed that the probability of a memory bit having a “0” or “1” is  $1/2$ , and so is the probability of a new data bit being a “0” or “1.” Therefore, the probability of the two bits having different values is  $1/2$  because the corresponding combinations are “0 (old bit) and 1 (new bit)” and “1 and 0” out of the four possible combinations. The probability of having  $i$  different bits among the old data and the new data is then  $(1/2)^N \binom{N}{i}$  based on the binomial distribution.

Similarly, the average number of bits to update with Flip-N-Write is calculated as follows:

$$\bar{K} = \sum_{i=0}^{N/2} i \cdot \frac{1}{2^{N+1}} \binom{N+1}{i} + \sum_{i=N/2+1}^{N+1} (N+1-i) \cdot \frac{1}{2^{N+1}} \binom{N+1}{i} \quad (2)$$

Notice that the counting is now split into two cases, of which the latter is for the flipped write case. Table 1 presents Equations (1) and (2) in a series format. The number of bits to update peaks at  $\frac{N}{2}$  with Flip-N-Write when the hamming distance of the old and the new data words reaches  $\frac{N}{2}$  and  $(\frac{N}{2} + 1)$ . The number of bits to update decreases with Flip-N-Write after that point while it keeps increasing with DCW.

Figure 2(a) plots the average number of bits to update per PRAM write using DCW and Flip-N-Write. Further, Figure 2(b) depicts the improvement brought by Flip-N-Write, again in terms of the number of bits to update per write, over the conventional write scheme and the DCW scheme. The average number of bits to update differs not greatly, but

Dist.	(a)	(b)	(c)	(d)
0	$\frac{1}{2}^N \binom{N}{0}$	0	$\frac{1}{2}^{N+1} \binom{N+1}{0}$	0
1	$\frac{1}{2}^N \binom{N}{1}$	1	$\frac{1}{2}^{N+1} \binom{N+1}{1}$	1
2	$\frac{1}{2}^N \binom{N}{2}$	2	$\frac{1}{2}^{N+1} \binom{N+1}{2}$	2
$\frac{N}{2} - 1$	$\frac{1}{2}^N \binom{N}{N/2-1}$	$\frac{N}{2} - 1$	$\frac{1}{2}^{N+1} \binom{N+1}{N/2-1}$	$\frac{N}{2} - 1$
$\frac{N}{2}$	$\frac{1}{2}^N \binom{N}{N/2}$	$\frac{N}{2}$	$\frac{1}{2}^{N+1} \binom{N+1}{N/2}$	$\frac{N}{2}$
$\frac{N}{2} + 1$	$\frac{1}{2}^N \binom{N}{N/2+1}$	$\frac{N}{2} + 1$	$\frac{1}{2}^{N+1} \binom{N+1}{N/2+1}$	$\frac{N}{2}$
$N - 2$	$\frac{1}{2}^N \binom{N}{N-2}$	$N - 2$	$\frac{1}{2}^{N+1} \binom{N+1}{N-2}$	3
$N - 1$	$\frac{1}{2}^N \binom{N}{N-1}$	$N - 1$	$\frac{1}{2}^{N+1} \binom{N+1}{N-1}$	2
$N$	$\frac{1}{2}^N \binom{N}{N}$	$N$	$\frac{1}{2}^{N+1} \binom{N+1}{N}$	1
$N + 1$			$\frac{1}{2}^{N+1} \binom{N+1}{N+1}$	0

**Table 1: Hamming distance of old and new data of  $N$  bits, its probability, and the number of bit updates with DCW ((a) and (b)) and Flip-N-Write ((c) and (d)). For Flip-N-Write, we include the flip bit so that the maximum distance is  $(N + 1)$ .**

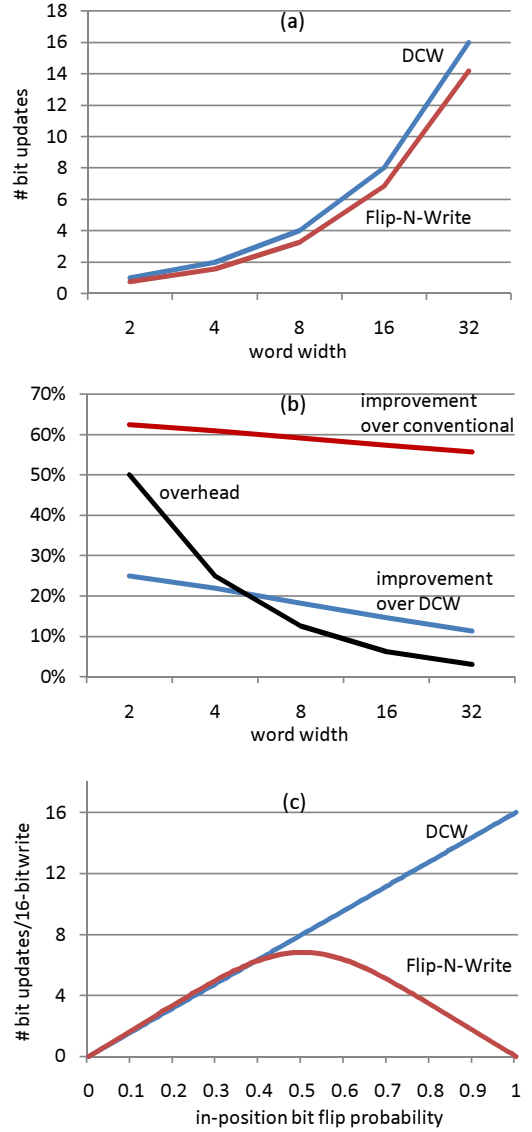
noticeably between DCW and Flip-N-Write. This is because of the binomial distribution; the probability of having many different bits between the old and new data words turns out to be quite small. The reduction in bit updates with Flip-N-Write over the conventional scheme is significant. In the word widths we examined, the reduction is at least 55%. Compared with DCW, Flip-N-Write’s advantage is between 11% (32-bit word) to 25% (2-bit word). The relative advantages we present here are based on the assumption of uniform random bit value distribution. With realistic workloads, we actually see larger benefits with Flip-N-Write (and DCW) over the conventional write scheme (Section 4).

The memory array area overhead of Flip-N-Write is  $1/N$  when the memory word width is  $N$ , and is plotted also in Figure 2(b). It is shown that the overhead decreases quickly as  $N$  increases; it is 12.5% with  $N = 8$  and becomes well less than 10% with  $N = 16$  or more. Considering the usual non-array contribution of 20% or more in the total memory chip area [2, 7, 12], the flip bit overhead is less than what we indicated in Figure 2(b). We will further discuss other design issues with Flip-N-Write in Section 3.4.

Finally, Figure 2(c) depicts the average number of bits to update given an in-position bit flip probability. The question is: How do DCW and Flip-N-Write perform if the number of different bits in the old and the new data words is given as probability? The result shows that Flip-N-Write writes fewer bits than DCW from the in-position bit flip probability of around 39% and above. It is also shown that DCW degenerates into the conventional write scheme if all data bits are different in consecutive writes (e.g., all-“0” data and all-“1” data are written in sequence). In such situations (pathological for DCW), Flip-N-Write offers the maximum relative benefit over DCW and the conventional scheme—either none or single bit update is required depending on how the old data word has been encoded in PRAM.

### 3.3 Write bandwidth

Assuming that address and data transfers are done in a pipelined manner, PRAM’s write bandwidth is strictly limited by the data unit programming time (especially the SET operation latency). Currently, high-density prototypes have



**Figure 2: (a) Average number of bits to update with DCW and Flip-N-Write per word width. (b) Improvement with Flip-N-Write over the conventional scheme and DCW. Flip-N-Write’s overhead is also shown. (c) Average number of bits to update given an “in-position bit flip probability,” which dictates how often a bit in the same position has different values in the old and the new data. The memory word is assumed to be 16 bits.**

a SET latency of 150ns–400ns [2, 7, 12]. PRAM bandwidth is also limited by the write data unit size. Unfortunately, the write data unit size, the number of bits to program simultaneously, is limited (especially in mobile platforms) by the instantaneous write current as we discussed in Section 2.2. Reducing the SET operation latency depends heavily on material, circuit, and integration technologies [20] and is beyond the scope of this paper.

Flip-N-Write improves the PRAM write bandwidth by doubling the write unit size under a write current constraint. For instance, the maximum write current of  $\times 16$  write with Flip-N-Write is the same as the maximum current of  $\times 8$



Number of cores	Four symmetric cores connected with a crossbar
Core pipeline	Intel’s ATOM-like two-issue in-order pipeline with 16 stages [6]
Branch predictor	Hybrid with 4K-entry gshare, local, and selector; 6 cycle misprediction penalty
Hardware prefetch	Four stream prefetchers per core [21]; 16 cache block prefetch distance and 2 prefetch degree
On-chip caches	32KB 4-way L1 I/D caches with 1-cycle latency; 2MB, 16-way shared L2 cache with 11-cycle latency; all caches use LRU replacement and have 64B block size
PRAM memory module	Has a variable number of banks each with a 2KB row buffer and 2KB write queue. Read latency: 1 (row address) + 27 (initial) + 4 (data transfer) = 32 memory cycles; read latency (row buffer hit): 1 (row address) + 1 (column address) + 1 (match) + 4 (data transfer) = 7 memory cycles; read busy time: 27 memory cycles; write busy time (64B cache block): 160 (set time) × 8 (data units) = 1280 memory cycles (conventional), 27 (read latency) + 160 × 8 = 1307 memory cycles (DCW), and 27 + 160 × 4 = 667 memory cycles (Flip-N-Write)

**Table 2: Baseline CMP configuration with timing parameters for system performance evaluation experiment.**

ence logic output and the count1 logic outcome by program enable logic. When the program enable signal for a bit is “low” (i.e., no programming), programming for the bit is suppressed and the SET/RESET enable signal for the bit becomes a don’t care condition.

A modified write driver circuit is presented in Figure 3(c). Note that the SET and RESET enable signals for a bit are AND-gated with the program enable signal. As discussed previously, only when the program enable signal is high, a programming action (SET or RESET) occurs. The changes we added to the original write driver design in [12] is minimal and we do not expect the delays of our extra logic to affect the PRAM clock frequency. The design in [12] reserves a 30ns initialization delay before actual programming sequence commences, which would more than subsume the extra logic delays.

## 4. SYSTEM-LEVEL EVALUATION

Our analysis in Section 3 used statistical methods based on certain assumptions about the bit-level data distribution in memory words. This section performs a system-level evaluation of Flip-N-Write PRAM using realistic workloads. As in Section 3, we will focus on evaluating the conventional write scheme, the data-comparison write (DCW) scheme, and the proposed Flip-N-Write scheme.

### 4.1 Evaluation methodology

We design and perform three experiments in this section: (1) Firmware update, (2) data update, and (3) multicore system performance. Each experiment uses a different workload set and PRAM configuration, as described in the following.

The firmware update experiment employs five embedded benchmark programs from the MiBench suite [4]: `basicmath`, `typeset`, `stringsearch`, `patricia`, and `pgp`. These programs represent relatively large programs in the five categories of the benchmark suite. To compile the programs, we use gcc 2.95.3 (ARM architecture) and gcc 3.4.6 (x86 architecture). First, the programs are compiled with the “-O1” optimization flag and written to the PRAM-based code memory we model. The same programs are then compiled with the “-O3” optimization flag and written again to the same code memory, mimicking a firmware update. The experiment is repeated, this time reversing the order of compiler optimization levels. We will report the average number of actual bit updates per 1,024 code bits, split into SET and RESET operations. The 133MHz PRAM chip we model in this experiment has a 16-bit word width, similar to [12].

The data update experiment uses a set of photo files (in the jpeg format) taken from publicly available New York

Times “Pictures of the Day” color photo series [17]. We selected seven dates from mid- to late-April of 2009. Each date contains 11 to 13 photos mostly having 600×400 pixels. During experiment, photos in individual dates are written into a PRAM device in sequence. As before, we will report the average number of actual bit updates per 1,024 data bits, split into SET and RESET operations. We also employ mp3 music files from two albums: Diana Krall’s *When I Look in Your Eyes* (Verve label) and Igor Stravinsky’s *The Rite of Spring and Firebird Suite* by Ozawa, Chicago Symphony and Leinsdorf, Boston Symphony (RCA Silver label). Krall has 13 tracks and Stravinsky 24 tracks. For generating the mp3 files, we use Nero 8 [16] to rip the CDs at “medium quality” (128Kbps bit rate) and “high quality” (192Kbps). We perform similar experiments with the mp3 files as we do with the photo files. We use the same PRAM chip model as used in the first experiment.

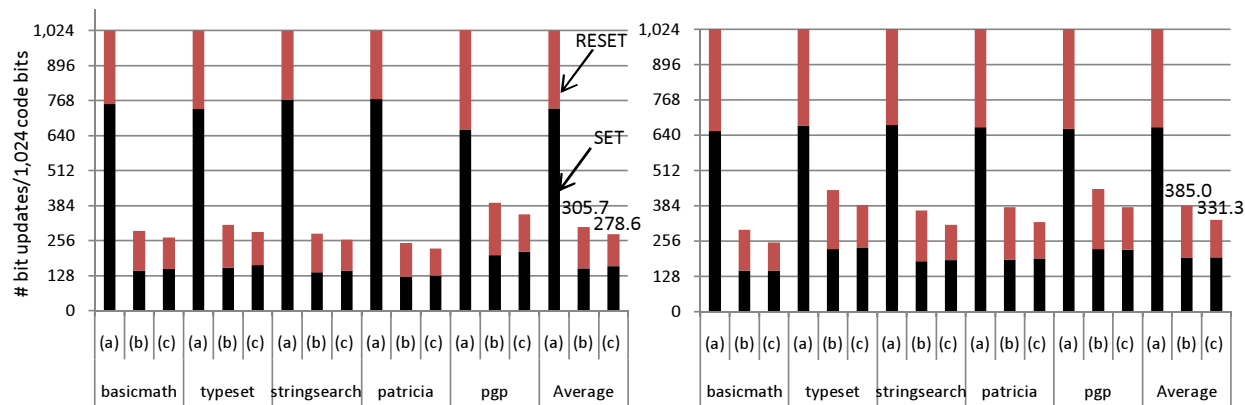
Finally, for the multicore system performance experiment, we use an in-house trace-driven simulator that models a four-core processor system [3]. We generate traces using the PIN binary instrumentation infrastructure [13] to feed the simulator. The processor model details are summarized in Table 2. Unlike the previous two experiments that used a discrete PRAM chip, we use a PRAM-based memory module in this experiment. The memory module has four ×16 PRAM chips and features a 64-bit data interface (e.g., SO-DIMM) running at 400MHz. Like DDR2 memories, the module’s data transfer occurs on both rising and falling edges of the clock, effectively doubling the I/O bandwidth to 800MHz. The read access latency, read busy time, and write busy time parameters are derived from the prototype of [12] and are summarized in Table 2. We choose to use this prototype because it is of a real high-density product grade [15]; we also change the parameters in our sensitivity study. Benchmark programs are drawn from the SPEC 2006 CPU suite [22] (“multiprogrammed” workload), the SPLASH-2 benchmark suite [23] (“multithreaded”), and the SPECjbb benchmark [22] (“server”). The workload details are summarized in Table 3. Performance is defined to be the instructions executed per cycle (IPC) measured during 10B cycles after initial phases (multiprogrammed), the reciprocal of the program execution time (multithreaded), and the reciprocal of the time needed to complete 10,000 transactions (server).

### 4.2 Results

**Firmware update experiment.** Figure 4 presents the result for the two instruction set architectures (ARM and x86) we examined, in terms of the number of bit updates

Workload name (type)	Description
LLMM (multiprogrammed)	464.h264ref, 435.gromacs, 473.astar, and 483.sphinx3 with the reference input
LLHH (multiprogrammed)	464.h264ref, 435.gromacs, 429.mcf, and 459.GemsFDTD with the reference input
MMHH (multiprogrammed)	473.astar, 483.sphinx3, 429.mcf, and 459.GemsFDTD with the reference input
cholesky (multithreaded)	Input: -p4 -B4 -C32768 tk29.0
lu (multithreaded)	Input: -p4 -n4096 -b16
ocean (multithreaded)	Input: -p4 -n258 -e1e-9 -r1500 -t28800
SPECjbb (server)	4 warehouses, total 10,000 transactions

**Table 3: Workloads for multicore system performance evaluation.** Multiprogrammed workloads are formed by co-scheduling four single-thread programs. We compose workloads by including programs with relatively low cache pressure (“L”), medium pressure (“M”), and high pressure (“H”).



**Figure 4: The number of actual bit updates per writing 1,024 bits of program code.** Results for the ARM architecture (left) and the x86 architecture (right) are shown. The three write schemes shown are: (a) Conventional write scheme, (b) DCW, and (c) Flip-N-Write.

per 1,024 code bits. The plots show that both DCW and Flip-N-Write significantly reduce the number of actual code bit updates during firmware update, compared with the conventional write scheme. For updating 1,024 code bits, DCW programs 305.7 bits on average and Flip-N-Write 278.6 bits for the ARM architecture and 385.0 bits (DCW) and 331.3 bits (Flip-N-Write) for the x86 architecture. Compared with the conventional scheme, Flip-N-Write reduces code bit updates by 73% and 68% on average for the ARM and the x86 architectures, respectively. Compared with DCW, Flip-N-Write achieves a reduction of 9% (ARM) and 14% (x86).

For the conventional write scheme, there are more SET operations performed than RESET operations. There are 739.4 SET operations and 284.6 RESET operations on average for the ARM architecture and 667.2 SET operations and 356.8 RESET operations for the x86 architecture. This asymmetry in the frequencies of the two operations is caused by the distribution of zero’s and one’s in the program codes; there are in general more zero’s than one’s in code and data [14]. However, the intrinsic bit value distribution does not directly affect the ratio of SET operations to RESET operations for DCW and Flip-N-Write. The frequencies of the two operations are more or less matched. This is because the two schemes update a memory bit only when its old data value differs from the new data value and skip other bit positions with no transition.

Comparing the two ISAs, we find that DCW and Flip-N-Write have fewer bit updates with the ARM architecture (i.e., RISC) than the x86 architecture (i.e., CISC). This is expected as the x86 architecture’s code density is higher

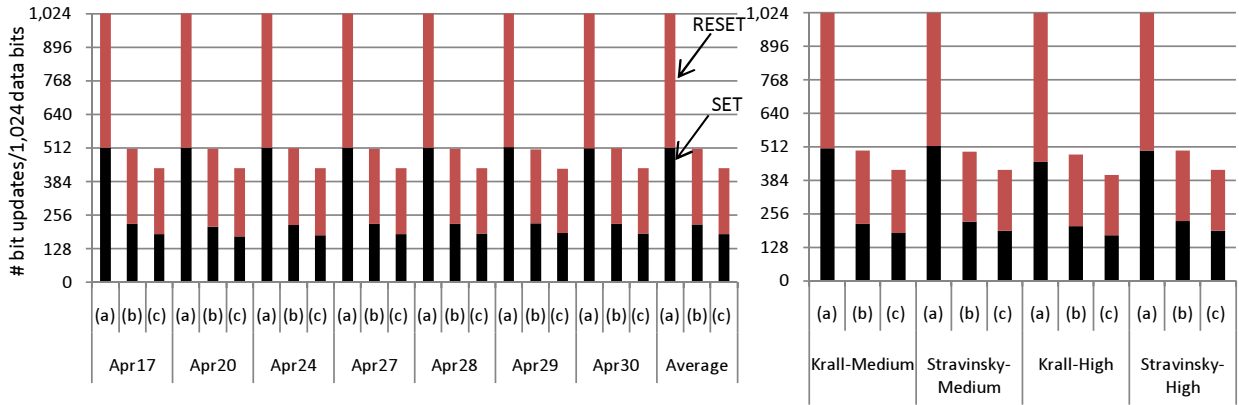
than that of the ARM architecture.<sup>1</sup> Higher code density implies that the bit distribution (“0” or “1”) is more random. This is indirectly evidenced by the ratio of SET frequency to RESET frequency with the conventional scheme; ARM shows wider asymmetry than x86 (SET frequency:RESET frequency = 2.60:1 (ARM) vs. 1.87:1 (x86)). Accordingly, the improvement with Flip-N-Write over DCW (14%) is close to the theoretical improvement rate (15%) in the case of x86 (Figure 2). On the other hand, the absolute reduction of bit updates is larger for the ARM architecture, with both DCW and Flip-N-Write.

In terms of firmware update bandwidth, Flip-N-Write has twice the bandwidth (9.28MB/s) compared with that of the conventional scheme and DCW (4.64MB/s) in the  $\times 16$  mode. Based on the read, SET, and RESET operation energy estimates of [11], the conventional scheme consumes 123.6nJ to update 1KB of firmware on average, DCW 44.8nJ, and Flip-N-Write 39.9nJ for the ARM architecture. For x86, they consume 126.9nJ (conventional), 56.4nJ (DCW), and 47.2nJ (Flip-N-Write). Lastly, in terms of write endurance, Flip-N-Write has over 67% less wear than the conventional scheme and about 9% (ARM) and 14% (x86) less than DCW.

**Data update experiment.** Figure 5 shows the result of our second experiment. As with the first experiment, DCW and Flip-N-Write achieve a large reduction of bit updates compared with the conventional write scheme.

We make several interesting observations. First, the re-

<sup>1</sup>We do not use the 16-bit Thumb instruction set in this experiment.



**Figure 5: The number of actual bit updates per writing 1,024 bits of photo data (left) and music data (right). The three write schemes shown are: (a) Conventional write scheme, (b) DCW, and (c) Flip-N-Write.**

duction with DCW and Flip-N-Write is smaller than that for the firmware update experiment. The reason is, the compressed file formats (jpeg and mp3) randomize the bit value distributions in the data files we examine. This is shown by the ratio of SET and RESET operation frequencies for the conventional write scheme; the ratio is close to 1:1. Slight asymmetry in the ratio is observed in high-quality mp3 files. In this case, “high-quality” implies less compression and less randomization.

The number of bit updates with DCW approaches 512 bits per 1,024 data bits (exactly the half), as the theoretical study suggests (equation (1)). The improvement with Flip-N-Write over DCW is 14%–15%, again very close to our theoretical analysis result (Figure 2(b)). We expect that uncompressed data will have much less randomized bit value distributions (across a sequence of data written to the same memory location) [14] and we predict the result with uncompressed data to be more like Figure 4 than Figure 5.

As discussed earlier, Flip-N-Write achieves twice the data update bandwidth compared with that of the conventional scheme and DCW under a given instantaneous write current constraint. In terms of write energy, the conventional scheme consumes 133.9nJ on average for 1KB of photo files whereas DCW and Flip-N-Write consume 76.3nJ and 65.5nJ each. For music files, they consume 134.8nJ (conventional), 73.5nJ (DCW), and 62.7nJ (Flip-N-Write) on average. In terms of write endurance, Flip-N-Write has over 57% less wear than the conventional scheme and at least 14% less wear than DCW.

**Multicore system performance experiment.** Figure 6 presents the average memory access latency (AMAL) and the relative performance of the three PRAM write schemes. In addition to the three write schemes we examine, we define an unrealistic yet intuitive scheme called “NoDelay” to put the studied schemes in perspective. With NoDelay, PRAM has perfect, zero-latency write capability with no bank lockup due to programming. In all other schemes, PRAM banks under programming are unavailable for read access. Performance of the three studied write schemes is presented relative to that of NoDelay.

It is clearly shown in Figure 6 that AMAL and the performance of a workload are significantly improved with Flip-N-Write, compared to the other two schemes; it cuts down AMAL by 47% on average, relative to the conventional scheme.

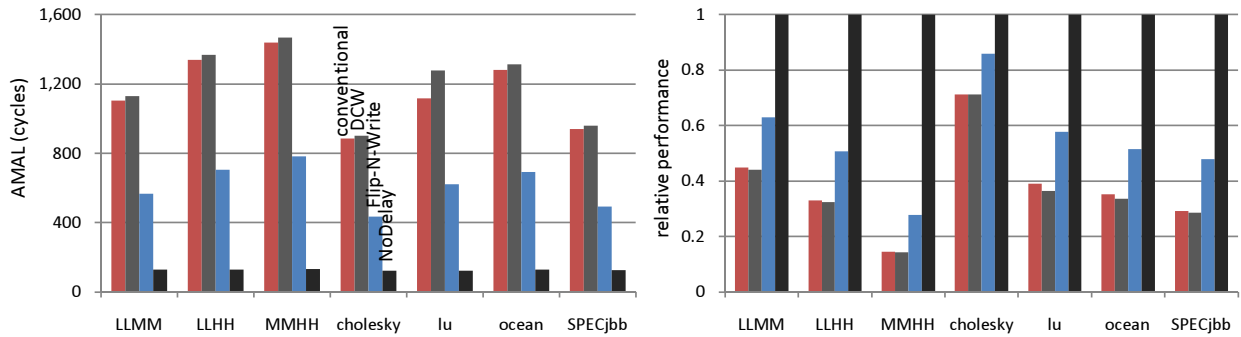
In contrast, DCW degrades AMAL, albeit slightly. We also saw similar boost in system performance with Flip-N-Write. With the slow SET operation, the performance of the examined workloads is dominantly affected by PRAM bank lockup, even with a generous write queue size of 2kB per bank. Well less than half of all read accesses find their target bank available immediately; the rest experience (usually significantly) longer memory latency because the bank has been locked up or the write queues are full.

As hinted above, both AMAL and system performance suffer with the three PRAM write schemes, compared with NoDelay. AMAL degradation (averaged over workloads) is as large as  $9.08\times$  (conventional),  $9.42\times$  (DCW), and  $4.81\times$  (Flip-N-Write). Performance degradation is  $2.88\times$  (conventional),  $2.95\times$  (DCW), and  $1.91\times$  (Flip-N-Write). The PRAM memory module we initially model based on a product-grade PRAM prototype, assisted with reasonable architectural support (8 banks, per-bank row buffer, per-bank write queue, and 2MB L2 cache), simply falls short of meeting the write bandwidth demand of a small-scale multicore processor.

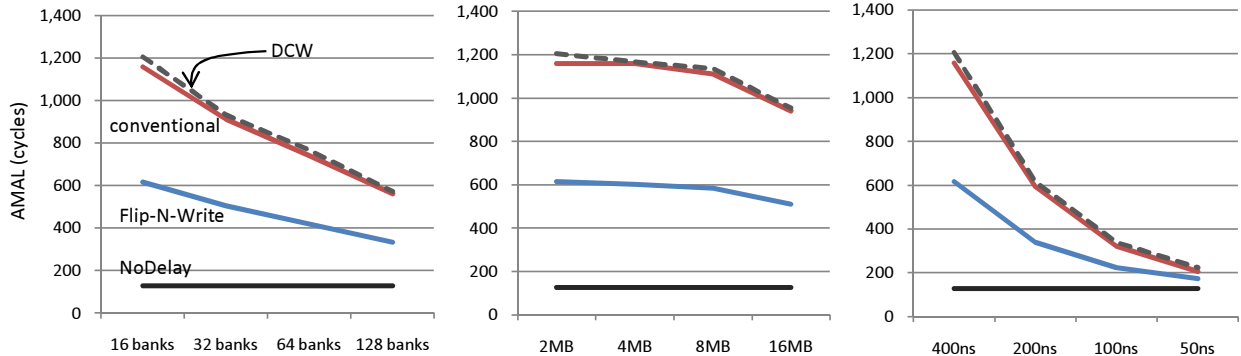
Lastly, Figure 7 presents the result of a sensitivity study on AMAL. We consider three parameters: Number of PRAM banks, L2 cache size, and SET operation latency. Each of these parameters affect the availability of a PRAM bank targeted by a memory read. The first two parameters represent potential microarchitectural improvements (at cost) that increase PRAM bandwidth and reduce traffic to PRAM. The third parameter is driven largely by the PRAM technology.

We derive three messages from the result. First, the microarchitectural techniques help reduce the impact of long SET latency. AMAL is reduced from 1159 (cycles) down to 560 when the number of banks is increased from 16 to 128 (conventional). By increasing the cache size from 2MB to 16MB, AMAL is improved by 19% (conventional).<sup>2</sup> While the obtained improvement in AMAL is respectable, the two microarchitectural techniques were unable to close the AMAL gap to NoDelay. For instance, the gap with 128 banks is still  $4.37\times$  or 432 cycles, using a modest 1.6GHz processor clock. Second, as expected, solving the very source of

<sup>2</sup>This result is rather disappointing; only cholesky and ocean saw significant improvement from the increased cache capacity.



**Figure 6:** Average memory access latency for reads (left). Relative performance to NoDelay (right). We show the result for PRAM with 16 internal banks.



**Figure 7:** AMAL sensitivity to the number of PRAM banks (left). AMAL sensitivity to L2 cache size (middle). AMAL sensitivity to SET operation latency (right). Except the parameter being varied, all other machine parameters follow the default setting used for Figure 6.

the problem—long SET latency—proves to be immediately rewarding. At 50ns SET latency, the gap between the conventional scheme and NoDelay closes to  $1.60\times$  or 77 cycles. With Flip-N-Write, the gap is only  $1.34\times$  and 44 cycles. This sharp contrast underscores the future direction in PRAM research and development—how to reduce the severe bank lockup impact caused by long SET operation latency if PRAM is to competitively serve main memory designs. Third, Flip-N-Write consistently provides benefits in all design points we examined. This is because the underlying mechanisms used by the microarchitectural techniques and Flip-N-Write are different and synergistic. Flip-N-Write works at the lowest level (bit cell) to filter out unnecessary bit programming.

## 5. RELATED WORK

Vast PRAM research so far has concentrated on electro-thermal properties of phase change materials, circuit-level device implementation, and process integration.<sup>3</sup> The exploration of microarchitectural and architectural optimizations for PRAM and PRAM-based systems has begun rather recently. We find that the work by Yang et al. [24], Lee et al. [11], Zhou et al. [25], and Qureshi et al. [19] are the most closely related work to ours. Because we’ve discussed in detail the work of Yang et al. throughout this paper, especially in Section 3.2 and Section 3.3, we will only summarize

<sup>3</sup>Interested readers are referred to Raoux et al. [20] for literature.

the three other papers here, all of which look at design and optimization issues for PRAM-based main memory.

Lee et al. show in their experiment with a baseline design using a DDR2-like configuration and a set of parallel workloads that PRAM gives a  $1.6\times$  delay penalty and a  $2.2\times$  energy penalty compared with DRAM. To close this gap, they explore two microarchitectural techniques: (1) Multiple row buffers (inside a PRAM chip) and (2) partial writes enabled by introducing multiple dirty bits in the cache blocks. Employing multiple row buffers helps improve the read latency and cut down the write energy and endurance through write coalescing. The multiple row buffers bring the PRAM delay and energy disadvantages down to  $1.2\times$  and  $1.05\times$ , respectively. Partial writes reduce the number of bit updates by not writing untouched, clean data portion in a dirty cache block to the main memory when the cache block is replaced. They report an average PRAM module lifetime of 5.6 years with the two techniques. Flip-N-Write is complementary to them because it exploits the actual content of old and new data words rather than architecturally visible events (e.g., writes occurring on the same address).

Zhou et al. propose and evaluate three architectural techniques to prolong the lifetime of 3D stacked PRAM-based main memory: Redundant bit write removal, an idea similar to [24], shifting of data in a PRAM row periodically to distribute hot spots, and segment swapping to distribute data writes with coarse granularity such as a multiple of pages. The proposed techniques, when used together, were shown to significantly increase PRAM’s lifetime to 13 years

(for MLC PRAM) to 22 years (for SLC PRAM) on average, from less than one year.

Qureshi et al. examine a hybrid main memory architecture having a DRAM buffer and PRAM main memory. The DRAM buffer becomes a cache to the lower-level PRAM main memory and filters out detrimental write traffic to the main memory. Their evaluation using a 16-core processor shows that the hybrid main memory can reduce page faults by  $5\times$  and provide a speedup of  $3\times$ . They also explore techniques to reduce write traffic: Lazy write, line level write-back, and page level bypass. They reduce the write traffic by  $3\times$  and increase the average lifetime of PRAM from 3 years to 9.7 years. It will be interesting to combine the techniques of Zhou et al. and Qureshi et al. with Flip-N-Write (except redundant bit write removal) as they will create synergy.

Lastly, while not sharing the goal and context of our work, researchers have examined strategies to exploit asymmetry in SRAM memory cells. Like our proposed approach, Kumar et al. [8] used a memory cell flipping technique to combat memory cell degradation caused by negative bias temperature instability (NBTI). In order not to continuously “stress” a specific PMOS device in a 6T SRAM cell, they periodically flip the cell content to “relax” the two PMOS devices in the cell in an alternating manner. Their design is however unaware of the memory content and simply flips all the memory bits from time to time. Moshovos et al. [14] observed that there are more zero bits in memory than are ones. Accordingly, they proposed and studied an asymmetric memory cell design that sheds less leakage current when the bit stored in a memory cell is zero. They do not consider flipping data bits to further cut down leakage current.

## 6. CONCLUSIONS

Based on the observation that write operation is much more expensive than read operation for PRAM, this paper proposed and evaluated Flip-N-Write, a simple microarchitectural technique to enhance the write performance, energy and endurance of PRAM. The following summarizes our contributions and conclusions:

- We analyzed in detail the two PRAM write optimization strategies using a read-modify-write sequence: DCW (data-compare write) [24] and the proposed Flip-N-Write scheme. With its data re-encoding capability, Flip-N-Write is strictly better than DCW in terms of the average number of bits to program per write and cuts the maximum number of bits to program per write by half. For the word widths of 2 bits to 32 bits, the reduction in bit updates with Flip-N-Write over the conventional write scheme is 56% (32-bit word) to 63% (2-bit word). Flip-N-Write also reduces the memory array busy time due to programming to nearly half. The deterministic write delay of Flip-N-Write is considered a crucial advantage, enabling improved PRAM write performance without changing the rest of the system.
- Using system-level simulation methods, we analyze the operating characteristics of the conventional PRAM write operation, the DCW write operation, and the Flip-N-Write write operation when the PRAM device is used as a program storage (replacement of NOR flash), a data storage (replacement of NAND flash), and main memory (replacement of DRAM). The ex-

perimental result agrees with the analytical study we perform. Additionally, we showed that reducing the bank lockup time due to memory writes will remain a critical problem for PRAM to be fully adopted as the main memory technology.

The proposed Flip-N-Write scheme can be implemented within a PRAM device and is completely transparent to the rest of the system. Accordingly, the benefits of other PRAM-aware architectural optimizations (e.g., Lee et al. [11]) will not be compromised when Flip-N-Write is in place. On the other hand, Flip-N-Write can help simplify the design of the architectural PRAM optimization strategies since it takes care of unnecessary PRAM updates at the lowest, bit level.

## Acknowledgment

We thank the anonymous reviewers for their constructive comments. This work was supported in part by NSF grants CCF-0702236 and CCF-0952273.

## 7. REFERENCES

- [1] S. J. Ahn et al. “Highly Manufacturable High Density Phase Change Memory of 64Mb and Beyond,” *Proc. Int’l Electron Devices Meeting (IEDM)*, May 2004.
- [2] F. Bedeschi et al. “A Multi-Level-Cell Bipolar-Selected Phase-Change Memory,” *Proc. IEEE Int’l Solid-State Circuits Conf. (ISSCC)*, February 2008.
- [3] S. Cho, S. Demetriades, S. Evans, L. Jin, H. Lee, K. Lee, and M. Moeng. “TPTS: A Novel Framework for Very Fast Manycore Processor Architecture Simulation,” *Proc. Int’l Conf. Parallel Processing (ICPP)*, September 2008.
- [4] M. R. Guthaus et al. “MiBench: A Free, Commercially Representative Embedded Benchmark Suite,” *Proc. Annual Workshop Workload Characterization (WWC)*, December 2001.
- [5] S. Hanzawa et al. “A 512kB Embedded Phase Change Memory with 416kB/s Write Throughput at 100 $\mu$ A Cell Write Current,” *Proc. IEEE Int’l Solid-State Circuits Conf. (ISSCC)*, February 2007.
- [6] Intel. *Intel Atom Processor*, <http://www.intel.com/technology/atom/>.
- [7] S. Kang et al. “A 0.1 $\mu$ m 1.8V 256Mb 66MHz Synchronous Burst PRAM,” *Proc. IEEE Int’l Solid-State Circuits Conf. (ISSCC)*, February 2006.
- [8] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar. “Impact of NBTI on SRAM Read Stability and Design for Reliability,” *Proc. Int’l Symp. Quality Electronic Design (ISQED)*, March 2006.
- [9] S. Lai. “Current status of the phase change memory and its future,” *Proc. Int’l Electron Devices Meeting (IEDM)*, May 2003.
- [10] C. Lam. “Cell Design Considerations for Phase Change Memory as a Universal Memory,” *Proc. Symp. VLSI Technology, Systems and Applications (VLSI-TSA)*, April 2008.
- [11] B. C. Lee, E. İpek, D. Burger, and O. Mutlu. “Architecting Phase Change Memory as a Scalable DRAM Alternative,” *Proc. Int’l Symp. Computer Architecture (ISCA)*, June 2009.
- [12] K.-J. Lee et al. “A 90nm 1.8V 512Mb Diode-Switch PRAM with 266MB/s Read Throughput,” *Proc. IEEE*

- Int'l Solid-State Circuits Conf. (ISSCC)*, February 2007.
- [13] C.-K. Luk et al. "Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation," *Proc. Int'l Conf. Programming Languages Design and Implementation (PLDI)*, June 2005.
- [14] A. Moshovos, B. Falsafi, F. N. Najm, and N. Azizi. "A Case for Asymmetric-Cell Cache Memories," *IEEE Trans. Very Large Scale Integration Systems (TVLSI)*, 13(7), July 2005.
- [15] M. Mouthaan. "Mass production phase-change RAM in June," <http://www.hardware.info>, May 2009.
- [16] Nero AG. <http://www.nero.com>.
- [17] New York Times. Pictures of the Day, <http://www.nytimes.com/pages/multimedia/>.
- [18] Numonix. "The basics of phase change memory (PCM) technology," *Technical White Paper*, [http://www.numonyx.com/Documents/WhitePapers/PCM\\_Basics\\_WP.pdf](http://www.numonyx.com/Documents/WhitePapers/PCM_Basics_WP.pdf), 2008.
- [19] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. "Scalable High Performance Main Memory System Using Phase-Change Memory Technology," *Proc. Int'l Symp. Computer Architecture (ISCA)*, June 2009.
- [20] S. Raoux et al. "Phase-change random access memory: A scalable technology." *IBM J. Res. & Dev.*, 52(4/5), July/September 2008.
- [21] B. Sinharoy et al. "POWER5 system microarchitecture," *IBM J. Res. & Dev.*, 49(4/5), July/September 2005.
- [22] Standard Performance Evaluation Corporation. <http://www.specbench.org>.
- [23] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. "The SPLASH-2 Programs: Characterization and Methodological Considerations," *Proc. Int'l Symp. Computer Architecture (ISCA)*, June 1995.
- [24] B.-D. Yang et al. "A Low Power Phase-Change Random Access Memory using a Data-Comparison Write Scheme," *Proc. IEEE Int'l Symp. Circuits and Systems (ISCAS)*, May 2007.
- [25] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," *Proc. Int'l Symp. Computer Architecture (ISCA)*, June 2009.